Git

O Git é um sistema de controle de versão. Desenvolvedores usam sistemas desse tipo para compartilhar código e gerenciar as versões dele. O Git é diferente do GitHub, o GitHub é onde o código fonte fica armazenado na nuvem, como uma espécie de Google Drive, o Git é o sistema que o GitHub e o desenvolvedor, na sua máquina, usam para gerenciar o código.

O valor do Git para desenvolvimento de software é poder ter um histórico das mudanças e reverter as mesmas quando necessário, seja apenas uma mudança ou um pacote delas, que geralmente seria uma versão. Desenvolvimento de software trabalha com versões, e a cada nova versão podem ocorrer erros, e muitas vezes as versões antigas continuam sendo usadas por usuários e os desenvolvedores precisam prestar suporte para ela. O Git ajuda a gerenciar tudo isso.

Para usar o Git você precisa instalar ele na sua maquina: https://git-scm.com/downloads

O Git trata os projetos como repositórios, que basicamente seria uma pasta onde você pode colocar seus arquivos (qualquer tipo). Por exemplo, o conteúdo da aula que você baixou (https://github.com/migradev/front-end) é um repositório.

Com o git instalado, para criar um repositório basta acessar uma pasta via terminal (Prompt de comando) e executar o comando init:

1 - cd <caminho da pasta>
2 - git init

Agora a pasta se tornou um repositório, mas ele ainda não está conectado a um repositório no GitHub, ele está apenas na sua máquina, para isso você precisa adicionar a url do repositório remoto (o repositório na nuvem), da seguinte forma:

git remote add <nome do repositório remoto> <url>

Nesse caso o nome será "origin", que é o nome padrão para adicionar o repositório remoto principal. Note que um repositório local pode ter N remotos.

git remote add origin <url>

Com isso o seu repositório local e o remoto estão conectados, agora basta fazer uma mudança e fazer o upload dela para remoto. No Git uma mudança se chama commit, um

commit pode ter mudanças em vários arquivos, e é obrigatório que ele tenha uma mensagem descrevendo a mudança, que você deve escrever.

Após mudar os arquivos desejados você deve adicioná-los para o próximo commit:

git add <caminho do arquivo>

Geralmente queremos adicionar todos os arquivos ao mesmo tempo, então um jeito mais fácil seria passar apenas um ponto no lugar do caminho exato, dessa forma ele vai adicionar todos os arquivos da pasta que o seu terminal está acessando no momento, se você estiver na pasta raiz do repositório ele vai adicionar todas as suas mudanças:

git add .

Com as mudanças adicionadas você pode fazer o commit:

git commit -m "descrição da mudança"

Sempre descreva a mudança em si, o que ela muda no projeto e não o que você teve que fazer, descreva de forma que alguém consiga entender sem precisar saber todo o histórico, mas ao mesmo de forma resumida, por exemplo:

Errado: Mudei o arquivo compra.html para consertar o erro do botão
Errado: Consertando o erro do botão da home
Correto: Consertando o erro do botão 'Saiba mais' da home que não aparecia quando o usuário estava logado

Com esse commit pronto você pode fazer mais commits ou mandar o que você tem para o repositório remoto, da seguinte forma:

git push

Ao fazer isso você estaria mudando diretamente o código fonte salvo no repositório remoto, e na maioria das vezes não queremos fazer isso, pois esse código ainda precisa ser testado ou ainda não está 100% pronto. Para resolver esse problema o Git tem um recurso chamado branches, uma branch seria como se fosse uma versão alternativa do código em que você pode mudar a vontade e salvar no repositório remoto sem alterar o código original e quando aquelas alterações estiverem prontas e testadas você pode unificar a sua branch com a branch principal, aquela onde código fonte original está, geralmente chamada de master ou main.

Para criar uma branch primeiro você precisa escolher a branch a qual você quer copiar para criar a sua, para isso você precisa “mudar” para a branch de sua escolha. O “mudar de branch“ no Git significa que o git irá transformar todo o conteúdo da pasta onde fica o seu repositório para o conteúdo da branch que você está mudando e todos os comandos executados a partir de então serão nessa branch:

git checkout <nome da branch>

Após trocar de branch é muito importante que você puxe as mudanças mais recentes dela antes de fazer qualquer coisa, para isso execute o comando:

git pull

Digamos que você queira fazer uma branch a partir da master:

1 - git checkout master
2 - git pull *
3 - git checkout -b minha-branch

*** Nunca se esqueça de puxar as mudanças mais recentes antes**

A opção “-b” indica que você quer criar uma branch nova e em seguida mudar para ela, ao invés de mudar para uma já existente. Note que com esse comando ele cria a branch apenas no seu repositório local e não no remoto:

git checkout -b <nome da nova branch a ser criada>

Com a branch feita você pode seguir o fluxo que vimos acima para realizar commits normalmente, mas a primeira vez que um push for ser realizado em uma branch que existe apenas no seu local ela irá precisar de um comando diferente:

git push -u origin minha-branch

O “-u” diz qual branch do remoto corresponde a sua local, o primeiro parâmetro é o nome do remoto, como falamos anteriormente o nome padrão para o repositório remoto principal é origin, dessa forma ela atribui a branch local **“**minha-branch**”** para uma no remoto de mesmo nome, caso não exista ele cria automaticamente:

git push -u <nome do remoto> <nome da branch remota (existente ou não)>

Com isso você tem a sua branch salva no remoto e pode continuar trabalhando nela normalmente. Uma branch geralmente (e é muito recomendado) é criada com o intuito de realizar 1 mudança ou 1 conserto de 1 bug (erro), quando o propósito dela foi cumprido é a hora de unificar ela com a branch master e abrir outra para as próximas tarefas. Para unificar branches se utiliza um comando chamado “merge”, por isso é muito comum desenvolvedores se referirem a isso como “mergear”, para mergear duas branches você precisa estar na branch de destino, aquela que irá receber a mudança e executar o comando apontando para a branch origem, a que irá realizar a mudança:

git merge <nome da branch com a mudança>

Sendo assim seria comum você trocar para a branch master e em seguida mergear com a sua:

1 - git checkout master
2 - git pull *
3 - git merge minha-branch

*** Nunca se esqueça de puxar as mudanças mais recentes antes**

Isso alteraria apenas a branch master no seu local, após o merge ser realizado você precisa mandar as mudanças para o remoto:

git push

Para trabalhar em um repositório já existente, por exemplo o que usamos para as aulas: https://github.com/migradev/front-end, basta escolher uma pasta para ele e executar o comando:

git clone <url do repositório>.git

Por exemplo com o repositório da aula seria:

git clone https://github.com/migradev/front-end.git

Uma pasta chamada “front-end” será criada dentro da pasta que você escolheu e ela será seu repositório local conectado ao remoto, daí em diante você poderia realizar qualquer um dos comandos que comentamos anteriormente e modificar o remoto sem problemas. Para se manter atualizado com o conteúdo das aulas basta executar o **git pull** na branch **main** após cada aula.