

The background features a light gray dashed line forming a large circle. Various solid-colored circles in teal, lime green, orange, and pink are scattered around the perimeter. Some circles are solid, while others are dashed outlines. A large teal ring is in the top left, and a large yellow ring is in the bottom right.

# Git Basics

Introducción al Control de Versiones

```
print “¡Hola Mundo!”;
```



# Manuel Alvarado

Ingeniero de Sistemas  
Desarrollador Web

@cubosensei  
info@manalco.co

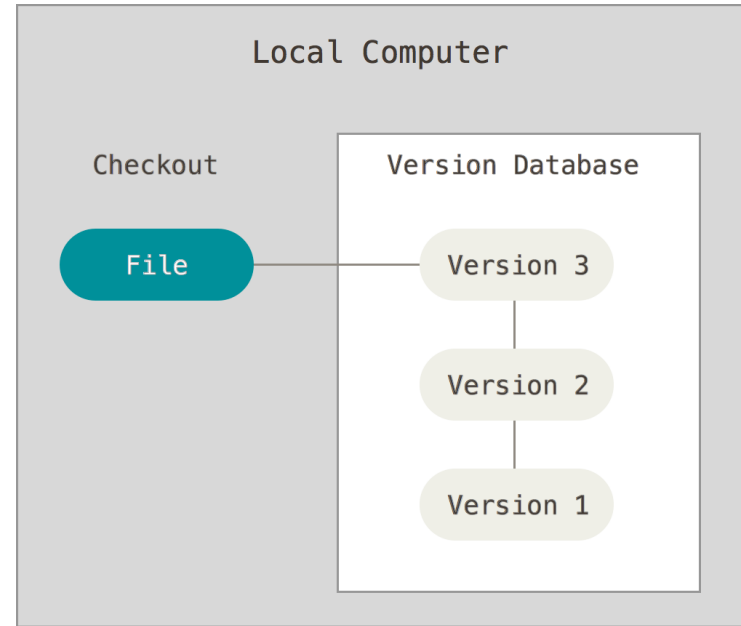
A decorative graphic consisting of various colored circles and rings in shades of pink, orange, teal, yellow, and green, scattered across the slide. Some are solid, some are dashed, and some are hollow rings.

# Control de Versiones

“Es un sistema que registra los cambios realizado a un archivo o conjunto de archivos a través del tiempo” \*

