**Primeiros passos com Github**

**Criando e clonando repositórios**

A duas formas de obter um repositório Git em nossa máquina: criando ou clonando.

**Criando ou transformando uma pasta local.**

1. Em c: crie uma pasta onde será organizado o projeto
2. Abra a pasta e clique com o botão direito do mouse
3. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
4. Digite **mkdir (mais o nome do repositório)** = para criar a pasta
5. Digite **cd (mais o nome do repositório criado)** = para entrar na pasta
6. Digite **git init** = para iniciar um repositório git

**COMANDOS:**

**cd .git =** mostra que é um diretório GIT\_DIR

**ls =** lista o que tem no repositório

**cat config =** configurações responsáveis pelo repositório

**Clonando um repositório**

1. Em c: crie uma pasta onde será organizado o projeto
2. Abra a pasta e clique com o botão direito do mouse
3. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
4. Digite  **git clone (mais o link HTTPS ou SSH)**

**Mudando o nome ao clonar um repositório**

1. Em c: crie uma pasta onde será organizado o projeto
2. Abra a pasta e clique com o botão direito do mouse
3. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
4. Digite  **git clone (mais o link HTTPS ou SSH)**  e em seguida o **nome desejado**

**COMANDOS:**

**cd (mais o nome de uma pasta)** = muda ou viaja entre as pastas

**. =** é usado para pastas ocultas. Exemplo: **.git**

**git remote -v =**  mostra os repositórios vinculados.

**cd .. =**  Volta a pasta local.

**Conectar repositório local ao remoto**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git remote add origin (mais o link HTTPS OU SSH)**

**Conferindo se um repositório está ou foi vinculado**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite  **cd .git**
4. Digite **cat config**

**Clonando apenas uma branch**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite  **git clone (mais o link HTTPS ou SSH) - - (a branch desejada) - -single-branch**

**Criação de repositórios remotos GitHub**

1. Acesse seu github
2. Clique no sinal de **+(mais)** ao lado do usuário
3. Escolha a opção **new repository**
4. Preencha o formulário

**COMANDOS:**

**git status =**  mostra como está seu repositório (commit, pull, push) alterações nele feitas.

**touch README.md =** cria um arquivo README no repositório

**Salvando alterações**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **mkdir (mais o nome do repositório)** = para criar a pasta
4. Digite **cd (mais o nome do repositório criado)** = para entrar na pasta
5. Digite **git init** = para iniciar um repositório git

**OBS:**  Marckdown é uma linguagem de marcação com o HTML, porém mais simples. Podemos utilizar o navegador para escrever, por exemplo o site readme.so é um deles.

1. Digite **git add .** = adiciona todos os arquivos na área de preparação
   1. É possível adicionar arquivos separados, para isso digite **git add (mais o nome do arquivo)**
2. Digite **git commit -m “nome do commit”**

**COMANDOS:**

**git log =** exibe todos os commits

**Diretórios vazios**

O Git não reconhece diretórios vazios, caso você tente clonar ou crie uma pasta vazia dentro do seu repositório. Há uma convenção que, caso você precise de um diretório vazio, cria-se um arquivo chamado gitkeep.

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **mkdir (mais o nome do repositório)** = para criar a pasta
4. Digite **touch (nome da pasta).gitkeep**

**Gitignore**

Caso você não queira que alguma pasta ou arquivo suba para seu repositório (local ou remoto) siga os seguintes passos:

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **echo (o nome da pasta ou arquivo)/ > .gitignore**

**Desfazendo Alterações no repositório local**

Devemos sempre estar atentos a onde estamos salvando nossos diretórios locais, porém pode acontecer de errarmos. Caso você de um **git init** no diretório errado é Só excluir o arquivo **.git**  da pasta ou:

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **rm -rf .git**
4. Confirme a exclusão digitando **git status**

**Restaurando um arquivo**

Caso você queira restaurar a um estado anterior.

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status** = para ver o status da arvore (tree)
4. Digite **git restore (mais o nome do arquivo que deseja restaurar)**

Quando você faz o **git status** no passo 3, se há alguma alteração no seu arquivo, aparece uma sugestão do **git restore** e abaixo os nomes dos arquivos que foram modificados.

**ATENÇÃO:** Quando utilizado este comando, tenha certeza de que você não quer salvar nenhuma das alterações do arquivo, pois ele descartara todas as modificações feitas até o último commit.

**Modificando a mensagem do último commit**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git log** para listar os commits feitos
4. Digite **git commit - - amend -m”(nova mensagem)”**

**Retornar a um commit anterior**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git log** para lista os feitos
4. Digite **git reset (opções)**  mais a **hash**  do commit desejado
   1. **-- soft** = adiciona os arquivos do commit na área de preparação
   2. **- -mixed** = adiciona os arquivos na arvore de trabalho (untracked)
   3. **- - hard** = elimina os arquivos e os commits atuais.

Quando a **opção - -hard** é usada ao digitar **git log** não aparecerá mais as hashs dos outros commits. Caso queira ver estes commits é só digitar **git reflog.**

**ATENÇÃO:** É importante fazer todas estas alterações antes de enviar para o repositório remoto para evitar conflitos, pois estamos mexendo com o histórico de commit.

**Retirando arquivos da área de preparação**

Caso você tenha feito um **git add .**  , mas quer retirar algum arquivo da área de preparação.

