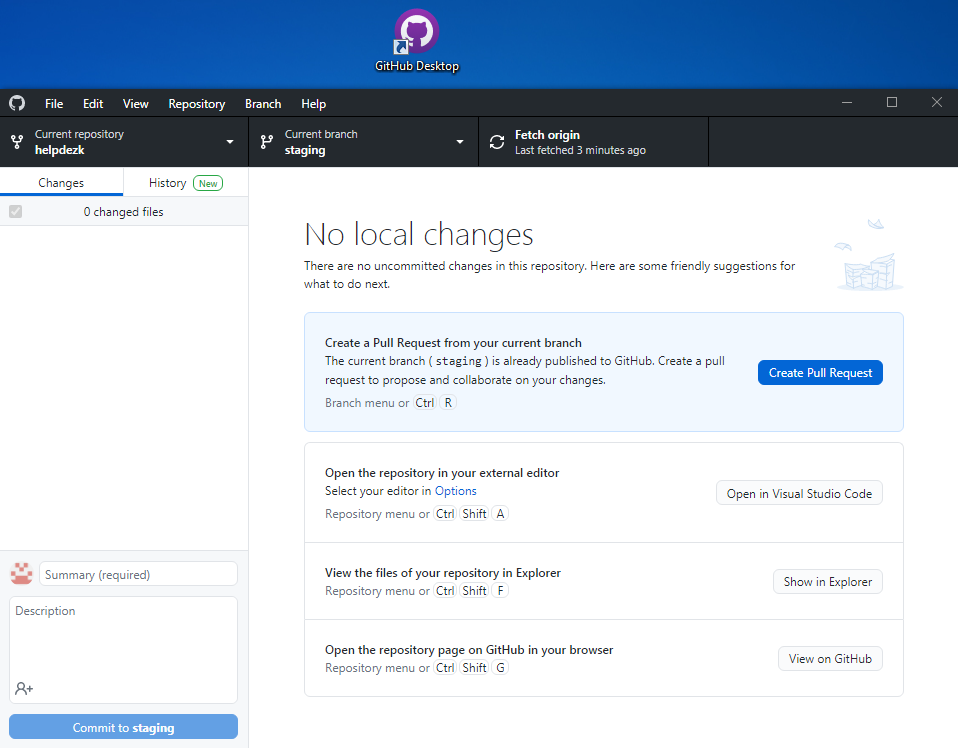
**1 - INTRODUÇÃO: INSTALANDO O GITHUB**

Embora possa ser utilizado a partir da linha de comando, neste guia será abordado o seu uso a partir da sua versão desktop, que disponibiliza uma interface gráfica, e permite que o trabalho com a ferramenta se torne mais eficiente.

Portanto, como primeiro passo, se não possuir o programa instalado em sua máquina, instale-o clicando neste link: <https://desktop.github.com/>

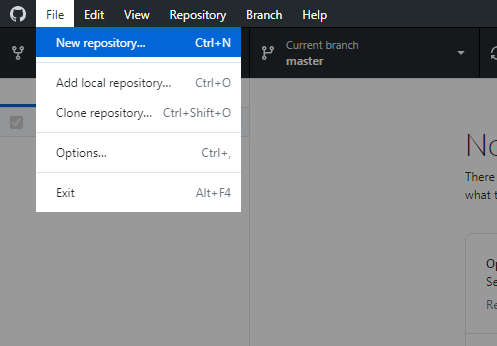


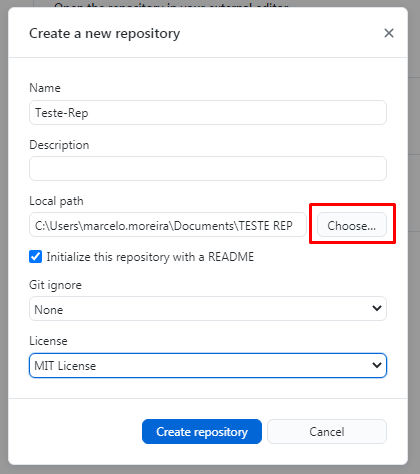
# **2 - WORKFLOW GITHUB**

**2.1 - Sobre a criação de repositórios**

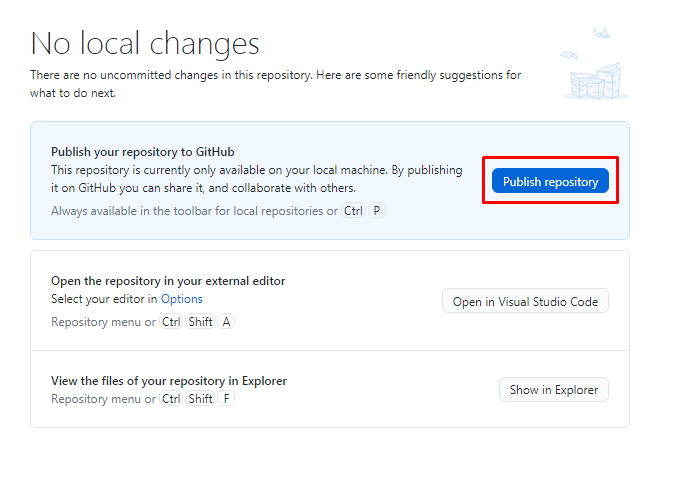
Tendo o programa em mãos, você poderia, agora, criar um repositório a partir de sua máquina local, criando uma pasta que será vinculada a um repositório no Github.