- Locales
- Centralizados
- Distribuídos

# Sistema de Control de Versiones Local



# Sistema de Control de Versiones Local



Proyecto.docx

Proyecto (final).docx

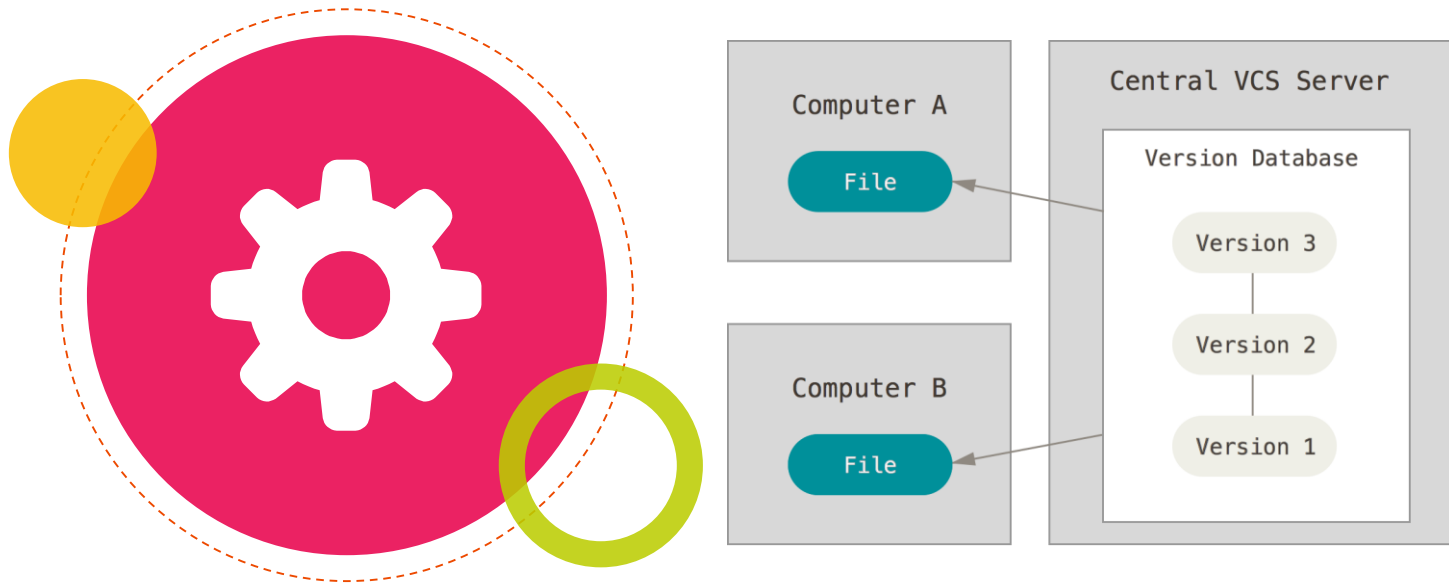
Proyecto (final final).docx

Proyecto (revisado).docx

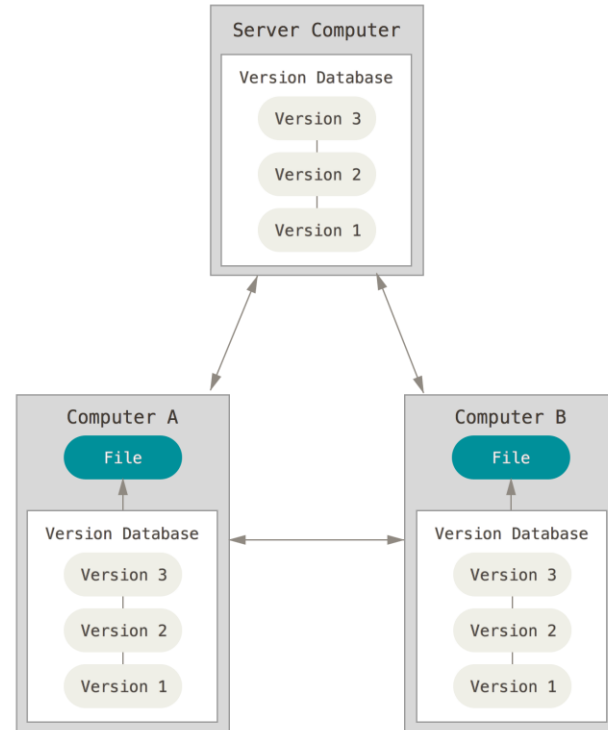
Proyecto (revisado 10-5-2016).docx

Proyecto (revisado 10-5-2016) final.docx

# Sistema de Control de Versiones Centralizado



# Sistema de Control de Versiones Distribuido



# Sistema de Control de Versiones Distribuído



Git  
Mercurial  
Bazaar  
Darcs



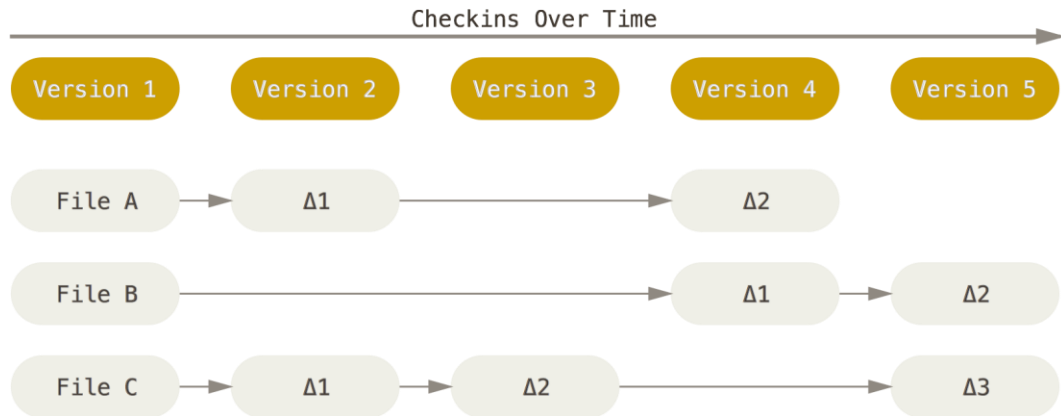


BitKeeper

- ◎ Velocidad
- ◎ Diseño sencillo
- ◎ Apoyo al desarrollo no lineal (ramas paralelas)
- ◎ Completamente distribuido
- ◎ Capaz de manejar grandes proyectos (como el núcleo de Linux) de manera eficiente.

# Git: ¿Cómo funciona?

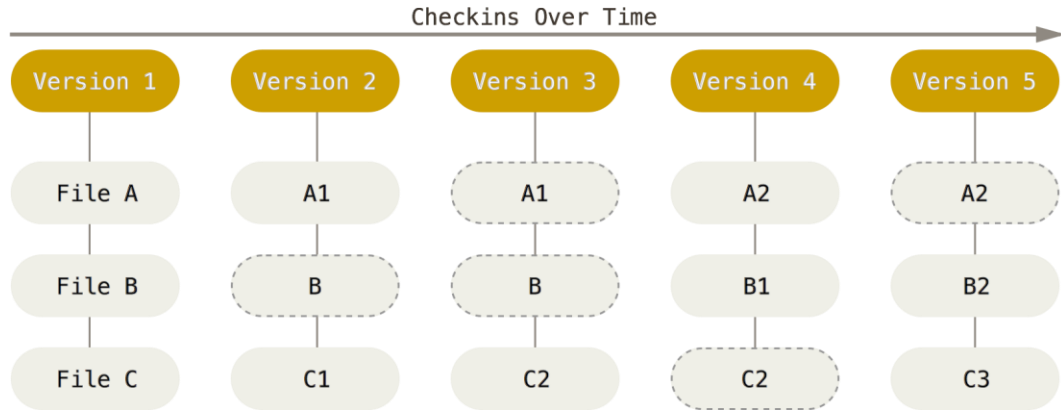
Instantáneas **!=** Diferencias



VCS que modelan datos con base en **diferencias**

# Git: ¿Cómo funciona?

Instantáneas != Diferencias



Git



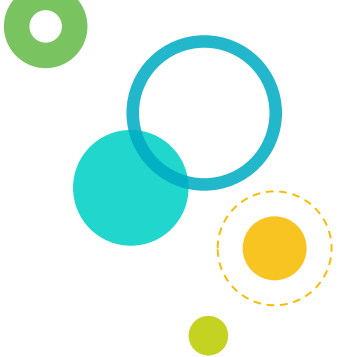
# Git: ¿Cómo funciona?

## Operaciones Locales (casi todas)

La mayoría de las operaciones en Git sólo necesitan archivos y recursos locales para operar.\*

Las operaciones parecen inmediatas.\*

Por ejemplo, para navegar por la historia del proyecto, Git no necesita salir al servidor para obtener la historia y mostrártela, simplemente la lee directamente de tu base de datos local.\*



# Git: ¿Cómo funciona?

## Integridad de Datos (Checksum)

Todo en Git es verificado mediante una suma de comprobación antes de ser almacenado, y es identificado a partir de ese momento mediante dicha suma.\*

Hash: una cadena de 40 caracteres hexadecimales (0-9 y a-f), y se calcula con base en los contenidos del archivo o estructura de directorios.\*

24b9da6552252987aa493b52f8696cd6d3b00373



## Los tres estados

Git tiene tres estados principales en los que se pueden encontrar tus archivos:

- ⦿ Modificado (modified)
- ⦿ Preparado (staged)
- ⦿ Confirmado (committed)



## Los tres estados

© Modificado (modified)

El archivo ha sido modificado pero todavía no se ha confirmado a la base de datos.



