GIT

Git é um software command-line interface (CLI) diferente dos softwares responsivos com o qual lidamos normalmente os Graphical User Interface (GUI).

**Comandos**

Windows

- cd: navega entre as pastas cd nome da pasta, para sair da pasta use cd ..

- dir: lista os diretórios na pasta selecionada

- mkdir: cria uma pasta make directory

- del / rmdir: del deleta apenas arquivos já o rmdir remove pastas

- cls: limpa a tela “clear screen”

- echo: printa no terminal o que você escrever

> redireciona o fluxo para um arquivo Ex. echo hello > hello. txt isso cria um arquivo txt contendo o hello

Como o GIT funciona por debaixo dos panos

SHA1

É um algoritmo de encriptação (Secure Hash Algorithm) que gera um conjunto de caracteres de 40 dígitos

Caso seja feita uma alteração no arquivo será gerado uma **nova hash** de 40 dígitos caso não haja alteração no arquivo será gerado a **mesma hash** de 40 dígitos.

Objetos Internos do GIT

**Blobs** (Bloco básico da composição)

Onde os arquivos ficam guardados (metadados)

Estrutura básica do Blob

Blob (tipo do objeto) 🡪 Tamanho do arquivo 🡪 \0 🡪 conteúdo

**Tree** (Armazena e aponta os tipos de blobs e commits)

Armazenam Blobs e seus nomes

É como os diretórios formados e os blobs são os conteúdos. Alterar o blob também modifica o hash da Tree

**Commits**

São os metadados formados na entrega/subida do arquivo/dados contendo tempo, mensagens, autor... alterações no blob também gerarão alterações na Tree e por consequência alterações no hash do commit.

**Segurança**

É por dessa estrutura que o GIT é tão seguro, qualquer alteração será registrada em todo o processo tornando uma alteração maliciosa praticamente impossível de não ser percebida.

Chave SSH e Tokens

**Chave SSH (Secure Socket Shell)**

Forma de estabelecer uma conexão segura e criptografada

Criando uma chave no BIT Bash

$ ssh-keygen -t ed25519 -C bruno\_works@outlook.com

Senha 210198

Pegando a chave criada (.pub pública)

cat id\_ed25519.pub

Inicializar o ssh\_agent que fica encarregado de pegar as chaves e lidar com elas

$ eval $(ssh-agent -s)

Passando a chave privada para o ssh\_agent

$ ssh-add id\_ed25519

$ git clone endereço

**Token de Acesso Pessoal**

Vai gerar um token e sempre que for fazer o commit na hora de colocar a senha será utilizado esse token

Iniciando o GIT e criando um Commit

Git init: inicia o repositório

Git add: mover arquivos

Git commit: criar commit

Ls -a(flag): mostra arquivos ocultos

Ciclo da Vida dos Arquivos no Git

**Tracked**

Arquivos os quais o Git tem ciência, são divididos em;

Unmodified – Arquivos não modificados

Modified – Arquivo Unmodified que sofreu alteração

Staged – Estado de espera de ação que fica entre o Modified e o Commit

**Untracked**

Arquivos os quais o Git não tem ciência.

Repositório

Servidor – Remote Repository

Ambiente de Desenvolvimento – Working Directory – Staging Area – Local Repository

Git status – mostra o estado dos aquivos

Mv arquivo ./nova pasta – move um arquivo para outro lugar

Git add \* - com o asterisco ele inclui todas as modificações

git config --global --unset user.name ou user.name

Ordem para o push

Git add .🡪 git commit -m “mensagem” 🡪 git push origin master/main(o nome da branche principal)

Resolvendo Conflitos

Conflito de Merge – Quando o GitHub tenta juntar duas alterações com alterações da mesma linha

Git pull origin master – puxa o que está no repositório remoto para o local para que sejam feitas as alterações

Git remote -v - traz para onde os repositórios remotos estão apontados

Git clone url do repositório que deseja clonar