Introdução ao Git e Github: Tudo que você precisa saber!

GUILHERME ABRIL 10, 2018 BACK-END, FRONT-END, FULL-STACK, PROGRAMAÇÃO, WEB DESIGN

TEMPO DE LEITURA: **13 MINUTOS**

Se você não sabe o que é um **sistema de controle de versão** você está no lugar certo, pois é disso que essa **introdução ao Git e GitHub** irão tratar.

E para você que já faz parte do universo da programação seja um desenvolvedor Full-Stack, Front-End, Back-End, ou Programador Desktop, ou quem sabe está iniciando neste novo mundo e ainda não domina sequer o básico dessas ferramentas, fica comigo nesse artigo pois elas são indispensáveis para você que vai trabalhar com qualquer tipo de código.

Nesse tipo de ferramenta existe um **repositório** que nos permite obter qualquer versão existente do nosso código.

Sempre que desejarmos controlar a versão de algum arquivo, temos que informar que queremos rastreá-lo no repositório. E a cada mudança que queremos de fato efetivar, devemos armazenar as alterações nesse repositório.

As alterações que são feitas nos mesmos arquivos são mescladas de forma automática sempre que possível.

Já os possíveis conflitos são identificados a cada vez que obtemos as mudanças dos nossos colegas de time.

Desde 1990, esse tipo de ferramenta está em ação. Alguns exemplos de sistemas de controle de versão mais antigos são o: CVS, ClearCase, SVN e Source-Safe.

Mas por volta do ano 2000, surgiram os sistemas de controle de versão bem mais modernizados, mais velozes e confiáveis, como o Mercurial, Bazaar e, é claro, o nosso famoso Git.

Provavelmente se você nunca viu nada a respeito do Git e Github, não deu para compreender muita coisa da teoria abordada nesta introdução, e com toda razão, pois conceitos como esses só aprendemos na prática, então se você deseja de fato se aprofundar e entender minuciosamente como funciona toda essa engrenagem veja o que nós vamos aprender nesse artigo.

// Introdução ao Git

// Como funciona o Git

// Comandos básicos do Git

// Introdução ao Github

// Iniciando o GitHub

// Como funciona o Github

// Como criar e copiar repositórios

// “Empurrando” e “Puxando” alterações com o Push e Pull

// Possíveis conflitos

// Branches

// Estude com o GitHub

Gostou do que vai aprender? então vamos lá que essa introdução do Git e GitHub ta sensacional.

Introdução ao Git

O Git, como já foi mencionado antes, é um sistema de controle de versão que, pela sua estrutura interna, é como se fosse uma máquina do tempo extremamente veloz, e um robô de integração muito competente.

Ele foi criado em 2005 por Linus Torvalds, que nada mais nada menos é o mesmo criador do Linux, que não estava muito feliz com o BitKeeper, o sistema de controle de versão utilizado no desenvolvimento do kernel do Linux.