Por exemplo, foi criada uma pasta de nome “TESTE REP”, em “Documentos”, e este caminho - o da pasta criada - foi selecionado para a criação do repositório:

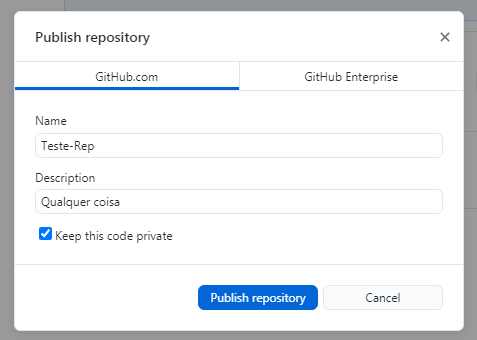




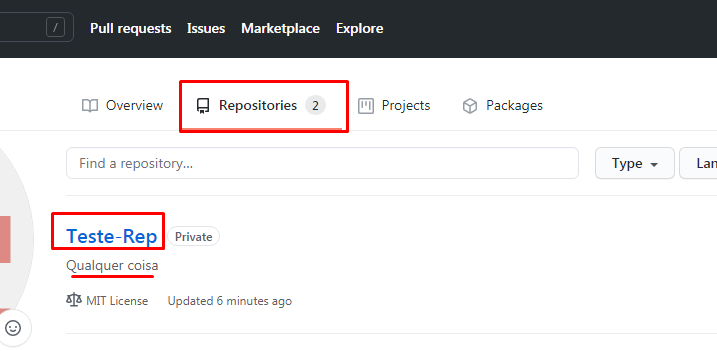
Assim, o repositório terá sido criado, mas ainda deverá ser publicado, para que seja acessível na sua página do Github.



Clicando no botão para publicar o repositório, uma janela modal irá surgir, e nela existirá um formulário para que você informe um nome para o repositório, e uma descrição, sendo ambos dados que serão utilizados para caracterizar este repositório na versão Web.



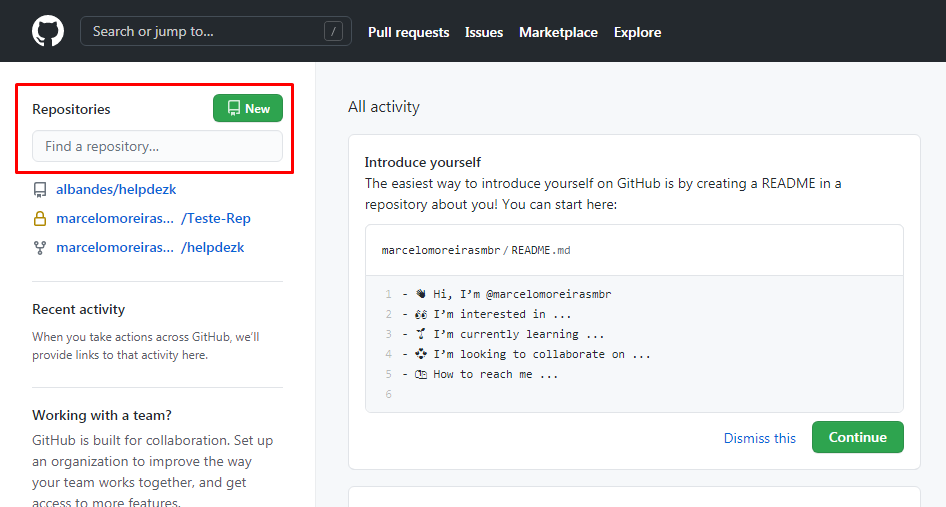
Na versão web, na aba “Repositories”, você pode conferir o repositório publicado, e com as definições escolhidas:



