

Introducción a GIT

Esta guía está diseñada para poder obtener el código del curso de GIT / GITHUB de una forma amigable y en español.

Aviso

Esta guía está en constante actualización, podría no estar completa.

Enlaces

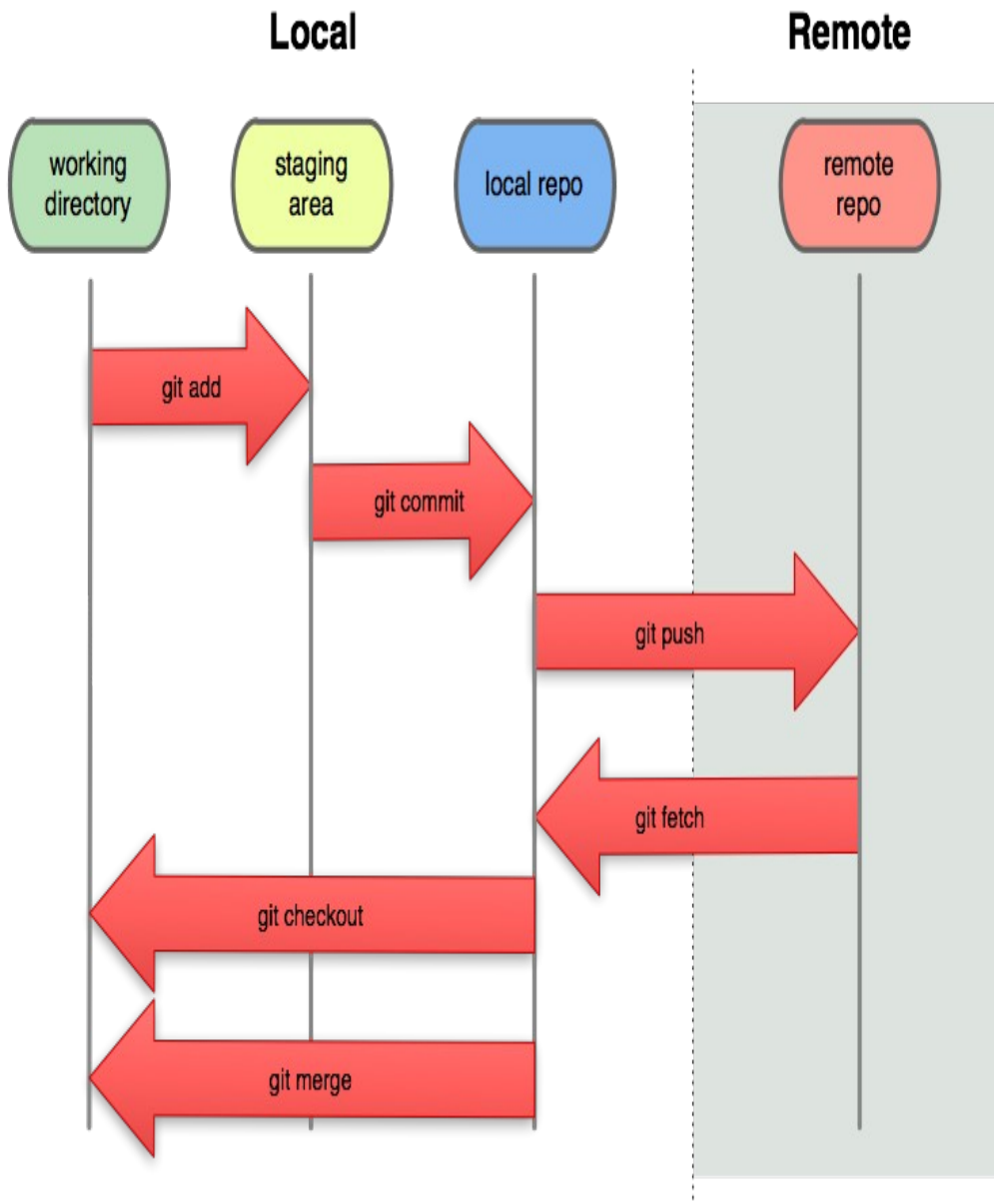
- [Instalar Git](#)

[Github](#)

¿Qué es GIT?

Es un software de control de versiones, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos (También puedes trabajar solo no hay problema). Existe la posibilidad de trabajar de forma remota y una opción es GitHub.

