

Git. Sesión Uno

Un vistazo rápido

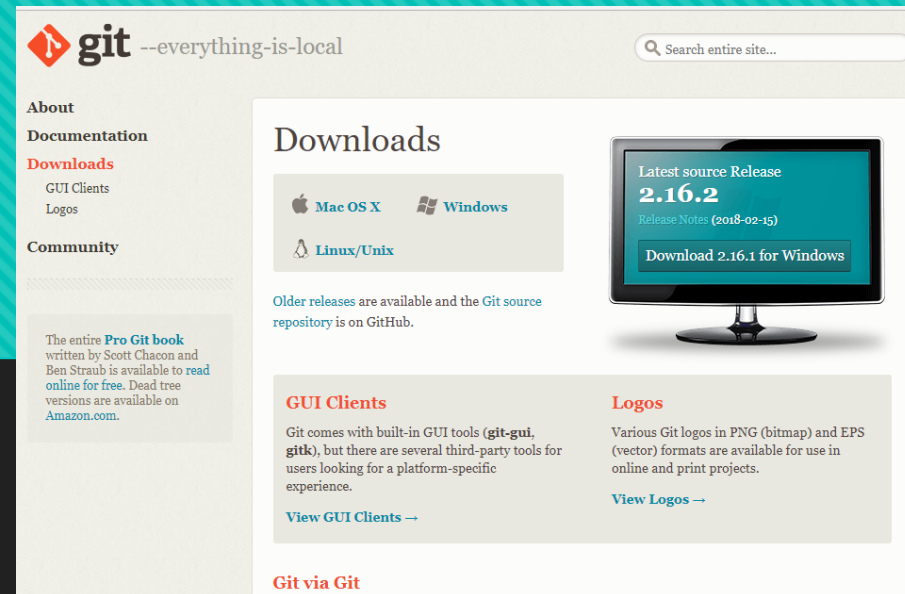
Alfredo González Gaviña, Walter Alejandro Moreno Ramírez, María Susana Ávila García.
Taller 3G
DEM Yuriria 21/02/2018

El principio.

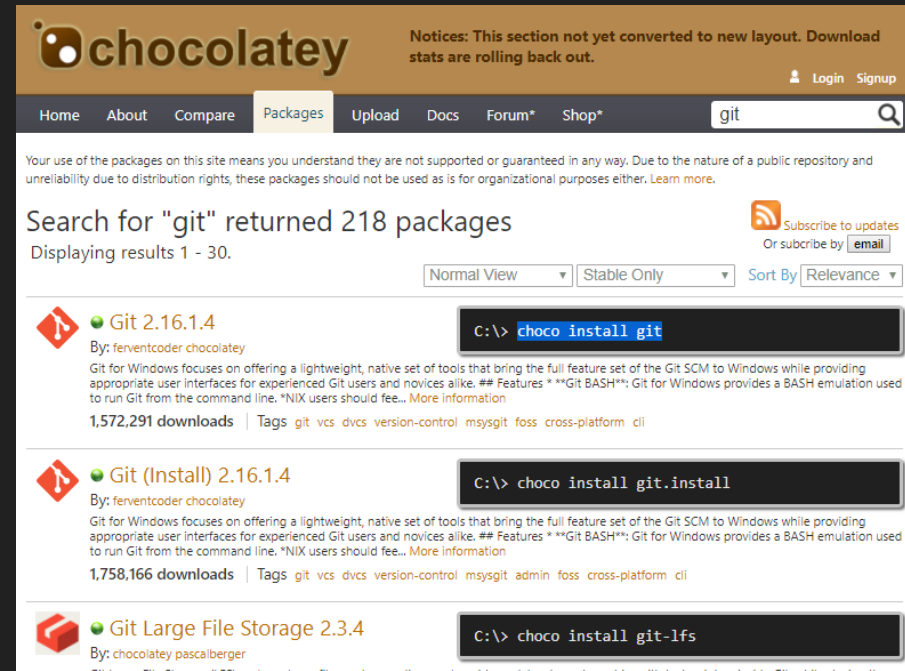
Antes que nada debe tener instalado el software controlador de versiones para poder trabajar con el. Git esta disponible para todos los sistemas operativos y puede descargarlo desde su pagina oficial o bien, usando su manager de paquetes favorito.

