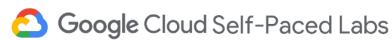


Começar o laboratório

01:15:00

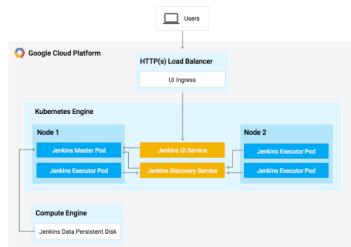
Entrega contínua com o Jenkins no Kubernetes Engine

1 hora 15 minutos Gratuito ★★★★☆ Avaliar laboratório

GSP051

Visão geral

Neste laboratório, você aprenderá a configurar um canal de entrega contínua usando o Jenkins no Kubernetes Engine. O Jenkins é o servidor de automação usado por desenvolvedores que integram com frequência o próprio código em um repositório compartilhado. A solução que você criará neste laboratório será semelhante ao diagrama abaixo:



[Veja mais detalhes](#) sobre como usar o Jenkins no Kubernetes.

Atividades deste laboratório

Neste laboratório, você realizará as seguintes tarefas:

- Provisionar um aplicativo do Jenkins em um cluster do Kubernetes Engine
- Configurar seu aplicativo do Jenkins usando o Gerenciador de pacotes Helm
- Conhecer os recursos de um aplicativo do Jenkins
- Criar e testar um canal do Jenkins

Pré-requisitos

Este é um laboratório de nível avançado. Antes de começar, você precisa saber pelo menos os conceitos básicos de programação de shell, do Kubernetes e do Jenkins. Veja alguns Qwiklabs para você se preparar:

- [Introdução ao Docker](#)
- [Hello Node Kubernetes](#)
- [Como gerenciar implantações com o Kubernetes Engine](#)
- [Como configurar o Jenkins no Kubernetes Engine](#)

Quando estiver tudo pronto, role para baixo e saiba mais sobre o Kubernetes, o Jenkins e a entrega contínua.

O que é o Kubernetes Engine?

O Kubernetes Engine é a versão hospedada do Kubernetes do Google Cloud, um gerenciador de cluster e sistema de orquestração avançado para contêineres. O Kubernetes é um projeto de código aberto que pode ser usado em muitos ambientes diferentes, desde laptops a clusters de vários nós com alta disponibilidade e de máquinas virtuais a bare metal. Como mencionado acima, os apps do Kubernetes são desenvolvidos em contêineres. São aplicativos leves com todas as dependências e bibliotecas necessárias para executá-los. Essa estrutura subjacente torna os aplicativos Kubernetes altamente disponíveis, seguros e rápidos de implantar, o que é ideal para desenvolvedores de nuvem.

O que é o Jenkins?

O Jenkins é um servidor de automação de código aberto que permite orquestrar com flexibilidade canais de versão, teste e implantação. Com o Jenkins, os desenvolvedores podem fazer iterações rápidas em projetos sem se preocupar com problemas de sobrecarga que podem ser gerados pela entrega contínua.

O que é a entrega/implantação contínua?

Quando é preciso configurar um canal de entrega contínua (CD, na sigla em inglês), a implantação do Jenkins no Kubernetes Engine oferece benefícios importantes em comparação com uma implantação padrão baseada em VM.

Se você usar contêineres no processo de criação, um host virtual poderá executar jobs em vários sistemas operacionais. O Kubernetes Engine oferece ephemeral build executors (executores de compilação temporários), que são usados somente quando as versões estão ativamente em execução, deixando recursos para outras tarefas do cluster, como jobs de processamento em lote. Outra vantagem do uso de executores de compilação temporários é a velocidade, já que eles são iniciados em questão de segundos.

Além disso, o Kubernetes Engine vem com o balanceador de carga do Google, que pode ser usado para automatizar o roteamento de tráfego da Web para suas instâncias. O balanceador de carga processa a terminação SSL e usa um endereço IP global configurado com a rede de backbone do Google. Com a interface da Web, esse balancer define para quais lavors seus usuários a sua instância no ambiente nlin

GSP051	-/100
Visão geral	
Configuração	
Baixe o código fonte	
Como provisionar o Jenkins	
Configurar Helm	
Configure e Instale o Jenkins	
Conecte-se ao Jenkins	
Entenda o aplicativo	
Como implantar o aplicativo	
Como criar um canal do Jenkins	
Como criar o ambiente de desenvolvimento	
Inicie a implantação	
Como implantar uma versão canário	
Como implantar a versão na produção	
Teste seu conhecimento	
Parabéns!	

caminho mais rápido possível.

Agora que você aprendeu um pouco sobre o Kubernetes, o Jenkins e como os dois interagem em um pipeline de CD, é hora de criar um.