**2.2 - Selecionando um repositório de terceiros**

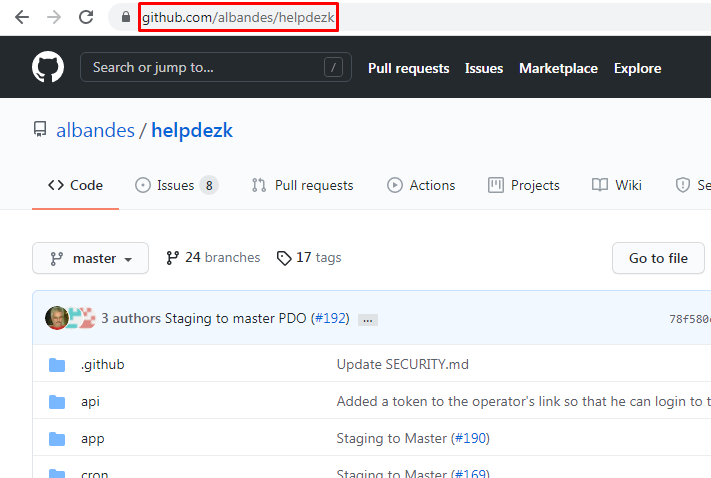
Além de ter seus próprios repositórios, de sua inteira autoria, você pode trabalhar com repositórios criados por outros usuários da plataforma.

Para isso, vá na página inicial da sua conta, clicando no ícone do Github, existente no canto superior esquerdo, e encontre o input de pesquisa cujo placeholder é “Find a Repository”, ou “Encontre um Repositório”.



Pois bem: neste campo você pode pesquisar por repositórios utilizando palavras-chave, como pelo nome do usuário, ou do repositório procurado. Ademais, se tiver, também pode procurar por meio da URL do repositório.

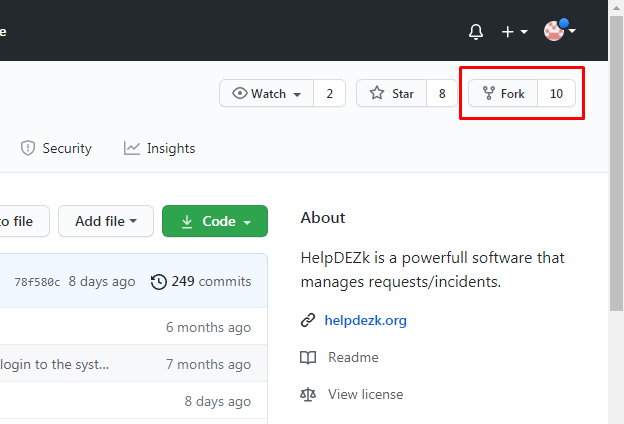
Considere de antemão a URL <https://github.com/albandes/helpdezk>, onde estão publicados os arquivos do sistema Helpdezk - o qual você irá trabalhar.



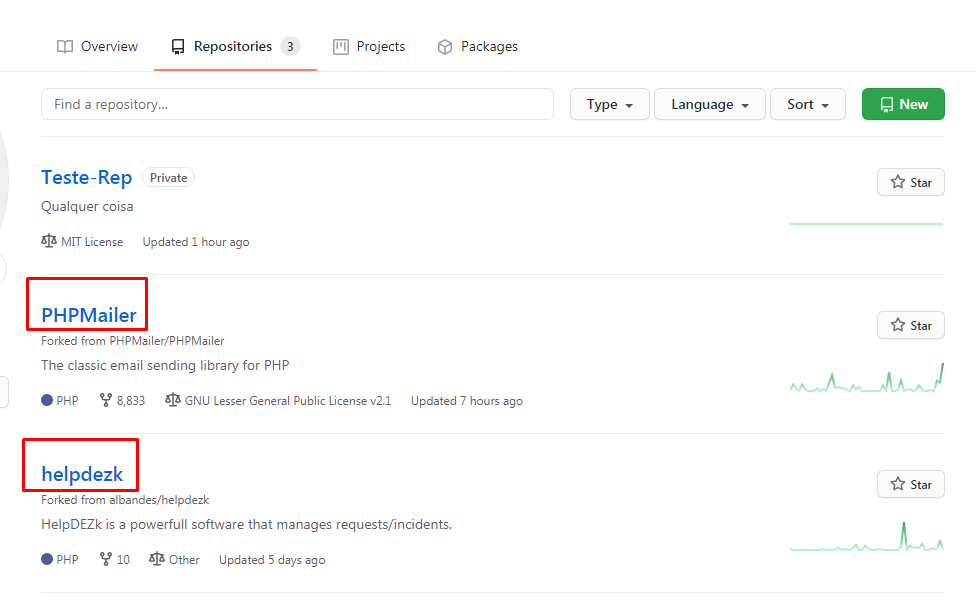
## **2.3 - Fork do repositório**

Tendo encontrado um repositório externo, pertencente a outro usuário, e querendo você ter ele para si, na sua máquina, pode realizar um “fork” do mesmo.

Criar um “fork” de um repositório significa apenas produzir uma cópia pessoal do projeto de outra pessoa. Além disso, o processo estabelece uma ponte entre o repositório original, e a cópia pessoal, permitindo que modificações sejam realizadas na cópia e inseridas no projeto original - mas claro, sob autorização do autor do projeto.