<https://git-scm.com/>

<https://git-scm.com/downloads>



The screenshot shows the official Git website. At the top, the Git logo is followed by the tagline "--everything-is-local". A search bar is in the top right. The left sidebar contains links for "About", "Documentation", "Downloads", "GUI Clients", "Logos", and "Community". The main content area has a "Downloads" section with buttons for "Mac OS X", "Windows", and "Linux/Unix". To the right, a monitor displays the "Latest source Release 2.16.2" with a "Download 2.16.1 for Windows" button. Below this, there are sections for "GUI Clients" and "Logos". At the bottom left, a note mentions the "Pro Git book".



The screenshot shows the Chocolatey website's search results for "git". The header includes the Chocolatey logo, a notice about the new layout, and links for "Login" and "Signup". A navigation bar contains "Home", "About", "Compare", "Packages", "Upload", "Docs", "Forum*", and "Shop*". A search bar shows "git". Below the navigation bar, a disclaimer states that packages are not supported or guaranteed. The search results show "Search for 'git' returned 218 packages" and "Displaying results 1 - 30.". There are three results visible: 1. "Git 2.16.1.4" by ferventcoder chocolatey, with 1,572,291 downloads and a command box showing "C:\> choco install git". 2. "Git (Install) 2.16.1.4" by ferventcoder chocolatey, with 1,758,166 downloads and a command box showing "C:\> choco install git.install". 3. "Git Large File Storage 2.3.4" by chocolatey pascalberger, with a command box showing "C:\> choco install git-lfs".

El principio.

Debian/Ubuntu

```
$ sudo apt-get install git-all
```

RedHat/Fedora

```
$ sudo dnf install git-all
```

Arch/Manjaro

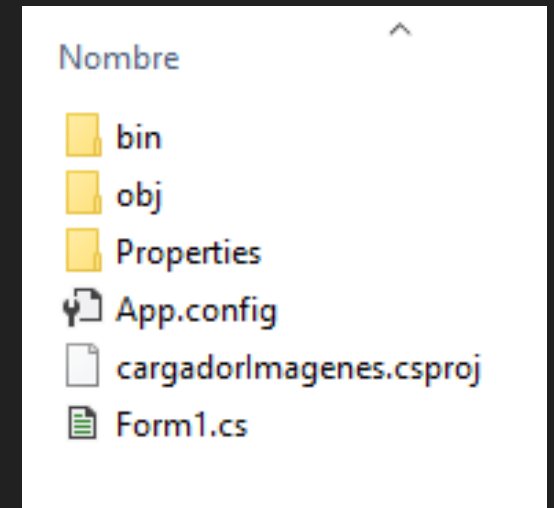
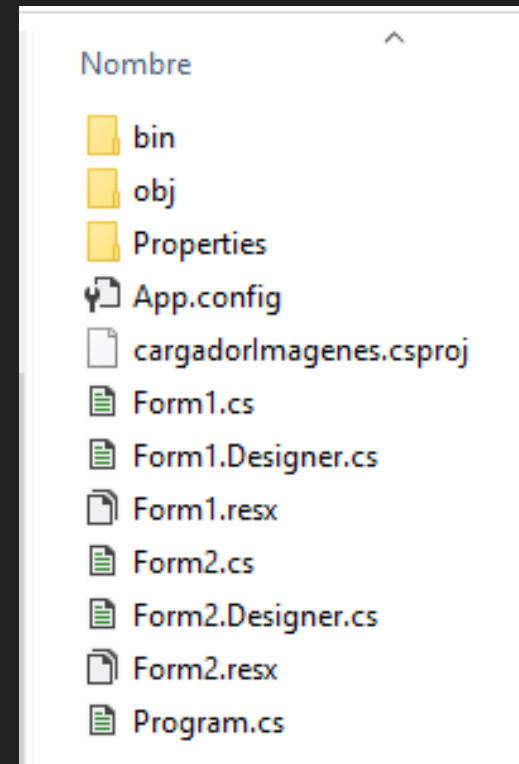
```
$ sudo pacman -S git
```