## Los tres estados

© Preparado (staged)

El archivo modificado ha sido marcado en su versión actual para que pueda ser confirmado.

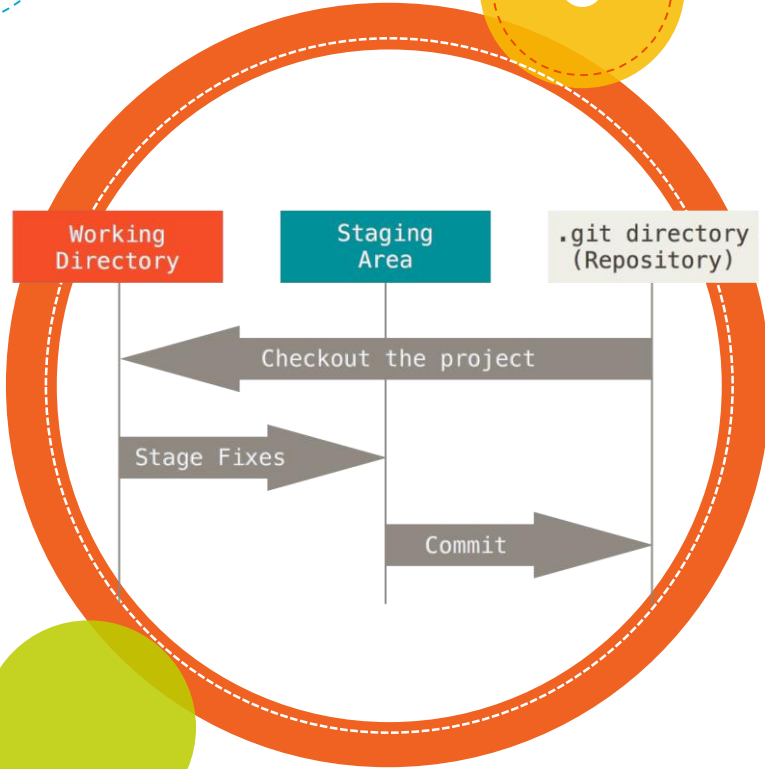




## Los tres estados

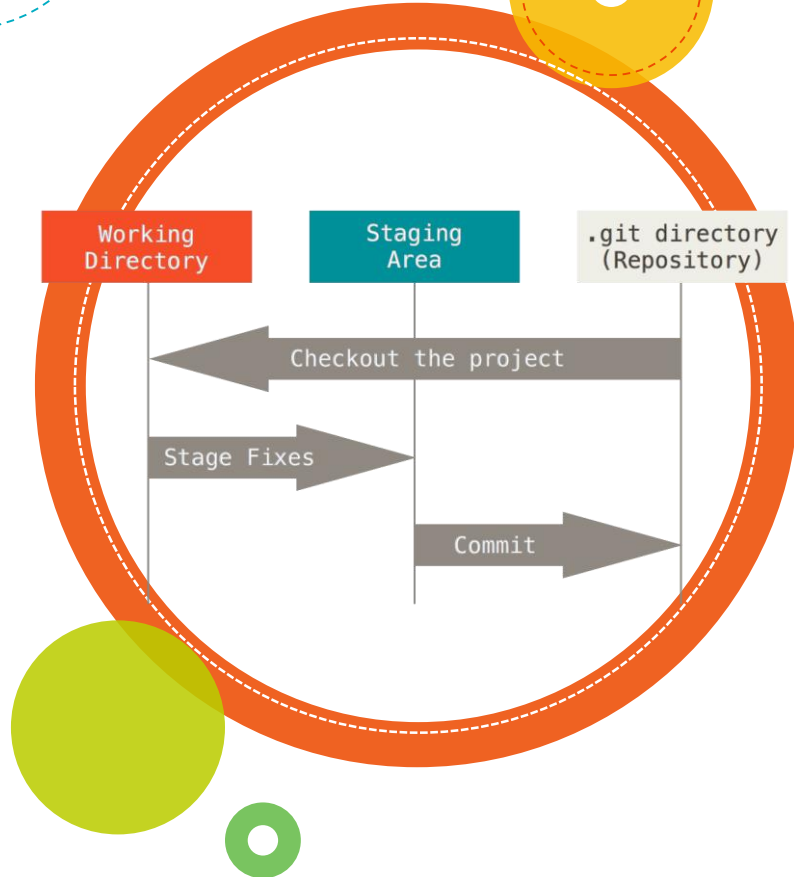
© Confirmado (committed)

