Versionado
Git y Github

...

by Arcángel Andrés Artigue
@Arcangel_617

¿Qué es el control de versiones?

¿Qué es el control de versiones?

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

¿Qué puedo versionar?

En nuestro ámbito hablamos de versionar código fuente, pero en realidad cualquier tipo de archivo que encuentres en un ordenador puede ponerse bajo control de versiones.

¿Qué es Git?

Los tres estados

Los tres estados

Git tiene tres estados principales en los que se pueden encontrar tus archivos: *confirmado* (**committed**), *modificado* (**modified**), y *preparado* (**staged**).

Confirmado significa que los datos están almacenados de manera segura en tu base de datos local.

Modificado significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.

Preparado significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Los tres estados

Esto nos lleva a las tres secciones principales de un proyecto de Git: el directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area).

El directorio de Git es donde Git almacena los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando clonas un repositorio desde otro ordenador.

El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

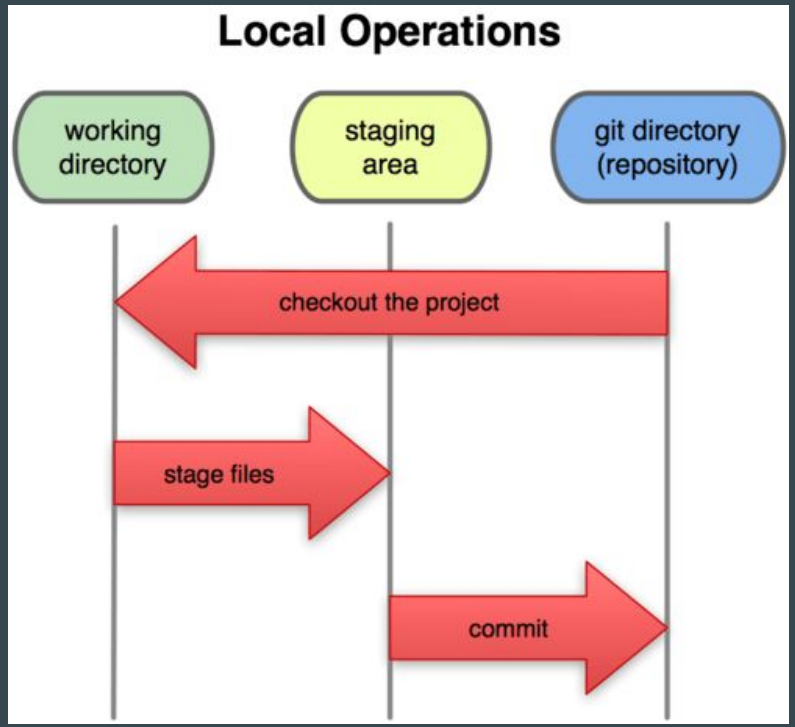
El área de preparación es un sencillo archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación

Los tres estados

El flujo de trabajo básico en Git es algo así:

Modificas una serie de archivos en tu directorio de trabajo.
Preparas los archivos, añadiendolos a tu área de preparación.
Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación, y almacena esas instantáneas de manera permanente en tu directorio de Git.

Los tres estados



Instalación

Instalación

Debian/Ubuntu (y derivados)

apt-get install git

CentOS 7

yum install git

Windows

https://git-for-windows.github.io/ o https://git-scm.com/download/win

Configurando Git por primera vez

Tu identidad

git config --global user.name “tu_nombre”
git config --global user.email “tu_correo”

