Guía de Git/Github

Comandos básicos

Objetivo: Comprender las bases de git para el control de versiones de código, aprender sus comandos básicos y cómo utilizar Github para subir nuestros repositorios.

Algunas referencias

Tal vez te convenga que visites las referencias al final de la guía, pero creo es importante que estén aquí al inicio por si después regresas a está guía puedes encontrar estas referencias muy útiles, y los comandos más usados.

1. Guía sencilla: comandos e instrucciones explicadas de manera breve y con dibujitos. http://rogerdudler.github.io/git-guide/index.es.html
2. HOJA DE REFERENCIA PARA GITHUB GIT https://github.github.com/training-kit/downloads/es_ES/github-git-cheat-sheet.pdf
3. Beginner's Guide to Using Git and GitHub https://www.codementor.io/git/tutorial/git-github-tutorial-for-beginners
4. Getting started with Git and GitHub: the complete beginner's guide https://towardsdatascience.com/getting-started-with-git-and-github-6fcd0f2d4ac6

Qué es y para qué Git

Git es un sistema de control de versiones, que registra la historia de los cambios realizados sobre un proyecto.

Gracias a Git los desarrolladores pueden:

- Ver los cambios realizados
- Identificar quien realizó los cambios
- Identificar cuando se realizaron los cambios
- Ver la razón de los cambios
- El 90% de los desarrolladores usa Git

Instalar git para el sistema operativo: https://git-scm.com

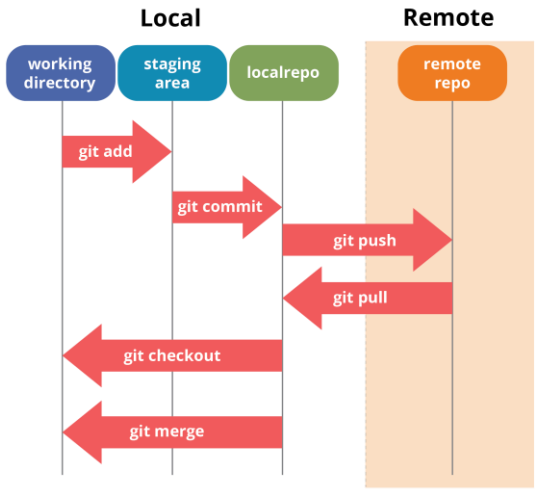
¡Instálalo!

Lista de comandos básicos

Nota: Igual que las referencias anteriores esta lista de comandos tomará sentido una vez leída la guía, y te servirán de referencia para cuando las necesites.

- **git init**: inicializa un repositorio de forma local. Crea un folder oculto para el control de versiones.
- **git clone**: crea una copia local de un proyecto que ya existe de forma remota. Incluye todos los archivos del proyecto, historia y branches (ramas).
- **git add**: Permite incluir archivos al control de versiones.
- **git commit**: guarda el proyecto al momento (como si le tomaran una foto). Se le pone en un comentario la razón del cambio.
- **git status**: ver los cambios que aun no han sido guardados (archivos, eliminados, cambiados o añadidos).
- **git branch**: muestra la rama actual o crear una nueva
- **git merge**: une las líneas de desarrollo (combina los cambios de 2 diferentes ramas)
- **git pull**: actualiza de forma local el código con lo que se tiene en el repositorio remoto (jala los datos)
- **git push**: actualiza el repositorio remoto con los cambios guardados de forma local usando commit (empuja o manda los cambios)

Operaciones y comandos en Git



En Git se trabaja con repositorios. Un **repositorio** puede verse como un folder que contiene archivos y se desea mantener un monitoreo de los cambios sobre ellos. Se puede trabajar de forma local en tu computadora y posteriormente se puede enviar a un servidor remoto como Github.

Sin embargo, tu puedes estar trabajando en tu proyecto en tu máquina (**working directory**) y no todo quieres que se guarde en el repositorio. Tú decides que cosas controlar con git y que no. A las cosas que deseas darle un control debes añadirlas primero en el área de preparación (**staging area**) y posteriormente las puedes pasar al repositorio local usando el comando **commit**. El repositorio local se puede mandar a GitHub usando **git push**, o se pueden

descargar cambios del repositorio usando **git pull**. Considera que en proyectos grandes muchos pueden estar subiendo cambios al repositorio remoto.

Nota: *Todo cambio realizado a los archivos se realiza en el working directory y no se registran los cambios en el repositorio hasta que se añada a la staging area y posteriormente con un commit al respositorio.*

Hay 2 formas de iniciar a usar Github:

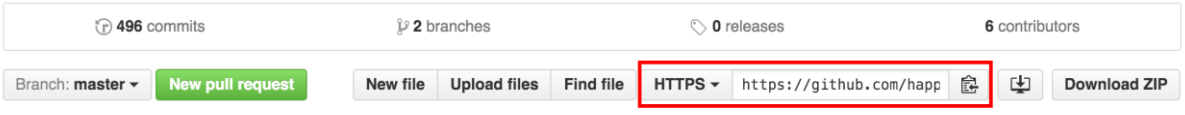
1. **Tengo un repositorio local**: Si tu inicias tu proyecto en tu máquina y creas un repositorio local, lo que sigue es crear en Github un repositorio vacío y subir los cambios.
2. **Tengo un repositorio en Github:** Si ya tienes un repositorio en Github con archivos y deseas trabajar con este repostorio, debes primero bajar el repositorio para tenerlo de forma local.

