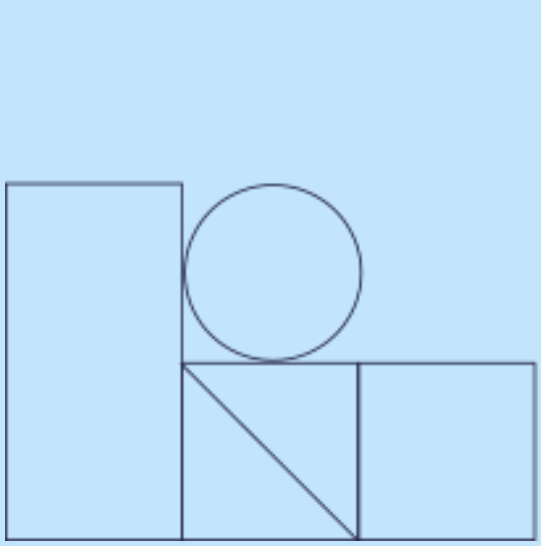
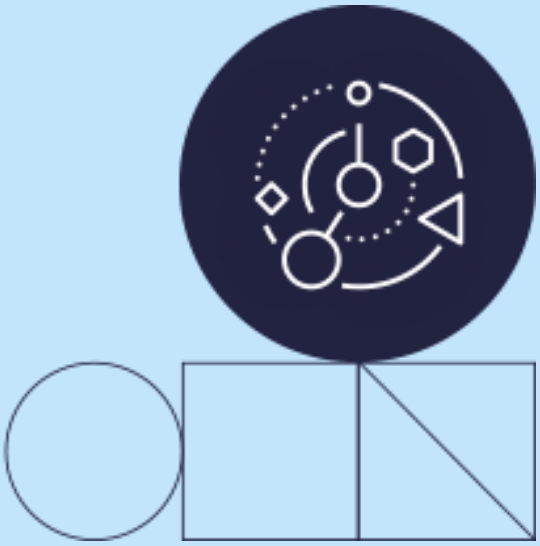
IBM SkillsBuild

IBM

Git y Github

Push, pull, fork, clone





Índice

Push

El comando *git push* se usa para cargar contenido del repositorio local a un repositorio remoto. El envío es la forma de transferir confirmaciones desde tu repositorio local a un repositorio remoto.

git push se usa sobre todo para publicar y cargar cambios locales a un repositorio central. Después de modificar el repositorio local, se ejecuta un envío para compartir las modificaciones con los demás miembros del equipo.

Uso de git push

La sintaxis para el uso del comando push es:

git push <remote> <branch>

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

Everything up-to-date
```

Envía la rama especificada a una , junto con todos los *commits* y objetos internos necesarios. De este modo se crea una rama local en el repositorio de destino. Para evitar que se sobrescriban los *commits*, Git no nos permitirá enviarlos cuando el resultado en el repositorio de destino sea un posible conflicto con un archivo ya existente.

git push <remote> --force

Es igual que el comando anterior, pero fuerza el envío incluso si el resultado es una fusión con posibles conflictos.

No debemos usar --force a menos de que tengamos absoluta certeza de lo que estamos haciendo.

Git evita que sobrescribamos el historial del repositorio central al negarse a enviar solicitudes cuando el resultado es una fusión con posibles conflictos. De este modo, si el historial remoto difiere de nuestro historial, tendremos que incorporar una rama remota y fusionarla con la local para después intentar enviarla de nuevo.

Este proceso es similar a la forma en la que SVN realiza la sincronización con el repositorio central mediante el comando *svn update* antes de confirmar un conjunto de cambios.

La marca --force anula este comportamiento y hace que la rama del repositorio remoto coincida con la local, y se eliminan así todos los cambios en el repositorio remoto que se hayan producido desde que realizamos la última incorporación de cambios.

La única situación en la que podríamos tener que forzar el envío es cuando nos damos cuenta de que las confirmaciones que acabamos de compartir no están del todo bien y las corregimos mediante cambios manuales o usando el comando:

git commit --amend

No obstante, antes de usar la opción --force, debemos tener la certeza absoluta de que ninguno de nuestros compañeros de equipo ha incorporado los cambios de esas confirmaciones.

git push <remote> --all

Envía todas nuestras ramas locales a una rama remota especificada.

git push <remote> --tags

Las etiquetas no se envían automáticamente cuando enviamos una rama o usamos la opción --all. La marca --tags envía todas las etiquetas locales al repositorio remoto.

Pull

Git pull es un comando de *Git* utilizado para actualizar la versión local de un repositorio desde otro remoto.

Podríamos decir que *git pull* es lo contrario de *git push*, es decir, con el comando *push* subimos los cambios de nuestro repositorio local a la nube y con *pull* descargamos los archivos desde la nube a nuestro repositorio local.

Es uno de los cuatro comandos que solicita interacción de red por *Git*.

Actualiza la rama de trabajo actual, es decir, la rama en la que estamos trabajando actualmente.

Esto significa que *git pull* recuperará los cambios desde el repositorio remoto y los integrará en la rama local actual.

La síntaxis de este comando es el siguiente:

