Git

Sobre - ferramenta/software para versionamento de código } confiável, eficiente
Proporciona trabalho em equipe.

Instalação -

No Linux basta utilizar o comando no Terminal - sudo apt-get install git

---

Configurando usuário e email:

git config --global user.name "nome_do_usuário"
git config --global user.email "email_normalmente_do_github"

Criando um novo Repositório:

1° passo - crie uma nova pasta ou na pasta de seu projeto execute o comando:

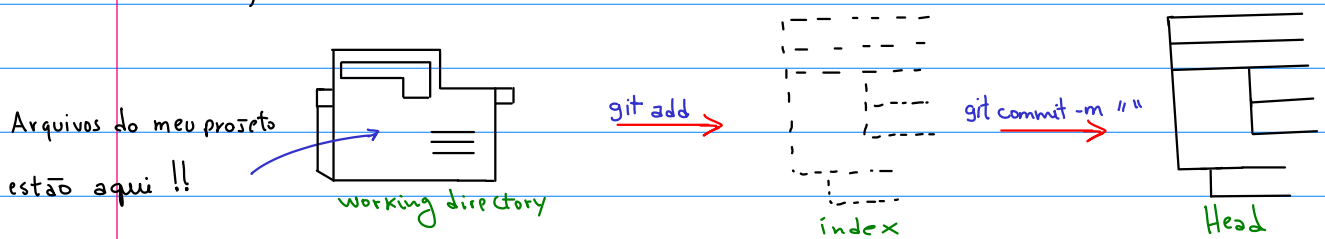
----- mkdir projeto
-> make directory (Faça diretório)

comando → git init } inicializa / cria um Repositório.

2° passo - execute o comando

comando → git status
* indica em qual repositório estou
* indica os untracked files (arquivos não monitorados/rastreados, arquivos que o Git ainda "não conhece". Vou precisar fazer um git add (explicação em breve).
arquivos não versionados

O fluxo de trabalho do Git:



Os repositórios locais consistem em 3 "árvores" mantidas pelo git. A primeira dela é sua Working directory (diretório de trabalho) que contém seus arquivos vigentes. A segunda é o Index (índice) que funciona como uma área temporária. Por fim temos a Head que aponta para o último commit que você fez.

3° Passo - vou adicionar meus arquivos que estavam untracked para o (index) usando:

comando → git add <arquivo>

ou

git add . } adiciona todos os arquivos

ou

git add *.c } adiciona todos os arquivos .c

pode ser qualquer formato.

4° passo - agora que meus arquivos estão no Index eles podem ser "comitados", isso porque somente os arquivos no Index podem ser Empacotados em um novo commit (arquivos untracked não entram nos commits por definição).

posso remover arquivos do Index com o comando. → git rm --cached <arquivo>

5º passo - realizar o commit:

comando → git commit -m "mensagem"

Agora os arquivos Empacotados por meio de commit são enviados para o Head, mas não ainda para um repositório Remoto, como no Github por exemplo.

Em Resumo:

init - cria repositório.

status - permite avaliarmos o estado dos arquivos no diretório de trabalho e no índice.

add - adiciona ao índice.

rm --cached - retira arquivo novo do índice.

reset HEAD - retira o arquivo que foi modificado do índice.

commit - vai gravar o conteúdo presente no índice e coloca no repositório /Head.

log - mostra o histórico de commits no repositório

Emendando commits:

Já fiz commit, mas acabei esquecendo um detalhe que preciso corrigir ou até mesmo um arquivo que precisava estar naquele commit empacotado. É evidente que quero corrigir porém, não quero fazer outro commit. O que fazer ???

1º corrija seu arquivo, se necessário
2º git add <arquivo>
3º git commit -m "messagem" --amend

O amend "emenda" os commits, ficando no mesmo rastro. Unindo em um único commit.
↳ faça somente antes de fazer um push, isto é, enviar para algum repositório REMOTO.

Andando na linha do tempo do nosso repositório:
↳ voltando commits

Vamos supor que eu "comitei" determinado arquivo e por algum motivo eu me arrependi e quero jogar isso fora. Como eu faço ?

comando → git reset HEAD~1 --soft
↳ níveis que eu quero voltar.

ou

comando → git reset HEAD~1 --hard

Retirando arquivos do índice:

comando → git reset HEAD <arquivo>

Adicionando ao índice e realizando "commits":

comando → git commit -a -m "mensagem"

Removendo arquivos que Não estão no índice:

Para remover arquivos untracked, isto é, que não estão no índice, basta executar:

comando → git clean -f

Trabalhando com branches

Os branches ("ramos") são utilizados para desenvolver funcionalidades isoladas uma das outras ou corrigir bugs. O branche master é o branche padrão quando você cria um repositório. Já os outros branches serão usados para desenvolver, após isso vamos mesclar esses branches, por meio do merge, ao branche master.

O comando para criar um branch é:

git branch nome_do_branch
ou
git checkout -b nome_do_branch.

O comando:

git branch mostra em qual branch estamos.

O comando:

git checkout nome_do_branch muda de branch.
master retorna para o master

Mesclando - Merge

Assim, para fazer um merge de outro branch ao seu branch ativo (Ex: master), use:

comando → git merge <branch>

OBS: o git sempre tenta fazer o merge das alterações automaticamente. Porém, isso nem sempre é possível e acaba resultando em conflitos. Nesse caso, somos responsáveis por resolver esses conflitos manualmente.