Configuração

Antes de clicar no botão Start Lab

Leia estas instruções. Os laboratórios são cronometrados e não podem ser pausados. O timer é iniciado quando você clica em **Começar o laboratório** e mostra por quanto tempo os recursos do Google Cloud ficarão disponíveis.

Este laboratório prático do Qwiklabs permite que você realize as atividades em um ambiente real de nuvem, não em uma simulação ou demonstração. Você receberá novas credenciais temporárias para fazer login e acessar o Google Cloud durante o laboratório.

O que é necessário

Para fazer este laboratório, você precisa ter:

- acesso a um navegador de Internet padrão (recomendamos o Chrome);
- tempo para concluir as atividades.

Observação: não use seu projeto ou sua conta do Google Cloud neste laboratório.

Observação: se estiver usando um dispositivo Chrome OS, abra uma janela anônima para executar o laboratório.

Como iniciar seu laboratório e fazer login no console do Google Cloud

1. Clique no botão **Começar o laboratório**. Se for preciso pagar, você verá um pop-up para selecionar a forma de pagamento. No painel **Detalhes do laboratório** à esquerda, você verá o seguinte:

- O botão **Abrir Console do Cloud**
- Tempo restante
- As credenciais temporárias que você vai usar neste laboratório
- Outras informações se forem necessárias

2. Clique em **Abrir Console do Google**. O laboratório ativa recursos e depois abre outra guia com a página **Fazer login**.

Dica: coloque as guias em janelas separadas lado a lado.

Observação: se aparecer a caixa de diálogo **Escolher uma conta**, clique em **Usar outra conta**.

3. Caso seja preciso, copie o **Nome de usuário** no painel **Detalhes do laboratório** e cole esse nome na caixa de diálogo **Fazer login**. Clique em **Avançar**.

4. Copie a **Senha** no painel **Detalhes do laboratório** e a cole na caixa de diálogo **Olá**. Clique em **Avançar**.

Importante: você precisa usar as credenciais do painel à esquerda. Não use suas credenciais do Google Cloud. Ensiná.

Observação: se você usar sua própria conta do Google Cloud neste laboratório, é possível que receba cobranças adicionais.

5. Acesse as próximas páginas:

- Aceite os Termos e Condições.
- Não adicione opções de recuperação nem autenticação de dois fatores (porque essa é uma conta temporária).
- Não se inscreva em testes gratuitos.

Depois de alguns instantes, o console do GCP vai ser aberto nesta guia.

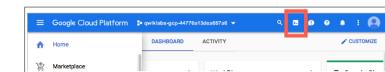
Observação: para ver uma lista dos produtos e serviços do Google Cloud, clique no **Menu de navegação** no canto superior esquerdo.



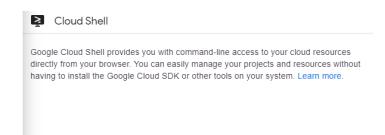
Ative o Google Cloud Shell

O Google Cloud Shell é uma máquina virtual com ferramentas de desenvolvimento. Ele conta com um diretório principal permanente de 5 GB e é executado no Google Cloud. O Google Cloud Shell permite acesso de linha de comando aos seus recursos do GCP.

1. No Console do GCP, na barra de ferramentas superior direita, clique no botão **Abrir o Cloud Shell**.



2. Clique em **Continue (continuar)**:



Demora alguns minutos para provisionar e conectar-se ao ambiente. Quando você está conectado, você já está autenticado e o projeto é definido como seu **PROJECT_ID**. Por exemplo:



gcloud é a ferramenta de linha de comando do Google Cloud Platform. Ele vem pré-instalado no Cloud Shell e aceita preenchimento com tabulação.

É possível listar o nome da conta ativa com este comando:

```
gcloud auth list
```

Saída:

```
ACTIVE: *
ACCOUNT: student-01-xxxxxxxxxx@qwiklabs.net
To set the active account, run:
$ gcloud config set account 'ACCOUNT'
```

É possível listar o ID de projeto com este comando:

```
gcloud config list project
```

Saída:

```
[core]
project = <project_ID>
```

Exemplo de saída:

```
[core]
project = qwiklabs-gcp-44776a13de367a6
```

A documentação completa do **gcloud** está disponível na página [Visão geral do gcloud](#) do Google Cloud.

