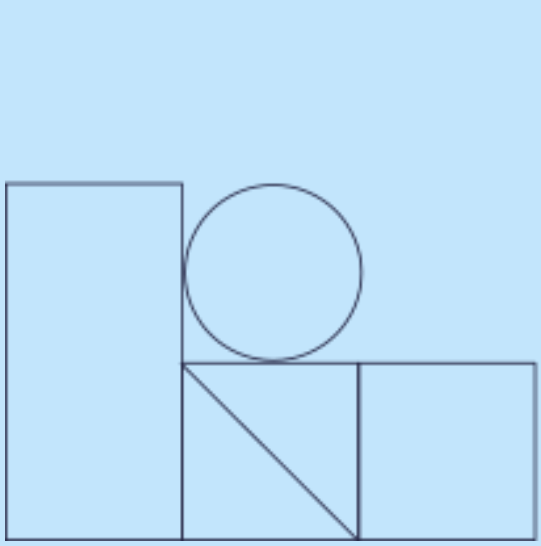
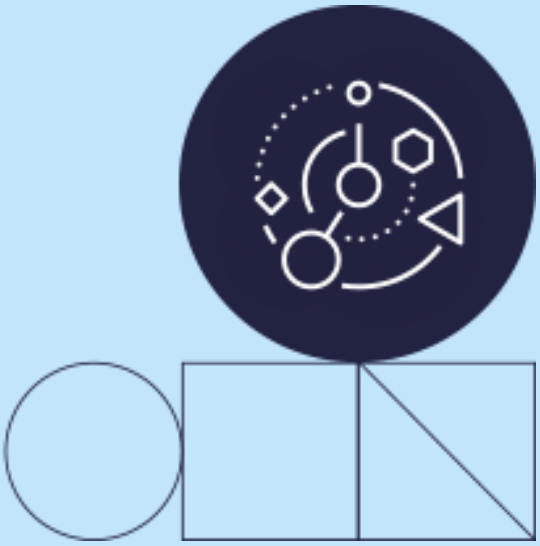
IBM SkillsBuild

IBM

Git e Github

Push, pull, fork, clone





Índice

Push

O comando *git push* é usado para carregar conteúdo do repositório local para um repositório remoto. Empurrar é a maneira de transferir os compromissos de seu repositório local para um repositório remoto.

O *git push* é usado principalmente para publicar e carregar mudanças locais em um repositório central. Após modificar o repositório local, um compromisso é executado para compartilhar as modificações com outros membros da equipe.

Usando o git push

A sintaxe para usar o comando push é:

git push <remote> <branch>

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)
$ git push

Everything up-to-date
```

Enviar o ramo especificado para um , juntamente com todos os *commits* e objetos internos necessários. Isto cria uma filial local no repositório alvo. Para evitar que os commits sejam sobrescritos, Git não permitirá que os commits sejam empurrados quando o resultado no repositório alvo for um conflito potencial com um arquivo existente.

git push <remote> --force

É o mesmo que o comando anterior, mas obriga o envio mesmo que o resultado seja uma fusão com possíveis conflitos.

Não devemos usar --force a menos que estejamos absolutamente seguros do que estamos fazendo.

Git nos impede de sobrescrever o histórico do repositório central, recusando-se a apresentar pedidos quando o resultado é uma fusão com conflitos potenciais. Portanto, se a história remota for diferente de nossa história, teremos que puxar uma filial remota e fundi-la com a filial local e depois tentar comprometê-la novamente.

Este processo é similar à maneira como o SVN sincroniza com o repositório central usando o comando de atualização svn antes de comprometer um conjunto de mudanças.

A marca --force anula este comportamento e faz com que a filial do repositório remoto coincida com a filial local, removendo assim todas as mudanças no repositório remoto que ocorreram desde o último compromisso.

A única situação em que podemos precisar forçar o compromisso é quando nos damos conta de que os compromissos que acabamos de compartilhar não são totalmente corretos e os corrigimos através de mudanças manuais ou usando o comando:

git commit --amend

Entretanto, antes de utilizar a opção --force, devemos ter absoluta certeza de que nenhum de nossos colegas de equipe incorporou as mudanças desses compromissos.

git push <remote> --all

Enviar todas as nossas filiais locais para uma filial remota especificada.

git push <remote> --tags

As etiquetas não são enviadas automaticamente quando enviamos uma filial ou utilizamos a opção --all. A marca --tags envia todas as tags locais para o repositório remoto.

Pull

Git pull é um comando Git usado para atualizar a versão local de um repositório a partir de um repositório remoto.

Poderíamos dizer que *git pull* é o oposto de *git push*, ou seja, com o comando push carregamos as mudanças de nosso repositório local para a nuvem e com o comando pull descarregamos os arquivos da nuvem para nosso repositório local.

Este é um dos quatro comandos que solicitam interação em rede por Git. Por padrão, o *git pull* faz duas coisas.

Ele atualiza o ramo de trabalho atual, ou seja, o ramo em que estamos trabalhando atualmente.

Ele atualiza as referências das filiais remotas para todas as outras filiais.

git pull recupera (*git fetch*)) os novos compromissos e funde-os *(git merge*) em sua filial local.

A sintaxe para este comando é a seguinte:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

 # Formato General

$ git pull OPCIONES REPOSITORIO REFSPEC


 # Pull de una rama específica

$ git pull NOMBRE-REMOTo NOMBRE-RAMA
```

Onde:

OPCIONES são as opções de comando, tais como como --quiet ou --verbose.

