

É um sistema de controle de versões distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registar o histórico de edição de qualquer tipo de arquivo.

O Git permite que a gente armazene as modificações feitas em cada arquivo em um servidor próprio para isso. Toda a gestão de alterações é feita pelo Git e nós só precisamos nos preocupar em criar código que funciona, e não em quem alterou o que antes.

Cada alteração que você faz fica gravada em um histórico, podendo ser visualizada e restaurada a qualquer instante.

O que são (e para que servem) sistemas de controle de versões e como eles podem ajudar o nosso fluxo de desenvolvimento

* Nos ajudam a manter um histórico de alterações;
* Nos ajudam a ter controle sobre cada alteração no código;
* Nos ajudam para que uma alteração de determinada pessoa não influencie na alteração realizada por outra;
* Etc.

Outros controles de versões conhecidos são: CVS, SVN, Mercurial e o próprio Git.

**O que é repositório?**

* Um repositório contém todos os arquivos do seu projeto e o histórico de revisão de cada arquivo. Você pode discutir e gerenciar o trabalho do projeto dentro do repositório.

**O que é branch?**

* Branches ("ramos") são utilizados para desenvolver funcionalidades isoladas umas das outras. A branch master é a branch "padrão" quando você cria um repositório.
* É interessante separar o desenvolvimento de funcionalidades em branches diferentes, para que as mudanças no código para uma não influenciem no funcionamento de outra.

**COMANDOS**

**git init**

Cria/inicializa um repositório a partir do diretório selecionado, assim, todas as alterações que forem realizadas no arquivo localizado dentro deste repositório poderão ser mostradas pelo Git, como indicações do que foi modificado, quem modificou, quando...

**git init --bare**

Com este comando nós criamos um repositório que não terá a working tree, ou seja, não conterá uma cópia dos nossos arquivos. Como o repositório servirá apenas como servidor, para que outros membros da equipe sincronizem seus trabalhos, poupamos espaço de armazenamento desta forma.

**git config --local user.name "nome"**

Adiciona nome do usuário na configuração do git.

**git config --local user.email "e-mail"**

Adiciona e-mail do usuário na configuração do git.

**git --version**

Verifica a versão do git que está sendo utilizada em sua máquina.

**git status**

Mostra o estado/status do nosso repositório.

**git add <arquivo>**

Adiciona o arquivo para ser monitorado/incluído nos commits. Para adicionar todos os arquivos do diretório basta inserir “.” no lugar de <arquivo>.

**git commit -m "Mensagem"**

Commita as alterações realizadas, onde -m "" é a mensagem de commit que será inserida.

**git log**

**git log –oneline**

**git log -p**

Monstra os logs de todos os commits realizados naquele repositório (<https://devhints.io/git-log>).

**git remote add <origin> <endereço>**

**git remote rename <origin> <local>**

Adiciona um repositório onde serão armazenados os arquivos do projeto (origin = repositório/repositório antigo | local = repositório novo | endereço = git/diretório...).

**git clone <endereço>**

Clona o repositório na máquina (endereço = git/diretório...).

**git push <origin> <master>**

Envia para o servidor as alterações realizadas no repositório/branch (origin = repositório | master = branch).

**git pull <origin> <master>**

Pega as alterações do repositório/branch especifica (origin = repositório | master = branch).

**git branch <features>**

Adiciona uma nova branch (features = branch).

**git checkout <features>**

Muda para trabalhar no ramo(branch)/linha de desenvolvimento local (features = branch);

**git checkout -b <features>**

Cria uma nova branch e já muda o ramo de trabalho para essa branch criada (features = branch).

**git checkout -- <index.html>**

Remove a alteração realizada no arquivo desejado (index.html = arquivo).

**git checkout <18ef648>**

Volta o projeto para o estado que era nesse commit. Caso queira realizar alterações a partir desse commit será necessário criar uma nova branch. (Para voltar, basta ir para branch desejável) (18ef648 = hash do commit).

**git merge <features>**

Unifica as alterações (branch master e features) em um commit. O merge junta os trabalhos e gera um merge commit. (Para sair da tela de merge necessário apertar :x + enter) (features = branch).

**git rebase <features>**

O rebase aplica os commits de outra branch na branch atual (features = branch). O comando git rebase não gera um commit de merge, simplificando o nosso log.

**git reset HEAD <index.html>**

**git restore --staged <index.html>**

Reverte arquivo adicionado para commit (index.html = arquivo).

**git revert <cc1410e7860f88430cff9c8d6587f5c7e986d392>**

Criar uma commit revertendo o commit desejado (alterações serão desfeitas) (cc1410e7860f88430cff9c8d6587f5c7e986d392 = hash do commit).

**git stash**

Salva a alteração na "nuvem" para que possa pega-la depois.

**git stash list**

Lista todas as alterações salvas.

**git stash apply <0>**

. Aplica as modificações, porém as mesmas continuam em stash list (0 = número da stash) (obter o número através do comando stash list [stash@{0}]).

**git stash drop**

Remove a alteração salva na stash.

**git stash pop**

Realiza o merge resgatando as alterações salvas e remove a alteração (última) salva na stash.

**git diff <69db27f>..<cc1410e>**

Verifica a diferença entre os códigos do commit x e y. Somente o git diff mostra o que foi alterado (antes de realizar o git add) (69db27f/ cc1410e = hash do commit).

**git tag -a <v0.1.0> -m "mensagem"**

Adiciona no commit atual uma tag informando que nesse commit há uma nova release (v0.1.0 = versão/nome).

**git tag**

Apresenta todas as tags criadas.

**git push <origin> <v0.1.0>**

Envia a tag gerada para 'servidor' (origin = repositório | v0.1.0 = versão/nome).

**.gitignore**

Para ignorar um arquivo (o mesmo não aparecera ao utilizar o comando git add) basta colocar o nome do arquivo/pasta a ser ignorado dentro do arquivo .gitignore

**Ignorar arquivo:**

ide-config

**Ignorar pasta completa:**

Temp/