HEROKU

LOGIN: [robson.luizsilva2019@gmail.com](mailto:robson.luizsilva2019@gmail.com)

SENHA: r10201431N@

Site : <https://robsonl-dio-bankline-api.herokuapp.com/swagger-ui/index.html>

## [Pré-requisitos: Instale o Git e a CLI do Heroku](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#prerequisites-install-git-and-the-heroku-cli)

Você deve ter o Git e a CLI do Heroku instalados para implantar com o Git.

* [Instruções de instalação do Git](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)
* [Instruções de instalação do Heroku CLI](https://devcenter-heroku-com.translate.goog/articles/heroku-cli?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#install-the-heroku-cli)

Antes de implantar seu aplicativo no Heroku, inicialize um repositório Git local e confirme o código do aplicativo nele.

O exemplo a seguir demonstra a inicialização de um repositório Git para um aplicativo que reside no example-appdiretório:

$ cd example-app

$ git init

Initialized empty Git repository in .git/

$ git add .

$ git commit -m "My first commit"

Created initial commit 5df2d09: My first commit

44 files changed, 8393 insertions(+), 0 deletions(-)

create mode 100644 README

create mode 100644 Procfile

create mode 100644 app/controllers/source\_file

...

Inicialize o repositório Git no diretório raiz do seu aplicativo. Se seu aplicativo estiver em um subdiretório do seu repositório, ele não será executado quando enviado para o Heroku.

Agora você está rastreando o código do seu aplicativo em um repositório Git local. Ele ainda não existe em nenhum servidor remoto.

## [Crie um controle remoto Heroku](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#create-a-heroku-remote)

[Os controles remotos](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=http://git-scm.com/book/en/Git-Basics-Working-with-Remotes) do Git são versões do seu repositório que residem em outros servidores. Você implanta seu aplicativo enviando seu código para um controle remoto especial hospedado no Heroku que está associado ao seu aplicativo.

Heroku Git é uma conveniência para implantação e não pretende ser um repositório git estável. Use o [**GitHub**](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=http://github.com/) (recomendado), [**GitLab**](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://about.gitlab.com/) , [**BitBucket**](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://bitbucket.org/) ou outro sistema de controle de versão para rastrear sua base de código.

### [Para um novo aplicativo](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#for-a-new-app)

O heroku createcomando CLI cria um novo aplicativo vazio no Heroku, junto com um repositório Git vazio associado. Se você executar este comando no diretório raiz do seu aplicativo, o repositório Heroku Git vazio será definido automaticamente como remoto para seu repositório local.

$ heroku create -a example-app

Creating app... done, ⬢ example-app

https://thawing-inlet-61413.herokuapp.com/ | https://git.heroku.com/example-app.git

Você pode usar o git remotecomando para confirmar que um controle remoto herokufoi definido para seu aplicativo:

$ git remote -v

heroku https://git.heroku.com/example-app.git (fetch)

heroku https://git.heroku.com/example-app.git (push)

### [Para um aplicativo existente](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#for-an-existing-app)

Adicione um controle remoto ao seu repositório local com o heroku git:remotecomando. Tudo o que você precisa é o nome do seu aplicativo Heroku:

$ heroku git:remote -a example-app

set git remote heroku to https://git.heroku.com/example-app.git

### [Renomear um controle remoto](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#rename-a-remote)

Por padrão, a CLI do Heroku nomeia todos os controles remotos do Heroku que cria para seu aplicativo heroku. Você pode renomear seus controles remotos com o git remote renamecomando. Por exemplo, renomeie herokupara heroku-staging:

$ git remote rename heroku heroku-staging

Renomear seu controle remoto Heroku pode ser útil se você tiver vários aplicativos Heroku que usam a mesma base de código. Nesse caso, cada aplicativo Heroku tem seu próprio controle remoto em seu repositório local.

A documentação do Dev Center pressupõe que seu aplicativo tenha um único controle remoto Heroku chamado heroku.

## [Implante seu código](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#deploy-your-code)

Para implantar seu aplicativo no Heroku, use o git pushcomando para enviar o código da [ramificação principal](https://devcenter-heroku-com.translate.goog/articles/git-branches?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) do repositório local para o herokucontrole remoto. Por exemplo:

$ git push heroku main

Initializing repository, done.

updating 'refs/heads/main'

...