Baixe o código fonte

Para fazer a configuração, abra uma nova sessão no Cloud Shell e execute o seguinte comando para definir sua zona `us-east1-d`:

```
gcloud config set compute/zone us-east1-d
```

Em seguida, clone o código de amostra do laboratório:

```
gsutil cp gs://splx/gsp051/continuous-deployment-on-kubernetes.zip .
unzip continuous-deployment-on-kubernetes.zip
```

Agora, mude para o diretório correto:

```
cd continuous-deployment-on-kubernetes
```

Como provisionar o Jenkins

Como criar um cluster do Kubernetes

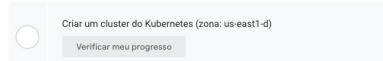
Agora, execute o seguinte comando para provisionar um cluster do Kubernetes:

```
gcloud container clusters create jenkins-cd \
--num-nodes 2 \
--machine-type n1-standard-2 \
--scopes "https://www.googleapis.com/auth/source.read_write,cloud-platform"
```

Essa etapa pode demorar vários minutos para ser concluída. Os escopos extras permitem que o Jenkins accesse o Cloud Source Repositories e o Google Container Registry.

Teste a tarefa concluída

Clique em **Verificar meu progresso** para analisar a tarefa realizada. Se o cluster do Kubernetes tiver sido criado corretamente, você verá uma pontuação de avaliação.



Antes de continuar, execute o seguinte comando para confirmar se o cluster está ativo:

```
gcloud container clusters list
```

Consulte as credenciais do cluster:

```
gcloud container clusters get-credentials jenkins-cd
```

O Kubernetes Engine usa essas credenciais para acessar seu cluster recém-provisionado. Confirme se você conseguiu se conectar a ele executando o seguinte comando:

```
kubectl cluster-info
```

Configurar Helm

Neste laboratório, você usará o Helm para instalar o Jenkins do repositório Charts. O Helm é um gerenciador de pacotes que facilita a configuração e a implantação de aplicativos do Kubernetes. Depois de instalar o Jenkins, você poderá configurar seu canal de CI/CD.

1. Adicione o repositório de gráfico estável do Helm:

```
helm repo add jenkins https://charts.jenkins.io
```

2. Certifique-se de que o repositório esteja atualizado:

```
helm repo update
```

Configure e instale o Jenkins

Ao instalar o Jenkins, um arquivo de `valores` pode ser usado como um modelo para fornecer valores que são necessários para a configuração.

Você usará um arquivo de `valores` personalizados para configurar automaticamente seu Kubernetes Cloud e adicionar os seguintes plug-ins necessários:

- Kubernetes:1.29.4
- Workflow-multibranch:latest
- Git:4.7.1
- Configuration-as-code:1.51
- Google-oauth-plugin:latest
- Google-source-plugin:latest
- Google-storage-plugin:latest

Isso permitirá que o Jenkins se conecte ao cluster e ao projeto do GCP.

1. Baixe o arquivo de `valores` personalizados:

```
gsutil cp gs://splx/gcp38/values.yaml jenkins/values.yaml
```

2. Use o Helm CLI para implantar o gráfico com suas definições de configuração.

```
helm install cd jenkins/jenkins -f jenkins/values.yaml --wait
```

Talvez leve alguns minutos para que o comando seja concluído.

Teste a tarefa concluída

Clique em **Verificar meu progresso** para analisar a tarefa realizada. Se o gráfico do Jenkins tiver sido configurado corretamente, você verá uma pontuação de avaliação.



3. Após a execução do comando, verifique se o estado do pod do Jenkins está definido como "Running" e o do container como "READY":

```
kubectl get pods
```

Exemplo de resposta:

NAME	READY	STATUS	RESTARTS	AGE
cd-jenkins-7c786475dd-vbhg4	1/1	Running	0	1m

4. Configure a conta de serviço do Jenkins para poder fazer a implantação no cluster.

```
kubectl create clusterrolebinding jenkins-deploy --clusterrole=cluster-admin --serviceaccount=default:cd-jenkins
```

Você verá a seguinte resposta:

```
clusterrolebinding.rbac.authorization.k8s.io/jenkins-deploy created
```

5. Execute o seguinte comando para configurar o encaminhamento de portas da IU do Jenkins no Cloud Shell:

```
export POD_NAME=$kubectl get pods --namespace default -l app.kubernetes.io/component=jenkins-master -l app.kubernetes.io/instance=cd -o jsonpath='{.items[0].metadata.name}'; kubectl port-forward $POD_NAME 8080:8080 > /dev/null &
```

6. Agora, verifique se o serviço do Jenkins foi criado corretamente:

```
kubectl get svc
```

Exemplo de resposta:

NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
cd-jenkins	10.35.249.67	<none>	8080/TCP	3h
cd-jenkins-agent	10.35.248.1	<none>	50800/TCP	3h
kubernetes	10.35.248.1	<none>	443/TCP	9h

Você está usando o [Plug-in do Kubernetes](#) para que os nós do builder sejam iniciados automaticamente, conforme necessário, quando o mestre do Jenkins os solicitar. Após a conclusão do processo, eles serão automaticamente desativados, e os recursos relacionados serão adicionados outra vez ao pool de recursos de clusters.