Los datos están almacenados de manera segura en la base de datos local.\*



Esto nos lleva a las tres secciones principales de un proyecto de **Git**

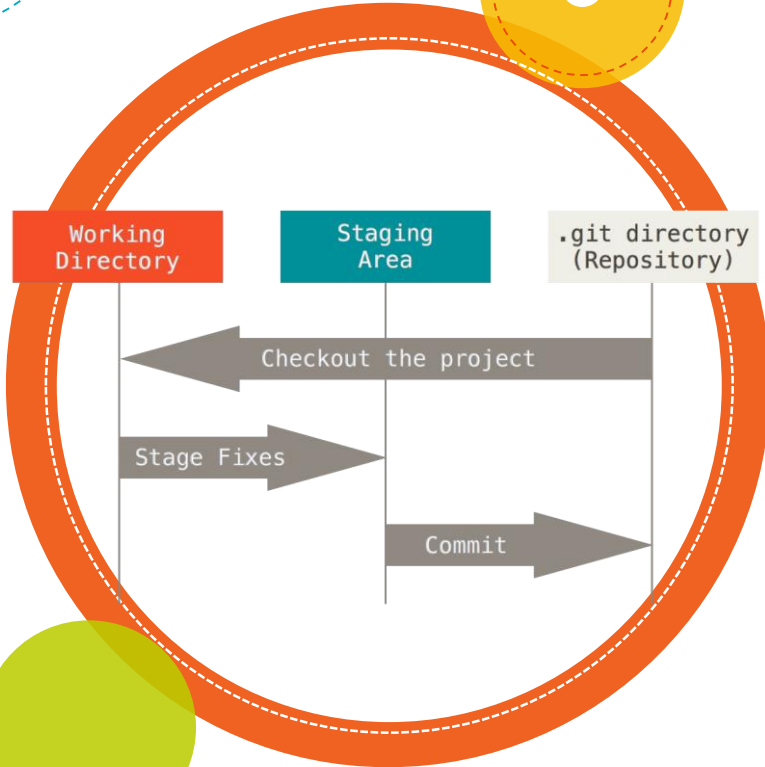
- ◎ El directorio de Git (**Git directory**).
- ◎ El directorio de trabajo (**working directory**).
- ◎ El área de preparación (**staging area**).



## ◎ El directorio de Git (Git directory).

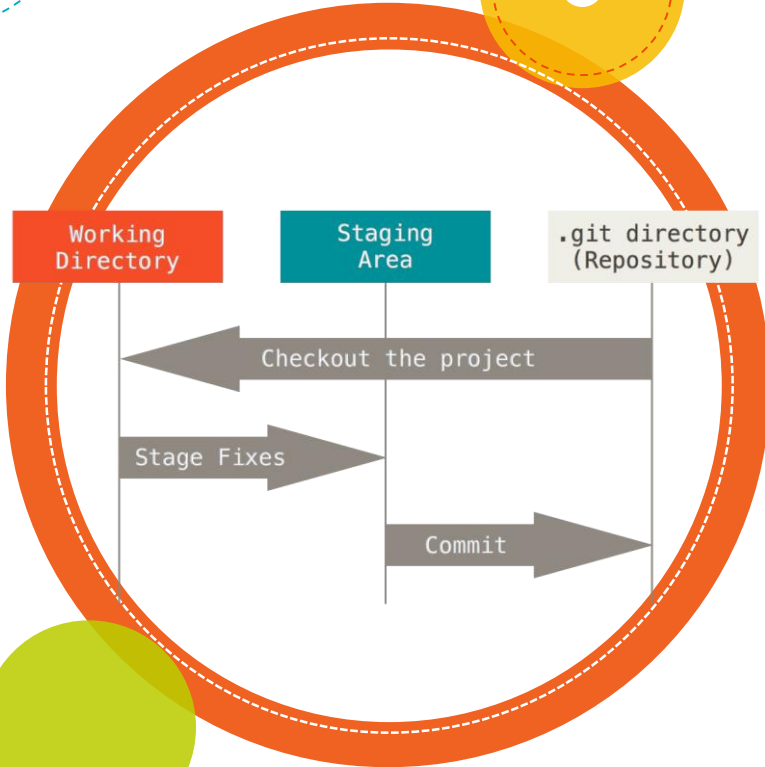
Git almacena los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando se clona un repositorio desde otro ordenador.\*

`/.git`



## ◎ El directorio de trabajo (working directory).

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 se puedan usar o modificar.\*



◎ El área de preparación (staging area).

Es un archivo, generalmente contenido en el **directorio de Git**, que almacena información acerca de lo que va a ir en la próxima confirmación.\*

**`/.git/index`**

# Flujo de Información



El flujo de trabajo básico en **Git** sería:

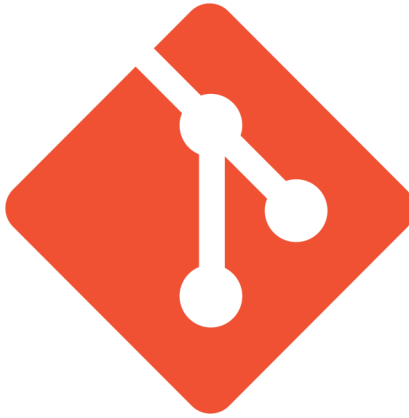


- ◎ Se modifican archivos en el directorio de trabajo.
- ◎ Se marcan los archivos modificados agregándolos al área de preparación.
- ◎ Se confirman los cambios. (Se crea una nueva versión permanente en el directorio de Git).



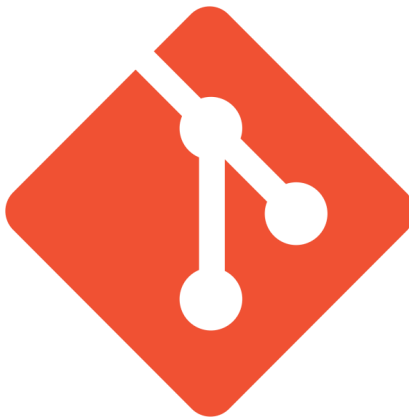
- © Se modifican archivos en el directorio de trabajo.
- © `$> git add file`
- © `$> git commit -m "mensaje"`