```
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
```

Donde:

OPCIONES son las opciones de comandos, como *--quiet* o *--verbose*.

REPOSITORIO es la URL de nuestro repositorio remoto. Por ejemplo:

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

REFSPEC especifica qué referencias recuperar y qué referencias locales actualizar.

NOMBRE-REMOTO es el nombre de nuestro repositorio remoto.

Por ejemplo: origin.

NOMBRE-RAMA es el nombre de nuestra rama.

Por ejemplo: develop.

Nota

Si tenemos cambios no confirmados, la parte de fusión del comando *git pull* fallará y nuestra rama local quedará intacta.

Por lo tanto, siempre deberíamos confirmar nuestros cambios en una rama antes de actualizar nuevas confirmaciones de un repositorio remoto.

Uso de git pull

Usaremos *git pull* para actualizar un repositorio local del repositorio remoto correspondiente. Por ejemplo: Mientras trabajamos localmente en *main*, ejecutaremos *git pull* para actualizar la copia local de *main* y actualizar las otras ramas remotas.

Sin embargo, hay algunas cosas que hay que tener en cuenta para que ese ejemplo sea cierto:

El repositorio local tiene un repositorio remoto vinculado.

- Confirma esto ejecutando *git remote -v*
- Si existen múltiples remotos, *git pull* podría no ser suficiente información. Es posible que debamos ingresar *git pull origin* o *git pull upstream*.

Si la rama a la que nos hemos movido tiene una rama de seguimiento remoto correspondiente deberemos revisa esto ejecutando *git status*.

Si no hay una rama de seguimiento remota, *Git* no sabe de dónde extraer la información.

Fork

El "*fork*" es una de las operativas comunes con el trabajo en *Git* y *GitHub*. Básicamente sirve para **crear una copia de un repositorio en nuestra cuenta de usuario**.

Ese repositorio copiado será básicamente un clon del repositorio desde el que se hace el *fork*, pero a partir de ese momento **el *fork* vivirá en un espacio diferente y podrá evolucionar de manera distinta.**

El *fork* lo podemos entender como una *rama externa de un repositorio*, colocando esa rama en un nuevo repositorio controlado por otros usuarios.

Una vez hecho el *fork* existirán dos repositorios distintos. Inicialmente uno era copia exacta del otro, pero a medida que se vaya desarrollando y publicando cambios en uno u otro repo, ambos repositorios podrán tender a ser tan distintos como quieran cada uno de los equipos de desarrollo que los mantengan.

Así pues, un *fork* es una copia de un repositorio, pero **creado en nuestra propia cuenta de *GitHub*, donde sí que tenemos permisos de escritura**. Por tanto, si tenemos intención de bajarnos un repositorio de *GitHub* para hacer cambios en él y ese repositorio no nos pertenece, lo más normal es que creemos un *fork* primero y luego clonemos en local nuestro propio *fork*.

Operativa de Pull request

El *fork* es un paso inicial para conseguir participar en los proyectos de las otras personas que publican código en *GitHub*. **Cualquier contribución comienza por la realización de un *fork*** del repositorio en el que queremos colaborar.

