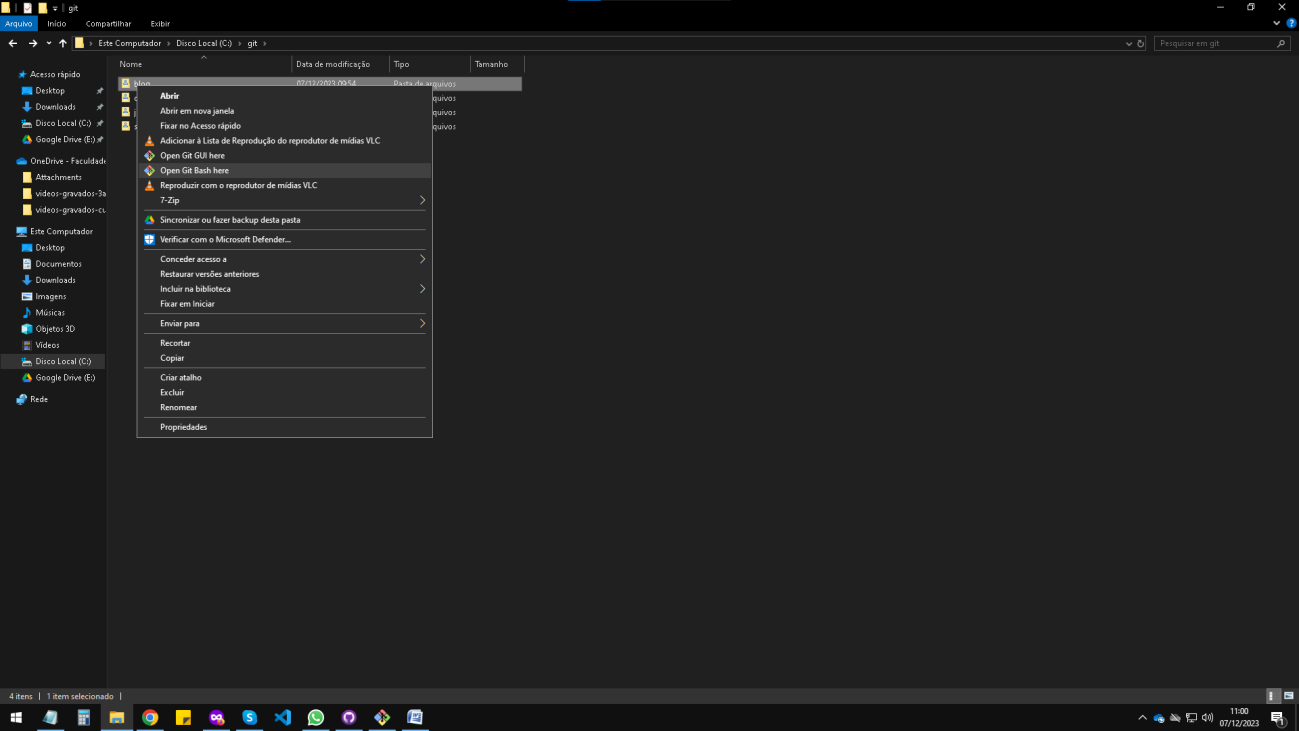
Para instalar o git, basta clicar em “next”...