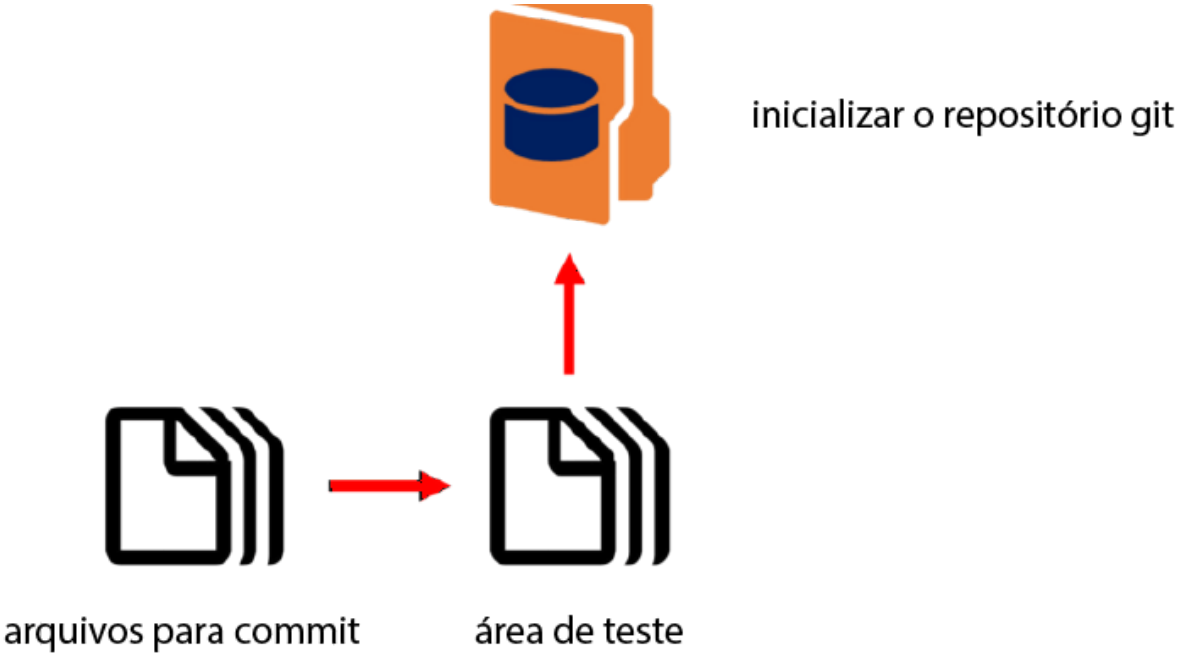
Mas hoje em dia, além do kernel do Linux, a ferramenta é usada em diversos outros projetos de código aberto.

E o mundo corporativo adotou o Git como uma ferramenta indispensável para se trabalhar com as equipes de desenvolvedores, tanto aqui no Brasil como no mundo.

Com o passar do tempo, o Git se tornou um padrão para o setor de desenvolvimento. A capacidade de capturar seu código em um determinado momento é incrivelmente útil à medida que sua base de código cresce e você precisa fazer referência a versões anteriores dele.

Nos dias de hoje, conhecer e compreender como utilizar o Git é uma habilidade indispensável para quem deseja ter uma carreira bem-sucedida no desenvolvimento de aplicações.

Como funciona o Git



Com o Git, você é capaz de gravar mudanças locais em seu código utilizando uma ferramenta de linha de comando, chamada *Git Shell* (O Git também pode ser utilizado por outras ferramentas de linha de comando).

Na linha de comando permite-se inserir comandos para visualizar, alterar e gerenciar seus arquivos e pastas em um terminal bem simples, em vez de se utilizar uma **interface gráfica de usuário (GUI**).

Se você nunca utilizou a linha de comando antes, pode ficar despreocupado, pois depois que você iniciar, verá que é incrivelmente simples.

Ao utilizar o Git, você fará alterações em seus arquivos de código como faria normalmente durante o processo de desenvolvimento.

Quando você tiver concluído uma etapa da codificação, ou quiser capturar certas alterações, basta adicionar os arquivos que você alterou em uma área de preparação e então os confirme no histórico de versões do seu projeto (repositório) utilizando o Git.

Abaixo vamos começar a entrar na linha de comando para você ver na prática o funcionamento de cada etapa.

Comandos básicos do Git

Enquanto você estiver utilizando o Git através da linha de comando, provavelmente você também irá utilizar alguns comandos básicos no terminal enquanto passa pelo seu projeto e arquivos / pasta do sistema, incluindo:

- **pwd** – que verifica onde você está no sistema de arquivos atual.
- **ls** – lista arquivos no diretório atual.
- **cd** [nome-do-diretório] – move-se para o nome ou caminho do diretório fornecido.
- **mkdir** [nome-do-diretório] – cria um novo diretório com o nome escolhido.

Criando os repositórios

Quando você tem o desejo de utilizar o Git para um projeto, o primeiro comando que você deve executar é o **git init**, com o nome do seu projeto:

git init [nome do projeto]

Você irá executar este comando na linha de comando do Git Shell no diretório principal do seu projeto, o qual você poderá navegar no Shell utilizando os comandos que listamos acima.

