GIT Tutorial Para Iniciantes

O que é GIT?

Em 2005, Linus Torvalds (o homem conhecido por criar o núcleo, ou kernel, do SO Linux) desenvolveu o GIT, que desde então tem sido ativamente mantido por Junio ​​Hamano, um engenheiro de software japonês.

Atualmente, o GIT é um dos mais famosos sistemas de controle de versão de código aberto e milhões de projetos no mundo inteiro o utilizam para seu controle de versão (incluindo projetos comerciais e de código aberto). O Git é totalmente grátis e pode ser instalado em Mac, Linux, Windows e Solaris diretamente do site oficial . Algumas das características essenciais do GIT são:

* Um sistema de controle de versão distribuído, o GIT segue uma abordagem peer to peer, contrário de outros como o Subversion (SVN) que segue um modelo baseado em cliente-servidor.
* GIT permite aos desenvolvedores ter uma infinidade de ramos de código completamente independente. Criação, exclusão e fusão desses ramos é simples e não leva tempo.
* No GIT, todas as operações são atômicas. Isso significa que uma ação pode ter sucesso ou falhar (sem fazer nenhuma alteração). Isso é importante porque em alguns sistemas de controle de versão (como o CVS) onde as operações não são atômicas, se uma operação de repositório é suspensa, ela pode deixar o repositório em um estado instável.
* No GIT, tudo é armazenado dentro da pasta .git. Isso não é o mesmo em outros VCS como SVN e CVS onde os metadadados de arquivos são armazenados em pastas ocultas (por exemplo, .cvs, .svn, etc.)
* GIT usa um modelo de dados que ajuda a garantir a integridade criptográfica de qualquer coisa presente dentro de um repositório. Cada vez que um arquivo é adicionado ou um commit é feito, suas somas de verificação são geradas. Da mesma forma, eles são recuperados através de suas somas de verificação também.
* Outra característica presente no GIT é sua área de teste ou índice. Na área de preparação, os desenvolvedores podem formatar commits e receber feedback ​​antes de aplicá-los.

O GIT é consideravelmente simples de usar. Para começar, você pode criar um repositório ou conferir um já existente. Após a instalação, um simples git-init irá deixar tudo pronto. Da mesma maneira, o comando git clone pode criar uma cópia de um repositório local para um usuário.

Resumo principais comando do Git via terminal (Linux)

Configuração Inicial

* **Setar usuário:** **git config --global user.name** "Fulano de Tal"
* **Setar email:** **git config --global user.email** fulanodetal@software-ltda.com.br
* **Setar editor:** **git config --global core.editor** vim
* **Setar ferramenta de merge:** **git config --global merge.tool** vimdiff
* **Setar arquivos a serem ignorados:** **git config --global core.excludesfile ~/.gitignore**
* **Listar configurações:** **git config –list**

Repositório Local

* **Criar novo repositório:** **git init**
* **Verificar estado dos arquivos/diretórios: git status**

Adicionar arquivo/diretório (staged area)

* **Adicionar um arquivo em específico:** **git add** meu\_arquivo.txt
* **Adicionar um diretório em específico:** **git add** meu\_diretorio
* **Adicionar todos os arquivos/diretórios:** **git add** .
* **Adicionar um arquivo que esta listado no .gitignore (geral ou do repositório):**

**git add -f** arquivo\_no\_gitignore.txt

Comitar arquivo/diretório:

* **Comitar um arquivo:** **git commit** meu\_arquivo.txt
* **Comitar vários arquivos:** **git commit** meu\_arquivo.txt meu\_outro\_arquivo.txt
* **Comitar informando mensagem:** **git commit** meuarquivo.txt -m "minha mensagem de commit"

Remover arquivo/diretório:

* **Remover arquivo:** **git rm** meu\_arquivo.txt
* **Remover diretório: git rm** -r diretório

Visualizar histórico

* **Exibir histórico:** **git log**
* **Exibir histórico com diff das duas últimas alterações: git log** -p -2
* **Exibir resumo do histórico (hash completa, autor, data, comentário e qtd alterações (+/-)): git log --stat**
* **Exibir informações resumidas em uma linha (hash completa e comentário):** **git log --pretty=oneline**
* **Exibir histórico com formatação específica (hash abreviada, autor, data e comentário):**

**git log --pretty=format:"%h - %an, %ar : %s"**

%h: Abreviação do hash;: %an: Nome do autor;

%ar: Data;

%s: Comentário.