Os forks, como pode ver abaixo, surgem na página dos seus repositórios:



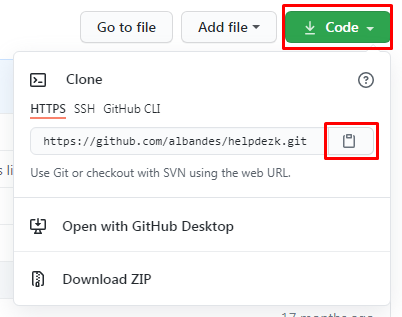
A requisição para a assimilação no projeto original das modificações realizadas em cópias, chama-se “Pull Request”. Ou seja, aqueles que desejam contribuir para um sistema, trabalham com suas cópias, e realizam requisições para que as alterações sejam consideradas e inseridas no projeto original.

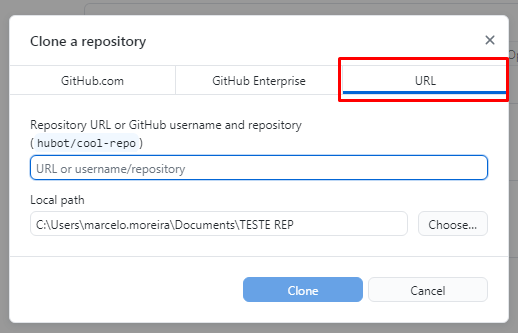
A tradução de “Pull Request” é “solicitação de puxar”, ou seja, é uma requisição para que um terceiro realize a ação de “puxar” as alterações para si. Desta forma, se você realiza modificações em um sistema de terceiros e deseja que sejam implementadas, ao invés de um “Push”, como deve realizar para “empurrar” alterações em um repositório próprio, solicita que os autores “puxem” seu trabalho, e por isso dá-se o nome “Pull Request”.

## **2.4 - Clonar o repositório**

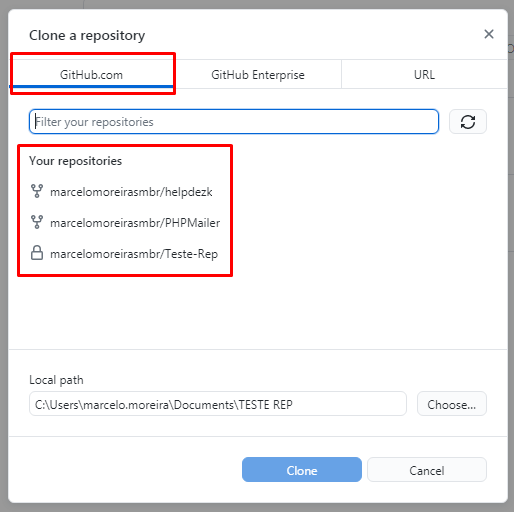
Ao criar um “fork” de um repositório, você cria para si uma cópia do mesmo, acessível a partir de sua aba “Repositories”, como mostrado anteriormente, mas ainda não uma acessível ou existente localmente. Para isso, isto é, para ter a cópia do repositório em sua própria máquina, você deve cloná-lo.

Para realizar este procedimento existem algumas vias possíveis, como (i) copiando a URL e a inserindo na sessão de clonagem da versão desktop.

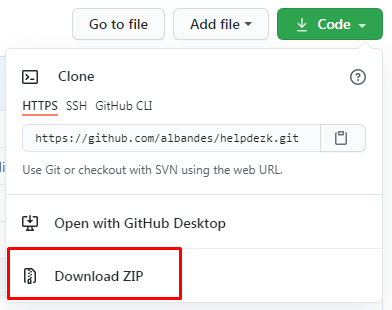




Ou, (ii) também na sessão de clonagem da versão desktop, clicando no repositório disponível, e que de fato só estará se existir entre os seus repositórios - próprios e publicados, ou forks.



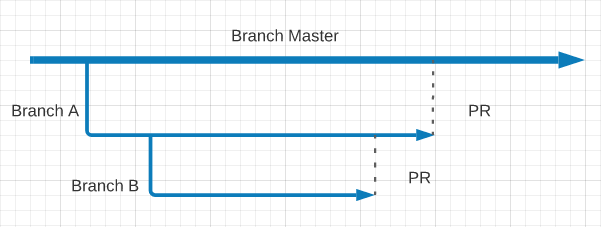
Ou, (iii) você pode realizar o download do repositório em formato .ZIP.



## **2.5 - Sobre Branches**

Para que um número maior de pessoas, por exemplo, trabalhe em um mesmo projeto, deve ser possível segmentar o projeto em partes para que cada entidade atue isoladamente, e “suba” alterações de forma que não haja conflitos com as realizadas por outras. É como uma solução para a necessidade de “desenvolvimentos paralelos” que existem os branches.