Depois de executar esse comando o Git então criou um arquivo .git oculto dentro do diretório principal do seu projeto.

Este arquivo rastreia o histórico de versões do seu projeto e é o que transforma o projeto em um repositório Git, permitindo que você execute os comandos do Git nele.

Fazendo alterações

git add [arquivo] ou git add *

Depois de fazer as alterações em seus arquivos e escolher por capturá-los para o histórico de versões do seu projeto, você deve colocá-los na área de preparação com o *git add,* pelo nome do arquivo ou incluindo todos os arquivos na sua pasta atual utilizando o *git add *.*

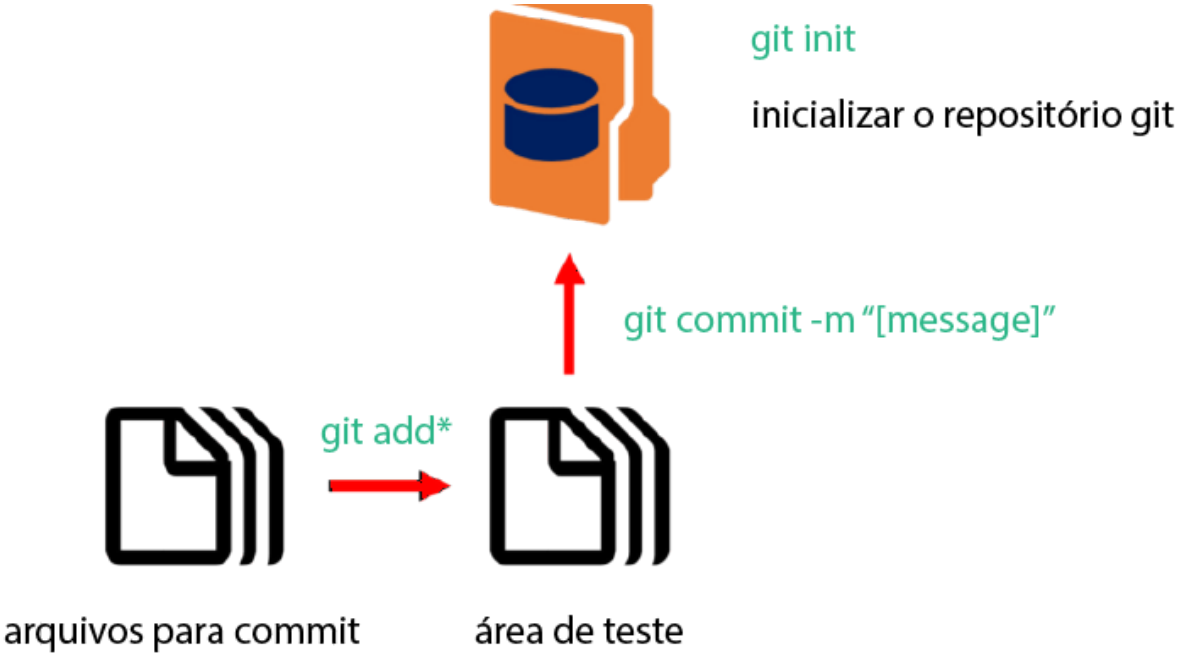
git commit -m "[message]"

Para finalmente fazer o commit das alterações feitas nos seus arquivos da área de preparação para o histórico de versões do seu repositório, você precisa executar o *git commit* com uma mensagem descritiva de quais alterações foram realizadas.

git status

Se, a qualquer instante, você desejar visualizar um resumo dos arquivos que você modificou e ainda não confirmou, simplesmente execute o **git status** no repositório do seu projeto na linha de comando do Git Shell.

Como funciona?



Agora, que temos os comandos básicos do Git no seu devido lugar, você pode utilizar o Git para capturar de forma instantânea o histórico de versões do seu projeto.

Basta que você inicialize um novo repositório executando o *git init* no diretório principal do seu projeto.