Use este mesmo comando sempre que quiser implantar a versão mais recente confirmada do seu código no Heroku.

O Heroku implanta apenas o código que você envia para o [mestre ou](https://devcenter-heroku-com.translate.goog/articles/git-branches?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) ramificações principais do controle remoto. Enviar código para outra ramificação do herokucontrole remoto não tem efeito.

### [Implantar de uma filial além de main](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#deploy-from-a-branch-besides-main)

Para implantar o código no Heroku de um não mainbranch do seu repositório local (por exemplo, testbranch), use a seguinte sintaxe e envie-o para o mainbranch remoto:

$ git push heroku testbranch:main

Esse método oferece suporte a aplicativos que dependem de submódulos Git, além de muitas outras [**estratégias de resolução de dependência**](https://devcenter-heroku-com.translate.goog/articles/git-submodules?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) .

Heroku não suporta [**git lfs**](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://git-lfs.github.com/) . O uso desse método pode fazer com que os pushes falhem.

## [Vários controles remotos e ambientes](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#multiple-remotes-and-environments)

Você pode usar as mesmas técnicas usadas para implantar na produção que podem ser usadas para implantar uma ramificação de desenvolvimento do seu aplicativo em um aplicativo de teste no Heroku. Consulte [Gerenciando vários ambientes para um aplicativo](https://devcenter-heroku-com.translate.goog/articles/multiple-environments?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) para obter mais informações.

## [Desvincular-se do processo de construção](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#detach-from-the-build-process)

Depois de iniciar uma implantação do Heroku com git push, você pode desanexar do processo de compilação resultante pressionando Ctrl + C. Desanexar não cancela a compilação ou a implantação. A compilação continua em segundo plano e cria uma nova [versão](https://devcenter-heroku-com.translate.goog/articles/releases?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) assim que é concluída.

## [Comportamento de implantações simultâneas](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#behavior-of-simultaneous-deploys)

É possível iniciar uma implantação antes da conclusão de uma implantação anterior do mesmo aplicativo. Por exemplo, dois colaboradores em um aplicativo enviam confirmações diferentes para o herokucontrole remoto aproximadamente ao mesmo tempo.

Se essa situação ocorrer, as diferentes versões do seu aplicativo serão implantadas no Heroku na ordem em que suas respectivas compilações forem concluídas. Essa ordem pode diferir da ordem em que os pushes ocorreram.

Por exemplo, considere duas compilações, A e B. Se a compilação B começar após a compilação A, mas terminar antes dela, o Heroku implantará a compilação B primeiro . Então, quando o Build A finalmente é concluído, o Heroku o implanta, substituindo o Build B.

## [Autenticação HTTP Git](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#http-git-authentication)

Por padrão, o Heroku usa HTTP como seu transporte Git. A CLI do Heroku coloca automaticamente as credenciais no .netrcarquivo em heroku login. O cliente Git usa cURL ao interagir com remotos HTTP e cURL usa as credenciais do .netrcarquivo. Consulte o [**artigo sobre autenticação da CLI**](https://devcenter-heroku-com.translate.goog/articles/authentication?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) para obter detalhes.

Você não pode autenticar com o endpoint Heroku HTTP Git usando seu nome de usuário Heroku (e-mail) e senha. O endpoint Heroku HTTP Git só aceita autenticação [**HTTP básica**](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=http://en.wikipedia.org/wiki/Basic_access_authentication) baseada em chave de API . Use uma chave de API conforme descrito nesta seção.

Se você se autenticar no serviço Git com credenciais incorretas, receberá este erro:

remote: ! WARNING:

remote: ! Do not authenticate with username and password using git.

remote: ! Run `heroku login` to update your credentials, then retry the git command.

remote: ! See documentation for details: https://devcenter.heroku.com/articles/git#http-git-authentication

Se você estiver usando outros clientes Git, como EGit ou Tower, configure-os para usar uma string vazia para nome de usuário e a chave de API da sua conta para senha. A chave de API está [disponível na CLI](https://devcenter-heroku-com.translate.goog/articles/authentication?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#retrieving-the-api-token) e no [painel](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://dashboard.heroku.com/account) .

