

# git - guia prático

apenas um guia prático para começar com git. sem complicação ;)

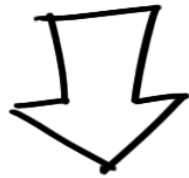
por Roger Dudler

créditos para @tfnico, @fhd and Namics

guia em english, deutsch, español, français, italiano, nederlands, русский,

日本語, 中文, 한국어

por favor informe problemas em github



## instalação

Baixe o git para OSX

Baixe o git para Windows

Baixe o git para Linux

## criando um novo repositório

crie uma nova pasta, abra-a e execute o comando

```
git init
```

para criar um novo repositório.

## obtenha um repositório

crie uma cópia de trabalho em um repositório local executando o comando

```
git clone /caminho/para/o/repositório
```

quando usar um servidor remoto, seu comando será

```
git clone
```

```
usuário@servidor:/caminho/para/o/repositório
```

## fluxo de trabalho

seus repositório local consiste em três "árvores" mantidas pelo git. a primeira delas é sua **Working Directory** que contém os arquivos vigentes. a segunda **Index** que funciona como uma área temporária e finalmente a **HEAD** que aponta para a última confirmação que você fez.



## adicionar & confirmar

Você pode propor mudanças (adicioná-las ao **Index**) usando

```
git add <arquivo>
```

```
git add *
```

Este é o primeiro passo no fluxo de trabalho básico do git. Para realmente

confirmar estas mudanças use

```
git commit -m "comentários das alterações"
```

Agora o arquivo é enviado para o **HEAD**, mas ainda não para o repositório remoto.

## enviando alterações

Suas alterações agora estão no **HEAD** da sua cópia de trabalho local. Para

enviar estas alterações ao seu repositório remoto, execute

```
git push origin master
```

Altere *master* para qualquer ramificação desejada, enviando suas alterações para ela.

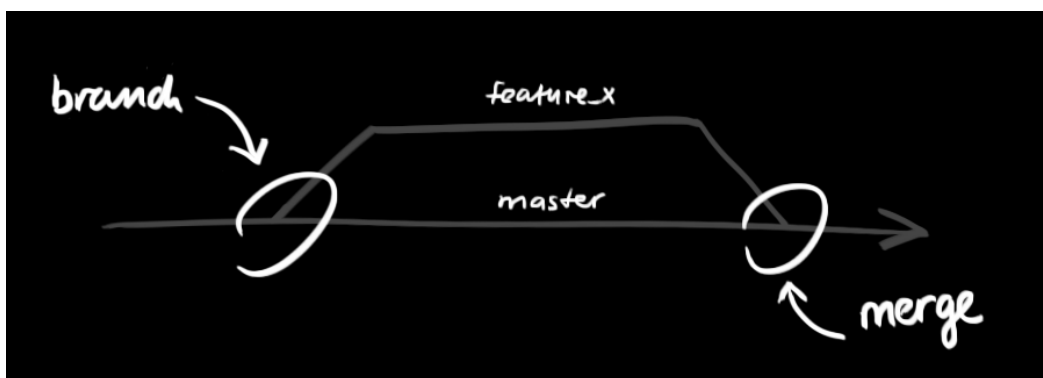
Se você não clonou um repositório existente e quer conectar seu repositório a um servidor remoto, você deve adicioná-lo com

```
git remote add origin <servidor>
```

Agora você é capaz de enviar suas alterações para o servidor remoto selecionado.

## ramificando

Ramos são utilizados para desenvolver funcionalidades isoladas umas das outras. O ramo *master* é o ramo "padrão" quando você cria um repositório. Use outros ramos para desenvolver e mescle-os ao ramo *master* após a conclusão.



crie um novo ramo chamado "funcionalidade\_x" e selecione-o usando

```
git checkout -b funcionalidade_x
```

retorne para o *master* usando

```
git checkout master
```

e remova o ramo da seguinte forma

```
git branch -d funcionalidade_x
```

um ramo *não está disponível a outros* a menos que você envie o ramo para seu

repositório remoto

```
git push origin <ramo>
```

## atualizar & mesclar

para atualizar seu repositório local com a mais nova versão, execute

```
git pull
```

na sua pasta de trabalho para *obter* e *mesclar* alterações remotas.

para mesclar um outro ramo ao seu ramo ativo (ex. master), use

```
git merge <ramo>
```

em ambos os casos o git tenta auto-mesclar as alterações. Infelizmente, isto nem sempre é possível e resulta em *conflitos*. Você é responsável por mesclar estes *conflitos* manualmente editando os arquivos exibidos pelo git. Depois de

alterar, você precisa marcá-los como mesclados com

```
git add <arquivo>
```

antes de mesclar as alterações, você pode também pré-visualizá-as usando

```
git diff <ramo origem> <ramo destino>
```

# rotulando

é recomendado criar rótulos para releases de software. Este é um conhecido conceito, que também existe no SVN. Você pode criar um novo rótulo chamado

*1.0.0* executando o comando

```
git tag 1.0.0 1b2e1d63ff
```

o *1b2e1d63ff* representa os 10 primeiros caracteres do id de confirmação que você quer referenciar com seu rótulo. Você pode obter o id de confirmação com

```
git log
```

você pode também usar menos caracteres do id de confirmação, ele somente precisa ser único.

## sobrescrever alterações locais

No caso de você ter feito algo errado (que seguramente nunca acontece ;) você

pode sobrescrever as alterações locais usando o commando

```
git checkout -- <arquivo>
```

isto substitui as alterações na sua árvore de trabalho com o conteúdo mais recente no HEAD. Alterações já adicionadas ao index, bem como novos arquivos serão mantidos.

Se ao invés disso você deseja remover todas as alterações e confirmações locais, recupere o histórico mais recente do servidor e aponte para seu ramo

master local desta forma

```
git fetch origin
```

```
git reset --hard origin/master
```

## dicas úteis

Interface gráfica padrão

```
gitk
```

usar saídas do git coloridas

```
git config color.ui true
```

exibir log em apenas uma linha por confirmação

```
git config format.pretty oneline
```

fazer inclusões interativas

```
git add -i
```

## recursos & links

clientes gráficos

GitX (L) (OSX, código aberto)

Tower (OSX)

Source Tree (OSX, gratuito)

GitHub for Mac (OSX, gratuito)

# guias

Livro da comunidade Git


Pro Git

Pense como um git

Ajuda do GitHub

Um guia visual do Git

## comentários

Like  and 9 others liked this.

add new comment

Login



showing 6 comments

Sort by newest first ▼



**Flávio micheletti**

Pô cara, de 0 a 10, nota 11.

Muito bom mesmo, claro, objetivo e preciso.

Parabéns

1 month ago

Like Reply



**Daniel Cambria**

Parabéns, está ótimo o guia. Layout 100%

1 month ago

Like Reply



**squíter**

Cara, você mandou muito bem!

Parabens!

Nunca mais perco meu tempo explicando GIT pra alguém, vou sempre mandá-los pra cá! :D

2 months ago

Like Reply



**Jeudi Prando**

show de bola! sugeri para ser recomendado para todo mundo lá da empresa!

2 months ago

Like Reply



**Rafael Sandrini**

Show! parabens excelente post

2 months ago

Like Reply



**Eduardo Rabelo**, O conhecimento é irresistível

Simples, eficiente e objetivo!






Eu aprendi a usar Git ;)

5 months ago

Like Reply

✉ [Subscribe by email](#)  [RSS](#)

reactions



Trackback URL