MAC OS

```
$ brew install git
```

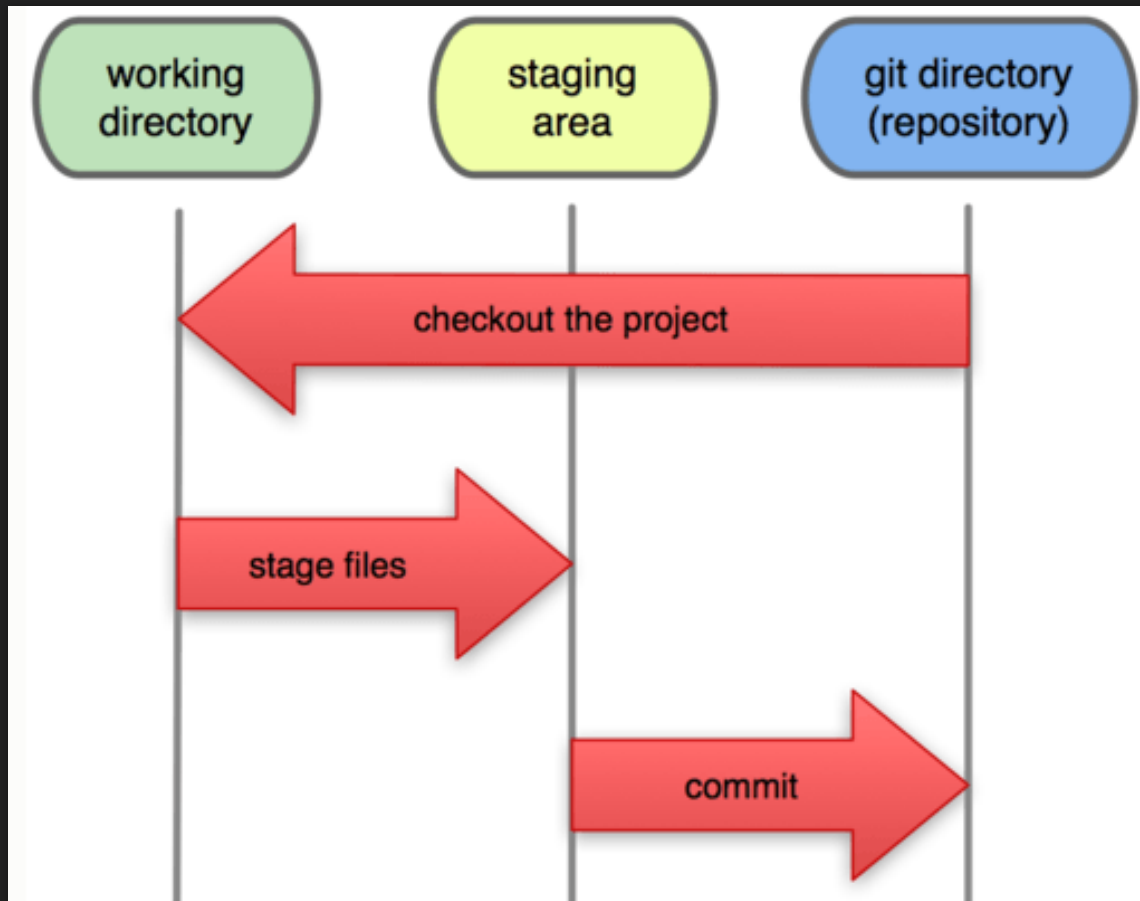
La mejor versión de ti.