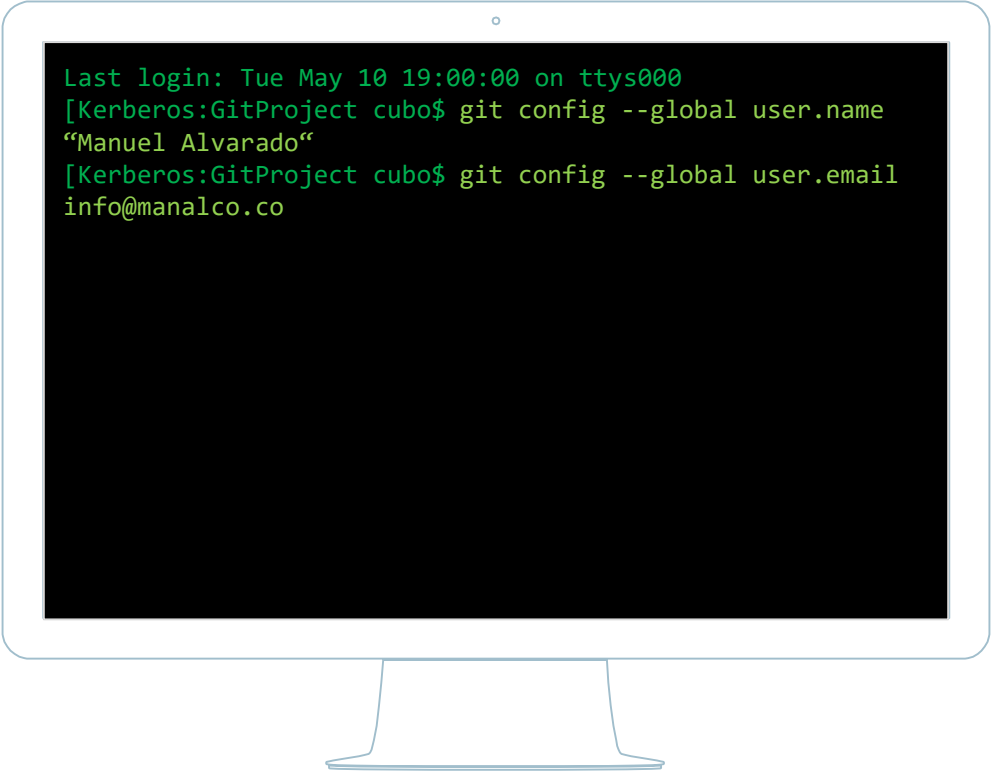
# ¿Cómo se usa?

## Comandos Vs GUI



¿Cómo se usa?

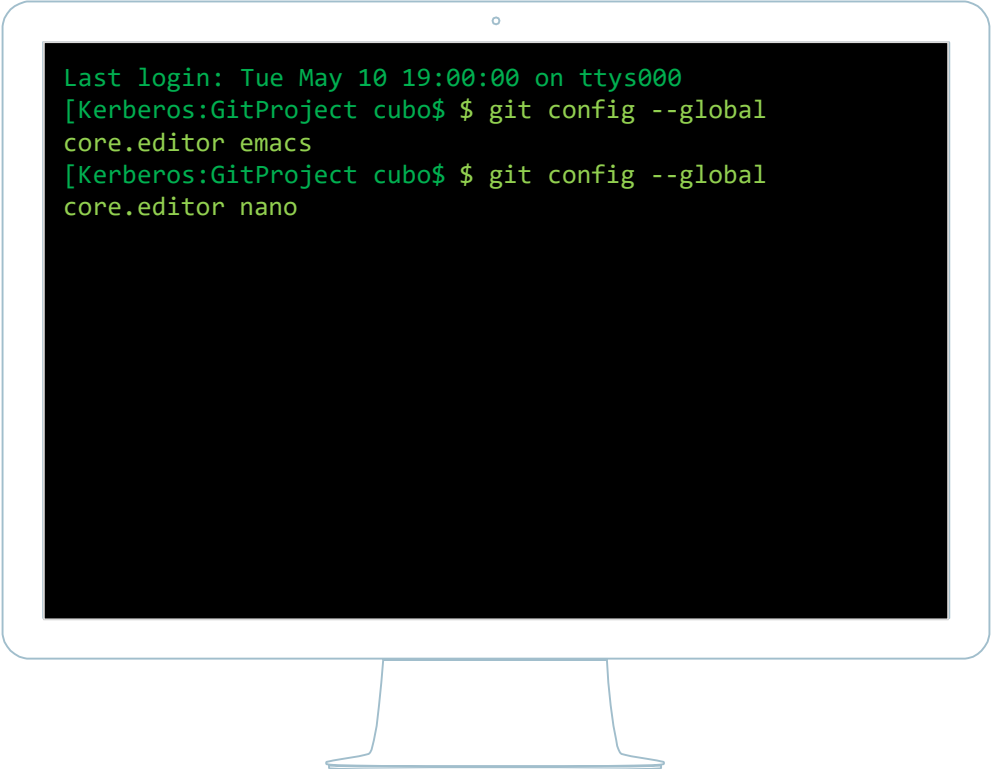
With Git commands you can use the `--force`

A computer monitor with a thin blue border and a stand, displaying terminal output on a black background. The text is in a green monospaced font. The output shows a successful login and the execution of two git config commands to set the user name and email.

```
Last login: Tue May 10 19:00:00 on ttys000
[Kerberos:GitProject cubo$ git config --global user.name
"Manuel Alvarado"
[Kerberos:GitProject cubo$ git config --global user.email
info@manalco.co
```

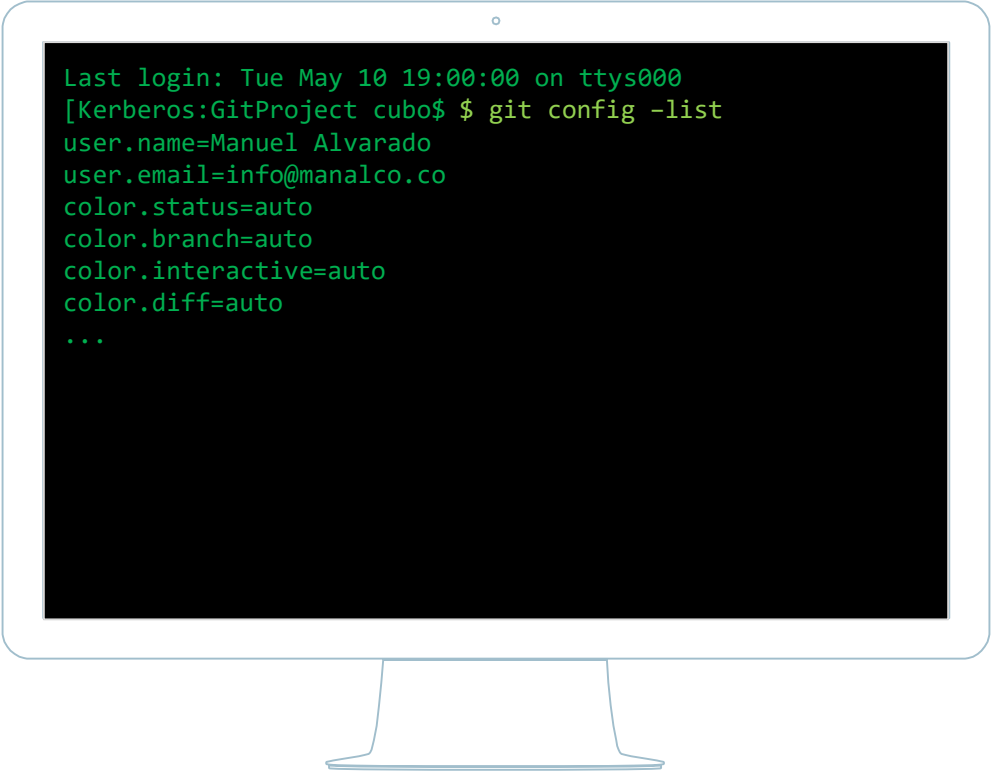
## Comandos para empezar Configurar el usuario

Esto te identifica en servicios de repositorios como GitHub.



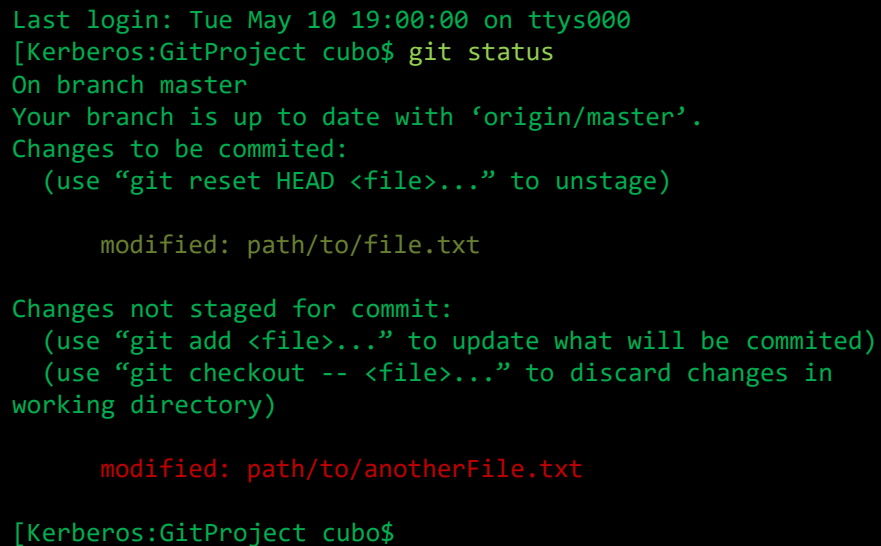
```
Last login: Tue May 10 19:00:00 on ttys000
[Kerberos:GitProject cubo$ $ git config --global
core.editor emacs
[Kerberos:GitProject cubo$ $ git config --global
core.editor nano
```

Comandos para empezar  
Configurar el editor  
Generalmente es **Vi** o **Vim**.

A computer monitor with a thin blue border and a small circle at the top center. The screen is black and displays green text from a terminal session. The text shows a login message, a command prompt, and the output of the 'git config -list' command, which lists various configuration settings like user.name, user.email, and color options. The monitor is supported by a simple stand.

```
Last login: Tue May 10 19:00:00 on ttys000
[Kerberos:GitProject cubo$ $ git config -list
user.name=Manuel Alvarado
user.email=info@manalco.co
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
...
```

Comandos para empezar  
Listar las configuraciones



```
Last login: Tue May 10 19:00:00 on ttys000
[Kerberos:GitProject cubo$ git status
On branch master
Your branch is up to date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    modified:   path/to/file.txt

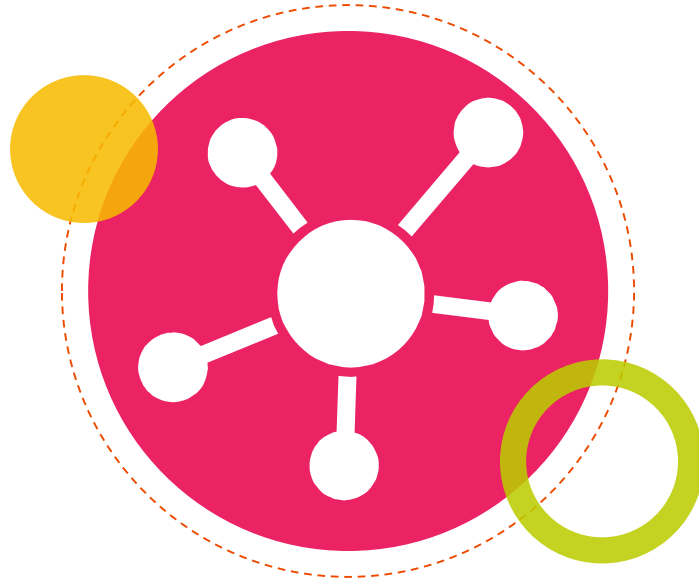
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in
working directory)

    modified:   path/to/anotherFile.txt

[Kerberos:GitProject cubo$
```

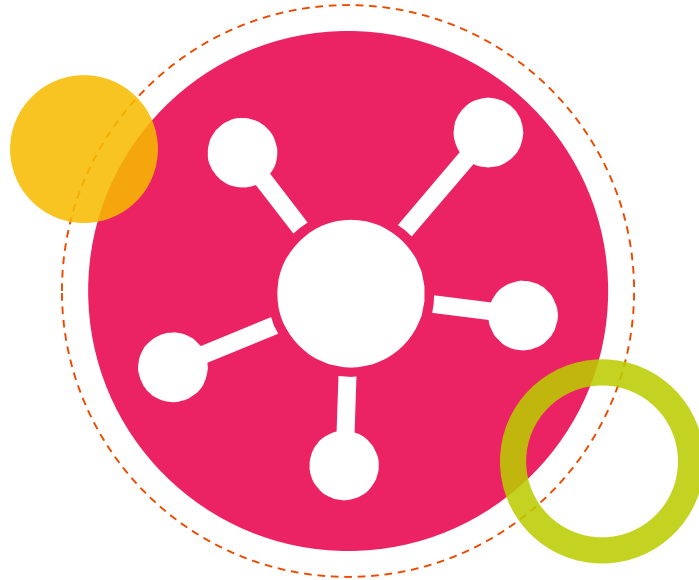
Comandos para empezar  
Colores

# Ramas (branches)



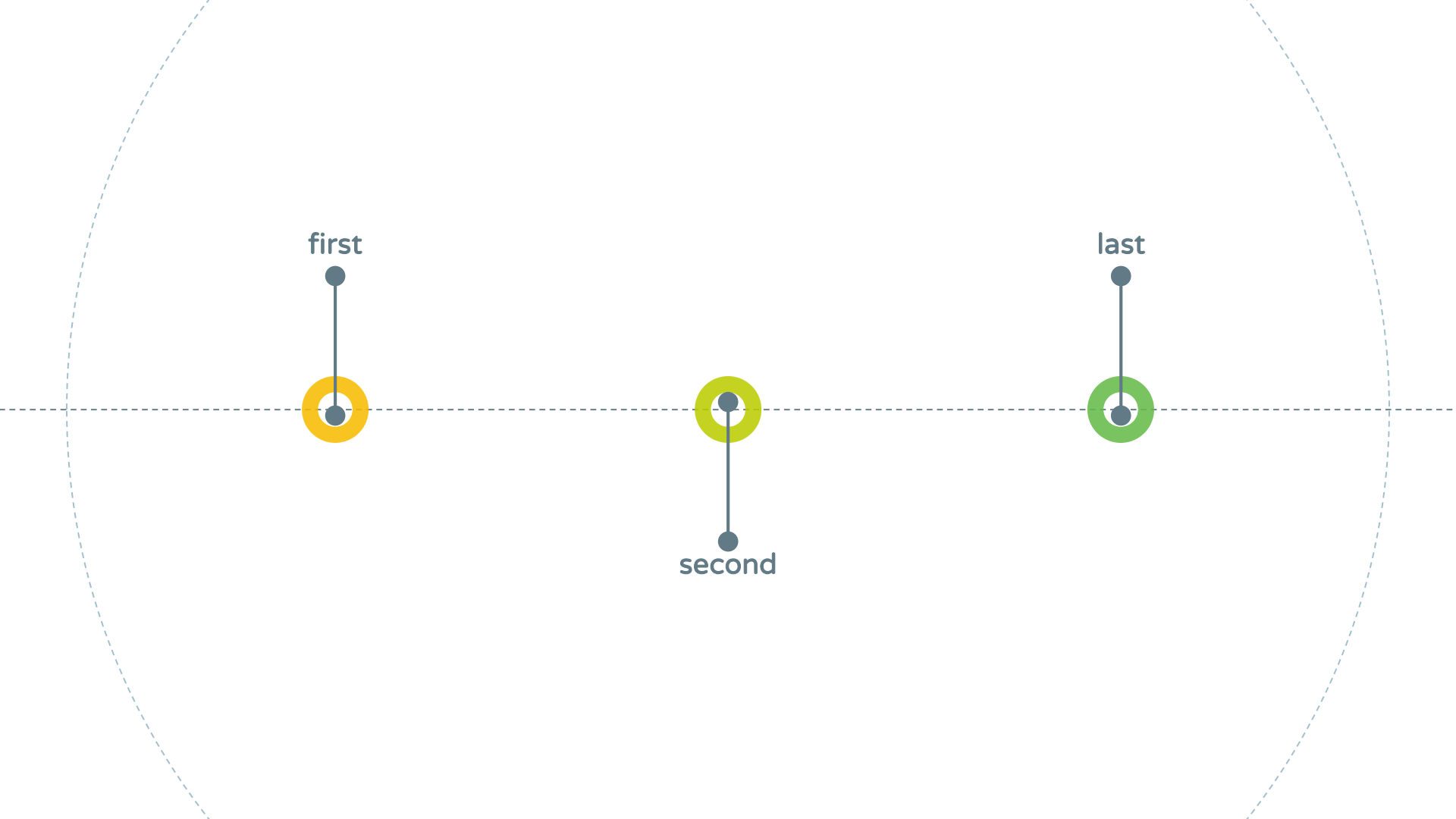
Cuando hablamos de ramificaciones, significa que tu has tomado la rama principal de desarrollo (**master**) y a partir de ahí has continuado trabajando sin seguir la rama principal de desarrollo.\*

# Ramas (branches)

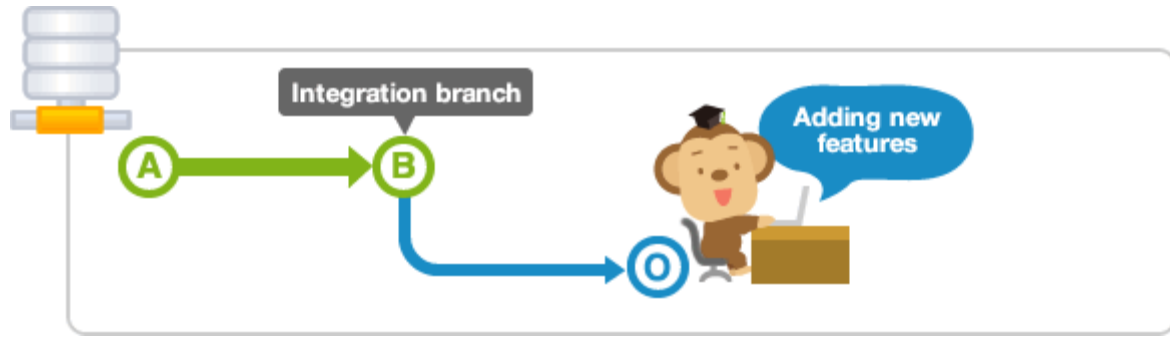


Git promueve un ciclo de desarrollo donde las ramas se crean y se unen entre sí, incluso varias veces en el mismo día.\*