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**
4. Digite **git reset (mais o nome da pasta/arquivo)**

Ou

1. Digite **git restore - -staged (mais o nome da pasta/arquivo)**

**Enviando e baixando alterações**

1. Crie no seu github um repositório com o mesmo nome do diretório
2. Abra a pasta e clique com o botão direito do mouse
3. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
4. Digite **git status**  para ver se há arquivos não rastreados
5. Digite **git add .**  para adicionar arquivos na área de preparação
6. Digite **git commit -m “mensagem do commit”**
7. Digite **git remote add origin (mais a url do repositório no github)**
8. Digite **git push -u origin main**

Atualizando a página do GitHub veremos os arquivos enviados.

**Editando no repositório remoto**

1. Conecte-se a sua conta do Github
2. Abra o repositório que esta o arquivo a ser editado
3. Abra o arquivo que irá editar
4. Clique no ícone de lápis

**Commitando alteração remota**

1. No final da página de edição a o botão commit Changes
2. Escreva a mensagem de commit
3. Clique no botão

**Usando o editor Web do Github**

O github tem um editor web que é praticamente um VSCODE

1. Conecte-se à sua conta do Github
2. Abra o repositório que está o arquivo a ser editado
3. Use a tecla de atalho . (literalmente aperte ponto no teclado)

Após fazer as alterações, assim como no Vscode, a um ícone na lateral direita que mostra que temos que enviá-las.

1. Clique no ícone
2. Escreva a mensagem
3. Clique em commit Changes

**Puxando as alterações do repositório**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git pull**  para puxar as alterações

**Trabalhando com branches**

Simplesmente e você não trabalhar com a **página principal** do seu projeto. Na principal ficará somente o que já está funcionando e vai será finalizado. Para isso criamos **Ramos (branches)**, de forma que editamos (a equipe envolvida) e depois juntamos (merging) tudo, antes de finalizar.

**Criando uma Branch e alternado para ela**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git chekout -b (mais o nome da brach)**

Se digitarmos **git log**  veremos que nossos commits estão apontando **(HEAD -> (nome da branch) , main)** para a principal **(main)**.

Agora estamos trabalhando dentro de uma **branch (ramo) que não e a main (principal)**, se fizermos um novo commit ele será salvo dentro da  **nova branch e a principal ficará no commit antigo.**

**Voltando para branch main (ramo principal)**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**  e verifique em qual branch você está
4. Caso não esteja na **main**, digite **git chekout** para voltar

**Listando os últimos commits de todas as branches**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git branch -v**

**COMANDOS:**

**git branch – v =**  lista os últimos commits de todas as branches

**Mesclando (juntando) branches**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**  e verifique em qual branch você está
4. Digite **git merge (mais a branch que será mesclada)**

**COMANDOS:**

**git merge=**  mescla as branches

**Excluindo uma branch**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**  e verifique em qual branch você está
4. Digite **git branch** para listar todas
5. Digita **git branch -d (mais a branch que será excluída)**

**COMANDOS:**

**git branch =**  lista todas as branches

**git branch – d =** deleta a branch

**Conflitos de Merge**

Se dois programadores estão no mesmo projeto e fazem alterações na mesma linha de programa e commitam, você receberá um aviso que deve primeiro **fazer um git pull** para comparar os arquivos e depois enviar para o repositório remoto **(git push)**.

**Trabalhando com Braches – comandos úteis no dia a dia.**

**Baixar repositório sem commits dos outros programadores**

Para não ter que ficar mesclando seu repositório toda hora com os de outros programadores através do **git pull**  que é a junção do **git fetch e git merge** fazemos assim.

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**  e verifique em qual branch você está
4. Digite **git branch** para listar todas
5. Digite **git fetch origin (nome da branch)** para puxá-la
6. Digite **git diff (nome da branch) /(nome da branch)** para comparar as alterações

**COMANDOS:**

**git fetch =** puxa o repositório sem os commits de outros programadores

**git diff =** compara as alterações de branches

**Comparando alterações e mesclando**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git status**  e verifique em qual branch você está
4. Digite **git branch** para listar todas
5. Digite **git diff (mais os nomes das branches separada por /)** para comparar as alterações
6. Digite **git merge (mais os nomes das branches separada por /)**

**Clonando repositório com várias branches**

1. Em c: crie uma pasta onde será organizado o projeto
2. Abra a pasta e clique com o botão direito do mouse
3. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
4. Digite **git branch** para listar todas
5. Digite  **git clone (mais o link HTTPS ou SSH) - - branch (nome da branch) -- single branch**

**Arquivando uma alteração**

1. Abra a pasta e clique com o botão direito do mouse
2. Selecione a opção **git bach** 
   1. Caso não esteja aparecendo clique em **Mostrar mais opções**
3. Digite **git stash** para arquivar a alteração

**COMANDOS:**

**git stash =** arquiva as alterações

**git stash list =** lista as alterações

**git stash pop =** traz as alterações e exclui a mais recente

**git stash apply =** traz as alterações e mantem a mais recente

**Links de materiais de apoio:**

[**https://github.com/elidianaandrade/dio-curso-git-github**](https://github.com/elidianaandrade/dio-curso-git-github)

[**https://academiapme-my.sharepoint.com/:p:/g/personal/renato\_dio\_me/EYjkgVZuUv5HsVgJUEPv1\_oB\_QWs8MFBY\_PBQ2UAtLqucg?rtime=ZyVA\_EDC3Eg**](https://academiapme-my.sharepoint.com/:p:/g/personal/renato_dio_me/EYjkgVZuUv5HsVgJUEPv1_oB_QWs8MFBY_PBQ2UAtLqucg?rtime=ZyVA_EDC3Eg)