iniciar repositorio

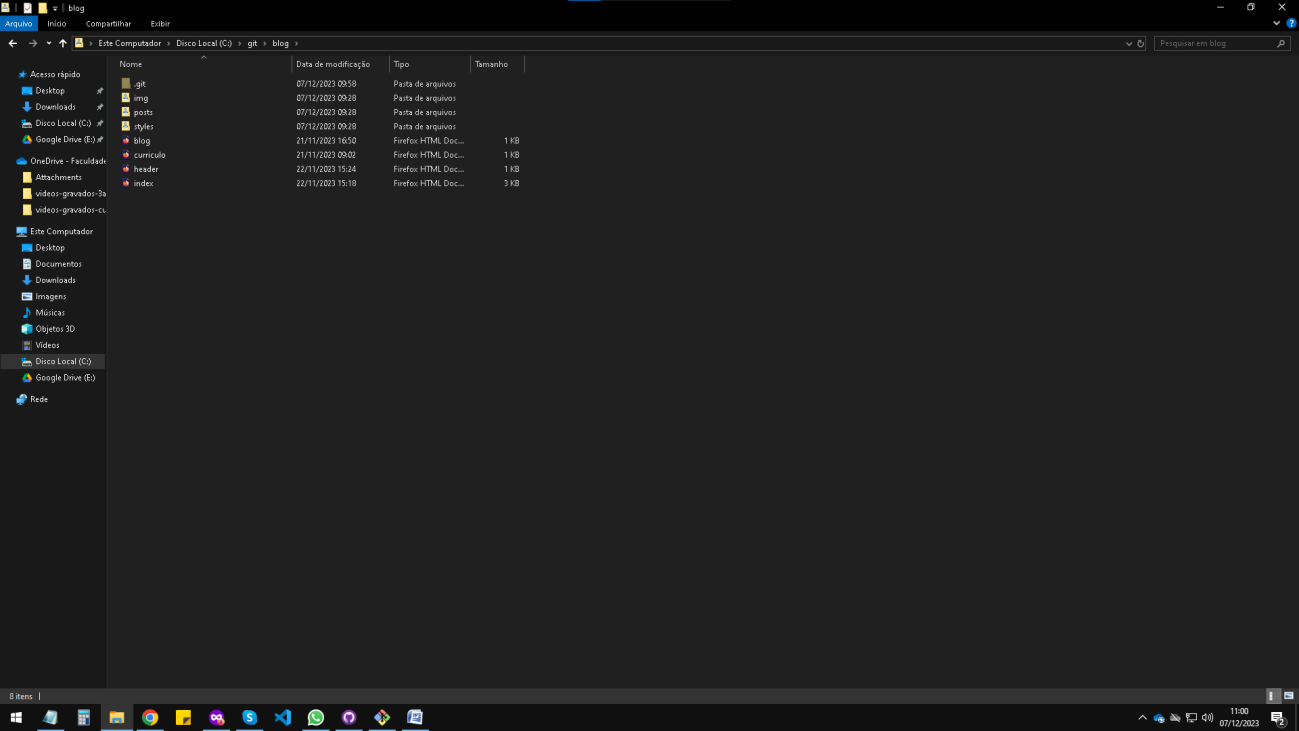
botao direito na pasta e selecione "Open Git Bash here” para abrir o terminal próprio do git.



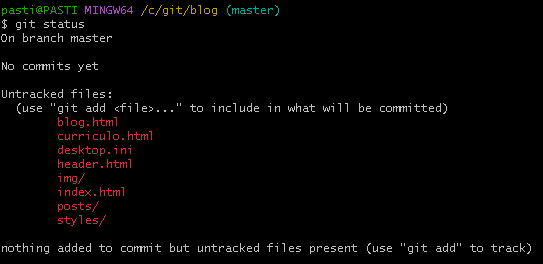
O comando “git init” inicia o repositório na pasta selecionada.



Será cirada uma pasta oculta ".git"



O comando "git status" exibirá o status do repositório, além de quais arquivos já foram ou não incluídos no controle de versão.



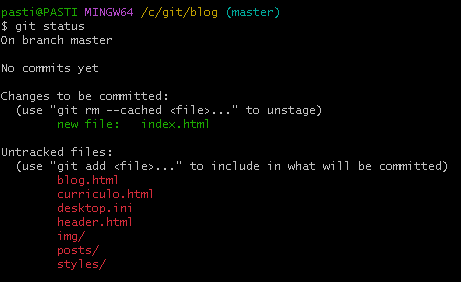
“No commits yet” informa que não há versões do código.

“Untracked files:” informa que arquivos não estão no controle de versão e quais arquivos sã esses.

“Git add “nome do arquivo.extensao” ” adicionara o arquivo no controle de versão.



Ao rodar o comando “git status” novamente, ele exibira o arquivo incluído recentemente ao controle de versão.



O comando “git add .” adicionará todos os arquivos contidos na pasta ao controle de versão.



Após adicionar os arquivos ao controle de versão, é necessário criar uma versão antes de disponibilizá-lo no github (ou qualquer outro).

O comando “git commit –m“mensagem”” criará a primeira versão do repositório.

No local do texto “mensagem”, após o termo –m“” é necessário adicionar a mensagem do porque foi criado essa versão (commit) do código.

