Sobre o Git

Aprenda sobre o sistema de controle de versões, Git e como ele funciona com GitHub.

Neste artigo

Sobre controle de versão e o Git

Um sistema de controle de versão, ou VCS, monitora o histórico de alteraçoes à medida que as pessoas e equipes colaboram em projetos em conjunto. Como os desenvolvedores fazem alterações no projeto, qualquer versão anterior do projeto pode ser recuperada a qualquer momento.

Os desenvolvedores podem revisar o histórico do projeto para descobrir:

- Quais alterações foram feitas?
- Quem fez as alterações?
- Quando as alterações foram feitas?
- Por que as alterações foram necessárias?

Os VCSs fornecem a cada colaborador uma visão unificada e consistente de um projeto, evidenciando o trabalho que já está em andamento. Ver um histórico transparente das alterações, quem as fez, e como eles contribuem para o desenvolvimento de um projeto ajuda os integrantes da equipe a manter-se alinhados enquanto trabalham de forma indepenc

Em um sistema de controle de versão distribuído, cada desenvolvedor tem uma cópia completa do projeto e do histórico do projeto. Ao contrário dos sistemas de controle de versão centralizados conhecidos, os DVCSs não precisam de uma conexão constante com um repositório central. Git é o sistema de controle de versão distribuída mais popular. O Git é comumente usado para o desenvolvimento de software de código aberto e comercial, com benefícios significativos para indivíduos, equipes e negócios.

- O Git permite que os desenvolvedores vejam toda a linha do tempo das suas alterações, decisões e progressão de qualquer projeto em um só lugar. Desde o momento em que acessam a história de um projeto, o desenvolvedor tem todo o contexto de que precisa para entender e começar a contribuir.
- Os desenvolvedores trabalham em todos os fusos horários. Com um DVCS como o Git, a colaboração pode acontecer a qualquer momento enquanto mantém a integridade do código-fonte. Ao usar branches, os desenvolvedores podem propor alterações no código de produção.
- As empresas que usam o Git podem derrubar as barreiras de comunicação entre equipes e mantê-las focadas em fazer o melhor trabalho. Além disso, o Git possibilita alinhar especialistas em todos os negócios para colaborar em grandes projetos.

um projeto, junto com o histórico de revisão de cada arquivo. O histórico de arquivos aparece como instantâneos no tempo denominados commits. Os commits podem ser organizados em várias linhas de desenvolvimento denominadas branches. Como o Git é um DVCS, os repositórios são unidades auto-confinadas e qualquer pessoa que tiver uma cópia do repositório pode acessar toda a base de código e seu histórico. Ao usar a linha de comando ou outras interfaces de uso, um repositório Git também permite a interação com o histórico, clonagem do repositório, criação de branches, commiting, merge, comparação de alterações entre versões de código e muito mais.

Por meio de plataformas como GitHub, o Git também oferece mais oportunidades para transparência e colaboração do projeto. Repositórios públicos ajudam as equipes a trabalhar juntas para criar o melhor produto final possível.

Como GitHub funciona

GitHub hospeda repositórios do Git e fornece aos desenvolvedores ferramentas para enviar um código melhor por meio das funcionalidades de linha de comando, problemas(discussões encadeadas), pull requests, revisão de código ou o uso de uma coleção de aplicativos grátis e

para compra em GitHub Marketplace. Com camadas de colaboração como o fluxo de GitHub, uma comunidade de 100 milhões de desenvolvedores, e um ecossistema com centenas de integrações, GitHub muda a forma como o software é construído.

GitHub cria colaboração diretamente no processo de desenvolvimento. O trabalho é organizado em repositórios onde os desenvolvedores podem definir os requisitos ou direção e definir expectativas para os integrantes da equipe. Em seguida, ao usar o fluxo GitHub, os desenvolvedores simplesmente criam um branch para trabalhar nas atualizações, enviar alterações para salvá-las, abrir um pull request para propor e discutir alterações, e fazer merge de pull requests quando todos estiverem na mesma página. Para obter mais informações, confira "Fluxo do GitHub".

Para planos e custos de GitHub, confira GitHub Pricing. Para obter informações sobre como GitHub Enterprise se compara a outras opções, confira Comparando o GitHub com outras soluções de DevOps.

