Git & GitHub

O que é e pra que serve?

Git - (**git init,** usaremos somente na PRIMEIRA vez, depois não mais, para cada repositório)

Um sistema de controle de versão distribuído (DVCS) que permite que um time trabalhe em um mesmo projeto ou em um mesmo arquivo e provê ferramentas para contornar a maioria dos problemas que podemos ter nesse tipo de situação.

Utilizando o ***Git*** , você e seu time vão conseguir colaborar de forma assíncrona e harmoniosa, ganhando produtividade no momento de *"juntar as partes"* que cada um fez e tendo segurança de que todo o trabalho realizado está salvo.

GitHub

Ferramenta online que hospeda repositórios Git, é uma excelente ferramenta para toda a gestão do seu código, tendo recursos para acompanhamento de projeto ágil de software, controle de *Issues* , quadros *Kanban* , *Pull Requests* que permitem um *Code Review* mais bem feito, entre outras dezenas de integrações que vão facilitar sua vida com desenvolvimento de software.

O que é controle de versão?

É um mecanismo de versão distribuído. Onde tem todo o código, todo mecanismo, toda engenharia pra garantir todo o funcionamento do mesmo.
Github, Gitlab, etc. São plataformas que hospedam repositório do Git. Deixar seu projeto visível e distribuir para outras pessoas o seu repositório. Com diversas funcionalidades, que ajudam no fluxo de desenvolvimento de um software de uma equipe.

Git

- guarda um conjunto de operações dos seus arquivos através dos commits. O conjunto desses commits, cria um histórico de alterações ao longo de todo momento cronológico que você viveu. Tendo assim um caminho claro, pra poder ir e voltar entre as diferentes versões que o arquivo viveu, seu projeto.

Importância

Melhor fluxo de trabalho, mais eficiente no ponto de vista de gestão de mudança, diferenças entre um commit e outro. Possível comparar duas versões. Bom para gestão de conteúdo. Fácil para trabalhar em equipe, cada um no seu computador.

3 conceitos importantes:

- Ramificação (Branching)
- Mesclagem (Merge)
- Resolução de conflitos

Branch

É uma versão independente e editável que se cria de um código. Ramificação.

Branch master (main)

É a principal do código, a "oficial" onde ele é todo consolidado. Onde fica o código que vai para o ar, deploy. Também é editável. "O que as pessoas estão vendo".

- git branch
- git branch -d "nome da branch" , para deletar uma branch

Merge

Unir as alterações feitas numa branch e na outra branch. Para fazer o merge é necessário ir na branch principal.

- git merge

Autenticação

Fazer uma ponte entre o Git **(local)** e Github**(remoto)** de maneira segura.
Permitindo assim proteger suas informações pessoais e mandar comandos para o Github diretamente pelo terminal.
Quando o processo é feito, você informa ao sistema remoto que é para utilizar as credenciais da sua conta ao executar algum comando do git e, ao mesmo tempos, comprova para o Github que você é exatamente quem diz que é.

Checkout

Sair de uma branch e ir pra uma outra branch exemplo: git checkout main

Add

git add . (git add "nome do arquivo")

commit

Registrar a alteração significativa feita no projeto. Cria um histórico

git commit -m “uma mensagem aqui”

Git Clone

Serve para copiar um repositório Git já existente. O git recebe uma cópia de quase todos os dados que o servidor possui. Cada versão de cada arquivo no histórico do projeto é obtida quando você roda o comando git clone.

Git Log

Exibe os registros log do commit.
Antes de usar o git log é necessário adicionar os arquivos (git add) e fazer o git commit.
Com o comando **git log**, você pode visualizar o que vem sendo feito em uma determinada branch ou avaliar as alterações de um arquivo em especial. Isso pode ser útil para você entender como alguma parte do código vem sendo evoluída, ou pode ajudar a avaliar os commits locais antes de dar **git** push.

Git remove:

Como remover/apagar arquivos:
git-rm “nome do arquivo”
ou
ressuscitar arquivos através do git:
git log - -diff-filter=D - -sumary
depois
git checkout + “os quatro primeiros digitos do numero do commit (hash)” ~1 “nome do arquivo”
exemplo: git checkout 2234~1 sobremesas.txt

Git ignore

Serve para ignorar arquivos que você não quer adicionar.
.gitignore

Git push, pull, fetch

Fazer o git clone antes de tudo

Push

Envia as informações/alterações feitas no projeto, para o repositório remoto

Fetch
Puxa do repositório remoto para o repositório local. Verifica se há alguma mudança em relação a branch que está no repositório remoto.

Pull
Verifica também se há alguma modificação e além disso faz o “merge” entre as duas branchs, automaticamente (Uni as branchs). E traz as informações do repositório remoto para o local.

Linhas de código:
Primeiro usuário:
fazendo alteração
git add
git commit -m
git push

_____
Segundo usuário:
git fetch
git pull
Fazendo segunda alteração:
git add
git commit -m
git push

_____
git pull

Pull Request no Github

Importante para fazer sugestões e contribuir com alterações em um repositório, basta ter a permissão de leitura.
A vantagem de se realizar o *Pull Request* é que você garante que a branch-padrão terá todo seu trabalho concluído e aprovado, realizando todas as novas alterações em uma branch separada.