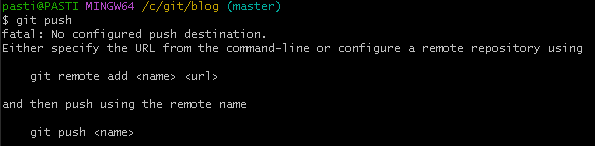
Ao rodar esse comando pela primeira vez, será pedido que configure o git na máquina.

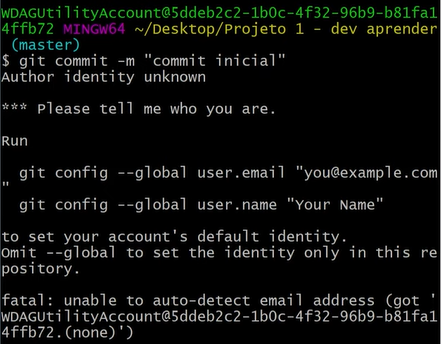
$ git config --global user.email fulanodetal@exemplo.br

$ git config --global user.name "Fulano de Tal"

Deverão ser utilizados os mesmos dados da conta no GitHub.

Essa informação é importante porque o nome ficará atrelado a cada versão (commit) enviado.



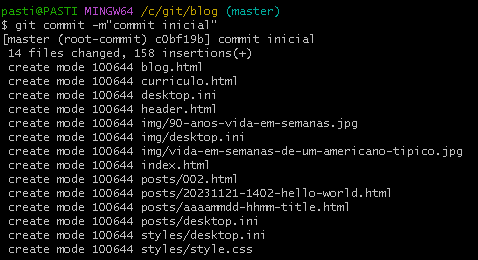


Após rodar o comando “git commit –m“mensagem”” são exibidas as seguintes informações:

Nome do commit;

Nome da brand para onde foram enviadas essas alterações. No caso, foi a brand master.”;

Arquivos adicionados à versão;



git push – enviar alterações para o servidor na nuvem. O erro se deu porque ainda não foi definido o local para onde será enviado. Nesse caso deve-se acessar o GitHub e criar um repositório. Após criar o repositório através do site do Github, basta copiar o link do mesmo (<https://github.com/tinhnpassos/blog>) para o terminal bash do git.

Git remote add origin <https://github.com/tinhnpassos/blog> - esse comando define o local onde para onde o código será enviado.



Após isso, execute novamente o comando “git push”. Ocorrerá um erro porque ainda não foi definido qual a branch ele deverá enviar o código.

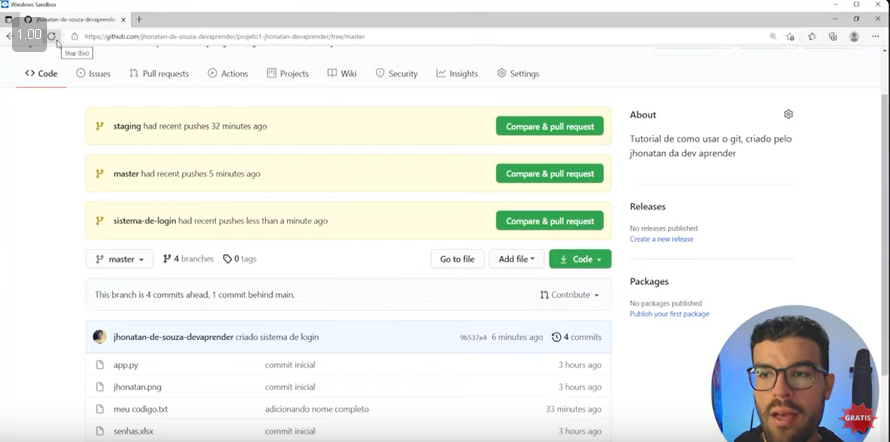


Git reflog exibe o histórico de atualizações do código, em ordem decrescente de versões.

