Comandos Git

* Criar uma pasta que irá comportar os códigos ou arquivos (adicione os arquivos)
* No terminal cmd digitar git init (botão direito git bash here)
* Registrar os arquivos por unidade com git add “nomedoarquivo”
* Ver a situação atual com o comando git status
* Para adicionar vários arquivos de uma única vez com git add . (tem que ter o espaço antes dos pontos)
* Criar a versão para subir ao git com git commit -m “nome da versão”
* Depois de criar o repositório na pagina git pega o endereço para subir os códigos
* Usar o comando para subir git remote add origin <https://...endereço>...
* Logo após usar comando git push (caso não funcione usar o código que ira indicar na mensagem (git push –-set -upstream origin master)
* Se autorizado irá subir os códigos
* Caso eu faça alguma modificação em algo nos arquivos com o comando git status irá indicar
* Com a situação acima tenho que subir novamente essa modificação com git add .
* Logo após preciso fazer um commit com comando e o que foi adicionado ou modificado git commit -m “adicionado-mudança”
* Logo após usar o git push para subir a mudança no servidor da nuvem
* Usar o comando git reflog mostrará as versões realizadas no repositório (a versão mais antiga sempre estará no final)
* Para retornar a versão anterior ou atualizar para versão mais velha usa o comando git reset –hard código da versão (ex: cd3bn89)
* Com o mesmo comando é possível retorna a versão mais atualizada que eu tinha retrocedido, basta colocar o código da versão mais atual

----------------------------Branch---------------------------------

* Seria uma espécie de lugar onde guardara os códigos funcionais ou processo de desenvolvimento, por exemplo o local master no git que fica como o principal onde ficaria os códigos que estão funcionais e ativos, poderia ser criado outro lugar sem ser o master onde poderia separar para códigos ainda em criação, isso vem a ser o Branch, caminhos diferentes
* Comando para ver as Branch é git branch
* Para criar uma nova Branch usa o comando git branch nome (ex: staging, que viria ser uma branch com esse nome em códigos que estão em estado de criação)
* Para verificar usa-se o comando git branch
* Para entrar na branch recém criada usa se o comando git checkout nomedabranch
* Caso comece a mudança em algum documento se faz necessário subir os documentos com comando git add . depois o git commit -m “ o que foi mudado” e depois o git push para subir o código, caso não suba com o comando irá mostrar o comando para ser usado na mensagem acima no terminal.
* Unir os códigos da branch diferentes ou transferir um código da branch 1 para a 2
* Primeiro vai para a branch que irá receber git checkout a-principal
* Para pegar os dados atualizados de outras pessoas do mesmo repositório usa-se o comando git pull, assim será baixado os arquivos mais atualizados do servidor para a minha máquina
* Depois para pegar os arquivos da branch 1 para a que estou no momento usa-se git merge nome-da-branch-que-tem-os-arquivos
* Criar uma nova branch com base da branch principal que estou utilizando usa-se git checkout -b nome-da-branch e-o-nome-da-branch-base
* Logo cria-se na pasta algum documento ou coloca, e depois aplica o comando git add ., logo depois pode ver o status e fazer um commit com o que foi adicionado em mensagem
* Logo depois vai para o branch que irá receber o arquivo novo ex:master com o checkout, e depois de atualizar com o pull, dá um merge com o nome da nova branch que tem os arquivos a adicionar e finalmente o push.
* Para adicionar uma regra de aprovação depois de fazer todos os passos de adicionar, mandar e subir para nuvem no branch, vai ate a pagina do git e clica no botão (Compare pull & request) e seleciona o branch principal que deseja aprovação do seu código para incorporar no código do dono, colocando o que está sendo solicitado (no caso pode ser a mensagem do commit) e caso precisa tem a caixa de texto
* Depois de fazer os passos, irá aparecer um botão com merge para aprovar, se tiver uma equipe irá precisar dos reviews e aprovação deles

---------------------------GIT IGNORE -------------------------------

* Usar para não mandar certos arquivos que não se fazem necessários
* Preciso usar um comando touch .gitignore isso criara um arquivo com este mesmo nome onde irá adicionar os arquivos ou pastas a serem ignorados (ex: pasta/ ou foto.png ) dentro do arquivo
* Logo após verificar com status, precisa subir o arquivo que bloqueia no caso o gitignore