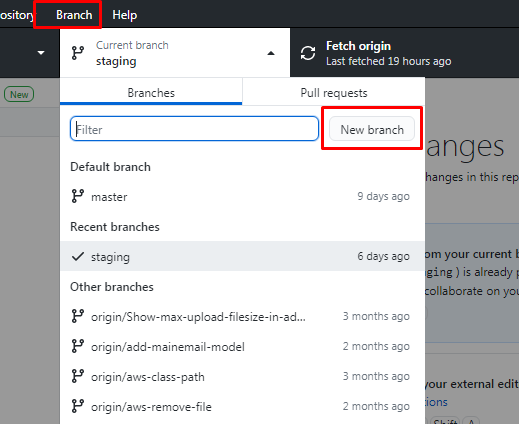
Pois bem: uma branch é uma ramificação do projeto, e um projeto com branches é um com ramificações de desenvolvimento.



O que ocorre então, na prática, é que se um projeto possui N desenvolvedores, e tendo cada um uma especialização, é interessante que cada um trabalhe no projeto em uma “dimensão própria”. Desta forma, se você desenvolve na branch A, e o seu colega na B, ambos trabalham no mesmo projeto, mas “sobem” as alterações não para o mesmo fluxo de desenvolvimento, e sim para seus próprios, e que ficam disponíveis para os outros desenvolvedores.

Quando dois fluxos de desenvolvimento se encontram, e se juntam, o que é realizado por meio de um “Pull Request” (PR), o processo, quando efetivado, é denominado de “merge”. Por exemplo, se existe uma branch para o desenvolvimento back-end de um site, e outra para o front-end, quando se juntam em uma só, na branch master, que é a principal, diz-se que foi realizado um merge.

Para criar um branch, acesse a versão desktop do Github e clique na guia “Branch”, depois em “New branch”. Na imagem abaixo você pode conferir as etapas descritas, e ainda ver, servindo como um exemplo de um sistema com múltiplas ramificações de desenvolvimento, as branches do sistema Helpdezk.



Para ficar atento às boas práticas, saiba que é recomendável que o nome do Branch esteja relacionado com a sua finalidade.

## **2.6 - Efetue as alterações necessárias e faça um Commit**

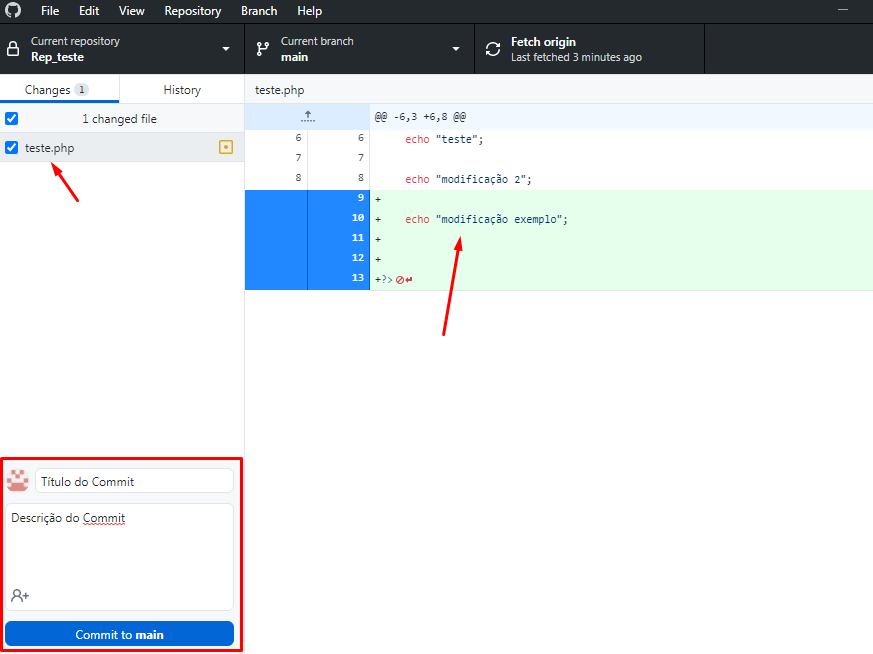
Após realizar alterações em um repositório, e pretendendo fazê-las “subir”, se for um repositório ou fluxo de desenvolvimento próprios, deve realizar apenas um “Push”, e se necessitar de autorização, um “Pull Request”.

No entanto, faltam detalhes na descrição sequencial destes processos. Um dos segmentos mais importantes do Github é o “Commit”, que é justamente um “pacote” das próprias alterações, e dentro de um “panorama histórico” dos estados dos arquivos. Ou seja, as alterações são agregadas em pacotes, e de forma que possa ser compreendido o que foi alterado e porquê.