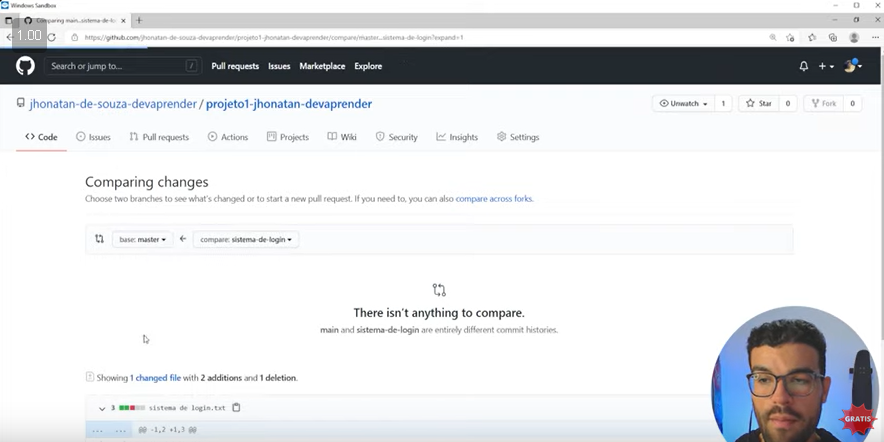


Git reset –hard id-version (onde id-version será por ex.: c0bf19b) retorna para a versão desejada do código. Esse comando permite navegar entre as versões do código.

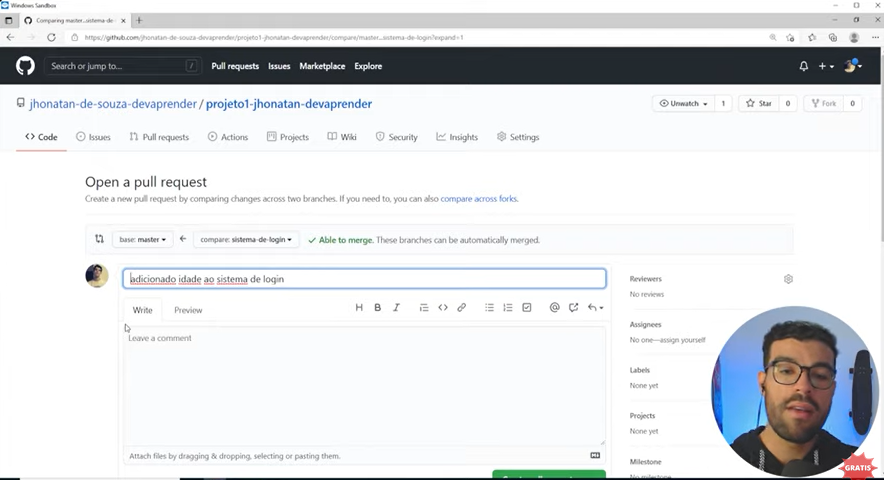
Pull request é utilizado no site do github após o envio das atualizações.



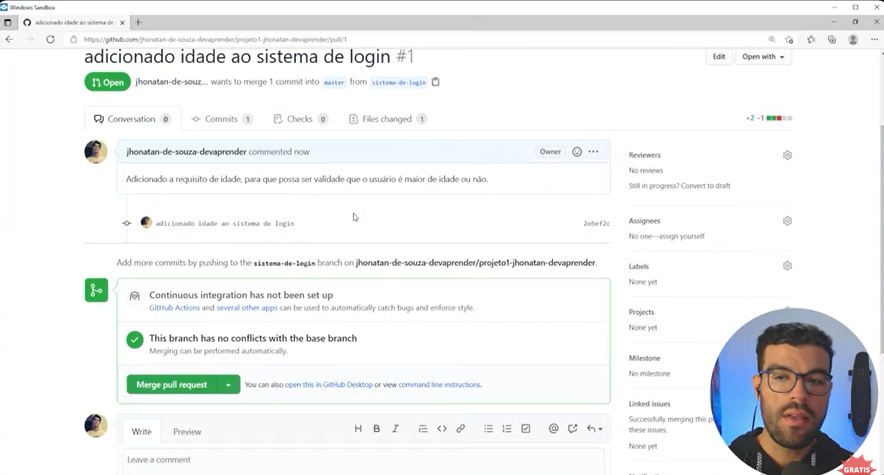
Após clicar em “Compare & pull request”, basta informar o que é mergiar com qual branch.



Após isso, poderá ser descrito o que foi feito além do nome do commit.

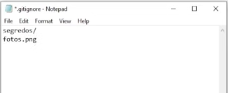


Na aba na lateral direita é possível verificar o menu “reviewrs” que são as pessoas que revisarão os seus projetos. Caso a atualização seja reprovada, basta verificar os comentários, realizar as alterações e enviar um novo commit, NÃO SENDO NECESSÁRIO ENVIAR UM NOVO PULL REQUEST. Basta enviar um comentário comentando o nome dos revisores que não aprovaração a nova versão, dizendo que realizou as alterações solicitadas.



