Tutorial de utilização do Git Hub

1º Passo: criar repositório;

- Para criarmos um repósitório, é necessário criar uma conta no site. Após a criação e configuração de sua conta, acesse o seu perfil.

- Lá, você encontrará uma opção nomeada de "Repositórios", clique na opção "Novo".

- Na tela de criação do seu repositório, você verá opções simples, como:

Definição do nome do seu repositório(sem espaços);

Definição da privacidade de seu repositório(Público ou Privado);

Adição de um README(arquivo que você explicará a finalidade do seu repositório);

Adição de um gitnore(Escolha de quais arquivos não rastrear em uma lista de templates);

E finalmente a escolha de uma licensa(para que você controle quem pode ou não utilizar seu código, e que seu repositório receba os devidos créditos caso seja utilizado por alguém);

* Para os próximos passos, é fundamental que você faça o download do programa GitBash (https://gitforwindows.org/)

2º Passo: Passos iniciais para utilização do Git Bash;

- Após a instalação do Git Bash, crie uma pasta no seu computador, e clique com o botão direito do mouse, você verá a opção "Git Bash here", clique nela e assim você terá definido o local o qual seu Git Bash armazenará seu repositório.

- Você verá que uma janela parecida com um CMD foi aberta, não a feche!! Digite git clone nesta janela mas não pressione nada.

- Voltemos ao site, entre no seu repositório criado e clique no botão verde "Code", copie o código do seu repositório em https e o cole na janela do Git Bash, utilize (shift + insert para colar).

- Feito isso, pressione enter. O que aconteceu aqui foi que você clonou(através do git clone) o seu repositório da internet para seu computador!!.

- Se nosso repositório for privado, nos será solicitado um token de acesso, que apenas o criador pode acessar, visto que será criado um clone do repositório. O login para este token é o mesmo informado para sua conta no site do github.

- Agora vamos acessar a pasta o qual o repositório se encontra. Para isso, precisamos utilizar a função "cd (nome da pasta)"

* Também podemos entrar na pasta, clicar com o botão direito do mouse e abrir outro gitbash

- Com este clone, temos os mesmos arquivos do nosso repositório

- Ao alterarmos QUALQUER coisa na pasta e utilizarmos o comando/função git status, o código nos mostrará que algo foi mudado no repositório """Untracked files: etc

- Para adicionarmos a modificação ao repositório online, usamos o comando "git add . " (neste caso, ao usarmos a sintaxe ponto, adicionamos tudo o que foi mudado) (caso quisermos, podemos adicionar apenas um arquivo que foi alterado/adicionado chamando-o pelo seu nome.{tipo_de_arquivo}) o arquivo alterado/adicionado aparecerá em verde, ao invés de vermelho. -podemos fragmentar os arquivos que queremos enviar, utilizando add apenas neles-

- Agora, utilizamos o comando commit para salvar de fato o que foi feito. a sintaxe é: "git commit -m "(NOME DA ATUALIZAÇÃO)""

- Agora, ao utilizarmos o git status novamente, aparecerá que não há nenhuma novidade. acabamos?

- Não, ainda falta executar o comando "git push" para "empurrarmos" as mudanças feitas para o servidor!

*** Considerações sobre git push e git pull:**

1. Se utilizamos o git push para enviar/exportar dados, utilizamos o git pull para receber/importar dados!

2. Antes de fazer qualquer coisa, é bom fazer um git pull apenas para prevenir erros de acontecerem.

3. Em suma, o pull reflete as alterações servidor_local - servidor_github

4. Caso haja algum erro, o console/prompt irá sinalizar o conflito.

3º Passo: O repositório é basicamente uma pasta;

- Inicialmente temos o ramo padrão/main branch. podemos criar mais deles para termos mais "áreas de trabalho" no repositório. a alteração que fazemos em um ramo x não altera o ramo principal. podemos até mesmo criar ramos dos ramos.

- Podemos mandar o tanto de mudanças/dados a serem incluídos que desejarmos. o mais comum é mandar todos os dados de uma vez.

- Para navegar entre esses ramos, utilizamos o símbolo logo abaixo da "aba" <> code

- Ao criarmos um branch, fazemos um clone/cópia do branch principal, além disso, ao criarmos um novo branch, devemos executar git pull para adicionar o ramo ao nosso computador.

- Se quisermos trabalhar em outro ramo, utilizamos o comando "git checkout (nome do branch)"

- Se alteramos algum arquivo, mudamos de ramo apenas com o checkout (sem fazer os procedimentos adequados para exportar a alteração para o github), o arquivo continuará na pasta na qual estamos trabalhando.

- Ao finalizar a exportação do arquivo alterado, se fizermos um checkout para outro ramo, o arquivo em questão irá "sumir"

- E quando queremos atualizar nosso ramo padrão/principal com as informações/dados importados do ramo alterantivo?

    * Selecionamos a opção/aba "pull request", então "new pull request" [NÃO É O COMPARE & PULL REQUEST]

    * Após isso, devemos selecionar a direção dos ramos cujas informações/mudanças queremos trocar.

- Enfim, conseguimos juntar os dois ramos.

- Também podemos usar o pull request para juntar ramos até de diferentes repositórios!!

4º Passo: Utilização de Commits como uma "máquina do tempo"!;

- O comando/opção view commit details serve como um histórico do que aconteceu. as linhas em vermelho são as linhas apagadas, as linhas em verde são as que foram adicionadas. podemos comentar o que fizemos em cada uma dessas linhas.

- Commits são utilizados para fazer anotações, salvar/criar arquivos e etc, tanto em markdown, quanto em código. vale ressaltar que nossos códigos não poderão ser executados, apenas escritos.

- Com o git, podemos utilizar os commits para voltar a uma versão anterior do nosso repositório! Selecionamos o símbolo <>, "browse the repository at this point in the history".

- Podemos utilizar isto apenas para OBSERVAR o código/commit, esta função não nos deixa editar o que aconteceu, pelo menos, não por agora.

5º Passo: Utilização de códigos de outras pessoas no Git Hub(Fork):

- E se quisermos pegar um código/repositório alheio? Bsta utilizar a opção Fork, na parte das abas no browser no github!

- Com isso, basicamente criamos um novo repositório no nosso servidor local, da maneira tradicional

- Por fim, enviamos o repositório e solocitamos o pull request do autor do repositório que modificamos.

- Vale ressaltar que não é possível completar o pull request se mais de alguém que utilizou o fork alterou um arquivo original do repositório original. dessa forma, cabe ao autor do repositório escolher qual das mudanças ele escolherá como a definitiva.