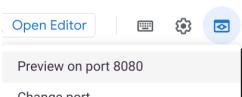
Esse serviço expõe as portas 8080 e 50800 dos pods que correspondem ao `seletor`. Isso vai expor as portas da IU da Web do Jenkins e as portas de registro do agente/construtor no cluster do Kubernetes. Além disso, os serviços `jenkins-ui` são expostos usando um ClusterIP para que não possam ser acessados de fora do cluster.

Conecte-se ao Jenkins

1. O gráfico do Jenkins criará automaticamente uma senha de administrador para você. Para vê-la, execute:

```
printf $(kubectl get secret cd-jenkins -o jsonpath='{.data.jenkins-admin-password}' | base64 --decode);echo
```

2. Para acessar a interface do usuário do Jenkins, clique no botão **Visualização na Web** no Cloud Shell e em **Visualizar na porta 8080**:



Orange port
About web preview

3. Agora você poderá fazer login com o nome de usuário do `admin` e sua senha gerada automaticamente.

A configuração do Jenkins no seu cluster do Kubernetes está pronta. O Jenkins gerenciará seus canais automatizados de CI/CD nas próximas seções.

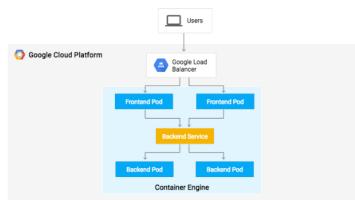
Entenda o aplicativo

Você implantará o aplicativo amostra `grome` no canal de implantação contínua. O aplicativo, escrito na linguagem Go, está no diretório `sample-app` do repositório. Quando você executa o binário `grome` em uma instância do Compute Engine, o app exibe os metadados da instância em um card de informações.

Backend that serviced this request	
Name	<code>gke-junkyard-default-pool-76087c0a-txu0</code>
ID	<code>9185001295255472551</code>
Hostname	<code>gke-junkyard-default-pool-76087c0a-txu0.c</code>
Zone	<code>us-west1-a</code>
Project	
Internal IP	<code>10.240.0.15</code>
External IP	<code>104.198.102.151</code>

O aplicativo simula um microserviço aceitando dois modos de operação.

- No modo de back-end: o `grome` detecta a atividade da porta 8080 e retorna metadados da instância do Compute Engine no formato JSON.
- No modo de front-end: o `grome` consulta o serviço de back-end e renderiza o JSON resultante na interface do usuário.



Como implantar o aplicativo

Você implantará o aplicativo em dois ambientes diferentes:

- **Produção:** é o site ativo que os usuários acessam.
- **Canário:** é um site de capacidade menor que só recebe uma porcentagem do tráfego do usuário. Use esse ambiente para validar seu software com o tráfego ativo antes de lançá-lo para todos os usuários.

No Google Cloud Shell, navegue para o diretório de aplicativos de amostra:

```
cd sample-app
```

Crie o namespace do Kubernetes para isolar logicamente a implantação:

```
kubectl create ns production
```

Use os comandos `kubectl apply` para criar as implantações das versões de produção e canário, além dos serviços:

```
kubectl apply -f k8s/production -n production
```

```
kubectl apply -f k8s/canary -n production
```

```
kubectl apply -f k8s/services -n production
```

Teste a tarefa concluída

Clique em **Verificar meu progresso** para analisar a tarefa realizada. Se as implantações tiverem sido criadas corretamente, você verá uma pontuação de avaliação.

Crie as implantações de produção e canário

Verificar meu progresso

Por padrão, somente uma réplica do front-end é implantada. Use o comando `kubectl scale` para garantir o mínimo de quatro réplicas em execução todo o tempo.

Escalone os front-ends do ambiente de produção executando o seguinte comando:

```
kubectl scale deployment grome-frontend-production -n production --replicas 4
```

Agora, confirme que você tem cinco pods em execução para o front-end, quatro para o tráfego de produção e um para versões canário. As alterações na versão canário afetarão somente um a cada cinco usuários (20%).

```
kubectl get pods -n production -l app=grome -l role=frontend
```

Além disso, confirme que você tem dois pods para o back-end, um para a produção e um

para a versão canário:

```
kubectl get pods -n production -l app=gceme -l role=backend
```

Consulte o IP externo dos serviços de produção:

```
kubectl get service gceme-frontend -n production
```

Observação: o endereço IP externo do平衡ador de carga talvez leve alguns minutos para aparecer.