## [Redefinir um repositório Git](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#reset-a-git-repository)

Para redefinir ou limpar o repositório Heroku Git de um aplicativo, use o [plugin heroku-repo](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=https://github.com/heroku/heroku-repo) CLI:

$ heroku plugins:install heroku-repo

$ heroku repo:reset --app appname

A redefinição do repositório Git exclui todo o código-fonte e o histórico do Git, portanto, certifique-se de ter outra cópia do repositório primeiro.

## [Mantenha o tamanho do seu repositório pequeno](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#keep-your-repository-size-small)

O tamanho descompactado de um check-out do HEADrepositório, combinado com o tamanho dos submódulos restaurados, não pode exceder 1 GB.

Não é recomendado implantar grandes repositórios com mais de 600 MB. Eles podem causar tempos limite e pushes lentos em geral. A execução heroku apps:infomostra o tamanho do seu repositório.

Causas comuns de grandes repositórios são arquivos binários verificados no repositório ou logs de desenvolvimento em constante mudança. Você pode remover arquivos enviados por acidente com [git filter-branch](https://translate.google.com/website?sl=en&tl=pt&hl=pt-BR&prev=search&u=http://git-scm.com/book/en/Git-Internals-Maintenance-and-Data-Recovery%23_removing_objects) . Depois de executá-lo, você deve enviar suas alterações com a --forceopção, exigindo coordenação entre sua equipe.

Depois de reduzir o tamanho do seu repositório localmente, você deve [redefinir o repositório Git do aplicativo](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#reset-a-git-repository) antes de enviá-lo para o Heroku novamente.

## [Limites](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#limits)

Para proteger o serviço Git, o Heroku impõe certos limites ao uso do repositório Git e ao tamanho do conteúdo.

O Heroku limita os usuários a uma janela contínua de 75 solicitações Git por hora, por usuário, por aplicativo. Depois de atingir esse limite, o Heroku nega solicitações do Git até que os níveis de solicitação fiquem abaixo do limite por alguns minutos. Você vê uma mensagem de erro como:

! Too many requests for this Git repo. Please try again later.

Se você atingir esse limite, certifique-se de que não haja processos ou scripts automatizados pesquisando o repositório Git.

Além disso, o tamanho descompactado de um check-out do HEADrepositório, combinado com o tamanho dos submódulos restaurados, não pode exceder 1 GB.

## [Implantar código rastreado no Subversion ou em outros sistemas de controle de revisão](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#deploy-code-tracked-in-subversion-or-other-revision-control-systems)

Embora o Git seja uma das melhores opções disponíveis para controle de revisão, você não precisa parar de usar seu sistema de controle de revisão atual. Você pode usar o Git puramente como um mecanismo de implantação, existindo lado a lado com sua outra ferramenta.

Você pode aprender muito mais sobre .gitignoreem [**nosso artigo sobre o tema**](https://devcenter-heroku-com.translate.goog/articles/gitignore?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc) .

Por exemplo, se você estiver usando o Subversion, inicialize seu repositório Git conforme descrito na seção [Pré-requisitos](https://devcenter-heroku-com.translate.goog/articles/git?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc#prerequisites-install-git-and-the-heroku-cli) . Em seguida, adicione um .gitignorearquivo para dizer ao Git para ignorar seus diretórios do Subversion.

$ git init

$ echo .svn > .gitignore

$ git add .

$ git commit -m "using git for heroku deployment"

Agora diga ao Subversion para ignorar o Git:

$ svn propset svn:ignore .git .

property 'svn:ignore' set on '.'

$ svn commit -m "ignoring git folder (git is used for heroku deployment)"

Recomenda-se usar o -f(force flag) para evitar conflitos com pushs de outros desenvolvedores. Como você está usando o Git para seu controle de revisão, mas apenas como transporte, usar o sinalizador force é uma prática razoável.

Cada vez que você deseja implantar no Heroku:

$ git add -A

$ git commit -m "commit for deploy to heroku"

...

$ git push -f heroku