¿Y que es una versión? De manera general una “Versión” es un modo particular de hacer algo. Una versión en el software es el nivel de desarrollo que tiene un programa informático. Para nosotros una **“versión”** será una **“fotografía instantánea”**, una “captura de pantalla” **de la carpeta en la que estamos trabajando** junto con todos los archivos que contiene y sus cambios correspondientes.



Dos versiones diferentes del mismo directorio de trabajo.

Simple y sencillo



Directorio de trabajo: Es una copia de una versión cualquiera. La información (archivos, imágenes, videos, música) se guarda en una base de datos dentro del directorio git y se extrae para ponerla en disco y pueda ser usada por personas.

Directorio git: Es donde reside la información que solo git utiliza, como bases de datos y metadatos. Este directorio contiene toda la información del repositorio y las versiones.

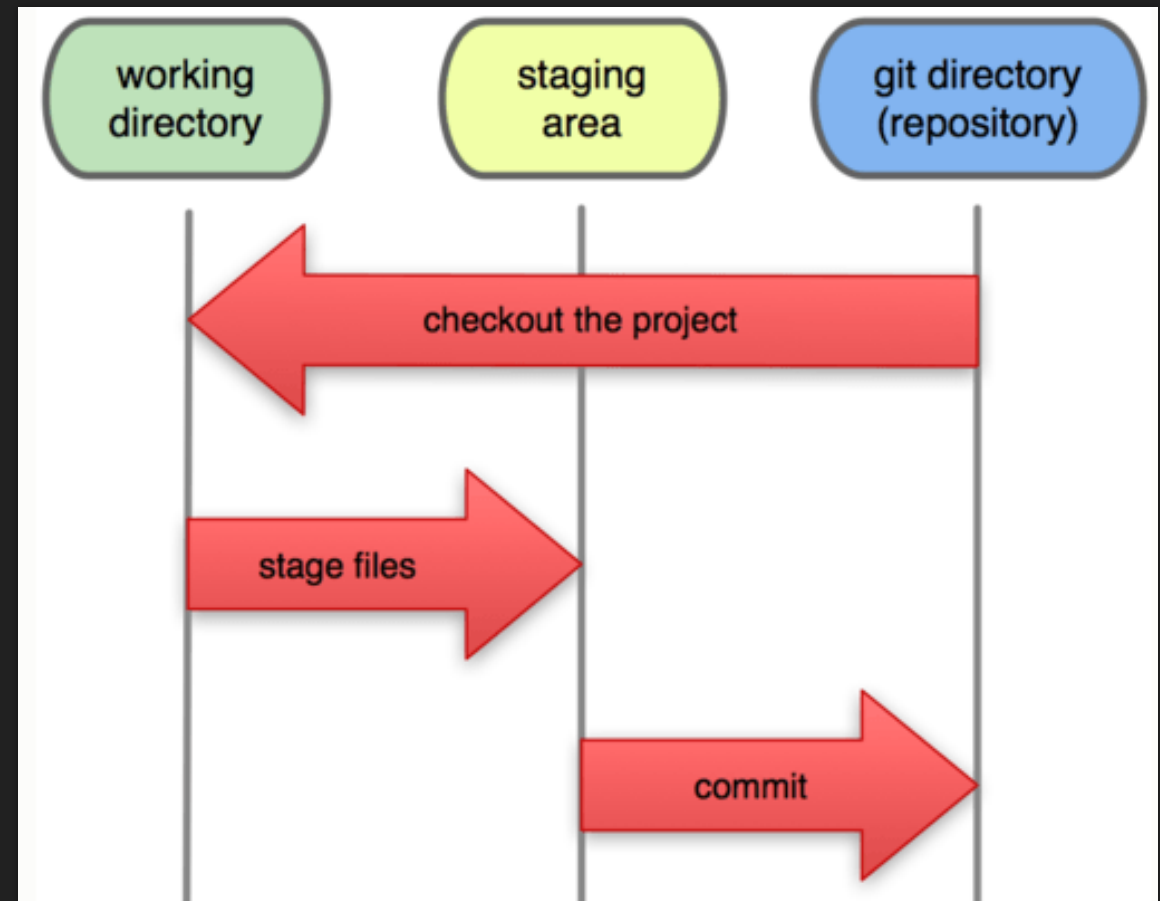
Área de preparación (índice, stage) : representa a los archivos cuyos cambios has sido “aceptados” como parte de la nueva versión del repositorio, puede ser un simple archivo hasta un proyecto completo.