REPOSITORIO é o URL de nosso repositório remoto. Por exemplo:

https://github.com/miRepositorio/repo.git

REFSPEC especifica quais referências a serem recuperadas e quais referências locais a serem atualizadas.

NOMBRE-REMOTO é o nome do nosso repositório remoto.

Por exemplo: origin.

NOMBRE-RAMA é o nome da nossa filial.

Por exemplo: develop.

Nota

Se tivermos mudanças não comprometidas, a parte de fusão do comando de puxar o gatilho falhará e nossa filial local permanecerá intacta.

Portanto, devemos sempre comprometer nossas mudanças em uma filial antes de atualizar novos compromissos a partir de um repositório remoto.

Usando o git pull

Usaremos o git pull para atualizar um repositório local do repositório remoto correspondente. Por exemplo: Enquanto trabalhamos localmente no main, vamos executar o git pull para atualizar a cópia local do main e atualizar as outras filiais remotas.

No entanto, há algumas coisas a serem levadas em conta para que esse exemplo seja verdadeiro:

O repositório local tem um repositório remoto vinculado.

- Confirme isso executando *git remote -v*
- Se houver vários controles remotos, *git pull* pode não ser informação suficiente. Talvez precisemos entrar *git pull origin* ou *git pull upstream*.

Se a filial para a qual nos mudamos tiver uma filial de rastreamento remoto correspondente, devemos verificar isso executando *git status*.

Se não houver um ramo de rastreamento remoto, Git não sabe de onde extrair as informações.

Fork

O "*fork*" é uma das operações comuns com o trabalho em Git e GitHub. Serve basicamente para **criar uma cópia de um repositório em nossa conta de usuário.**

Este repositório copiado será basicamente um clone do repositório do qual o *fork* é feito, mas a partir desse momento o *fork* viverá em um espaço diferente e será capaz de evoluir de uma maneira diferente**.**

O *fork* pode ser entendido como um ramo externo de um repositório, colocando esse ramo em um novo repositório controlado por outros usuários.

Uma vez que o garfo estiver pronto, haverá dois repositórios diferentes. Inicialmente um era uma cópia exata do outro, mas como o desenvolvimento e a liberação de mudanças em um ou outro repositório, os dois repositórios podem tender a ser tão diferentes quanto cada uma das equipes de desenvolvimento que os mantêm querem.

Portanto, um fork é uma cópia de um repositório, mas criado em nossa própria conta GitHub, onde temos permissões de escrita. Portanto, se pretendemos baixar um repositório do GitHub para fazer alterações nele e esse repositório não nos pertence, a coisa mais normal a fazer é criar primeiro um fork e depois clonar nosso próprio fork localmente.

Operação de Pull request

O fork é um passo inicial para se envolver nos projetos de outros que publicam código no GitHub. **Qualquer contribuição começa fazendo um fork do repositório** com o qual queremos contribuir.

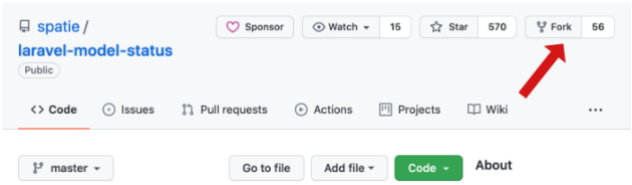
Uma vez criado o *fork*, podemos fazer alterações e solicitar a solicitação de pull request através da página do GitHub, que é basicamente uma solicitação para que nosso código seja fundido com o código no repositório onde contribuímos.

Esta operação Pull Request é um pouco mais complicada do que um simples fork e o que resumimos nas linhas anteriores.

Como fazer um fork no GitHub

Agora que sabemos o que é um *fork* e quando podemos precisar dele, vamos explicar o processo de fabricação de um fork no GitHub.

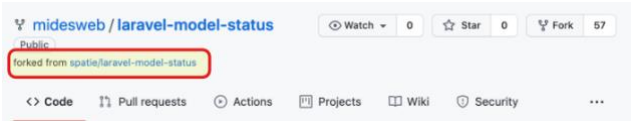
Basta acessar o repositório que queremos fazer fork, e clicar no botão "Fork".