Para remover um determinado branch, execute o comando:

git brand -d <branch>
-D

OBS: vc precisa estar fora deste branch

Em resumo:

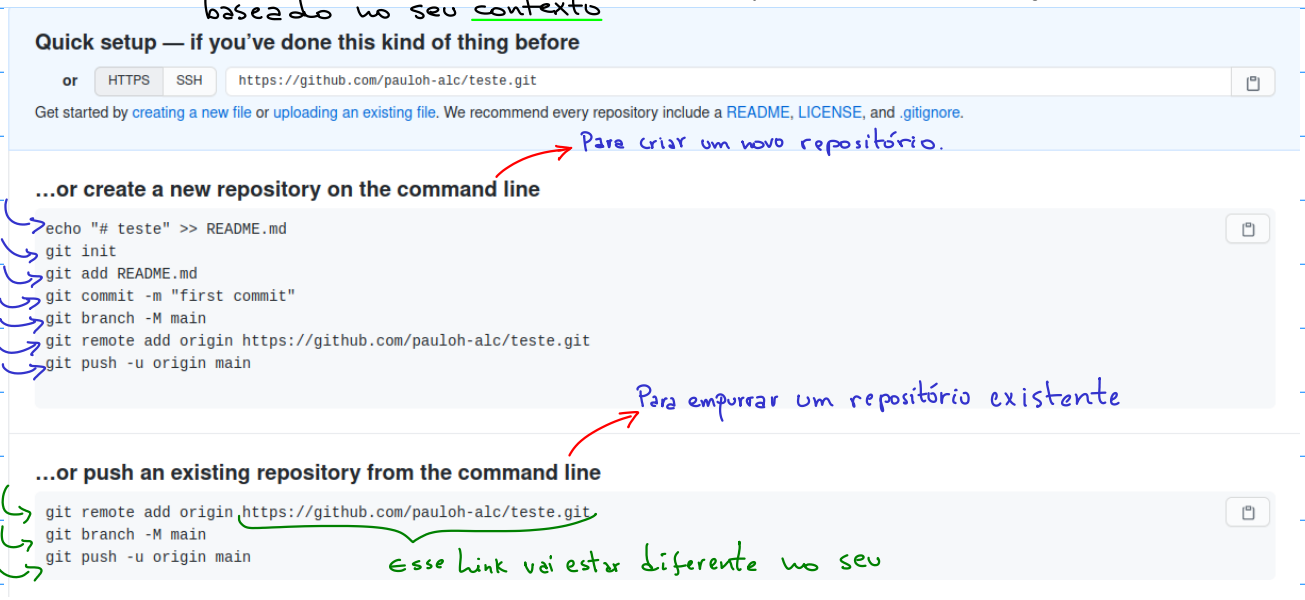
branch - lista os branches criados e mostra seu brand atual.
branch -d <nome> - vai apagar branch.
checkout -b <nome> - cria um novo branch a partir do branch atual.
merge <branch> - vai mesclar no branch atual a partir de outro branch.
rebase <branch> - desfaz commits atuais, traz os commits do outro branch e reaplica os commits.

▨ Criando um repositório remoto no GitHub

Rede Social < Une Repositório Remoto ⊕ Rede Social

1º passo - vá na opção New repository
2º passo - escolha um nome para seu repositório
3º Passo - você pode colocar uma descrição no seu repositório. (opcional)
4º Passo - seu repositório pode ser público ou privado.
5º Passo - você pode inicializar seu repositório com: README file, .gitignore, ou até escolher um licença. (opcional)

Exemplo: após os passos explicados na página anterior, digite os comandos baseado no seu contexto



Enviando Nossas alterações para repositório Remoto:

Nossas alterações agora estão no Head da sua cópia de trabalho local. Para enviar estas alterações ao Repositório Remoto, execute:

comando → git push <repositório_remoto> <nome_branch>

Ex: git push origin master

Para atualizar o repositório local com a mais nova versão, execute:

comando → git pull <repositório_remoto> <nome_branch>

Obtendo um repositório:

comando → git clone <caminho_do_repositorio>

☑ Criando um novo branch para voltar no tempo por meio do Id de um commit:

comando → git checkout -b <nome_do_branch> <id_do_commit>

Ex: git checkout -b inicio 95db45987...

Id do 1º commit do projeto.

☑ Criando uma tag apontando ao commit do branch atual:

↳ interessante para criar versões do projeto.

comando → git tag <nome> Ex: git tag V0.1

comando → git push --tags Envia todas as tags ao repositório remoto.

comando → git push <tag> Envia apenas a tag específica ao repositório remoto.

✳ Comandos do GIT - Resumo:

Para criar um novo repositório local: git init
Para adicionar arquivos ao índice: git add . OU git add ~nome do arquivo~
Para verificar / checar status do índice: git status
Para realizar um commit das novas mudanças: git commit -m "mensagem"
Para criar um novo branch: git checkout -b ~nome do branch~
Para mudar para um branch: git checkout ~nome do branch~
Para mesclar branches: git merge ~nome do branch~
Para adicionar um repositório remoto: git remote add ~nome do repo remoto~ ~https://sua_url_do_repo_remo
Para puxar mudanças do repositório remoto: git pull ~repo remoto~ ~nome do branch~
Para empurrar mudanças para repositório remoto: git push ~repo remoto~ ~nome do branch~

Por Paulo Henrique Diniz de Lima Alencar.

Referências :

BRILHANTE, H. Iniciando com Git por Fabio Akita. Disponível em: <https://www.youtube.com/watch?v=jdinE9SkUnE&t=3s>. Acesso em: 27 dez. de 2020.

DUDLER, R. git - guia prático. Disponível em: <https://rogerdudler.github.io/git-guide/index.pt_BR.html>. Acesso em: 29 dez. de 2020.