repositorio: projeto dentro do git

**Versão do git no computador**

git --version

**Definindo o nome do nosso usuario no git**

git config --global user.name "William Rolim"

**Definindo email no git hub**

git config --global user.email "williamrolimads@gmail.com"

**Visualizar o nome cadastrado**

git config --global user.name

**Visualizar o email cadastrado**

git config --global user.email

**Qual editor que estou usando**

git config --global core.editor sub (completa a sentença utilizando o compilador ao qual trabalha)

**Visualizar todas as informações cadastradas no git**

git config --list

**Inicializando o repositorio**

git init

**Branch, README e Commit**

Branch: Versões diferentes do meu sistema

Master - versão principal do meu sistema

**Commit: Fazer alterção, criar arquivos novos, modificando arquivos**

dica comentar as alterações do commit (breve e explicativo). Ex: alterado login que mexe com cookie

**Untraked** : Não está monitorando os arquivos, precisa adicionar ao monitoramento do git

**adicionando arquivos (Um por um)**

git add (nome do arquivo)

**adicionando Todos os arquivos que foram modificados e inseridos (untraked = monitorados)**

git add -A

**Fazendo alterção no arquivo e enviando via commmit**

git commit -m "Primeiro Commit"(aspas é exemplo)

**Adicionando todos os arquivos modificados e alteração no arquivo enviando via commit**

git commit -am "Digite aqui o que foi mudado"

**Historico das coisas que foram feitas(LogDeCommits)**

Lista de todos os commits (alterações do sistema) que foram feitas naquele branch(versões diferentes do sistema)

git -log

**Mostrar status das mudanças que foram feitas (alteração, criação exclusão)**

git status

obs: enquanto eu não der commit os arquivos só estarão disponiveis no meu computador.

**Saber qual Branch que estamos( (HEAD -> master)**

git branch

**Revertendo modificações para outra data.**

git reset bb00c9210489d733b3e01d11be2b28759e4015bd (copiar o codigo inteiro ou apenas os 7 primeiros digitos)

**get reset**

O Mixed (Default) reseta inclusive os arquivos, mas mantém a versão atual para você poder voltar para ela se quiser.

**Git reset -solid**

volta para as atualizações do primeiro, porém adiciona o anterior tbém nesse mesmo arquivo não comitadas

**get reset --soft**

Realiza as alerações no git porém, os arquivos permanecem na sua maquina local (localhost) sem realizar alterações

**Git reset -hard**

todas as atualizações são descartadas deixando somente o arquivo original

**Criando um novo branch**

**Git branch (escreve o nome do arquivo – sem parênteses) ex teste**

\*máster (versão principal que está no ar)

Teste (outra versão, teste com nova funcionalidade antes de passar para o branch principal)

C:\wamp\www\modulogit>git branch

\* máster (\* significa qual branch eu estou)

teste

**Trocando de Branch**

git checkout (nome do arquivo criado – sem parênteses) ex teste

C:\wamp\www\modulogit>git branch

master

\* teste **estou no git teste agora**

**Vizualizar o que realmente foi mudado em cada arquivo (todos os arquivos)**

**Git diff**

**Vizualizar somente o nome dos arquivos que foram modificados**

**Git diff --name-only**

**Vizualizar o que realmente foi mudado em um unico arquivo**

**Git diff (“digite o nome do arquivo “)**

**Cancelar a modificação de um arquivo**

**Git checkout HEAD – (espaço e nome do arquivo)**

**Enviando de local para remoto**

Generating a new SSH key and adding it to the ssh-agent

[**https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/**](https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/)

ssh-keygen -t rsa -b 4096 -C "*your\_email@example.com*"

***Local que o git salva os arquivos***

(/c/Users/willi/.ssh/id\_rsa):

Senha powerback

Fetch capacidade que eu tenho de puxar meu conteúdo no conteúdo remoto para meu local.

Puch é levar as coisas do meu repositório local para o repositório remoto

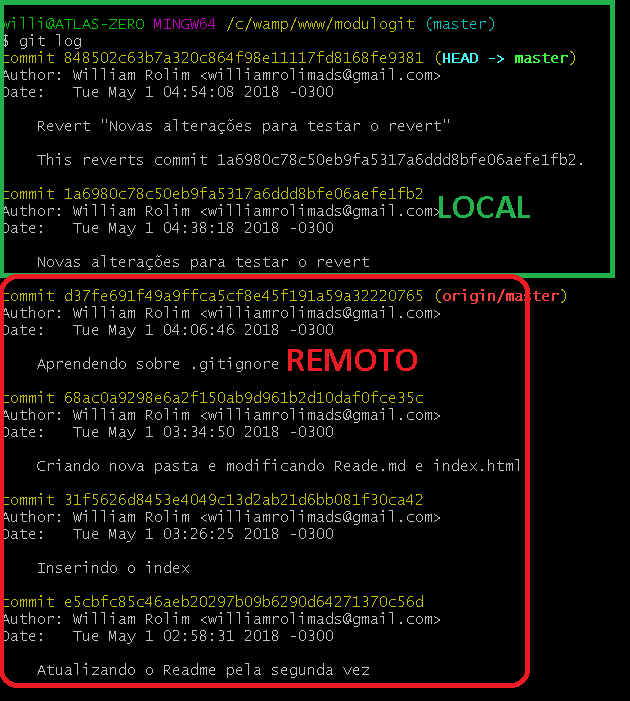
origin https://github.com/WilliamRolim/modulogit.git (fetch)

Comando PUSH (ENVIAR LOCAL PARA REPOSITORIO ONLINE)

Enviar online o arquivo

git push -u origin master

**GIT LOG**

****

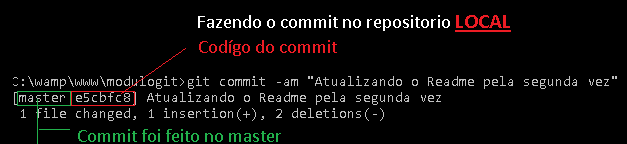
**Fazendo Alterações no repositório remoto**

Commit: Toda vez que enviar um bloco de alterações (fazer alterações, faço um commit)

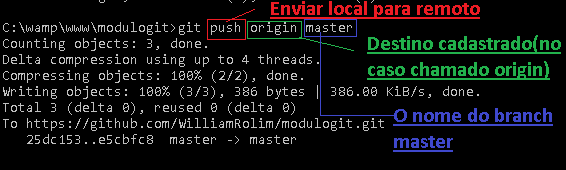
-a (adicionar todos os arquivos que foram alterados e modificados)

M (para adicionar uma mensagem) exemplo o que foi alterado

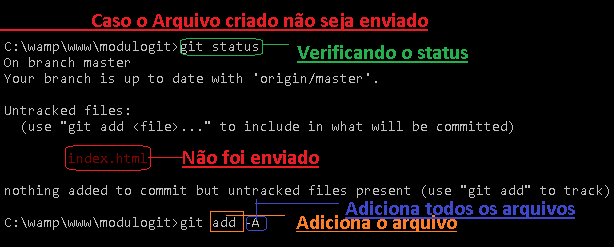
Lembrando: git commit –am



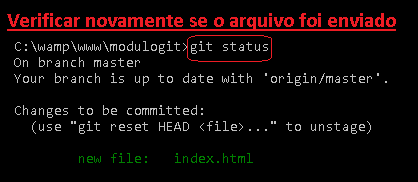
**Enviando os arquivos de local para remoto:**

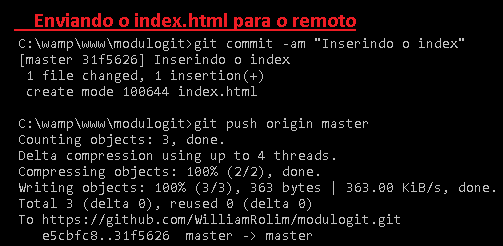
****

**Caso o arquivo não for enviado**

****

**Verificar novamente se o arquivo foi enviado**

****

**Enviando index.html (arquivo criado) ----Repeat: Reforçando aprendizado**

## Ignorando arquivos do repositório (gitignore)

## 

## Revertendo sem perder o código (Revert)

## Obs: git reset (vc perde todas as alterações que você fez).

## Revert: retorna ao estado original preservando seu commit (códigos que foram feitos) para ser alterado outra hora utilizando o código.

## Voltando o commit sem perder o código

## 

## Deletando branches locais e remotos

## Remover branches remoto

## 

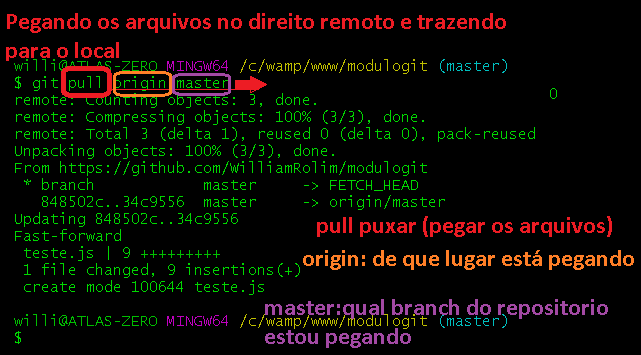
**Remover Branch Localmente**

****

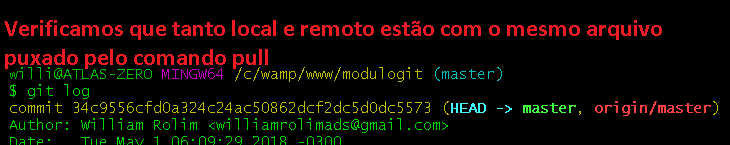
**Dica senha:**

 git config credential.helper store

## Puxando alterações de outras pessoas (pull)

****

Obs: Assim que efetuar o comando pull (puxar os arquivos remoto) ele irá aparecer no seu projeto.



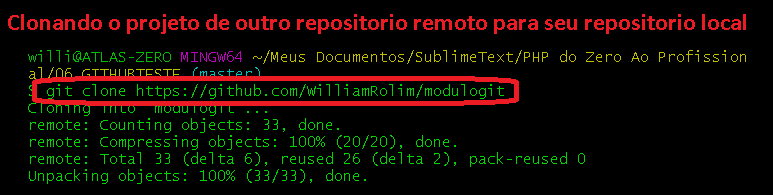
Dica mercado de trabalho: sempre que for dar um push você da um pull ou seja, vc pega as atualizações remotamente , junta com a local. Dessa forma o sistema fique atualizado

## Clonando repositórios remotos

## Criar uma copia fidedigna de repositório de terceiros no meu repositório local para alterações

Copiar o link:

Exemplo: <https://github.com/WilliamRolim/modulogit>



## Contribuindo com outros repositórios (fork / pull request)

## Aprenderemos a contribuir com outros projetos que não são os nossos.

## 