Exemplo de resposta:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gceme-frontend	LoadBalancer	10.79.241.131	104.196.110.46	80/TCP	5h

Cole o **IP externo** no navegador para ver o card de informações exibido. Você deverá ver uma página semelhante a esta:

Backend that serviced this request	
Name	gke-jenkins-cd-default-pool-ea8dd5ef-pgt0
Version	1.0.0
ID	359491858067761855
Hostname	gke-jenkins-cd-default-pool-ea8dd5ef-pgt0.c.quickstart-gcp-847acf74b55ab2.internal
Zone	us-central1-f
Project	quickstart-gcp-847acf74b55ab2
Internal IP	10.128.0.5
External IP	104.197.237.15

Agora, armazene o IP do平衡ador de carga do *serviço de front-end* em uma variável de ambiente para usar mais tarde:

```
export FRONTEND_SERVICE_IP=$(kubectl get -o jsonpath='{.status.loadBalancer.ingress[0].ip}' --namespace=production services gceme-frontend)
```

Para confirmar que os dois serviços estão funcionando, abra o endereço IP externo do front-end no navegador. Verifique a resposta da versão do serviço (que deve ser 1.0.0) executando o seguinte comando:

```
curl http://$FRONTEND_SERVICE_IP/version
```

Pronto. Você implantou o aplicativo de amostra. Agora, configure um canal para implantar as alterações de maneira contínua e confiável.

Como criar um canal do Jenkins

Como criar um repositório para hospedar o código-fonte do app de amostra

Crie uma cópia do app de amostra `gceme` e envie por push ao [Cloud Source Repository](#):

```
gcloud source repos create default
```

Ignore o aviso. Esse repositório não será cobrado.

Teste a tarefa concluída

Clique em **Verificar meu progresso** para analisar a tarefa realizada. Se o repositório de origem tiver sido criado corretamente, você verá uma pontuação de avaliação.



```
git init
```

Inicialize o diretório `sample-app` como o próprio repositório Git:

```
git config credential.helper gcloud.sh
```

Execute este comando:

```
git remote add origin https://source.developers.google.com/p/$DEVSHELL_PROJECT_ID/r/default
```

Defina o nome de usuário e o endereço de e-mail para as confirmações do Git. Substitua `[EMAIL_ADDRESS]` pelo endereço de e-mail do Git e `[USERNAME]` pelo nome de usuário do Git:

```
git config --global user.email "[EMAIL_ADDRESS]"
```

```
git config --global user.name "[USERNAME]"
```

Adicione, confirme e envie os arquivos por push:

```
git add .
```

```
git commit -m "Initial commit"
```

```
git push origin master
```

Como adicionar as credenciais da conta de serviço

Configure as credenciais para permitir que o Jenkins accesse o repositório do código. O Jenkins usará as credenciais da conta de serviço do seu cluster para fazer o download do código do Cloud Source Repositories.

Nota 1: Na interface de usuário do Jenkins, clique em **Manane Jenkins** na navegação à

esquerda e clique em **Manage Credentials**.

Etapa 2: clique em Jenkins

The screenshot shows the Jenkins 'Credentials' management interface. It lists several entries under 'Stores scoped to Jenkins'. One entry, 'Global credentials (unrestricted)', is highlighted with a red box. This entry has a sub-section titled 'Global credentials (unrestricted)' which contains a table with columns 'Name', 'Kind', and 'Description'. One row in this table is also highlighted with a red box.

Etapa 3: clique em **Global credentials (unrestricted)**.

Etapa 4: clique em **Add Credentials** no painel de navegação à esquerda.

Etapa 5: selecione **Google Service Account from metadata** na lista suspensa **Kind** e clique em **OK**.

As credenciais globais foram adicionadas. O nome da credencial é ID do projeto, encontrado na seção DETALHES DA CONEXÃO do laboratório.

Global credentials (unrestricted)

This screenshot shows the 'Global credentials (unrestricted)' table. It has one row, which is highlighted with a red box. The row contains information about a 'Google Service Account from metadata' kind of credential.

Como criar o job do Jenkins

Navegue até a interface do usuário do Jenkins e siga estas etapas para configurar um job de canal.

Etapa 1: clique em **New Item** no painel de navegação à esquerda:

The screenshot shows the Jenkins dashboard. The 'New Item' button, located in the main navigation bar, is highlighted with a red box.

Etapa 2: nomeie o projeto como **sample-app**, escolha a opção **Multibranch Pipeline** e clique em **OK**.

