**O que é Git?**

É um sistema opensource para versionamento de código, ao usar git você cria um histórico do seu trabalho e torna possível (ou melhor) o trabalho em equipe.

**Principais comandos GIT**

1. **git init:** inicializa um novo repositório vazio.
2. **git clone:** serve para você clonar/baixar um repositório que já existe.
3. **git branch:**
   1. **git branch --all** serve para nós visualizarmos todas as branchs do projeto
   2. **git branch -d NOME\_DA\_BRANCH** serve para deletarmos uma branch.
4. **git checkout** 
   1. **git checkout -b NOME\_DA\_NOVA\_BRANCH** cria uma nova branch de trabalho, seria como se nós criássemos uma nova cópia do projeto para trabalharmos.
   2. **git checkout NOME\_DA\_BRANCH:** nos permite trocar de branch (para branchs já existentes).
5. **git status:** fornece as informações sobre o estado atual da branch, nela podemos ver as áreas de:
   1. **stage:** são os arquivos com alterações que já dizemos que vamos comittar
   2. **Not stage:** são arquivos que têm alterações mas ainda não adicionamos na área de stage.
6. **git add:**
   1. **git add NOME\_DO\_ARQUIVO:** Serve para adicionarmos um arquivo específico na área de stage.
   2. **git add .** ou **git add --all:** serve para adicionarmos todas as alterações que estão na área de não stage para a área de stage.
7. **git restore --staged** **CAMINHO/NOME\_ARQUIVO:** Serve para remover um arquivo da área de **stage**.
8. **git commit:** Esse comando serve para que nós criemos um marco no nosso desenvolvimento, por exemplo: acabamos de criar uma feature de validação de token, portanto o correto é nós usarmos o **git add** para adicionar os arquivos correspondentes a essa feature na área de **stage** e depois usarmos o **git commit** para criar um “marco” na história do nosso projeto.
9. **git remote add origin**: Esse comando vincula o repositório local com um repositório remoto, no caso usamos o github, geralmente o “link” do repositório remoto segue o seguinte formato: [git@github.com](mailto:git@github.com):daniel-cubos/test.git, daniel-cubos é o nome da organização ou usuário e **test.git** é o nome do repositório.
10. **git push:** Após adicionar os arquivos na área de stage usando o **git add**, devemos criar um “marco” com uma mensagem usando o comando do **git commit**, após essa sequência de comandos vamos enviar o nosso código para o repositório que “hospeda” os nossos códigos, no caso o **github.**
    1. **git push -u origin NOME\_BRANCH:** Envia as informações da branch atual e cria uma versão dela no repositório remoto.
    2. **git push:** envia as informações de uma branch já existente no repositório remoto.
11. **git pull:** é usado para obter atualizações do repositório remoto referente a branch atual.
12. **git merge:** Após finalizar um trabalho em uma branch específica podemos “**mergear”** ou seja, mesclar uma branch com outra. Digamos que estamos trabalhando numa branch chamada **auth**, terminamos o trabalho e agora queremos “juntar” com as informações que estão na branch **master**, como faremos?
    1. Estando na branch **auth,** digitamos o comando no terminal: **git merge master,** assim os nossos códigos irão se mesclar.

Mas nem tudo são flores, vamos imaginar que duas pessoas criem suas próprias branchs, trabalhem no mesmo arquivo, pior ainda, editem a mesma linha, imagina só a confusão que vira? Pois é, quando essas pessoas forem fazer o “**merge”** das suas branchs, teremos alguns conflitos, mas nós poderemos resolvê-los usando o nosso editor de texto.

Links úteis:

<https://blog.geekhunter.com.br/comandos-git-mais-utilizados/>

<https://woliveiras.com.br/posts/comandos-mais-utilizados-no-git/>

<https://www.treinaweb.com.br/blog/comandos-do-git-que-voce-precisa-conhecer-parte-1>

<https://rogerdudler.github.io/git-guide/index.pt_BR.html>

<https://www.treinaweb.com.br/blog/git-e-github-quais-as-diferencas>