Verifique as demais opções de formatação no Git Book

* **Exibir histório de um arquivo específico: git log** -- <caminho\_do\_arquivo>
* **Exibir histórico de um arquivo específico que contêm uma determinada palavra:**

**git log --summary -S**<palavra> [<caminho\_do\_arquivo>]

* **Exibir histórico modificação de um arquivo: git log --diff-filter=M** -- caminho\_do\_arquivo>

(O “M” pode ser substituido por: Adicionado (A), Copiado (C), Apagado (D), Modificado (M), Renomeado (R), entre outros.)

* **Exibir histório de um determinado autor:**  **git log --author=**usuario
* **Exibir revisão e autor da última modificação de um bloco de linhas: git blame -L** 12,22 meu\_arquivo.txt

Desfazendo operações

* **Desfazendo alteração local (working directory):** **git checkout** -- meu\_arquivo.txt

(Este comando deve ser utilizando enquanto o arquivo não foi adicionado na staged area.)

* **Desfazendo alteração local (staging area):** **git reset HEAD** meu\_arquivo.txt

(Este comando deve ser utilizando quando o arquivo já foi adicionado na staged area. Se o resultado abaixo for exibido, o comando reset não alterou o diretório de trabalho.)

* **Unstaged changes after reset:**M meu\_arquivo.txt
* **A alteração do diretório pode ser realizada através do comando:** **git checkout** meu\_arquivo.txt

Repositório Remoto

* **Exibir os repositórios remotos:** **git remote**

**git remote** -v (Vincular repositório local com um repositório remoto)

**git remote** add origin git@github.com:fulanodetal/curso-git.git

* **Exibir informações dos repositórios remotos:** **git remote show** origin
* **Renomear um repositório remoto:** **git remote rename** origin curso-git
* **Desvincular um repositório remoto:** **git remote rm** curso-git
* **Enviar arquivos/diretórios para o repositório remoto:** **git push -u** origin master

O primeiro push de um repositório deve conter o nome do repositório remoto e o branch.

Os demais pushes não precisam dessa informação: **git push**

Atualizar repositório local de acordo com o repositório remoto

* **Atualizar os arquivos no branch atual:** **git pull**
* **Buscar as alterações, mas não as aplicar no branch atual:** **git fetch**
* **Clonar um repositório remoto já existente:** **git clone** git@github.com:fulanodetal/curso-git.git

Tags

* **Criando uma tag leve:** **git tag** vs-1.1
* **Criando uma tag anotada:** **git tag -a** vs-1.1 -m "Minha versão 1.1"
* **Criando uma tag assinada:** **git tag -s** vs-1.1 -m "Minha tag assinada 1.1"

Para criar uma tag assinada é necessário uma chave privada (GNU Privacy Guard - GPG).

* **Criando tag a partir de um commit (hash):** **git tag -a** vs-1.2 9fceb02
* **Criando tags no repositório remoto:** **git push origin** vs-1.2
* **Criando todas as tags locais no repositório remoto:** **git push origin**–tags

Branches

O master é o branch principal do GIT.

O HEAD é um ponteiro especial que indica qual é o branch atual. Por padrão, o HEAD aponta para o branch principal, o master.

* **Criando um novo branch: git branch** bug-123
* **Trocando para um branch existente:** **git checkout** bug-123

Neste caso, o ponteiro principal HEAD esta apontando para o branch chamado bug-123.

* **Criar um novo branch e trocar:** **git checkout** -b bug-456
* **Voltar para o branch principal (master): git checkout** master
* **Resolver merge entre os branches:** **git merge** bug-123

Para realizar o merge, é necessário estar no branch que deverá receber as alterações. O merge pode automático ou manual. O merge automático será feito em arquivos textos que não sofreram alterações nas mesmas linhas, já o merge manual será feito em arquivos textos que sofreram alterações nas mesmas linhas.

* **A mensagem indicando um merge manual será:** Automerging meu\_arquivo.txt

CONFLICT (content): Merge conflict in meu\_arquivo.txt

* **Automatic merge failed;** fix conflicts and then commit the result.
* **Apagando um branch:** **git branch -d** bug-123

Listar branches

* **Listar branches:** **git branch**
* **Listar branches com informações dos últimos commits:** **git branch -v**
* **Listar branches que já foram fundidos (merged) com o master:** **git branch --merged**
* **Listar branches que não foram fundidos (merged) com o master: git branch --no-merged**

Criando branches no repositório remoto:

* **Criando um branch remoto com o mesmo nome:** **git push origin** bug-123
* **Criando um branch remoto com nome diferente: git push origin** bug-123:new-branch
* **Baixar um branch remoto para edição:** **git checkout -b** bug-123 origin/bug-123
* **Apagar branch remoto:** **git push** origin:bug-123