A grandes rasgos.

El flujo de trabajo básico en Git es algo así:

- Modificas una serie de archivos en tu directorio de trabajo. Archivos modificados (**modified**)
- Preparas los archivos, añadiéndolos a tu área de preparación. Archivos preparados (**staged**)
- Confirmas los cambios tomando los archivos tal y como están en el área de preparación. Estos cambios se “Guardan” permanentemente en tu directorio de Git. Archivos “**Guardados**” (**committed**)

Fuente [<https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git>]



Lo primero es lo primero

Antes de comenzar a escribir comandos en la terminal debemos crear un repositorio local con el cual podamos trabajar.

```
$ git init <nombreRepo>
```

Después debemos configurar nuestra identidad para poder ser reconocidos y poder comenzar a trabajar.

```
$ git config --global user.name <nombreUser>
```

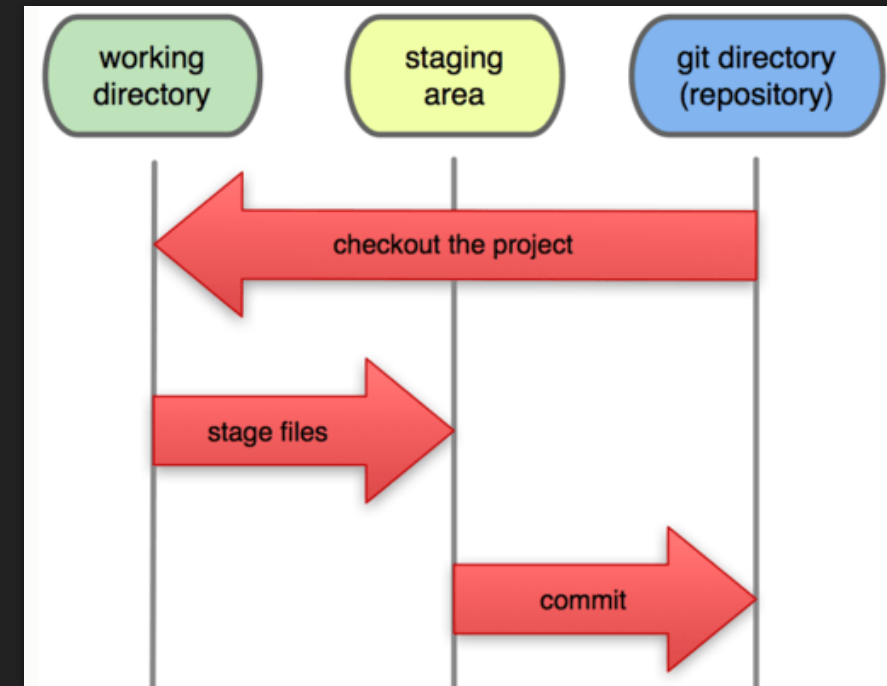
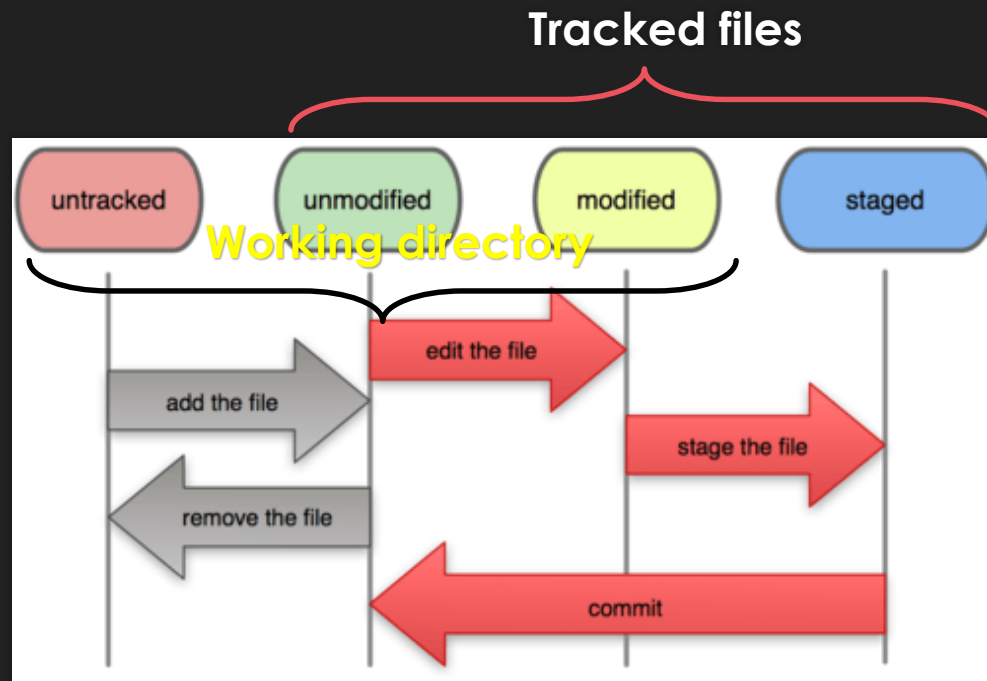
```
$ git config --global user.email <correo>
```