Usando git add * ou git add com os nomes dos arquivos específicos, você adiciona as suas alterações à área de preparação.

Finalmente, utilizando o *git commit,* você pode adicionar suas alterações ao histórico de versões do repositório.

A medida que você continua desenvolvendo o seu código, continue adicionando e confirmando suas alterações no seu repositório.

Introdução ao Github

Na sua essência, o GitHub é um serviço que permite hospedar seus repositórios Git on-line e colaborar com uma série de outros usuários.

Você tanto pode utilizar o GitHub por meio de seu portal na web, bem como a GUI (Interface Gráfica do Usuário) da área de trabalho do GitHub e o Git Shell.

O serviço prestado pela GitHub hoje é utilizado por mais de 12 milhões de desenvolvedores e organizações, e acabou se tornando um padrão popular para colaboração em projetos de código fonte aberto.

Iniciando o GitHub

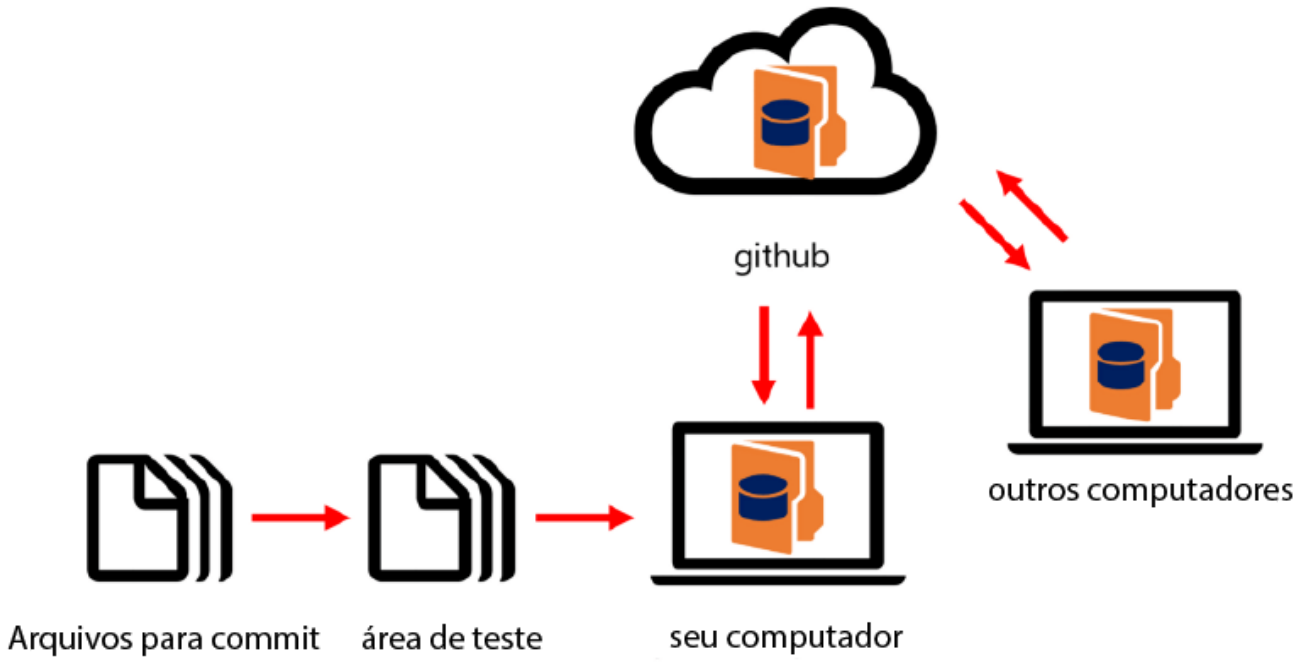
Para iniciar no Git e o GitHub, inscreva-se em uma conta do GitHub em github.com

Logo em seguida, baixe a GUI de desktop do GitHub em desktop.github.com. Esta é uma versão do aplicativo que usa o Git na linha de comando. Assim também deve baixar o Git Shell.