Flujo de trabajo de GIT



Tratando de explicar la imagen: Tenemos nuestro directorio local (una carpeta en nuestro pc) con muchos archivos, Git nos irá registrando los cambios de archivos o códigos cuando nosotros le indiquemos, así podremos viajar en el tiempo retrocediendo cambios o restaurando versiones de código, ya sea en Local o de forma Remota (servidor externo). En la práctica quedará más claro.

¿Qué es GitHub?

Es una plataforma de desarrollo colaborativo para alojar proyectos (en la nube) utilizando el sistema de control de versiones Git, Además cuenta con una herramienta muy útil que es GitHub Pages donde podemos publicar nuestros proyectos estáticos (HTML, CSS y JS) gratis.

Fundamentos de GIT

En este apartado podrás comenzar a trabajar con git.

Comandos básicos

Aprendamos los primeros comandos con git

Versión de git

```
git version
```

Registrar nuevo usuario asociado a git:

WARNING

No colocar como nombre de usuario el correo de su cuenta de Github, podría traer problemas a futuro.

```
git config --global user.name "mi nombre"
```

Es recomendable utilizar el correo asociado a Github

```
git config --global user.email "myemail@example.com"
```

Ayuda

```
// Ayuda sobre los comandos  
git help
```

Mi primer repositorio

```
// Iniciar un nuevo repositorio  
// Crear la carpeta oculta .git  
git init
```

```
// Ver que archivos no han sido registrados  
git status
```

```
// Agregar todos los archivos para que esté pendiente de los cambios  
git add .
```

```
// Crear commit (fotografía del proyecto en ese momento)  
git commit -m "primer commit"
```

```
// Muestra la lista de commit del mas reciente al más antiguo  
git log
```

En resumidas cuentas nosotros realizamos cambios en nuestros archivos, el comando `status` verificará que archivos han sido modificados. Cuando deseemos registrar esos cambios tendremos que agregarlos con `add` . así ya estará listo para poder hacer un `commit`. El `commit` realiza la copia de ese instante para poder volver en el tiempo si es que es necesario.

Trucos

```
// Muestra en una línea los commit realizados
git log --oneline
```

```
// Muestra en una línea los commit realizados pero más elegante
git log --oneline --decorate --all --graph
```

```
// Solo muestra los archivos modificados
git status -s
```

Diferencias entre -- y -

- **--decorate** hace referencia a una palabra

- **-s** hace referencia al comando o a varios comandos, **-sa** serían dos comandos diferentes

```
// Vemos información de la rama maestra
git status -s -b
git status -sb //Hace lo mismo que el comando anterior
```

Creando alias globales

Los alias nos sirven para crear atajos de comandos, podemos guardar diferentes alias de forma global y quedarán guardados en la configuración de git.

```
// Guardamos el alias "lg" que ejecutará todo lo que está entre comillas
git config --global alias.lg "log --oneline --decorate --all --graph"
```

```
// Para ver el archivo config con los alias creados
git config --global -e
```

Vim es el editor de código en la línea de comandos

Salir del modo edición "Vim"

Para salir del modo edición de la líneas de comando precionar **:q**, en caso de realizar algún cambio sin guardar escribir **:qa**

:q! también sirve para salir sin guardar

```
// Modo lectura sin poder modificar
git config --global -l
```

```
// Realiza el add . y commit más mensaje al mismo tiempo
git commit -am "más comandos agregados"
```

```
// Para editar un commit, como por ej: el mensaje
git commit --amend
```

Trucos de editor Vim

i para comenzar a editar

esc para salir del modo edición

wq para guardar y salir

q! salir sin guardar cambios

Viajes a través de los commit

Vamos a conocer como podemos movernos entre los diferentes commit que tengamos registrados, supongamos que tenemos los siguientes commit:

- f82f457 (HEAD -> master) mas comandos agregados
- f52f3da nuevos comandos en fundamentos.md
- e4ab8af mi primer commit

```
// Viajamos al commit en específico f52f3da
git reset --mixed f52f3da
```

```
// Viajamos al commit en específico f52f3da y eliminamos los cambios futuros
git reset --hard f52f3da
```

```
// Muestra todos los cambios incluso si borramos los commit
git reflog
```

```
// Viajamos al commit en específico f52f3da y podemos restaurar los archivos
git reset --hard f52f3da
```

