**Git – Git Bash (são diferentes)**

um sistema de versionamento de arquivos. Conseguimos ter o controle de tudo aquilo que a gente já escreveu, apagou ou alterou.

Conseguimos ter o controle de tudo aquilo que a gente já escreveu, apagou ou alterou.

O Git Bash é um terminal estendido para otimizar o uso do Git.

Palavras importantes:

Branch: é uma ramificação que você vai fazer no seu projeto.

É como se você fosse criar uma “nova” linha cronológica dentro do projeto, sem apagar/alterar a linha principal. E posteriormente, se quiser, pode inclui-lo na linha principal.

Para fazer uma Branch é preciso criar esse arquivo no Git primeiro, para depois editar no VsCode

Commit: postar uma nova alteração do projeto. (são os pontos na história). tira print/screeshot do meu status

Merge: é a junção da branch com a linha princial.

Remote: uma ligação do Git com o GitHub, ao criar um remote, pegamos o link do repositório que criamos no Git e transferimos para o GitHub. O remote faz a ligação/conexão entre os dois (GitHub com Git).

Push: ele é utilizado para colocar o commit que fez na máquina, para o GitHub. Porque a nova versão que eu fizer na máquina ela não vai automaticamente para o GitHub, vou precisar fazer o Push no remote.

Pull: é ao contrário do Push. Puxar o que está no repositório do GitHub para a máquina.

Comandos para usar no Git

Ls – para listar

Ls -a – para listar pastas ocultas

Git init - para inicializar/criar um repositório git (vazio)

Git add – serve pra gente dizer qual arquivo vai para a área de staged.

* Utilizado tanto para passar de Untracked (não rastreado) para Staged(pronto), quanto de Modified (modificado) para Staged (pronto).
* Caso eu faça novas alterações, quando for fazer um novo commit, posso colocar "git add ." o ponto significa para colocar TODOS os arquivos.
* Git add \* - adiciona os arquivos não rastreados (Untracked)
* Git add nomeArquivo

Git status - saber o status do repositório;

Git commit -m “nomeArquivo” - para fazer o commit. Entre aspas coloca a mensagem do commit que também serve como título.

Git config --list - serve para listar as configurações do Git;

Git remote add origin + o link do GitHub - "Git, faça essa conexão do nosso repositório local com o repositório do GitHub que tem o nome de origin.”

Git push -u origin master - ele vai fazer o "empurrão", de um lugar para o outro.

Para fazer um novo Push, porém no mesmo arquivo, não precisa colocar a letra "u”.

Git checkout -b "novo-botao" (nome da pasta/arquivo) - com esse comando, cria uma nova Branch e o checkout sai da brend atual (master) e vai para a nova "novo-botao"

Git checkout master - para voltar a brend, voltar ao que estava antes.

Git merge novo-botao - para unir as duas linhas, usa esse comando merge com o nome da Branch, depois disso, usar o comando push

clear - para limpar.

Git clone + link.git - para clonar algum projeto do Git.

Git pull - puxar alterações feitas do arquivo clonado.

\*Para ajudar a lembrar a continuação de algum comando, usar: git + nome de algum comando + -h. Exemplo: git commit (esqueci qual letra coloca depois, então uso o -h e ele me mostra as opções que podem ser usadas)

\*No git aparece o nome Master entre aspas, significa que essa pasta ou esse novo repositório é a Linha;

Principal do projeto

\* md - linguagem de marcação, todas as informações do projeto podem ser escritas nesse MD, que não será passado para o projeto em si.

* Acredito que podemos criar primeiro um repositório no GitHub, onde podemos já selecionar a opção de iniciar com o README, fazer um clone e começar a editar, fazer o que deseja.

Comandos para navegar no Terminal (cmd), mas que também pode ser usado no Git Bash:

cd – usado para navegar entre as pastas. (voltar um nível)

Para entrar em uma pasta especifica, usa cd + nome da pasta.

Para voltar a pasta anterior: cd ..

dir – lista de diretórios contidos na pasta em que estou. Ex.: C:\Users\Victoria

mkdir – Para criar uma pasta, coloca o comando + nome da pasta

ls - listar

del – para deletar repositórios (algo que está dentro de alguma pasta) Ex.: del workspace – vai deletar tudo que está dentro da pasta workspace.

Rmdir – para deletar a pasta/arquivo. Ex.: rmdir workspace /S (se não colocar o /S, ele não vai deletar). Se colocar: rmdir workspace /S /Q (colocando o /Q, o terminal deleta direto, sem perguntar se tem certeza.)

Mv (nome do arquivo) ./(nome da pasta) - para mover o arquivo para a pasta

cls - Para limpar o terminal

echo – ele printa de volta no terminal a palavra que você digitou. Para criar a pasta, usamos: echo hello > hello.txt

\*atalho no teclado

Tab – ele completa as palavras. Ex.: cd W+tag, ele vai completar a palavra Windows.

Objetos internos do Git

Blob(tem sha1 do arquivo) – Tree(aponta para a Blob) – Commit(aponta para arvore/tree, parente, autor e mensagem)

**GitHub**

plataforma para hospedar os arquivos / uma rede social para os desenvolvedores, onde compartilhamos os nossos projetos e códigos.

Interessante saber que o GitHub pode ser usado como portifólio por empresas;

GitHub trabalha com repositórios que são os diretórios (pastas do git) para a gente armazenar esses projetos

Para criar um repositório:

1. Clicar em Your repositories;
2. Clicar no botão New/Novo;
3. Dar um nome ao repositório no Repository name;
4. Se quiser, adicionar uma descrição;
5. Escolher se quer deixá-lo Público ou Não;
6. Se o seu projeto já tiver um README, não precisa marcar a opção que aparece;
7. Clicar em Create Repository.

Agora temos que apontar/empurrar ao repositório local: Git, para isso:

1. Copiar o link que aparece na tela do GitHub;
2. Abre o GitBash;
3. Verificar se está na pasta certa;
4. Usar o comando: git remote add origin + o link copiado;
5. Caso queira verificar se deu certo, pode usar o comando: git remote -v . Ele vai listar a lista de repositórios remotos cadastrados. Ou usar o comando: git status;
6. Pós isso, usamos o comando: git push origin master
7. Nessa parte o git pode pedir alguma senha e pronto.