Etapa 3: na próxima página, na seção **Branch Sources**, clique em **Add Source** e selecione **git**.

Etapa 4: no campo **Project Repository**, cole o URL de clone **HTTPS** do repositório sample-app no Cloud Source Repositories. Substitua **[PROJECT_ID]** pelo **ID do projeto**:

`https://source.developers.google.com/p/[PROJECT_ID]/r/default`

Etapa 5: na lista suspensa **Credentials**, selecione o nome das credenciais que você criou ao adicionar sua conta de serviço nas etapas anteriores.

Etapa 6: na seção **Scan Multibranch Pipeline Triggers**, marque a caixa **Periodically if not otherwise run** e defina o valor de **Interval** como 1 minuto.

Etapa 7: a configuração do seu job ficará assim:

This screenshot shows the 'Branch Sources' configuration for a Multibranch Pipeline job. It includes sections for 'Git', 'Project Repository', 'Credentials', 'Behaviors', and 'Property strategy'. The 'Project Repository' field contains the URL 'https://source.developers.google.com/p/[PROJECT_ID]/r/default'. The 'Credentials' dropdown is set to 'gke-source-gcp-03d72ab4-acab7541/default'. The 'Behaviors' section has 'Discover branches' selected. The 'Property strategy' is set to 'All branches get the same properties'.

Etapa 8: clique em **Save** e mantenha as definições padrão das outras opções.

Depois que você concluir essas etapas, um job denominado **Branch indexing** será executado. Esse "meta-job" identifica os branches do repositório e garante que não ocorreram alterações neles. Se você clicar no **sample-app** no canto superior esquerdo, o job mestre será exibido.

Observação: a primeira execução do job mestre poderá falhar enquanto você não fizer algumas alterações no código na próxima etapa.

O canal do Jenkins está pronto. Em seguida, você criará o ambiente de desenvolvimento para integração contínua.

Como criar o ambiente de desenvolvimento

Um branch de desenvolvimento é um conjunto de ambientes que os desenvolvedores usam para testar as alterações no código antes de enviá-las para integração ao site ativo. Esses ambientes são versões reduzidas do aplicativo, mas precisam ser implantados com os mesmos mecanismos do ambiente ativo.

Crie um branch de desenvolvimento

Para criar um ambiente de desenvolvimento em um branch de recursos, você pode enviá-lo por push ao servidor Git e deixar o Jenkins implantar o ambiente.

Crie um branch de desenvolvimento e envie-o por push ao servidor Git:

```
git checkout -b new-feature
```

Como modificar a definição do pipeline

O `Jenkinsfile` que define o pipeline é escrito com a [sintaxe do Groovy para pipelines do Jenkins](#) (página em inglês). Com um `Jenkinsfile`, é possível expressar todo um pipeline de versão em um único arquivo que coexiste com o código-fonte. Os canais são compatíveis com recursos eficientes, como o carregamento em paralelo, e exigem a aprovação manual do usuário.

Para que o canal funcione como esperado, é preciso modificar o `Jenkinsfile` para definir o código do seu projeto.

Abra o `Jenkinsfile` no seu editor de terminal, por exemplo, `vi`:

```
vi Jenkinsfile
```

Inicie o editor:

```
i
```

Adicione o `PROJECT_ID` ao valor `REPLACE_WITH_YOUR_PROJECT_ID`. (O `PROJECT_ID` é o ID do projeto encontrado na seção DETALHES DA CONEXÃO do laboratório. Você também pode executar o comando `gcloud config get-value project` para encontrá-lo.)

```
PROJECT = "REPLACE_WITH_YOUR_PROJECT_ID"
APP_NAME = "$PROJECT"
FE_SVC_NAME = "$APP_NAME"-frontend
CLUSTER = "jenkins-cd"
CLUSTER_ZONE = "us-east1-d"
IMAGE_TAG =
"grc.io/$PROJECT/$APP_NAME:$env.BRANCH_NAME.$env.BUILD_NUMBER"
JENKINS_CRED = "$PROJECT"
```

Salve o arquivo `Jenkinsfile` pressionando a tecla `Esc`, depois (para usuários do `vi`):

```
:wq
```

Modifique o site

Para demonstrar como alterar o aplicativo, mude a cor dos cards do gcmee de **azul** para **laranja**.