Teremos que entrar no GitHub com nossa conta para criar um *fork*.

Se você estiver em uma organização, uma imagem pode aparecer para você selecionar onde deseja criar o fork, em sua conta pessoal ou em uma de nossas organizações.

Uma vez que o fork estiver pronto, um novo repositório será criado em nossa conta, com uma cópia do reporte original. A página do GitHub também mostrará que este repositório é um garfo de outro repositório, o original.



Agora podemos perfeitamente fazer um clone deste *fork* localmente, para baixá-lo para nosso ambiente de desenvolvimento.

Poderemos fazer mudanças no projeto e depois fazer commit, e depois fazer o upload das mudanças para o GitHub.

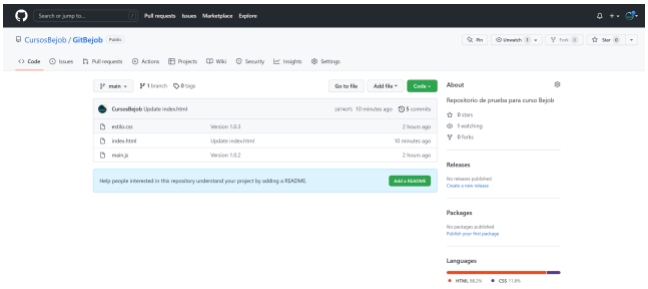
Como estamos carregando as modificações em nosso próprio repositório, já que o fork é feito por nossa conta, o GitHub nos permitirá publicar as modificações feitas..

Clone

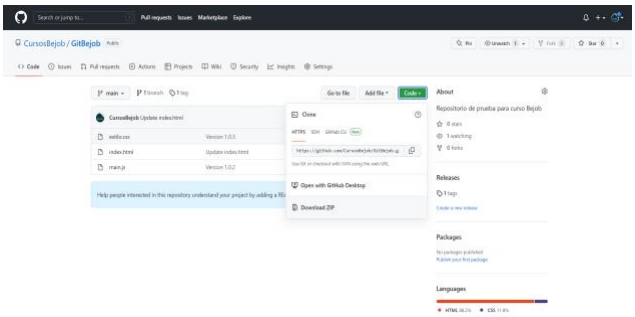
Ocasionalmente, podemos nos deparar com a desagradável surpresa de termos apagado acidentalmente nosso projeto local.

Se tivermos uma versão no GitHub, podemos recuperá-la de várias maneiras.

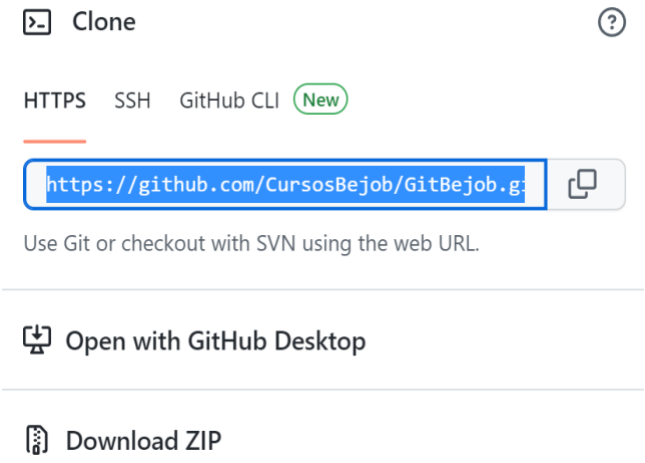
Podemos baixar uma versão compactada do menu *Tags*.



Você também pode baixá-lo da opção "Download zip" no menu "<>código":



Ou podemos copiar a URL que aparece logo acima da opção anterior e cloná-la.



Para fazer isso, usaremos nosso console com o comando “git clone URL”:

```
miPC@miPC MINGW64 /g/WORKSPACE/011
GitHub/Proyecto GIT (master)

$ git clone
https://github.com/CursosBejob/GitBejob.
git


Cloning into 'GitBejob'...
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17),
done.
remote: Compressing objects: 100%
(11/11), done.
remote: Total 17 (delta 2), reused 13
(delta 1), pack-reused 0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (2/2), done.
```

Diferença entre *fork* e *clone*

Se fizermos um clone normal de um repositório, o espaço GitHub desse clone permanecerá associado ao repositório que clonamos. Portanto, se fizermos mudanças no clone e quisermos publicá-las no GitHub, provavelmente não conseguiremos carregá-las.

Obviamente, se clonarmos um repositório que fosse nosso, podemos fazer mudanças localmente e carregá-las para o GitHub sempre que quisermos. Mas se o repositório pertencesse a outro desenvolvedor e não tivéssemos permissões de escrita sobre ele, então não seríamos capazes de carregar as mudanças, porque o GitHub não nos permitirá fazê-lo. É aqui que precisamos de um garfo.