A etapa final desse processo é inserir suas credenciais do GitHub. Você pode fazer isso a partir do Git Shell ou usando o aplicativo desktop GitHiub.

Você pode ver o artigo de configuração do GitHub aqui, se deseja mais informações.

Como funciona o Github



No GitHub, você verá o mesmo processo local de adicionar e confirmar arquivos em um repositório Git que foi inicializado em seu computador.

No entanto, você pode utilizar o GitHub para enviar suas modificações para o serviço de hospedagem do GitHub. Isso possibilita que outras pessoas trabalhem de maneira semelhante em um mesmo projeto, obtenham suas alterações em seus computadores e enviem suas próprias alterações para o GitHub.

Abaixo mostro os comandos que você irá precisar para utilizar o Git com o GitHub

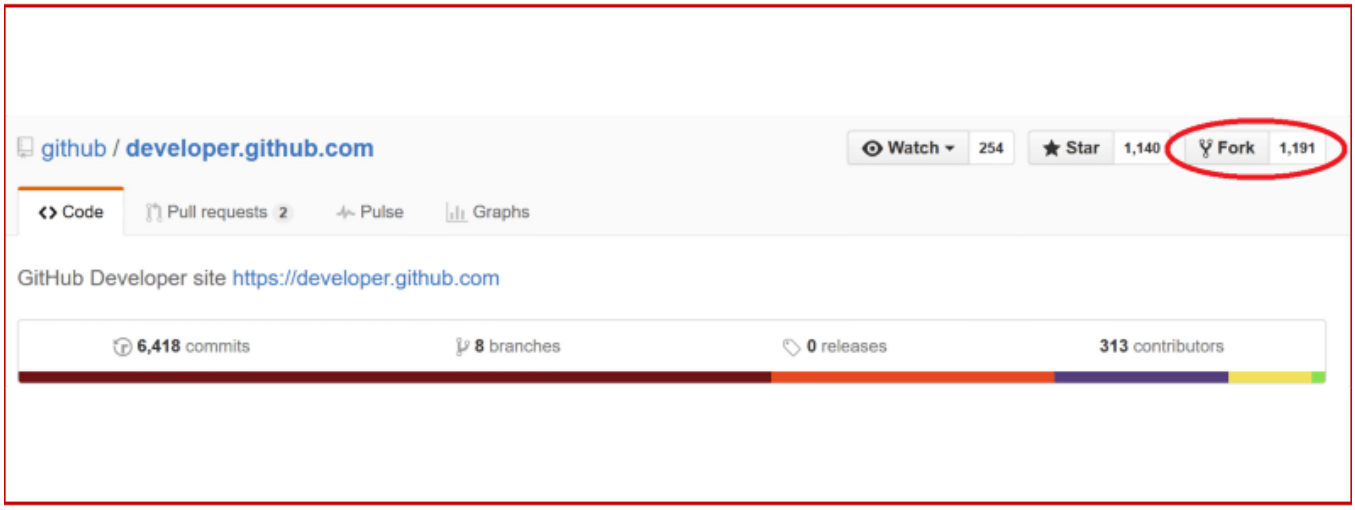
Como criar e copiar repositórios

fork

Com o git no seu ambiente local, se você quiser criar um repositório novo, você deve executar o git init.

No entanto, você também pode trabalhar em projetos hospedados no GitHub que já foram inicializados.

Uma das maneiras de copiar um repositório para a sua conta do GitHub é utilizando o fork, que está disponível no site do GitHub.