Git permite obtener información acerca de los comandos que existen y los parámetros que reciben.

```
$ git help <comando>    $ git <comando> --help    $ man git-<comando>
```

Manos a la obra

Fuente [<https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git>]

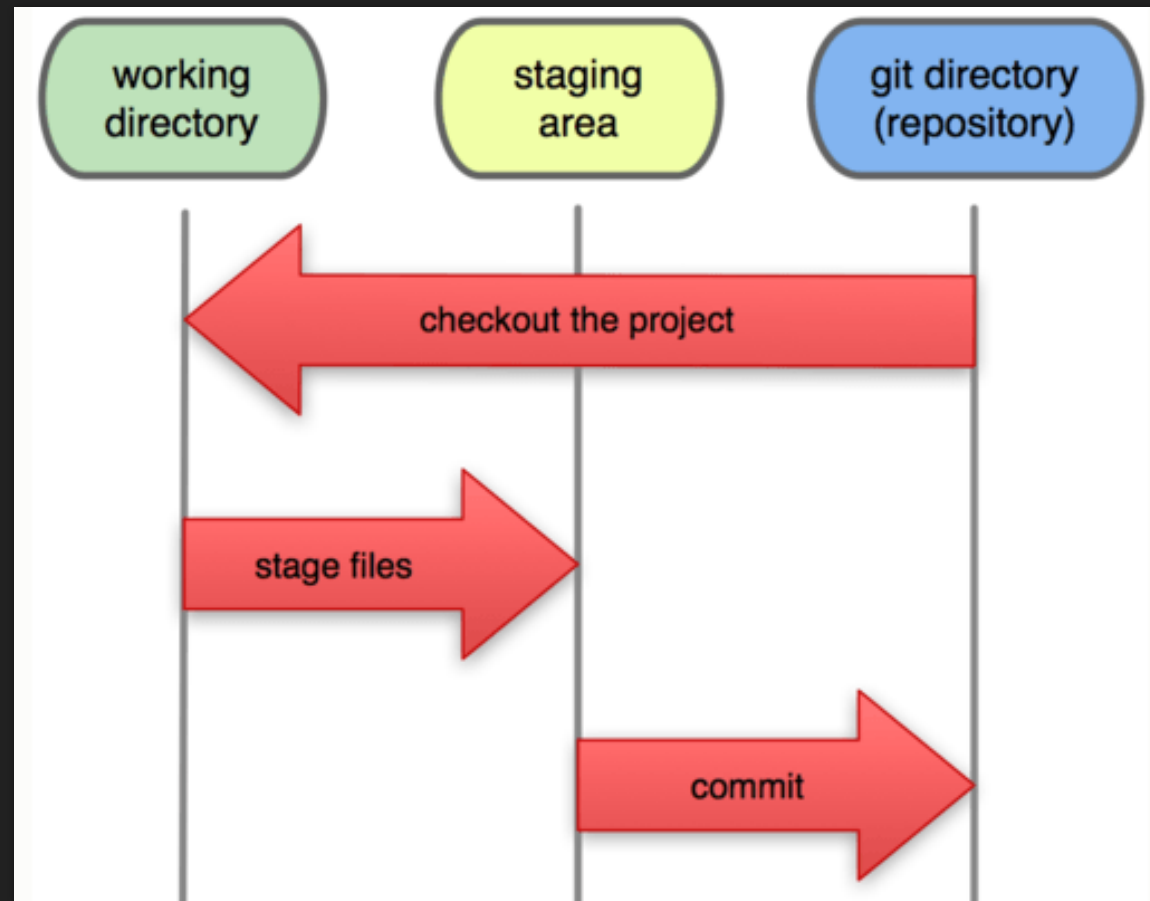


Existen dos tipos de archivos **en seguimiento (tracked)** y **sin-seguimiento (untracked)**. Los archivos con **seguimientos** pueden estar **sin modificaciones, modificados, o preparados**. Mientras que los **archivos sin-seguimiento son cualquier otro archivo en el repositorio y son ignorados por git**.

Protege lo que importa

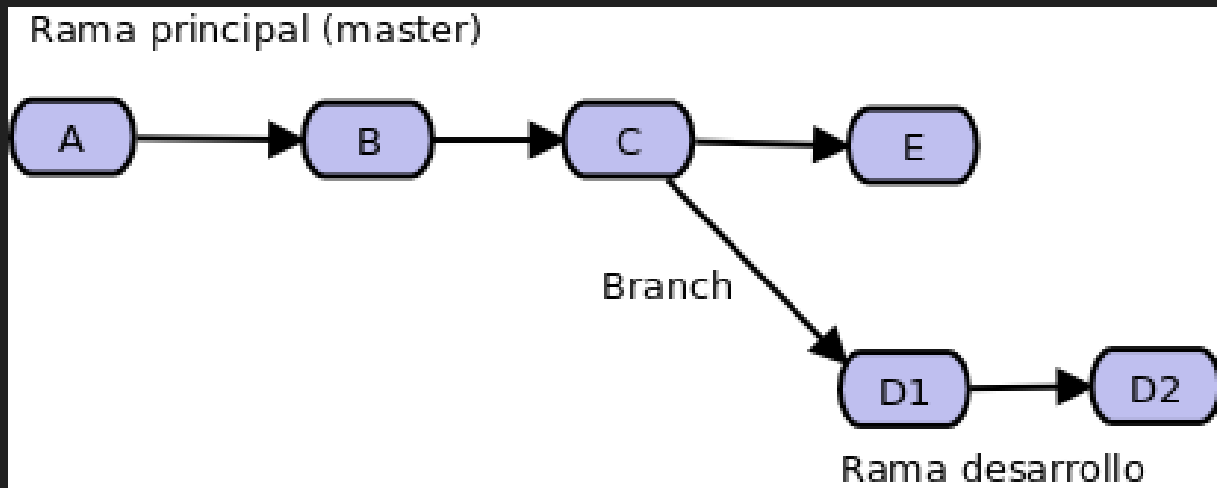
Al tener archivos “preparados” puedes proceder a “guardarlos”. Imagina que tomas una “selfie” del área de preparación, guardando cada detalle de los archivos que tienes preparados. Esta “selfie” del área de preparación se llama “**commit**” y su función es exactamente esa, crear una copia del área de preparación que es almacenada en el directorio git.