Si no hicimos un commit pero aún así queremos revertir los cambios en un archivo específico podríamos utilizar el siguiente comando:

```
git checkout -- nombreArchivo.conExtensión
```

Si deseamos destruir todos los cambios sin haber realizado un commit podemos utilizar:

```
git reset --hard
```

Para mayor información visite: [Click aquí](#)

Renombrar archivos

Puede que queramos renombrar un archivo, es recomendable hacerlo directamente en la línea de comandos para registrar los cambios con git.

```
// Cambiar nombre
git mv nombreOriginal.vue nombreNuevo.vue
```

Recuerden hacer el commit para registrar los cambios en git.

Eliminar archivos

```
// Cambiar nombre
git rm nombreArchivo.vue
```

También recordar hacer el commit para salgar cambios en git.

Ignorando Archivos

Para no hacer seguimiento de carpetas o archivos, debemos crear el siguiente archivo:

- .gitignore
- Su estructura de ejemplo sería así:

```
archivo.js // Ignora el archivo en cuestion
*.js // Ignora todos los archivos con extensión .js
node_modules/ //Ignora toda la carpeta
```

Ramas o branch

Hasta el momento solo hemos trabajado en la rama "master" pero puede que necesitemos crear diferentes ramas para los seguimientos de git.

```
// Crea una nueva rama
git branch nombreRama
```

```
// Nos muestra en que rama estamos
git branch
```

```
// Nos movemos a la nueva rama
git checkout nombreRama
```

Podemos unir la rama master con la nueva, para eso tenemos que estar en la master para ejecutar el siguiente comando:

```
// Nos movemos a la nueva rama
git merge nombreRama
```

```
// Eliminar una rama
git branch -d nombreRama
```

Atajos

Podemos utilizar `git checkout -b nuevaRama` para crear la nuevaRama y viajar a ella.

Tags

Con los tags podemos hacer versiones de nuestro proyecto.

```
// Crear un tags
git tag versionAlpha -m "versión alpha"
```

```
// Listar tags
git tag
```

```
// Borrar tags
git tag -d nombreTags
```

```
// Hacer una versión en un commit anterior ej: f52f3da
git tag -a nombreTag f52f3da -m "version alpha"
```

```
// Mostrar información del tag
git show nombreTag
```

GITHUB

Github es la plataforma online para trabajar con proyectos en git, en simples palabras es utilizar git de forma remota pero entiendase que son cosas totalmente distintas

Crear una cuenta en Github

Para comenzar a trabajar con github tienes que crear una nueva cuenta, es gratis así que no pierdes nada en este punto

Crear una nuevo repositorio

Para subir nuestro proyecto debemos crear un nuevo repositorio, al momento de la creación nos mostrará una serie de comandos para subir el proyecto.

```
git remote add origin https://github.com/bluuweb/tutorial-github.git
git push -u origin master
```

Al ejecutar estas líneas de comando te pedirá el usuario y contraseña de tu cuenta de github.

```
// Nos muestra en que repositorio estamos enlazados remotamente.
git remote -v
```

Subir los tags

Por defecto si creaste un proyecto con diferentes versiones no subirá los tags, para eso tenemos el siguiente comando.

```
git push --tags
```

Push

Al ejecutar el comando `git push` estaremos subiendo todos los cambios locales al servidor remoto de github, ten en cuenta que tienes que estar enlazado con tu repositorio, para eso puedes utilizar `git remote -v` luego ejecuta:

```
git push
```

Pull

Cuando realizamos cambios directamente en github pero no de forma local, es esencial realizar un pull, donde descargaremos los cambios realizados para seguir trabajando normalmente.

Es importante estar enlazados remotamente, puedes verificar con: `git remote -v`, luego ejecuta:

```
git pull
```

Fetch

Este comando hace la comparación de nuestros archivos locales con los del servidor, si existiera alguna diferencia nos pediría realizar un `get pull` para realizar un match de nuestros archivos locales.

```
git fetch
```

Clonar repositorio

Para descargar un repositorio completo basta con tomar la url ej:

`https://github.com/bluuweb/tutorial-github.git` y ejecutar el siguiente comando en alguna carpeta de su computadora.

```
git clone https://github.com/bluuweb/tutorial-github.git nombreCarpeta
```