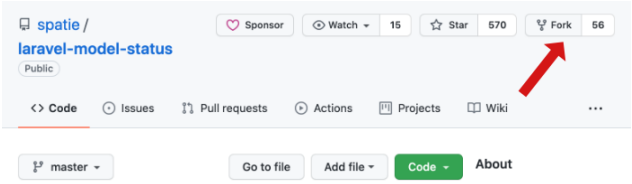
Una vez creado el *fork*, podemos realizar cambios y solicitar el *pull request* a través de la página de *GitHub*, que es básicamente una solicitud para que nuestro código se fusione con el código del repositorio donde hemos colaborado.

Esta operativa de *Pull Request* es un poco más complicada que realizar un simple *fork* y de lo que hemos resumido en las anteriores líneas.

Cómo hacer un fork en GitHub

Ahora que ya sabemos qué es un *fork* y cuándo lo podemos necesitar, vamos a explicar el proceso de realizar un *fork* en *GitHub*.

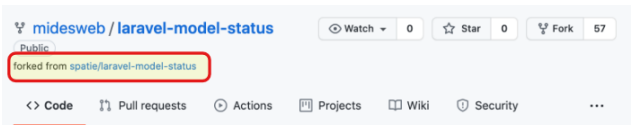
Simplemente accedemos al repositorio del que queremos hacer un *fork*, y pulsas el botón "*Fork*".



Tendremos que hacer *login* en *GitHub* con nuestra cuenta para crear un *fork*.

Si estamos dentro de alguna organización nos podrá aparecer una imagen para que seleccionemos en qué lugar queremos crear el *fork*, en nuestra cuenta personal o en alguna de nuestras organizaciones.

Una vez realizado el *fork* se creará un nuevo repositorio en nuestra cuenta, con una copia del repo original. En la página de *GitHub* se mostrará además que este repositorio es un *fork* de otro repositorio, el original.



Ahora podremos perfectamente hacer el clon de este *fork* en local, para descargarlo en nuestro entorno de desarrollo.

Podremos realizar cambios en el proyecto y luego hacer el *commit*, para seguidamente subir los cambios a *GitHub*.

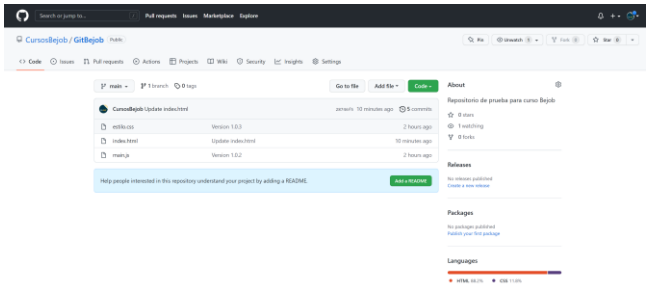
Como los cambios los estamos subiendo en un repositorio de nuestra propiedad, ya que el *fork* está realizado en nuestra propia cuenta, *GitHub* nos permitirá publicar las modificaciones realizadas.

Clone

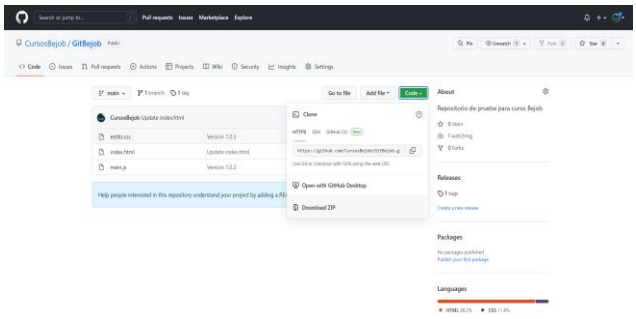
En alguna ocasión nos podemos encontrar con la desagradable sorpresa de que hemos borrado por accidente nuestro proyecto en local.

Si tenemos una versión en *GitHub* podemos recuperarlo de varias formas.

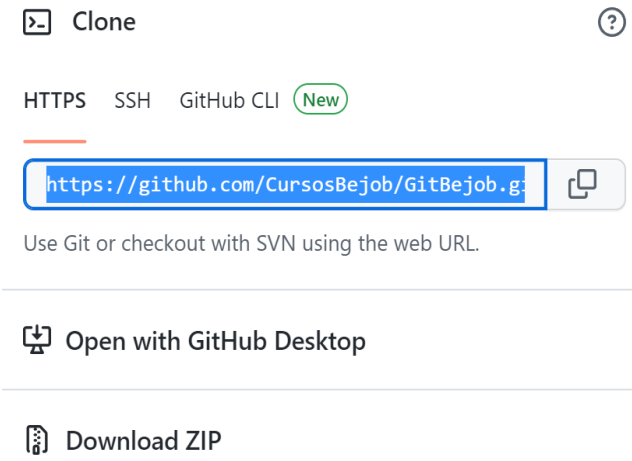
Podemos descargarnos una versión comprimida desde el menú *Tags*.



Podemos descargarlo también desde la opción “Download zip” del menú “<>code”:



O podemos copiar la URL que nos sale justo encima de la opción anterior y clonarla.



Para ello, usaremos nuestra consola con el 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.
```

Diferencia entre *fork* y *clone*

Si hacemos un clon normal de un repositorio, el espacio en GitHub de ese clon seguirá asociado al repositorio que hemos clonado. De este modo, si realizamos cambios sobre el clon y los queremos publicar en GitHub, probablemente no los podremos subir.

Obviamente, si clonamos un repositorio que era nuestro, podremos realizar cambios en local y subirlos a GitHub siempre que queramos. Pero si el repositorio era de otro desarrollador y nosotros no teníamos permisos de escritura sobre él, entonces no podremos subir cambios, porque GitHub no nos lo permitirá. Para este caso es donde necesitamos un fork.