

Tutorial Git

Yuri Dimitre Dias de Faria

Fevereiro 2020

1 O que é Git?

Git é um sistema de controle de versões distribuído desenvolvido por Linus Torvalds.

Seu objetivo inicial era o versionamento e a geração de um histórico de alterações do desenvolvimento do kernel Linux. Rapidamente o Git foi sendo adotado por outros projeto, sendo eles de desenvolvimento de software ou com propostas gerais.

Git é distribuído sob a licença GNU General Public License . Atualmente sua manutenção é supervisionada por Junio Hamano.

2 Como instalar o Git(Linux)

O Git pode vir por padrão em varias distribuições. Para ver se sua distro já possui ele, basta apenas rodar o comando “git - -version”.

Para instalar, verifique qual gerenciador de pacotes sua distribuição utiliza.

DPKG

```
# apt-get install -y git
```

RPM

```
# dnf install -y git
```

PACMAN

```
# pacman -Sy git
```

3 Configurações do Git

Agora que o Git está instalado no sistema, podemos personalizar seu ambiente. As suas customizações se manterão mesmo depois do programa ser atualizado.

Alem disso, as configurações poderão ser alteradas a qualquer momento, bastando apenas rodar os comandos novamente.

O Git vem com uma ferramenta já pronta para configuração, bastando apenas rodar o comando *git config* e atribuir valores as variáveis.

Estas variáveis ficam armazenadas em três lugares distintos:

- **/etc/gitconfig:** Valido para todos os usuários do sistema. Para isso, utilize a flag `-system` do comando *git config*.
- **~/.gitconfig ou ~/.config/git/config:** Valido apenas para o usuário atual. Para isso, utilize a flag `-global` do comando *git config*.
- **config(.git/config) de cada repositório:** Especifico do repositório.

Você pode ver suas configurações com o comando *git config -list*.

Sua identidade

A primeira coisa a ser feita após o Git ser instalado é a configuração de identidade. Ela é importante já que todo commit é carimbado com essas informações, que são imutáveis.

```
$ git config --global user.name "Nome"
```

```
$ git config --global user.email Nome@domain.br
```

Note a flag `-global`, o que significa que essa configuração será válida para o usuário atualmente logado.

4 Repositórios

Repositórios são a maneira do Git trabalhar. Você pode ter mais de um em sua máquina, mas cada projeto possui o seu exclusivo.

Existem duas formas de adquirir um repositório, criando ou clonando.

Esse tutorial irá abranger comandos que irão ser executados em emuladores de terminal ou janelas de comando. Existem programas, como o GitKraken, que é possível criar e gerenciar repositórios por meio de interfaces gráficas. No entanto, é preciso um conhecimento prévio e bem estabelecido dos comandos Git para sua total usabilidade.

Criando

Para criar um repositório, primeiro precisa navegar até a pasta onde está seu projeto. O comando *cd* é usado para mudar o diretório atual da sua janela de comandos.

```
cd /users/user/Projects/Tutorial
```

Assim que estiver na pasta do seu projeto, um repositório Git pode ser criado pelo comando *git init*. Esse comando irá criar um subdiretório **.git** dentro da pasta do seu projeto.

Clonando

Para clonar um repositório existente, navegue até a pasta que deseja salva-lo e execute `git clone [url]`.

Por exemplo, se quiser clonar o repositório do youtube-dl no Github, basta apenas executar o comando:

```
git clone https://github.com/ytdl-org/youtube-dl/
```

5 Alterando um repositório

O Git trabalha com status quando se modifica o conteúdo do repositório:

- **Untracked:** Arquivos que não foram vinculados ao repositório ainda.
- **Unmodified:** Arquivos do repositório que não sofreram alterações, seja uma modificação no conteúdo ou sua exclusão.
- **Modified:** Arquivos que foram modificados, mas suas alterações ainda não foram definidas e enviadas para o repositório.
- **Staged:** Arquivos que foram modificados e serão enviados para o repositório.

Para verificar o status do arquivo, execute o comando `git status`

Diff

É possível visualizar as alterações de arquivos marcados como “Modified” antes de eles passarem para “Staged”. Basta executar `git diff` que será possível visualizar as alterações. O que está de vermelho e a linha possuir um “-” significa que foi removido, e o que estiver de verde e a linha possuir um “+” significa que foi adicionado.

Você ainda pode visualizar as modificações dos arquivos marcados como “Staged” com o último commit adicionando o flag “- staged”.

Commit

Commits são a maneira de gerencia que o Git utiliza. Um commit possui todas as alterações que serão enviadas para o repositório. Todos os arquivos com status “Modified” serão enviados pelo mudados para “Staged”.

Para se adicionar arquivos no commit, utiliza-se o comando `git add [file]`.

Para se realizar um commit basta usar o comando `git commit`. Você ainda pode incluir uma mensagem com uma informação junto com a flag `-m` e a mensagem a frente.

```
git add *.c
```

```
git commit -m “Adicionando todos os sources codes”
```

Lembre-se que o commit ainda não foi enviado para o repositório, ou seja, eles são armazenados localmente.

Para mais informações, consulte o manual utilizando o comando `man git-commit`

.gitignore

É possível ignorar arquivos ou até pastas inteiras quando montar seu commit. Para isso, basta criar um arquivo chamado *.gitignore* e colocar o nome dos arquivos ignorados dentro.

O repositório <https://github.com/github/gitignore> possui alguns *.gitignore* prontos de várias linguagens.