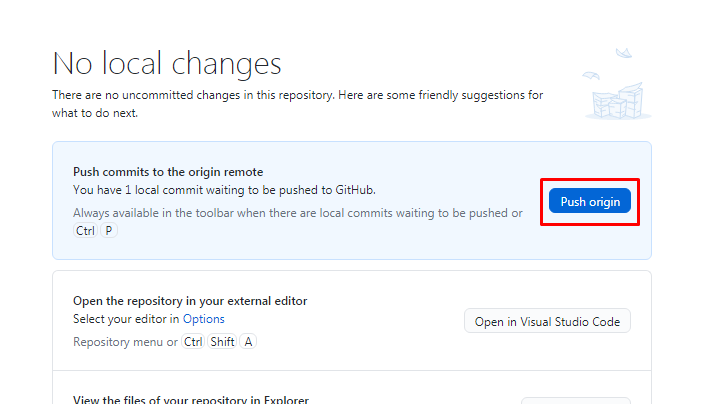
Um “commit” possui sempre uma nomeação, uma descrição, o nome do autor, e as modificações realizadas em uma espécie de “snapshot”. Os arquivos não são enviados apenas em sua nova forma, mas com as alterações expostas em comparação com o seu estado anterior.

Veja a imagem abaixo

. A linha verde contém o que foi adicionado, se em comparação com o estado anterior do arquivo “teste.php”, e no canto inferior esquerdo devem ser informados o título e descrição do commit.



Clicando em “commit to main”, este pacote de alterações é confirmado para envio ao repositório em questão, e na sua branch “main”. Em seguida, para realmente realizar o envio, deve ser clicado o botão “Push origin”, para que assim o pacote seja “empurrado para a origem”, isto é, para o repositório em que o arquivo modificado existe.



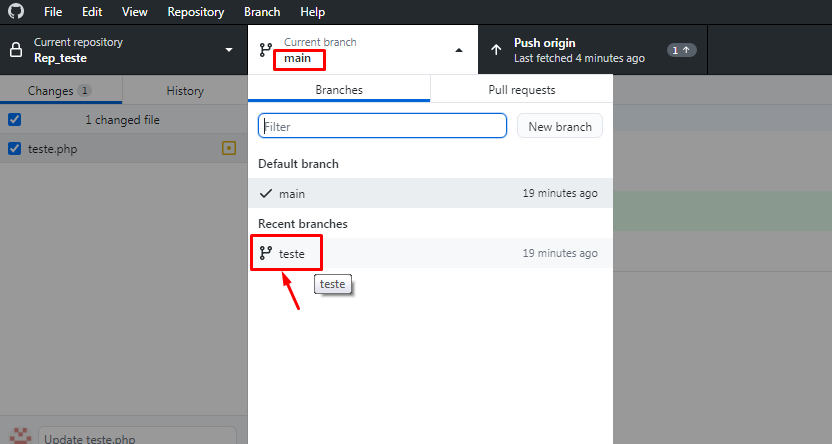
Em alguns casos você pode encontrar o termo “commit” flexionado como verbo, “do a commit”, “commitar”, que são formas comuns de comunicar que um commit será enviado.

## **2.7 - Faça um Push das alterações para o GitHub**

Desta forma, entendendo que as modificações são enviadas em pacotes, cujo nome é “Commit”, tanto o “Pull Request” quanto o “Push” podem ser entendidos como procedimentos para enviar commits. Além disso, os commits são enviados em branches, isto é, em fluxos ou ramificações de desenvolvimento.

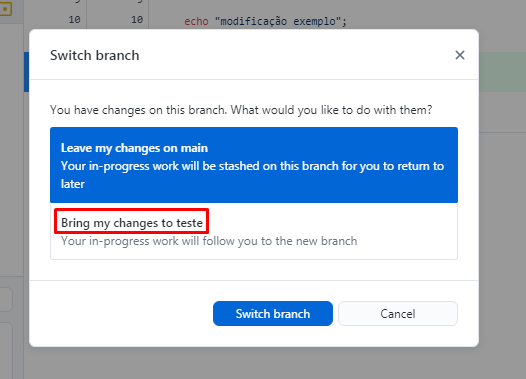
Assim, quanto ao “Push” em específico, e agora servindo de reforço para o que ficou claro, este consiste, como você já sabe, para empurrar alterações para um repositório, e em uma branch.

Primeiro, então, você realizará as modificações estando em uma branche, ou pode até mesmo trazer as alterações capturadas em uma, para outra. Veja na imagem abaixo:



O arquivo “teste.php” foi modificado, e a branche que estava definida era a “main”. Neste caso poderia realizar um commit, e um push origin, que as modificações “subiriam” para o repositório remoto, na branche main.

Mas, é possível, e para fins de exemplo, trocar para outra branche sem perder a captura das alterações realizadas no arquivo.



Agora, se definir o commit, e clicar em “Push origin”, o pacote será enviado para o repositório na branche “teste”. Ademais, fazendo isto, as alterações só existirão no arquivo, e poderão ser consultadas, se escolhida a branche “teste” no repositório remoto.