GitHub e a linha de comando

Comandos básicos do Git

Para usar o Git, os desenvolvedores usam comandos específicos para copiar, criar, alterar e combinar código. Esses comandos podem ser executados diretamente na linha de comando ou usando um aplicativo como GitHub Desktop. Aqui estão alguns comandos comuns para usar o Git:

- `git init` inicializa um novo repositório Git e começa a acompanhar um diretório existente. Ele adiciona uma subpasta oculta dentro do diretório existente que contém a estrutura de dados interna necessária para o controle de versão.
- `git clone` cria uma cópia local de um projeto que já existe remotamente. O clone inclui todos os arquivos, histórico e branches do projeto.
- `git add` prepara uma alteração. O Git controla as alterações na base de código de um desenvolvedor, mas é necessário testar e tirar um instantâneo das alterações para incluí-las no histórico do projeto. Este comando executa o teste, a primeira parte do processo de duas etapas. Todas as mudanças que são testadas irão tornar-se parte do próximo instantâneo e parte do histórico do projeto. O teste e o commit separados dão aos desenvolvedores total controle sobre o histórico do seu projeto sem alterar como eles codificam e funcionam.
- `git commit` salva o instantâneo no histórico do projeto e conclui o processo de controle de alterações. Em suma, um commit funciona como tirar uma foto. Qualquer item que

tenha sido preparado com `git add` passará a fazer parte do instantâneo com `git commit`.

- `git status` mostra o status das alterações como não controladas, modificadas ou preparadas.
- `git branch` mostra os branches que estão sendo trabalhados localmente.
- `git merge` mescla as linhas de desenvolvimento. De modo geral, esse comando é usado para combinar alterações feitas em dois branches distintos. Por exemplo, um desenvolvedor faria merge quando quisesse combinar alterações de um branch de recurso no branch principal para implantação.
- `git pull` atualiza a linha de desenvolvimento local com atualizações do equivalente remoto. Os desenvolvedores usam este comando se um colega fez commits em um branch remoto, e eles gostaria de refletir essas alterações no seu ambiente local.
- `git push` atualiza o repositório remoto com todos os commits feitos localmente em um branch.

Para obter mais informações, confira o guia de referência completo de comandos do Git.

Exemplo: Contribuir para um repositório existente

```
# download a repository on GitHub to our machine
# Replace `owner/repo` with the owner and name of the repository to clone
git clone https://github.com/owner/repo.git

# change into the `repo` directory
cd repo

# create a new branch to store any new changes
git branch my-branch

# switch to that branch (line of development)
git checkout my-branch

# make changes, for example, edit `file1.md` and `file2.md` using the text editor

# stage the changed files
git add file1.md file2.md

# take a snapshot of the staging area (anything that's been added)
git commit -m "my snapshot"

# push changes to github
git push --set-upstream origin my-branch
```

Exemplo: Inicie um novo repositório e publique-o em GitHub

Primeiro, você deverá criar um novo repositório em GitHub. Para obter mais informações, confira "Olá, Mundo". **Não** inicialize o repositório com um arquivo LEIAME, .gitignore ou License. Este repositório vazio irá aguardar seu código.

```
# create a new directory, and initialize it with git-specific functions
git init my-repo

# change into the `my-repo` directory
cd my-repo

# create the first file in the project
touch README.md

# git isn't aware of the file, stage it
git add README.md

# take a snapshot of the staging area
git commit -m "add README to initial commit"

# provide the path for the repository you created on github
git remote add origin https://github.com/YOUR-USERNAME/YOUR-REPOSITORY-NAME.git

# push changes to github
git push --set-upstream origin main
```

Exemplo: contribuir para um branch existente em GitHub

Este exemplo pressupõe que você já tenha um projeto chamado `repo` no computador e que um novo branch tenha sido enviado por push para o GitHub desde a última vez que as alterações foram feitas localmente.

```
# change into the `repo` directory
cd repo

# update all remote tracking branches, and the currently checked out branch
git pull

# change into the existing branch called `feature-a`
git checkout feature-a

# make changes, for example, edit `file1.md` using the text editor

# stage the changed file
git add file1.md
```

```
# take a snapshot of the staging area
git commit -m "edit file1"

# push changes to github
git push
```

Modelos para desenvolvimento colaborativo

Existem duas formas principais por meio das quais as pessoas colaboram em GitHub:

1. Repositório compartilhado
2. Bifurcação e pull

Com um repositório compartilhado, os indivíduos e as equipes são explicitamente designados como contribuidores com acesso de leitura, gravação ou administrador. Esta estrutura de permissão simples, combinada com funcionalidades como branches protegidos, ajuda as equipes a progredir rapidamente ao adotarem GitHub.

Para um projeto de código aberto, ou para projetos para os quais qualquer um pode contribuir, o gerenciamento de permissões individuais pode ser desafiador, mas uma bifurcação e um modelo de extração permite que qualquer pessoa que possa visualizar o projeto contribua. Um fork é uma cópia de um projeto na conta pessoal de um desenvolvedor. Cada desenvolvedor tem controle total da sua bifurcação e é livre para implementar uma correção ou novo recurso. O trabalho concluído nas bifurcações é mantido separado ou é retornado para o projeto original por meio de um pull request. Lá, os mantenedores podem revisar as alterações sugeridas antes de serem mesclados. Para obter mais informações, confira "Contribuindo com um projeto".