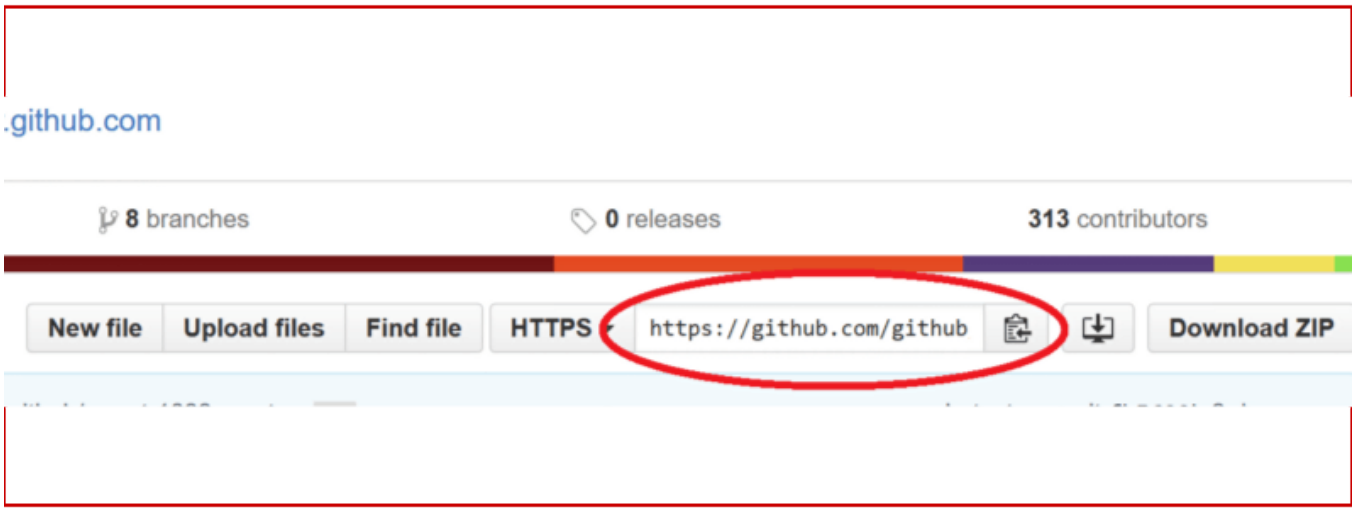
Dar Forking em um repositório essencialmente copia esse projeto para sua conta on-line do GitHub.

clone git [url]

O clone de um projeto simplesmente faz a cópia de um repositório Git com seu histórico de versões, por sua URL, para o computador local a partir do GitHub.

A partir daí, você já pode fazer e confirmar todas as suas alterações nesse repositório.

Qualquer alteração que você confirmar e enviar ao GitHub (conforme na imagem abaixo) serão salvas para a cópia que você fez desse projeto.



“Empurrando” e “Puxando” alterações com o Push e Pull.

git push [repo] [branch]

Para publicar um novo repositório inicializado ou qualquer alteração confirmada em seu repositório local para o GitHub, use o **git push** com o repositório e o branch do seu código.

A sintaxe padrão que geralmente você vai utilizar é **git push origin master**. Que enviará o seu código para o branch master (mostro logo abaixo) do seu repositório.

git pull

Se alguma alteração ocorrer em algum repositório que você queira extrair do GitHub para o seu computador local, por exemplo, ao colaborar em um projeto com outros desenvolvedores, basta utilizar o *git pull*.

Possíveis conflitos

É muito importante lembrar que, por mais que você tenha um repositório hospedado no GitHub, o histórico de versões de sua cópia local pode ser diferente de versões do repositório que você tem on-line.

Portanto, se você tentar extrair ou enviar modificações para arquivos que já foram modificados por outras pessoas e essas modificações já estiverem no GitHub, você poderá entrar em um ***merge conflict*** (*conflito de mesclagem).*

Um merge conflict ou conflito de mesclagem, essencialmente acontece quando o histórico de versões do repositório em seu computador local é diferente do histórico de versões do repositório no GitHub.

Você precisará resolver o conflito de mesclagem manualmente para reunir todas as alterações do repositório.

Se você quiser saber mais sobre os conflitos de mesclagem, acesse o artigo de ajuda do GitHub.

Branches

Ao utilizar o Git, você consegue visualizar o histórico de versões do desenvolvimento do seu projeto. No entanto, você pode escolher por desenvolver recursos, corrigir erros ou experimentar formas de separar o código do seu projeto principal de outra variante.

Você pode fazer isso com os *branches*, que são essencialmente versões paralelas do código principal do seu repositório – esse código é desenvolvido na “master” branch.