Veja o código do arquivo na branche “teste”:



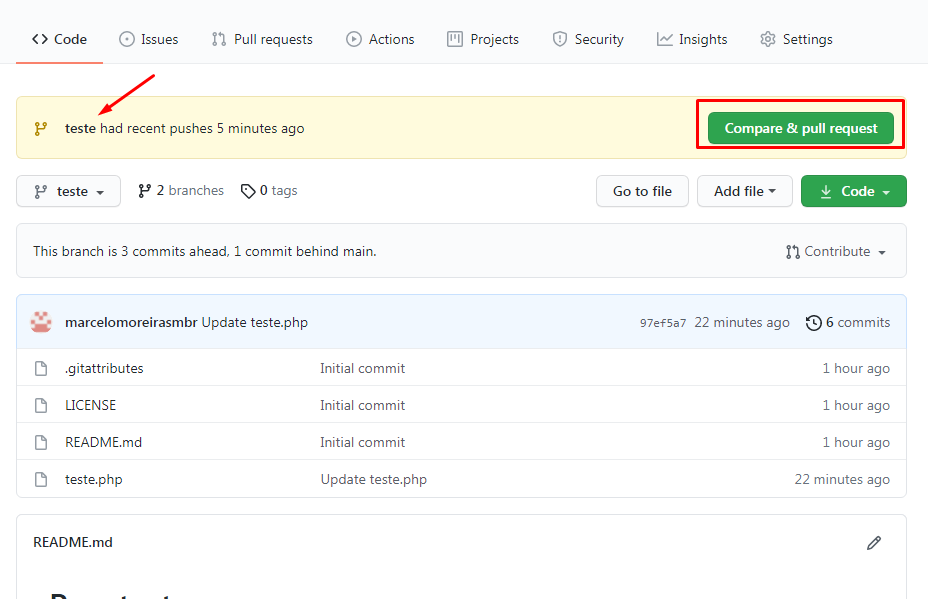
Agora veja o código na branche “main”:



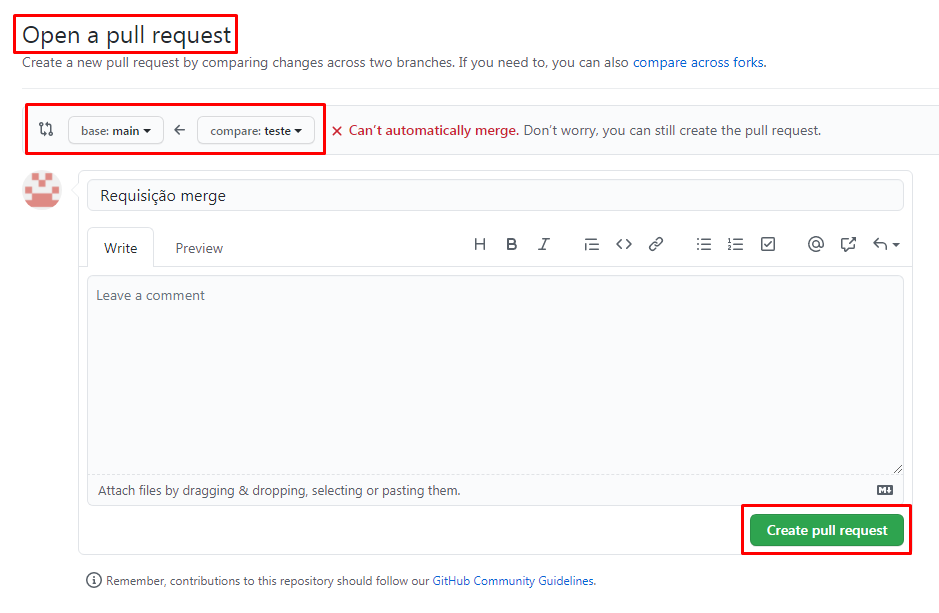
São diferentes porque as alterações foram “empurradas” para a branche “teste”, e não para a “main”.

## **2.7 - Envie suas alterações para serem revisadas**

Se você for para o seu repositório no GitHub, verá um botão “Compare & pull Request”.