6 Enviando e recebendo arquivos

Push

Para enviar informações para o repositório remoto, utilizaremos o comando *git push*. Para isso, é preciso que exista algum commit pendente para ser enviado.

É possível passar alguns argumentos para o comando, como o link do repositório remoto ou qual a branch será enviado o commit. Os argumentos mais usados são *origin master*, onde “origin” é o repositório padrão que foi configurado e “master” é a branch.

Existe uma infinidade de argumentos e flags para o comando *push*, você pode verificar utilizando o comando *man git-push* ou acessando <https://git-scm.com/docs/git-push>

Pull

Para atualizar os arquivos do repositório local basta utilizar o comando *git pull*. Isso irá mesclar os arquivos da branch atual do repositório remoto para o repositório local. Esse comando é uma mescla dos comandos *git fetch* e *git merge FETCH_HEAD*.

Checkout

Caso tenha feito alguma alteração indevida, e precisa sobrescrever as alterações locais, basta usar o comando *git checkout [file]* que ele retornará o arquivo para o estado mais recente no HEAD.

Tags

É possível colocar uma etiqueta em um commit para melhor gerenciamento do seu repositório. Por exemplo, você pode colocar a etiqueta “1.0.0” em um commit para simbolizar o lançamento de uma versão.

Para adicionar uma tag a um commit basta usar o comando *git tag [message] [id commit]*, onde o id de commit são os 10 primeiros números que você quer referenciar com seu rótulo. Para conseguir o id do commit utilize o comando *git log*

7 Branch

Branch, assim como o nome sugere, é uma ramificação no desenvolvimento do software. Muitas vezes precisamos acrescentar um recurso novo mas não queremos prejudicar o projeto principal com bugs, ou precisamos dividir o desenvolvimento do projeto para vários contribuidores.

Branches podem ser criadas com o comando *git checkout -b [branch name]*. Para voltar para a branch “master” basta usar o comando *git checkout master*. Para deletar a branch basta usar a flag “-d” junto do nome: *git checkout -d [branch name]*.

Merge

Quando se trabalha com varias branches, chega uma hora que é preciso juntar todas em uma só. Para isso, basta utilizar o comando *git merge [branch]*. Esse comando irá “mesclar” o conteúdo da branch atual com a especificada no comando.

Resolvendo conflitos

Quando se mescla duas branches é possível que ocorra conflitos entre os arquivos. Para resolver esses conflitos basta editar os arquivos e adiciona-los ao proximo commit.

O git é irá marcar onde ocorre o conflito, com o conteúdo da branch mesclada com o próprio nome e o conteúdo da atual com o HEAD.

Rebase

Enquanto o merge mescla os arquivos de duas branches, o rebase “refaz” a base delas, unificando-as. Isso irá produzir um histórico linear, mais limpo e fácil de ser lido.

Como o rebase mexe nas estruturas das branches envolvidas, reescrevendo o histórico de commits, pode levar a perda de trabalho e histórico se não for bem executada. Além disso, como não se gera um commit de merge, apenas junta as branches, fica mais difícil de rastrear quando foi feito o rebase.

Para usar o rebase basta usar o comando *git rebase*

8 Repositórios remotos

Repositórios remotos são servidores que possuem o seu repositório para a sincronização e trabalho em equipe. Dezenas de sites oferecem serviços de host de repositórios, como o Github e o Gitlab.

Para adicionar um link do repositório remoto ao seu repositório atual basta usar o comando *git remote add origin [url]*.

Github

Github é uma plataforma de hospedagem de código-fonte utilizando o Git. Amplamente usada para divulgação de trabalhos, o site permite a criação de times e contribuições a repositórios Open Sources.

Foi desenvolvida por Chris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon usando o framework Rails da linguagem Ruby e seu lançamento foi no ano de 2008. Em 2018 a plataforma foi adquirida pela Microsoft por U\$ 7,5 bilhões.

Criando um repositório

A criação de um repositório no site é bem intuitiva.

Depois de criado a conta e logado, na tela *dashboard* inicial, vá ao canto direito superior e clique na sua foto de perfil. Em seguida, no menu *dropdown*, clique em “*Your Repositories*”. Depois, clique em “*New*” no canto direito.

Na tela de criação, será necessário colocar o nome do repositório para a exibição no site e se o repositório será público ou privado. Você ainda pode colocar uma breve descrição, adicionar automaticamente um *.gitignore* e uma licença, assim como gerar automaticamente um arquivo *Markdown* *Readme.md*.

Fork

Um fork de um repositório é uma cópia, no estado atual, para uma modificação de outros usuários. Vários projetos são bifurcados de outros, como o Unity DE que é um fork do Gnome, ou LineageOS que é um fork do Cyanogenmod.

Para criar um fork de um software para o uso próprio basta entrar no repositório e clicar em “*Fork*” no canto superior. Lembre-se de ler a licença quanto a modificação e distribuição do mesmo.

README.md

O arquivo README.md é um arquivo escrito em *Markdown* utilizado para se informar sobre o repositório.

Markdown é uma linguagem de marcação que converte seu texto para html válido. Para mais informações acesse o link <https://www.markdownguide.org/>.

9 Conclusão

A ferramenta Git é muito poderosa e é de extrema importância seu aprendizado. Nesse artigo foi abordado apenas o básico, mas você já consegue gerenciar um repositório por conta própria.

Para mais informações, consulte a *man page* de cada comando usando o comando *man [command]*, ou verifique a documentação oficial no site <https://git-scm.com/docs/>