Nota: Si no se pasa ningún parámetro, muestra la información que posee

git config --global --list

Obtener ayuda

git help <comando>
git <comando> --help
man git-<comando>

Ejemplos: git help clone, git help add, git help commit

Fundamentos de Git

Obtener un repositorio

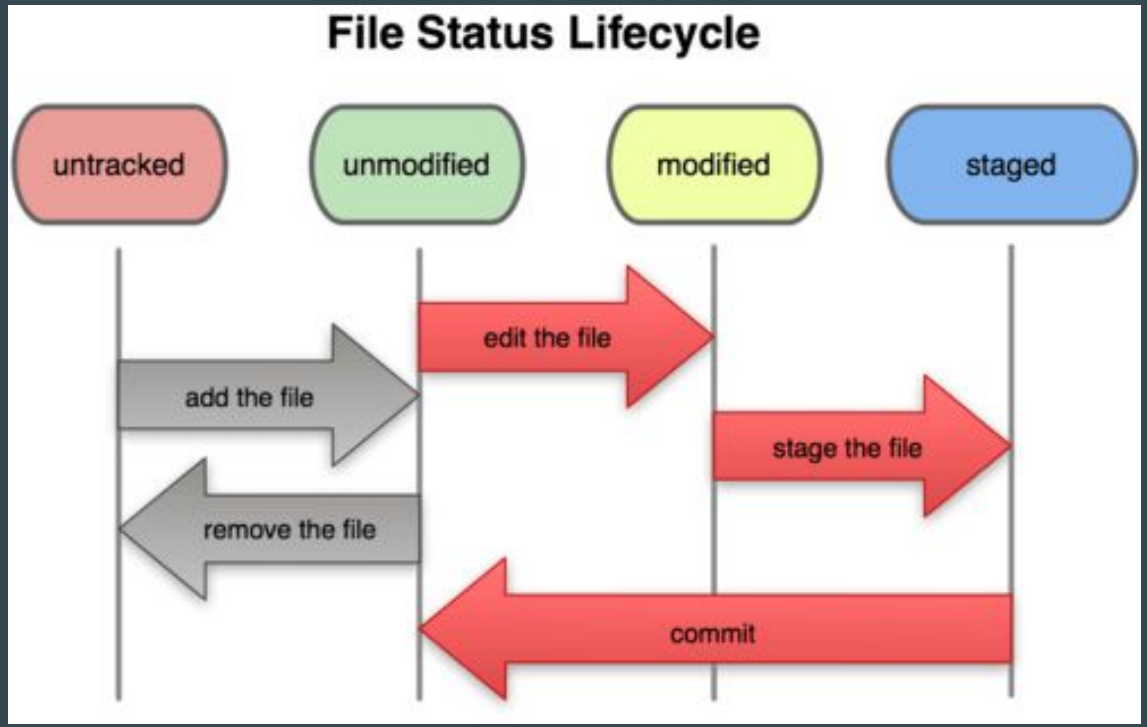
Inicializar un repositorio en un directorio existente

git init

Clonar un repositorio existente

git clone [url]

Guardando estados en el repositorio


File Status Lifecycle
untracked
unmodified
modified
staged
edit the file
add the file
stage the file
remove the file
commit

Estado, seguimiento, preparación y confirmación de archivos

git status

git add

git commit

git diff

git diff --chached

git rm

git mv

Estado, seguimiento, preparación y confirmación de archivos

git commit --amend

git cheackout

Ignorando archivos

Archivo .gitignore

Las reglas para los patrones que pueden ser incluidos en el archivo son:

Las líneas en blanco, o que comienzan por #, son ignoradas.
Puedes usar patrones glob estándar.
Puedes indicar un directorio añadiendo una barra hacia delante (/) al final.
Puedes negar un patrón añadiendo una exclamación (!) al principio.

Ignorando archivos

Los patrones glob son expresiones regulares simplificadas que pueden ser usadas por las shells. Un asterisco (*) reconoce cero o más caracteres; [abc] reconoce cualquier carácter de los especificados entre corchetes (en este caso, a, b o c); una interrogación (?) reconoce un único carácter; y caracteres entre corchetes separados por un guión ([0-9]) reconoce cualquier carácter entre ellos (en este caso, de 0 a 9).

Ignorando archivos

```
# a comment – this is ignored
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a files above
!lib.a
# only ignore the root TODO file, not subdir/TODO
/TODO
# ignore all files in the build/ directory
build/
# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt
# ignore all .txt files in the doc/ directory
doc/**/*.txt
```

¿Qué es GitHub?

Por último y no por eso menos importante

Tipos de Sistemas de control de versiones

Tipos de sistemas de control de versiones

Sistemas de control de versiones locales

Sistemas de control de versiones centralizados

Sistemas de control de versiones distribuidos

Enlaces útiles

https://help.github.com/articles/creating-a-pull-request/

https://codigofacilito.com/articulos/buenas-practicas-en-commits-de-git

https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet