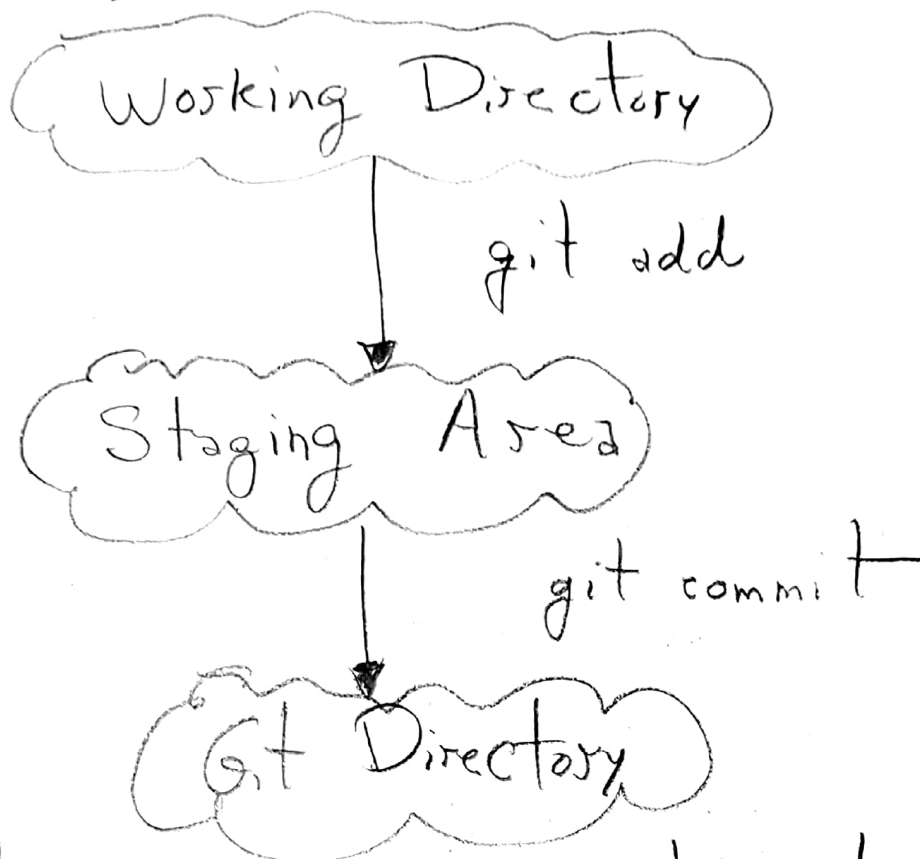


Git



`git init` → inicia um repositório git no diretório atual.

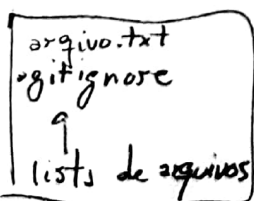
`git status` → status atual do repositório.

`git add` → "trackear" os arquivos. Obs: `git add.` adiciona todos os arquivos.

`git commit -m` "mensagem do commit"

`git commit -a` -m "mensagem do commit" mas precisa fazer o `git add`

Nota: Arquivos que não terão versões controladas devem ser adicionado no arquivo `.gitignore`



arquivo .gitignore

esse arquivo tb pode ser ignorado

arquivo oculto

`git diff` → mudanças nos arquivos no working directory
↳ não dá p ver no staging Area

git diff --staged → ver as mudanças nos arquivos na staged área

git log → mostra todos os commits, (ordem dos mais recentes p/ os mais antigos)
↳ note que há uma chave.

git log -p → todos os commits com as mudanças

git log -p -2 → últimos dois commits.

Nota: gitk → interface com relatórios. Poderá ser instalado novos programas de visualização.

git log --pretty=oneline → mostra apenas as chaves

git commit --amend -m "Nova mensagem"

↳ altera o commit anterior. Poderia antes fazer "git add ." para colocar novas funcionalidades p/ acrescentar no commit anterior realizado por engano. Note que a chave do commit é mudada.

git reset HEAD arquivo.c → remove um arquivo da staging área.

git checkout -- arquivo.c → recupera o arquivo.c da forma do commit anterior.

git rm arquivo.c → remove um arquivo. Ou seja, reverte ao estado do último commit.

Obs: Usar tags ajuda a reverter o status do sistema à uma data e hora associada à tag.

git tag → lista as tags criadas. Nota: tags são ponteiros para o sistema.

git tag -a v1.0 -m "versão 1.0" nome da tag mensagem
↳ cria uma tag anotada, com informações do usuário, data e hora, etc.

git tag -a v0.0 chave -m "versão 0.0" → mudando ou add uma tag a commits anteriores. ②

git show v.0.0 → saber detalhes a respeito de um ^{nome da tag} tag.

git checkout v.0.0 → volta o sistema ao status da tag v.0.0

git checkout ^{branch padrão} master → volta para o último commit.

git tag -d v.0.0 → remove a tag v.0.0

git branch teste → criando um branch de nome "teste".

git checkout teste → mudando para o branch de nome "teste".

git checkout -b teste → cria o branch de nome "teste" e faz a troca de ambiente.

git merge teste → estando no branch master, estão trazendo e mesclando os arquivos do branch teste.

git branch -d teste → removendo o branch de nome teste.

