Comandos:

Configurando Git:

- git config --global --unset-all user.email
- git config --global --unset-all user.name
- git config --global user.name "Seu Nome"
- git config --global user.email "seu@email.com"
- git remote add origin https://github.com/SEU_USUARIO/meu-projeto.git

Basicos:

- **git init** // Inicializa o repositório em uma pasta
- **git status** // Mostra se existem projetos na pasta e status caso existam projetos
- **git add <nomeDoArquivo>** // Git começa a monitorar esse novo arquivo
    - **git add .** // Monitora todos os aquivos
- **git commit – m "mensagem"** // Faz o commit com as alterações que foram feitas no projeto podendo ser adicionado uma mensagem, o commit é como um checkpoint
- **git log** // Mostra as alterações que foram feitas no projeto (commits) com o código de cada commit
- **git revert <hash do commit ou HEAD> --no-edit** // Cria um novo commit revertendo as alterações feitas no commit especificado pelo hash
- **git reset <hash do commit>** // Vai VOLTAR para o commit especificado no hash, ignorando todos os commits para frente (não recomendável em repositórios em nuvem), volta com as alterações pré feitas ou seja não altera arquivos.
    - **git stash** // O git reset volta apenas para o commit, mas não volta diretamente com as alterações feitas no commit, exemplo se você apagou um arquivo, o git stash serve para reseter as alterações que você fez.
- **git diff <hash do commit> <hash do commit a comparar>** // Compara dois commits de acordo com o hash, mostrando arquivos que foram criados e alterações.

Branchs:

- **git branch** // Mostra qual branch estou utilizando no meu projeto
- **git branch <nome da branch>** // Cria uma nova branch no sistema
- **git switch <nome da branch>** // Altera a branch que você está utilizando
- **git merge <nome da branch que você quer copiar>** // Junta duas branchs, por exemplo se criar uma branch para corrigir bugs, depois tem que subir as correções para a master
- **git branch -D <nome da branch>** // Deleta branch que não está sendo utilizada mais
- **git push -u origin <nome-da-branch>** // Passar novas branchs para repositório remoto

Repositorios Remotos:

- **gh auth login** // Faz o login no github
- **git push** // Envia as alterações para o github
    - **git push origin <nome da branch>** // Especifica para qual branch deve enviar as alterações.
    - **git push <nome da branch local>:<nome branch remota>**// Caso as branchs remotas e locais tenham nome diferentes
- **git pull** // Puxa as alterações que foram commitadas no seu repositório remoto (sempre fazer quando for trabalhar em um repositório antes de qualquer coisa)

Git Ignore:

- Crie um arquivo .qitignore na pasta do projeto, esse arquivo serve o git ignorar algumas pastas ou arquivos e não fazer o commit delas. Ex: Não quero que o arquivo anotações.txt seja commitado, então eu digito seu nome no meu gitignore.

Anotações:

- Um commit é como se fosse um checkpoint dentro do nosso projeto.

- Arquivos que estão verdes no vscode são arquivos novos criados que temos que adicionar ao repositório.

- Arquivos que estão laranja são arquivos que foram alterados e não estão commitados ainda.

- **U (Untracked)** → Arquivo novo que ainda não foi adicionado ao Git (git add).
- **A (Added)** → Arquivo foi adicionado ao staging (git add), mas ainda não foi commitado.
- **M (Modified)** → Arquivo já está sendo rastreado pelo Git, mas foi modificado após o último commit.

- Branchs são versões do nosso projeto, a branch master é o aplicativo na sua versão final que está rodando.

- Quando criamos uma nova branch ela clona tudo que foi feito na branch master e a nova branch passa a ter seu próprio histórico.