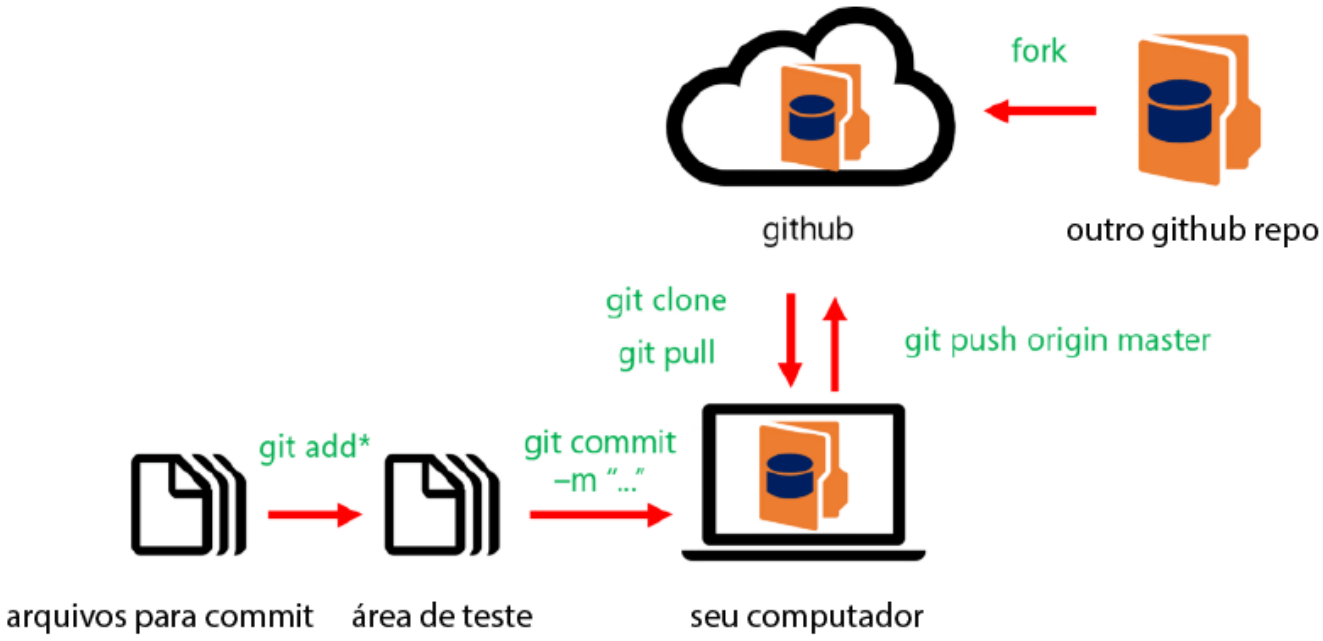
Você também pode criar vários *branches* para colaboração e outro exclusivo para o desenvolvimento do seu código.

Todas as alterações feitas nos arquivos em um branch permanecem apenas no histórico de versões desse branch.

Se, a qualquer momento, você quiser fazer um merge (mesclar) no código de um branch para outro, incluindo o branch principal, faça uma solicitação pull que mescle as alterações.

Mas, semelhante a utilização do *git pull* e *git push*, se modificações diferentes tiverem sido feitas nos mesmos arquivos em mais de um branch, você terá conflitos de mesclagem que devem ser resolvidos manualmente.

Como tudo isso funciona?



Com o GitHub, você estará totalmente habilitado para usar colaborativamente o Git para construir seus próprios repositórios e os dos outros.

Você pode distribuir um repositório inicializado para sua própria conta e clonar qualquer repositório do GitHub para o seu computador.

À medida que você continua *adicionando* e *confirmando as* alterações feitas nos arquivos em seu computador local com o Git, você pode *enviar* essas mudanças e *extrair* qualquer outra confirmação para o GitHub.

Estude com o GitHub

Aqui nós temos algo bem interessante para lhe mostrar, como aluno você pode acessar o GitHub Student Developer Pack, que inclui uma série de benefícios do GitHub e de outros parceiros.