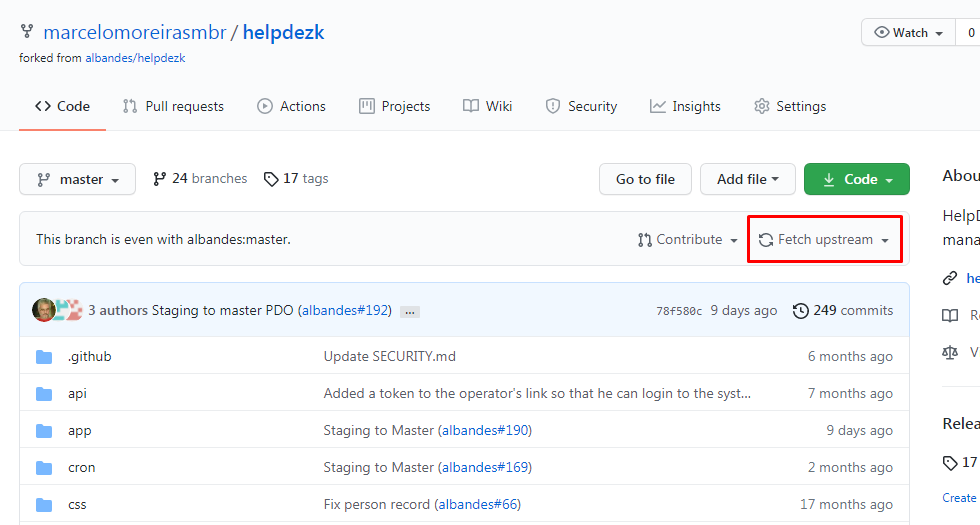
Clicando neste botão, você pode solicitar um “Pull Request” para realizar um merge entre uma branch e outra - neste caso, entre “teste” e “main”, e da primeira para a segunda. Desta forma, as duas serão comparadas, analisadas pelos desenvolvedores que administram a outra branch, e estes decidirão se aceitarão ou não o merge das branches.



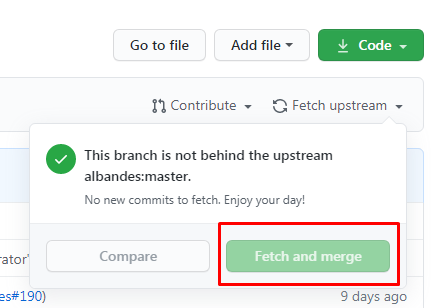
## **2.8 - Atualize seu Fork**

É importante que você realize sincronizações constantes do seu fork com o projeto original, para que não fique desatualizado em relação às atualizações mais recentes da branch principal do projeto, isto é, aquela do fluxo de desenvolvimento “principal” do sistema, em que existem os recursos que já foram analisados, testados e disponibilizados para uso.

Para isso, primeiro acesse o seu fork, clique em “Fetch Upstream” e em seguida em “Fetch and Merge”, se o botão não estiver desabilitado; ele estará se o fork estiver atualizado. Além disso, tenha em mente que esta ação é denominada de “Git Fetch”.



Neste caso, o fork está atualizado, porquanto além do botão estar desabilitado, está escrito “This branch is not behind the upstream albandes:master”, o que você pode entender como “Esta branch não está desatualizada em relação a branch master (albandes:master) do projeto”.



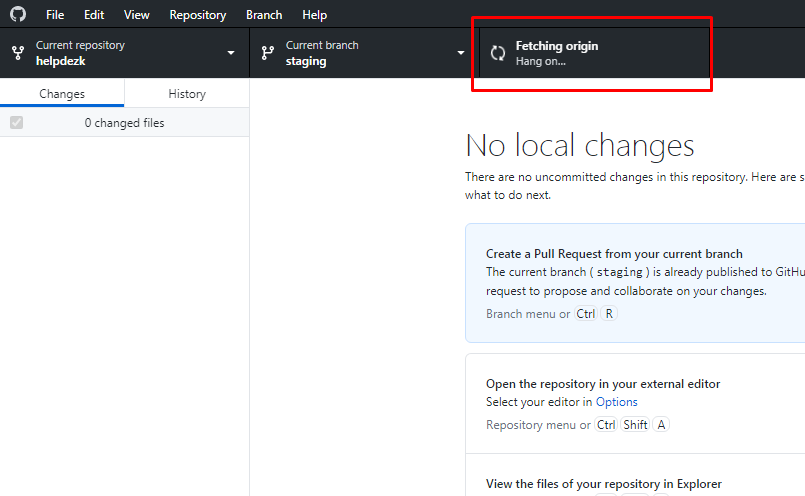
Uma via alternativa ao “Fetch and merge”, é a opção “Compare”. A diferença entre ambos, em termos gerais, é esta: o que é realizado manualmente no “compare”, no “fetch and merge” é executado automaticamente.

Clicando na opção “compare”, uma interface é apresentada, onde pode ser realizada a comparação analítica entre o projeto original, e o seu fork, e em seguida um PR deve ser realizado para que os dois sejam sincronizados. Já, em contraste, no “fetch and merge” estes passos são realizados de forma automática.

## **2.9 - Atualize seu repositório local**

Após, por exemplo, atualizar o seu repositório remoto, como o seu fork, você deve atualizar a sua versão local, isto é, aquela que existe na sua máquina.

Assim, o que deve fazer é realizar um “Fetch” na versão desktop do Github, para que as atualizações mais recentes sejam trazidas para o seu repositório local.



Caso houver atualizações, e neste caso será utilizado o repositório de teste para fins de demonstração, aparecerá a opção “Pull Origin”, isto é, “Puxar da origem”, ou “puxar as atualizações do repositório remoto que não existem no local, na máquina”.

