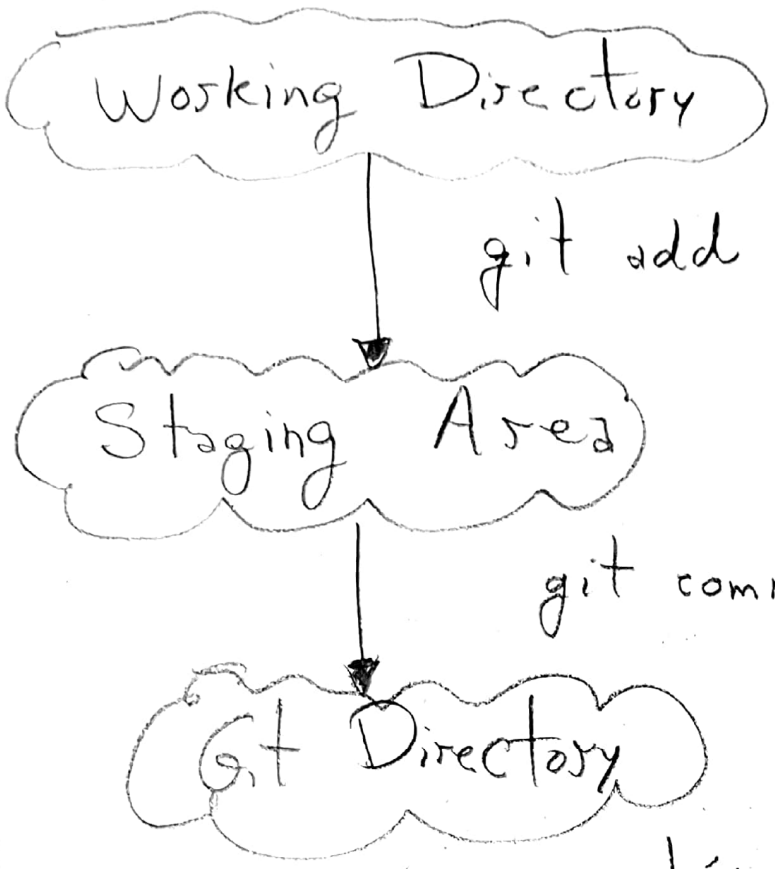
# Git




git init → inicia um repositório git no diretório atual.

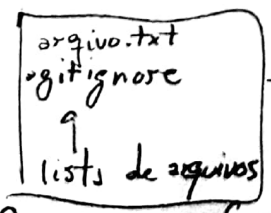
git status → status atual do repositório.

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"
↳ não precisa fazer o git add

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




→ arquivo .gitignore
↳ arquivo oculto

esse arquivo tb pode ser ignorado

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



Scanned by CamScanner

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. Ou seja, reverte ao padrão do último commit

git rm arqivo.c → remove um arquivo.

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.



Scanned by CamScanner

git show v0.0 → saber detalhes a respeito de uma tag.
(o nome da tag)

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

git checkout master → (branch padrão) 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 branche de nome "teste" e faz a troca de ambiente.

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

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

↳ alterações na mesma linha em branchs distintos irá gerar conflitos. Na hora de fazer o merge, ocorrerá um problema (o merge não será feito!). Daí, deverá ir nos arquivos com conflitos e escolher o que irá ficar. No arquivo terá opções comentadas. Delete o que não 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 p/ máquina local ← nome na máquina local.



Scanned by CamScanner

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 estão no branch master.

git pull origin master
↳ pega as atualizações do servidor que por ventura 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 → atualizaçao para um branch diferente do master. Para isso, fazer:
git fetch oring teste
↳ branche de nome "teste" na máquina local.

ssh-keygen → gerando uma chave de acesso p/ permitir que a máquina local venha interagir com o github.

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, ...



Scanned by CamScanner

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

No github, fazer um fork é 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 e ao finalizar, enviamos ao fork do projeto em nossa conta. Depois, no github, devemos ir em "New pull request" e clicar por 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".



Scanned by CamScanner

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 branchs

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

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



Scanned by CamScanner