Tutorial Git

Yuri Dimitre Dias de Faria

Fevereiro 2020

1 O que é Git?

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

Seu objetivo inicial era o versionamento e a geração de um historico 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 e distribuido sob a licensa GNU General Public License **GNULicense**. Atualmente sua manutenção e supervisionada por Junio Hamano.

2 Como instalar o Git(Linux)

O Git pode vir por padrão em varias distribuições.Para ver se sua distro ja 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 ja pronta para configuração, bastando apeanas rodar o comando *git config* e atribuir valores as variaveis.

Estas variaveis ficam armazenadas em três lugares distintos:

- **/etc/gitconfig:** Valido para todos os usuarios do sistema.Para isso, utilize a flag –system do comando git config.
- **˜/.gitconfig ou ˜/.config/git/config:** Valido apenas para o usuario atual. Para isso, utilize a flag –global do comando git config.
- **config(.git/config) de cada repositorio:** Especifico do repositorio.

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

Sua identidade

A primeira coisa a ser feita após o Git ser instalado e a configuração de identidade. Ela é importante ja que todo commit e carimbado com essas informações, que sao imutaveis.

$ 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 valida para o usuario atualmente logado.

4 Repositorios

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

Exitem duas formas de adquirir um repositorio, 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 possivel criar e gerenciar repositorios por meio de interfaces graficas. No entanto, é preciso um conhecimento previo e bem estabelecido dos comandos Git para sua total usabilidade.

Criando

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

cd /users/user/Projects/Tutorial

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

Clonando

Para clonar um repositorio existente, navegue ate a pasta que deseja salva-lo e execute *git clone [url]*.

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

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

5 Alterando um repositorio

O Git trabalha com status quando se modifica o conteudo do repositorio:

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

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

Diff

É possivel visualizar as alterações de arquivos marcados como "Modified" antes de eles passarem para "Staged". Basta executar *git diff* que será possivel visualizar as alterações. O que esta 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 ultimo 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 repositorio. 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 repositorio, ou seja, eles são armazenados localmente.

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

.gitignore

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

O repositorio `https://github.com/github/gitignore` possui alguns .gitignore prontos de varias linguagens.

6 Enviando e recebendo arquivos

Push

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

É possivel passar alguns argumentos para o comando,como o link do repositorio remoto ou qual a branch será enviado o commit. Os argumentos mais usados são *origin master*, onde "origin" e o repositorio 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 repositorio local basta utilizar o comando *git pull*. Isso irá mesclar os arquivos da branch atual do repositorio remoto para o repositorio 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 possivel colocar uma etiqueta em um commit para melhor gerenciamento do seu repositorio. Por exemplo,você pode colocar a etiqueta "1.0.0" em um commit para simbolizar o lançamento de uma versão.

Par adicionar uma tag a um commit basta usar o comando *git tag [message] [id commit]*, onde o id de commit são os 10 primeiros numeros que você quer

referenciar com seu rotulo. Para conseguir o id do commit utilize o comando *git log*

7 Branch

Branch,assim como o nome sugere,e 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 varios contribuidores.

Branchs 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 branchs, 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 conteudo da branch atual com a especificada no comando.

Resolvendo conflitos

Quando se mescla duas branchs é possivel 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 conteudo da branch mesclada com o proprio nome e o conteudo da atual com o HEAD.

Rebase

Enquanto o merge mescla os arquivos de duas branchs, o rebase "refaz" a base delas,unificando-as. Isso irá produzir um historico linear, mais limpo e facil de ser lido.

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

Para usar o rebase bastar usar o comando *git rebase*

8 Repositorios remotos

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

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

Github

Github é uma plataforma de hospedagem de codigo-fonte utilizando o Git. Amplamente usada para divulgação de trabalhos,o site permite a criação de times e contribuições a repositorios 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 repositorio

A criação de um repositorio 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 *dropdowm*, clique em *"Your Repositories"*. Depois,clique em *"New"* no canto direito.

Na tela de criação,será necessario colocar o nome do repositorio para a exibição no site e se o repositorio sera publico ou privado. Você ainda pode colocar uma breve descrição,adicionar automaticamente um *.gitignore* e uma licensa,assim como gerar automaticamente um arquivo *Markdowm* Readme.md .

Fork

Um fork de um repositorio é uma copia , no estado atual, para uma modificação de outros usuarios. Varios 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 proprio basta entrar no repositorio e clicar em *"Fork"* no canto superior. Lembre-se de ler a licensa 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 repositorio.

Markdowm é uma linguagem de marcação que converte seu texto para html valido. Para mais informações acesse o link `https://www.markdownguide.org/`.

9 Conclusão

A ferramenta Git é muito poderosa e é de extrema importancia seu aprendizado. Nesse artigo foi abordado apenas o basico,mas você ja consegue gerenciar um

repositorio por conta propria.