Abra o `html.go`:

```
vi html.go
```

Inicie o editor:

```
i
```

Altere as duas instâncias de `<div class="card blue">` com o seguinte código:

```
<div class="card orange">
```

Salve o arquivo `html.go` pressionando a tecla `Esc`, depois:

```
:wq
```

Abra o `main.go`:

```
vi main.go
```

Inicie o editor:

```
i
```

A versão é definida nesta linha:

```
const version string = "1.0.0"
```

Atualize-a para:

```
const version string = "2.0.0"
```

Salve o arquivo `main.go` novamente pressionando a tecla `Esc`, depois:

```
:wq
```

Inicie a implantação

Confirme e envie suas alterações por push:

```
git add Jenkinsfile html.go main.go
```

```
git commit -m "Version 2.0.0"
```

```
git push origin new-feature
```

Isto iniciará a criação do seu ambiente de desenvolvimento.

Depois de enviar a alteração por push ao repositório Git, navegue até a interface do usuário do Jenkins. Você verá que a criação do branch `new-feature` foi iniciada. Pode levar até um minuto para que as alterações sejam detectadas.

Depois que a versão estiver em execução, clique na seta para baixo ao lado do `build` na navegação à esquerda e selecione `Console Output`:

```
Build Executor Status
```

Observação: em um cenário de desenvolvimento, você não usaria um平衡ador de carga voltado para o público. Para proteger o aplicativo, você pode usar o [proxy kubectl](#). O proxy faz a própria autenticação com a API Kubernetes e retransmite as solicitações da máquina local para o serviço no cluster sem expor seu serviço à Internet.

Se nada for exibido em Build Executor, não se preocupe. Acesse a página inicial do Jenkins → app de amostra. Verifique se o pipeline new-feature foi criado.

Assim que tudo estiver pronto, inicie o proxy em segundo plano:

```
kubectl proxy &
```

Se o processo parar, pressione as teclas **Ctrl + C** para sair. Verifique se o aplicativo pode ser acessado enviando uma solicitação para `localhost` e permitindo que o proxy `kubectl` o encaminhe para seu serviço:

```
curl \n http://localhost:8081/api/v1/namespaces/new-feature/services/gceme-\n frontend:80/proxy/version
```

Você verá a resposta "2.0.0", que é a versão em uso no momento.

Se você receber um erro semelhante a este:

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {
    ...
  },
  "status": "Failure",
  "message": "no endpoints available for service \"gceme-frontend:80\"",
  "reason": "ServiceUnavailable",
  "code": 503
}
```

Isso significa que o endpoint do front-end ainda não foi propagado. Aguarde um pouco e tente o comando `curl` novamente. Prossiga quando receber a seguinte resposta:

```
2.0.0
```

Pronto. Você configurou o ambiente de desenvolvimento. Agora, você usará o que aprendeu no módulo anterior para implantar uma versão canária e testar um novo recurso.

Como implantar uma versão canária

Você confirmou que seu app está executando o código mais recente no ambiente para desenvolvedores. Agora, implante esse código no ambiente canário.

Crie um branch canário e envie-o por push para o servidor Git:

```
git checkout -b canary
```

```
git push origin canary
```

No Jenkins, você verá que o pipeline `canário` foi iniciado. Depois de concluído, verifique o URL de serviço para garantir que parte do tráfego seja atendido pela nova versão. O esperado é que cerca de 1 em cada 5 solicitações (sem ordem específica) retornem a versão 2.0.0.

```
export FRONTEND_SERVICE_IP=$(kubectl get -o \n jsonpath='{.status.loadBalancer.ingress[0].ip}' --namespace=production\n services gceme-frontend)
```

```
while true; do curl http://$FRONTEND_SERVICE_IP/version; sleep 1; done
```

Se você continuar vendo a versão 1.0.0, tente executar os comandos acima novamente. Depois de garantir que está tudo certo, encerre o comando com **Ctrl + C**.

Pronto. Você implantou uma versão canária. Agora, implante a nova versão na produção.

Como implantar a versão na produção

Agora que a versão canária está pronta e não houve reclamações de clientes, implante-a no restante da sua frota de produção.

Crie um branch canário e envie-o por push para o servidor Git:

```
git checkout master
```

```
git merge canary
```

```
git push origin master
```

No Jenkins, você verá que o pipeline mestre foi iniciado. *Depois de concluído* (o que pode levar alguns minutos), verifique o URL de serviço para garantir que parte do tráfego está sendo atendida pela nova versão, 2.0.0.