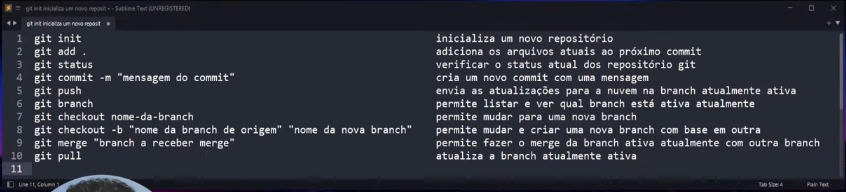
Touch .gitignore – criar um arquivo .gtignore no repositório do projeto.

Após criado, abra o arquivo através do notepad para definir os arquivos a serem ignorados.



Após isso, execute os comandos git add . , git status, git commit –m”tituloCommit” e git push para adicionar o arquivo .gitignore ao repositório.

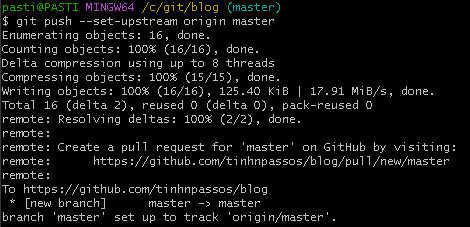
Resumo:



Vocabulário:

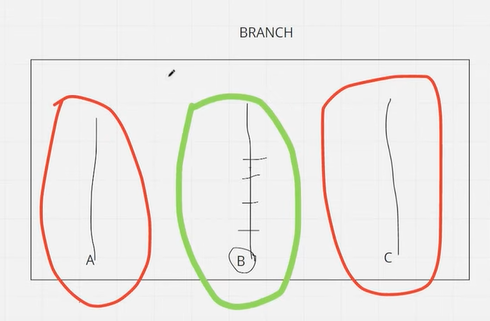
commit é versão do código

branch (galho) são divisões ou versões diferentes do código que está sendo versionado de forma separada. Inicialmente, enviaremos sempre para a master, Logo, você deverá executar o comando sugerido: “git push --set-upstream origin master”. Após executar o comando, aparecerá uma janela para realizar login, caso ainda não o tenha realizado. Feito, login, o comando fará o envio do código ao Github.



Ao retornar na página do github e atualiza-la, será exibida uma mensagem informando que houve envios recentes “master had recent pushes 1 minute ago”. Basta mudar da brand main para brand master para acessar o código.

Em projetos onde mais de uma pessoa trabalha ou aqueles já utilizados por clientes, é comum deixar um branch com a versão estável e outras branchs onde são inseridas as edições e melhorias antes de ser enviadas para a versão estável.



Git branch exibe as branchs disponíveis.



O \* e a cor verde no nome indicam qual a branch em uso, atualmente.

Git branch nomeASerUsado onde nomeASerUsado será o nome da branch. Esse comando é utilizado para a criação de uma nova branch. “Staging” é comumente utilizado para branchs de desenvolvimento, testes e afins.

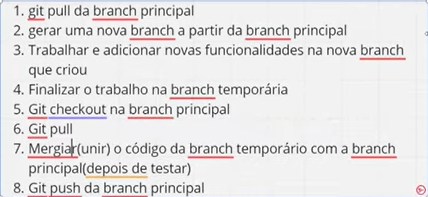


Git checkout Diego é utilizado para mudar para a branch citada no comando.



Para unir as alterações realizadas em branchs diferentes. Você deverá ir na branch que receberá a nova versão (git branch master) e depois executar o comando git merge staging. ANTES DE REALIZAR A UNIÃO DOS CÓDIGOS (MERGE), RECOMENDA-SE FORTEMENTE EXECUTAR O COMANDO git pull PARA TRAZER À MÁQUINA A VERSÃO MAIS RECENTE DO CÓDIGO DISPONÍVEL NO REPOSITÓRIO.





Git checkout –b nomeASerUtilizado master é usado para sair da branch atual para a branch a ser criada em um único comando e o “master” é o nome da branch utilizada como base para a criação da nova branch.

Conforme a imagem abaixo, será exibida as alterações realizadas.

LEMBRE-SE SEMPRE DE UTILIZAR O COMANDO GIT PUSH PARA ENVIAR AS ALTERAÇÕES AO SERVIDOR, NO CASO PARA O GITHUB.