↳ alterações na mesma linha em branches distintos irá gerar conflitos. Na hora de fazer o merge, ocorrerá um problema (o merge nao será feito!). Daí, deverá ir nos arquivos com conflitos e escolher o que irá ficar. No arquivo terá opções comentadas. Delete a que nao quiser considerar!

git init --bare → cria o versionamento permitindo o compartilhamento em rede local.

↳ no servidor. (fazer no servidor!)

git clone diretório do servidor (todo o caminho) novo nome

nome do projeto clonado → nome na máquina local.
nome na máquina local.

git remote → diz o nome do servidor. Note que
isso é possível uma vez que estamos
trabalhando em um clone. (projeto
clonado do servidor remoto). O padrão
é "origin".

git push origin master → enviando os arquivos
para o servidor origin que
git pull origin master está no branch master.

↳ pega as atualizações do servidor que
porventura foram feitas depois
do clone e coloca no branch master.
Se fizer o clone irei perde mudanças
que fiz no clone localmente!

O pull faz um merge mas pode
não ser legal. O melhor seria fazer
atualizações para um branch diferente
do master. Para isso, fazer:

git fetch origin teste

ssh-keygen → gerando uma chave de
acesso p/ permitir que a má-
quina local venha interagir com
o github. ↳ branch de
nome "teste" na
máquina local.

git reset HEAD~1 --hard → desfazendo o último
commit e as alterações nos arquivos.

git reset HEAD~1 --soft → desfazendo o último
commit. As alterações nos arquivos não serão
perdidas.

↳ ~2 faz o mesmo penúltimo
commit e assim por diante: ~3,
~4, ...

git push --delete origin tagname → remove a tag remotamente.

No github, fazer um fork p/ e' fazer um clone de projeto que ^{queremos} colaborar. Esse clone é feito no ambiente remoto do github. Depois, para se ter na máquina local, basta clonar com git clone.

Com os arquivos na máquina, basta fazer as alterações e ao finalizar, enviamos ao fork do projeto em nossa conta. Depois, no github, devemos ir em "New pull request" e clicar para enviar ao autor original do projeto.

git log --reflog → lista todos os commits realizados no projeto mesmo estando em um commit anterior acessado através do comando git checkout "chave". Tal comando serve p/ obter as chaves dos commits mais recentes do que estamos.

git log --pretty=fuller → mostra mais detalhes no commit como, por exemplo, data, nome do commit e autor do commit.

Nota: Ao criar um novo branch, basta mudar p/ ele utilizando git checkout. Para levar esse novo branch ao Github faça: git push origin "nome do novo branch".

(5)

Já p/ deletar o branch no GitHub, faça:
`git push --delete origin "nome do branch"`

Nota. Para acrescentar mudança em um branch no GitHub basta trabalhar normalmente no diretório do computador (diretório do projeto) e fazer um commit. Depois basta fazer:

`git push origin "nome do branch"`

↳ mesmo que o branch já exista, o comando acima enviará o diretório "commitado" com as novas mudanças.

Clonando um projeto e indo p/ uma tag específica de um branch específico:

↳ `git clone <diretório>`
`git checkout <tag name>`

ou
`git checkout <tag name> -b <branch name>`

`git branch -a` → lista todos os branches

`git clone <url> -b <branch name>` → iniciando em um branch específico.

`git tag -l` → lista todas as tags do projeto.

git log -p -2 → exibe o histórico com diff das duas últimas alterações.

git remote -v → exibe os repositórios remotos

git remote add origin git@github.com:prdmo/repo
↳ vincula um repositório local com um repositório remoto. O repositório remoto deverá ter sido criado no Github.

git pull → atualiza o branch atual

git fetch → busca pelas alterações, mas não aplica no branch atual

git push origin nome-tag → criando uma tag no repositório remoto

git push origin --tags → criando todas tags locais no repositório remoto.

git help comando → acessa a documentação de um comando.

git config --list → lista as configurações do git.

Obs: O HEAD é um ponteiro especial que indica o branch atual. Por padrão, HEAD aponta para o branch principal (o master).

Atualizando o master de um fork:

1. git remote add upstream "link do projeto oficial"
2. git fetch upstream
3. git merge upstream/master master sem aspas

Suponha que tenhamos atualizado o master de um projeto que é um fork de um projeto oficial. Depois nos arrependemos das mudanças e desejamos voltar o "master" para o estado oficial do projeto oficial, isto é, desejamos resetar nossas alterações. Para isso, faça:

- 1 - git remote add upstream "link do master oficial"
↳ sem aspas
- 2 - git fetch upstream
- 3 - git checkout master
- 4 - git reset --hard upstream/master
- 5 - git push origin master --force

Dica: Toda vez que for propor um pull request, para evitar poluição de commits, em cada nova proposta, crie um novo branch com as mudanças e ao fim, faça um pull request desse branch para o branch master do projeto oficial.

em uma nova proposta, se já houve um merge dessa proposta antiga, comece em um novo branch e crie as novas funcionalidades.