Saiba mais sobre o Student Developer Pack clicando aqui.

Conclusão da Introdução ao Git e GitHub

Gostou do artigo? Tem alguma sugestão? Utilize os comentários para deixar suas duvidas ou sugestões.

Para saber mais sobre como usar o Git e o GitHub, confira os recursos nos seguintes links:

git-scm.com – O site principal e a documentação do Git

help.github.com – Documentação de ajuda do GitHub

Sobre Guilherme

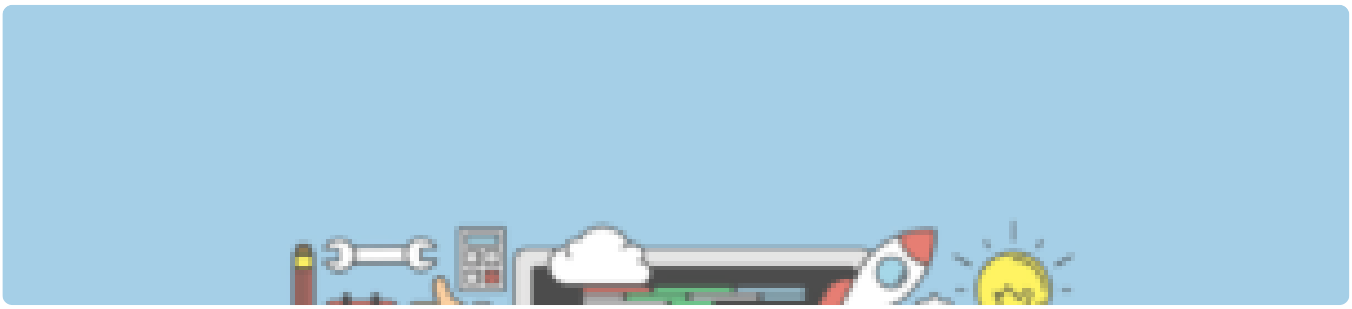
Desenvolvedor Full-Stack Web há 10 anos. Game Developer com grandes títulos na AppStore e Google Play. Empreendedor Digital e apaixonado por marketing.

Artigos relacionados

Pré-processadores CSS: O Guia Definitivo

Curso de Desenvolvimento de Games Completo

Pure CSS: Conheça esse incrível Framework CSS



Aprender a Programar do Zero: Dicas para Iniciar sua Jornada

CSS Grid Layout: Guia Absolutamente Completo

Criando sua Primeira Aplicação com AngularJS

ARTIGO ANTERIOR
Como fazer um site responsivo: Guia Completo e Definitivo

PRÓXIMO ARTIGO
5 Formas de Ganhar Dinheiro Programando, Mesmo para Iniciantes!

Buscar por:

Posts recentes

- Dez principais habilidades para se tornar um desenvolvedor full-stack em 2021
- O que é Python, e como ela se tornou uma Linguagem Poderosa e Dinâmica
- 7 Regras para desenvolver um CSS inteligente e robusto
- Os melhores jogos para PC de 2020: Momento de Distração
- Monetização de sites: do tráfego a rentabilidade mensal

Comentários

- Guilherme em 7 motivos para você trabalhar com desenvolvimento de games
- Guilherme em O que dá para fazer com JavaScript + Dicas de Framework
- ANTONIO MUEMBANZA em 7 motivos para você trabalhar com desenvolvimento de games
- Guilherme em Qual a melhor linguagem de programação para desenvolver Jogos?
- Guilherme em O que é Game Design e o que um game designer faz

Foco, impacto e criatividade

Insira o seu endereço de e-mail abaixo para receber **gratuitamente** as atualizações do blog!

Insira seu e-mail aqui

ASSINE JÁ!

Categorias

- Back-End
- Desenvolvimento de Jogos
- Empreendedorismo
- Front-End
- Full-Stack
- Marketing Digital
- Programação
- Uncategorized
- Web Design

Pacote Full-Stack