Antes de empezar, conviene configurar git en tu máquina, estos comandos debes ponerlos en la terminal de tu sistema operativo.
git config --global user.name "Your Name"
git config --global user.email mail@example.com

Tengo un repositorio local

Una vez instalado git en tu máquina puedes abrir una terminal (VS Code tiene el menú Terminal y abres una nueva).

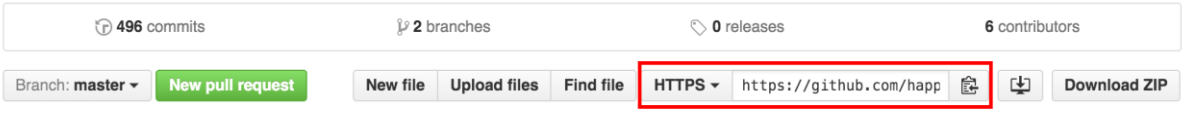
1. Crea un directorio que será el que usaremos para crear un repositorio local
2. Añade uno o varios archivos a este directorio
3. En la terminal entra al folder con: **cd folderName**
4. Vamos a crear el repositorio usando el comando **git init**
5. Vamos a añadir los archivos a la fase de preparación: **git add .** (el punto añade todos los archivos) otra forma es poner el nombre de los archivos a añadir **git add file1.txt file2.txt**
6. (paso opcional) Ver los cambios realizados: **git status**
7. Vamos a subir los cambios del staging area al repositorio local con el comando **git commit -m** "Comentario del commit, subir archivos por primera ocasión"
8. Entrar a Github, crea un repositorio vacío (sin readme) y copia el enlace al repositorio



9. En nuestra máquina vamos a configurar el repositorio remoto (el repo de Github) **git remote add origin** https://github.com/YOUR-USERNAME/YOUR-REPOSITORY.git
10. Finalmente subiremos los archivos a github (el –u es por si se han creado nuevas ramas) **git push -u origin master** (la palabra master es de la rama maestra)

Tengo un repositorio en Github

1. Entrar a Github, crea un repositorio vacío (sin readme) y copia el enlace al repositorio



2. En la terminal ejecuta el siguiente comando para clonar todo lo que está en github
   git clone https://github.com/me/repo.git
3. Nos movemos al directorio del repositorio
   cd repo
4. Creamos un nuevo archivo o modificamos alguno existente
5. Ver los cambios con **git status**
6. Añadir los cambios con **git add .**
7. Guardar los cambios en el repositorio con **git commit -m** "comentario"
8. Subir los cambios al repositorio **git push origin master** (push -u si hay nuevas ramas)

Cuando hay conflictos

2 personas en el equipo han modificado la misma línea de código. Esto ocasionará un conflicto que no permitirá subir cambios hasta que se resuelva.

Nota: Antes de hacer cualquier commit al repositorio remoto es conveniente hacer un pull (git pull origin master).

1. Entrar a Github, modifica una línea de un archivo ya existente en tu repositorio, esto lo puedes hacer directamente en la plataforma web de Github
2. En tu proyecto local también modifica esa misma línea
3. Añade los cambios a tu repositorio
   git add .
   git commit –m "cambios"
4. Antes de subir los cambios al repositorio remoto, realiza un pull
5. Observa que ha ocurrido un conflicto, con VS Code abre el archivo con conflicto y veras algo similar:

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes |
<<<<<<< HEAD (Current Change)
cambio que voy a borrar
=======
cambio3
>>>>>>> adabd0cca6742b654d1ed8153a1bcad7c12e5e3b (Incoming Change)
```

6. Arriba hay varias opciones se debe elegir una, si quedarte con lo que tu hiciste, con lo que ya estaba, o dejar ambas. Al final esto solo quitar las líneas especiales: <<<< HEAD , ====, >>>>.
7. Finalmente es guardar los cambios con git add, git commit, git push

Uso de ramas

Para evitar que todos le estén moviendo al mismo código se pueden hacer ramas y cada quien se enfoca en su parte en su propia rama. De vez en cuando se puede ir fusionando los avances probados sin errores de la rama actual a la rama maestra (master).

Para crear una rama y cambiarse a la rama nueva

- **git branch** nombreRama
- **git checkout** nombreRama
- Otra forma en un solo comando
  - **git checkout –b** nombreRama #crea y te cambia a la rama indicada

Para combinar los cambios de 2 ramas

1. Cambiar a la rama en la que se desea unir el trabajo, en este caso queremos unirla a master.
   - **git checkout** master
2. Realizar el merge (unir ambas ramas)
   - **git merge nombreDeLaOtraRama**
3. Resolver los conflictos (en caso de existir)
4. Hacer el add, commit y el push

Ignorar archivos o carpetas

En ocasiones hay archivos que no se quieren que se suban al servidor

Posibles razones:

- Son configuraciones locales del IDE (workspace, colores, etc.)
- Archivos para hacer pruebas locales
- Archivos zip

Es necesario crear un archivo **.gitignore** y en el archivo poner todo lo que se desea ignorar por ejemplo:

- archivo1.txt #se refiere a un archivo especifico
- *.zip #se refiere a todos los archivos con extensión zip
- resultados/ #se refiere al directorio resultados

- https://swcarpentry.github.io/git-novice-es/06-ignore/

Resumen

- Los 3 pasos básicos son:
    - **git add .** #añadir cambios
    - **git commit –m** "comentario" #guardar los cambios localmente
    - **git push origin** nombreBranch #subir los cambios
- Conviene revisar el status para ver si hay cambios y el branch actual
    - **git status**
- Para ignorar cambios con el archivo **.gitignore**
- Guía sencilla: comandos e instrucciones explicadas de manera breve y con dibujitos. http://rogerdudler.github.io/git-guide/index.es.html