```
export FRONTEND_SERVICE_IP=$(kubectl get -o \n jsonpath='{.status.loadBalancer.ingress[0].ip}' --namespace=production\n services gceme-frontend)
```

```
while true; do curl http://$FRONTEND_SERVICE_IP/version; sleep 1; done
```

Mais uma vez, se você vir instâncias de 1.0.0, tente executar os comandos acima novamente. Se quiser parar esse comando, pressione **Ctrl + C**.

Exemplo de resposta:

```
gcp-tutorial@854-student-qwiklabs-gcp-df99aba9e6ca114a:~/continuous-deployment-on-kubernetes$ sample-app$ while true; do curl http://$FRONTEND_SERVICE_IP/version; sleep 1; done
2.0.0
2.0.0
2.0.0
2.0.0
2.0.0
^C
```

Você também pode navegar para o site em que o aplicativo gomei exibe os cards de informações. A cor do card mudou de azul para laranja. Veja abaixo o comando que foi usado para consultar o endereço IP externo e confira a mudança:

```
kubectl get service gomei-frontend -n production
```

Exemplo de resposta:

Backend that serviced this request	
Name	gke-jenkins-cd-default-pool-c7ted012-3qb7
Version	2.0.0
ID	396367411415644538
Hostname	gke-jenkins-cd-default-pool-c7ted012-3qb7.c.qwiklabs-gcp-3ac85c6d0ecc505.internal
Zone	us-central1-f
Project	qwiklabs-gcp-3ac85c6d0ecc505
Internal IP	10.128.0.2
External IP	35.224.235.170

Teste seu conhecimento

Responda às perguntas de múltipla escolha a seguir para reforçar sua compreensão dos conceitos abordados neste laboratório. Use todo o conhecimento adquirido até aqui.

Quais são os seguintes namespaces do Kubernetes usados no laboratório?

kube-system
 helm
 default
 jenkins
 production

Enviar

O gráfico Helm é uma coleção de arquivos que descreve um conjunto relacionado de recursos do Kubernetes.

Verdadeiro
 Falso

Pronto.

Muito bem! Você implantou seu aplicativo na produção.

Parabéns!

Isso conclui este laboratório prático sobre implantação e uso do Jenkins no Kubernetes Engine para ativar o canal de entrega/implantação contínua. Você teve a oportunidade de implantar uma ferramenta DevOps importante no Kubernetes Engine e configurá-la para uso na produção. Você trabalhou com a ferramenta de linha de comando kubectl e as configurações de implantação em arquivos YAML, além de aprender um pouco sobre a configuração de canais do Jenkins para um processo de desenvolvimento/implantação. Com essa experiência prática, você deverá se sentir mais à vontade para usar essas ferramentas no seu próprio trabalho de DevOps.

Termine a Quest



Este laboratório autoguiado faz parte das Quests do Qwiklabs [Kubernetes no Google Cloud](#), [Cloud Architecture](#) e [DevOps Essentials](#). Uma Quest é uma série de laboratórios relacionados que formam o programa de aprendizado. Concluir esta Quest dá a você o selo acima como reconhecimento pela sua conquista. Você pode publicar os selos e incluir um link para eles no seu currículo on-line ou nas redes sociais. Caso você já tenha feito este laboratório, inscreva-se em uma Quest para ganhar os créditos de conclusão imediatamente. [Veja outras Quests do Qwiklabs disponíveis](#).

Comece o próximo laboratório

Faça o laboratório [Hello Node Kubernetes](#) para continuar a Quest ou confira estas sugestões:

- [Orquestração na nuvem com o Kubernetes](#)
- [Como gerenciar implantações com o Kubernetes Engine](#)

Próximas etapas / Saiba mais

- Leia mais sobre a solução [Jenkins no Kubernetes Engine](#).
- Saiba como usar o [Jenkins](#) para ativar a entrega contínua no [Kubernetes Engine](#).
- Leia mais em [Guias e soluções de DevOps](#) na documentação do Google Cloud.
- Conheça a [comunidade do Jenkins](#) (página em inglês).

Treinamento e certificação do Google Cloud

...ajuda você a aproveitar as tecnologias do Google Cloud ao máximo. [Nossas aulas](#) incluem habilidades técnicas e práticas recomendadas para ajudar você a alcançar rapidamente o nível esperado e continuar sua jornada de aprendizado. Oferecemos treinamentos que vão do nível básico ao avançado, com opções de aulas virtuais, sob demanda e por meio de transmissões ao vivo para que você possa encáixá-las na correria do seu dia a dia. As [certificações](#) ajudam você a validar e comprovar suas habilidades e conhecimentos das tecnologias do Google Cloud.

Manual atualizado em 27 de janeiro de 2022
Laboratório testado em 27 de janeiro de 2022
Copyright 2020 Google LLC. Todos os direitos reservados. Google e o logotipo do Google são marcas registradas da Google LLC. Todos os outros nomes de produtos e empresas podem ser marcas registradas das respectivas empresas a que estão associados.

Continuar a Quest