\$ git commit



Realidad alternativa.

En todos los SCV existe un concepto denominado “**rama**”, versiones, “capturas”, “selfies” del estado de un repositorio en un momento dado, es decir son conjuntos independientes de commits.



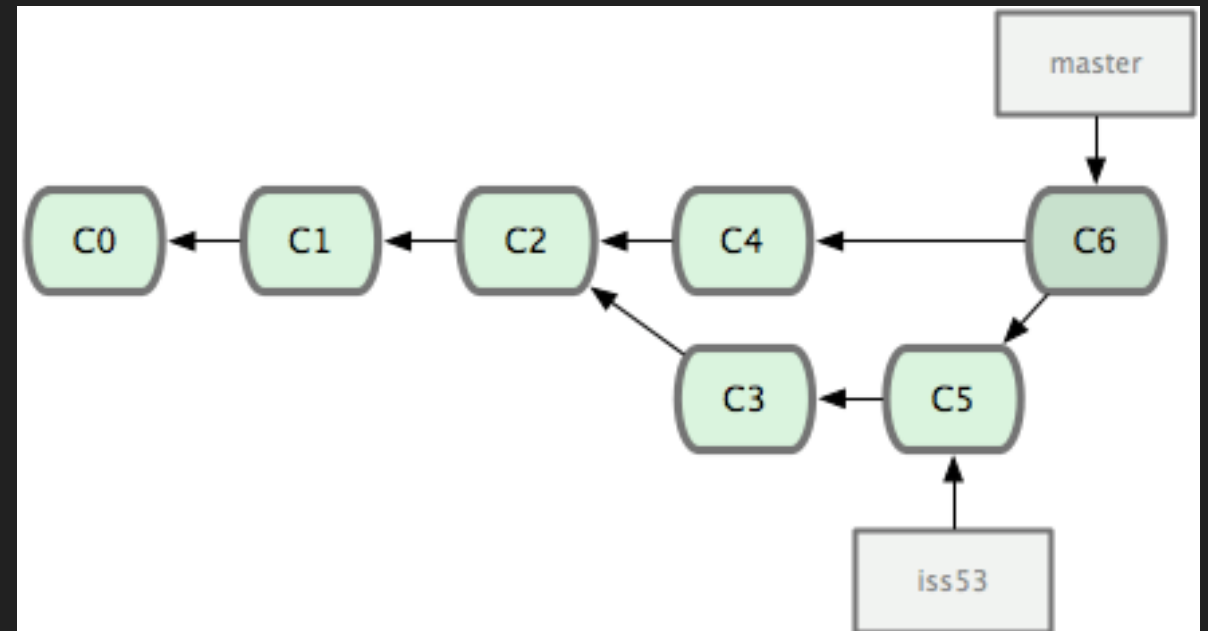
Una rama en git se puede crear con el comando:

```
$ git branch <nombreRama>
```

Fuusión haad!

Las ramas pueden contener información idéntica, similar, o completamente distinta, como se ha visto, el contenido de una rama es completamente independiente del de otra ya que las ramas son secuencias de commits. Entonces ¿Qué pasa si necesito archivos de un rama distinta? ¿Tengo que crearlos de nuevo en mi rama actual? Por fortuna existe un procedimiento que permite unir el contenido de dos ramas, la fusión. La fusión es la unión de los dos últimos commits de las ramas a fusionar y da como resultado un nuevo commit que contiene la información de los commits de las ramas involucradas

Fuente [<https://git-scm.com/book/es/v1/Ramificaciones-en-Git-Procedimientos-b%C3%A1sicos-para-ramificar-y-fusionar>]



El comando para crear una rama es:

\$ git branch <nombreRama>

Tu turno

A continuación se presenta un pequeño ejercicio donde se sugiere que intente realizarlo solo, si necesita ayuda puede pedir apoyo a los instructores. Suerte