Introdução ao Git e ao GitHub

Download e Instalação no Linux

> **Download do GIT**
>
> https://git-scm.com/

Instalação do GIT no Linux

> **apt-get install git**
>
> Para instalar no linux

> **git --version**
>
> Obtém a versão instalada

Comandos básicos para um bom desempenho no terminal (Windows e Linux)

> **dir**
>
> Lista os diretórios de onde está situado - ls no linux

> **cd**
>
> Possibilita a navegação nas pastas - cd no linux

> **cd **
>
> Leva para o diretório c: do windows

> **cd ..**
>
> Volta um nivel de pastas - No linux tambem

> **cls**
>
> Limpa o terminal - No linux é o comando clear ou ctrl+l

> **mkdir nomepasta**
>
> cria uma pasta - No linux é igual o comando

> **echo hello > hello.txt**
>
> Pega o valor hello e cria um arquivo txt através do > que é um redirecionador de fluxo - No linux tambem funciona

> **del nomearquivo**
>
> remove arquivos

> **rmdir nomediretorio**
>
> Deleta o diretório - no linux rm - rf nome diretório onde -rf é para deletar recursivamente e forçada outras pastas contidadas dentro da mesma

Tópicos fundamentais para entender o funcionamento do Git

> **SHA**
>
> É um algoritmo de encriptação. Gera um conjunto de caracteres de 40 dígitos unicos.
> Após a instalação do Git .
> Para abrir um terminal do Git Bash basta com o botão direito em qualquer pasta que deseja clicar na opção " GIT Bash Here " Assim não é necessário navegar até um diretório manualmente.

> **No GIT Bash**
>
> openssl sha1 arquivo.txt
> Ele irá encriptar o arquivo em sha1 (Uma chave com 40 caracteres)
> cada mudança no arquivo gera uma chave diferente, é isso que permite o git identificar que houve uma mudança no arquivo

Objetos internos do Git

> **Blobs**
>
> É o tipo do objeto que o git armazena as informações/metadados. Eles contem o tipo, tamanho, \0 e o conteúdo do arquivo mas na guarda o nome
> exemplo: echo -e 'blob 9\0conteudo' | openssl sha1

> **Tree**
>
> Armazena blobs e aponta para outros blobs e commits, ela tambem contem metadados, ela guarda o nome dos arquivos ja o blob não.
> A arvore monta toda estrutura onde esta os arquivos, apontando para outras arvores ou blobs, parecido como no S.O diretórios apontam para outros diretórios

> **Commit**
>
> É o objeto que junta tudo.
> Aponta para uma arvore ou commit realizado antes dele(parent), um autor, uma mensagem, contem um timestamp e um sha1
> (Tree, parente, autor, mensagem, timestamp) sendo o SHA1 deste comit o hash de toda essa informação. Ele é um sistema distribuído e seguro pois permite que varias pessoas contribuam em um script e ele garante que as versões seja sempre atualizadas e seguras

Iniciando o Git e criando um commit

> **git init**
>
> Inicia o repositorio do git

> **git add**
>
> para mover arquivos e começar o projeto em si ao versionamento

> **git commit**
>
> para commitar a versão

Quando é usado a primeira vez o git é necessario criar um usuario

> **git config --global user.email "email@gmail.com"**
>
> **git config --global user.name seunome**

No GIT BASH

- ir na pasta desejada e com botão direito apertar GIT "Bash Here"

> **mkdir livro-receitas**
>
> Para criar uma nova pasta dentro

> **git init**
>
> Para inicializar este repositorio

> **ls -a**
>
> Lista os arquivos e pastas ocultas do GIT Bash

> **git add ***
>
> Adiciona todos os arquivos que precisam ser commitados

> **git add arquivo.txt pasta/**
>
> **git commit -m "mensagem commit inicial"**
>
> Commita os arquivos
>
> **echo > README.md**
>
> Cria o arquivo README

Passo a passo no ciclo de vida

> **git init**
>
> inicializa um repositorio

> **git status**
>
> Serve para monitorar o status dos arquivos para saber em qual situação os arquivos estão (unmodified, modified, staged)

> **mv nomearquivo.txt ./diretoriodestino**
>
> Move um arquivo

- Arquivos Untracked: Arquivos que o git NÃO TEM conhecimento dele
- Arquivos Tracked: Aquivos que o git TEM conhecimento dele, possui 3 tipos (unmodified, modified, staged) o stage são os arquivos que estão sendo preparados para um outro tipo de agrupamento no caso um commit.
- Repositório remoto(GIT HUB) so recebe arquivos que estão commitados no repositorio local

Trabalhando com o GitHub

> **git config --list**
>
> Mostra suas configurações de GIT, como usuario, nickname

> **git config --global --unset user.email**
>
> Caso deseje mudar o email associado, este comando irar zerar o usuario. para acresentar novamente git config --global user.email "email@gmail.com"

> **git config --global --unset user.name**
>
> Caso deseje mudar o nome de usuario associado, este comando irar zerar o usuario. para acresentar novamente
> git config --global user.name denis

> **git remote -v**
>
> Lista os repositorios remotos que eu tenho

> **git remote add origin https://github.com/denisvr46/livro-receitas.git**
>
> Para apontar para um repositorio remoto e enviar do repositorio local para lá. Esta URL é fornecida pelo GITHUB ao criar um repositório em:
> ...or push an existing repository from the command line
> git remote add origin https://github.com/USUARIO/livro-receitas.git
> origin é apenas um alias, por convenção

> **git push origin master**
>
> Ele envia a brench para o repositorio

Como os conflitos acontecem no GitHub e como resolve-los

> **git pull origin master**
>
> puxa o arquivo que esta no repositório remoto github
> *origin é apenas um alias, por convenção

> **git clone https://github.com/python/cpython.git**
>
> Copia um repositorio

Observações

- O botão TAB tem a função de auto completar