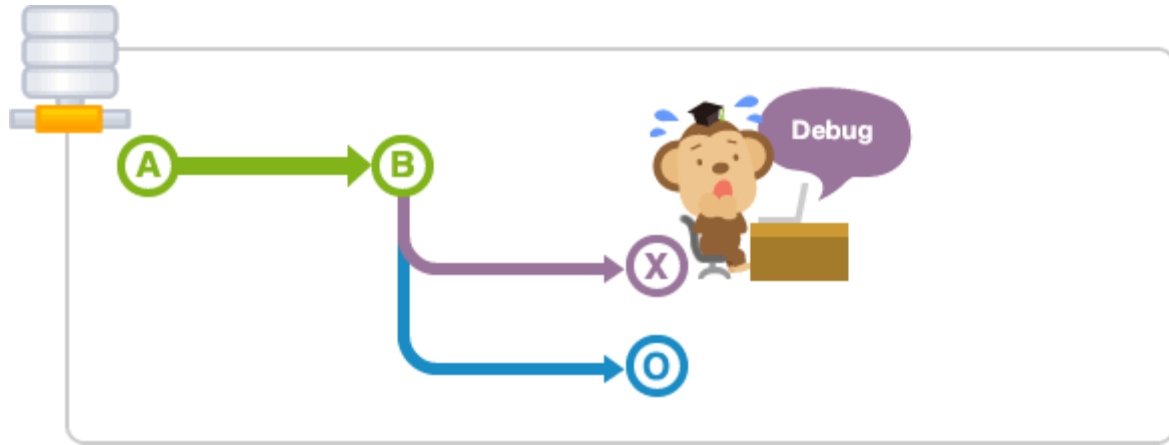




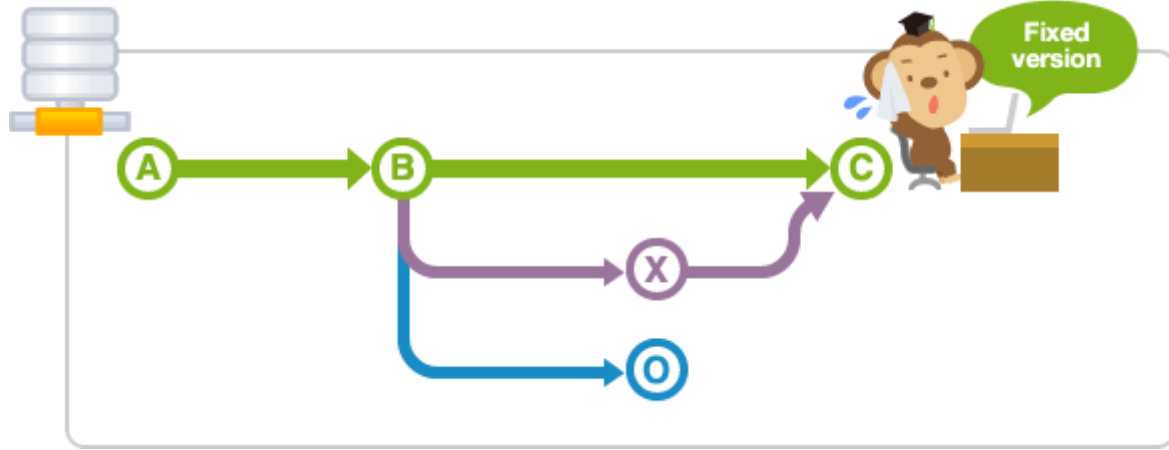
# Ramas (branches)



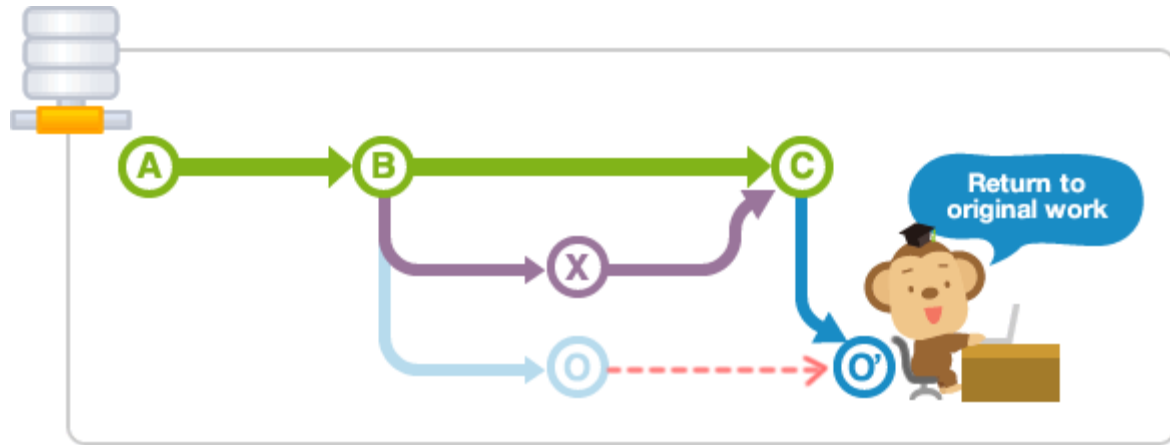
# Ramas (branches)



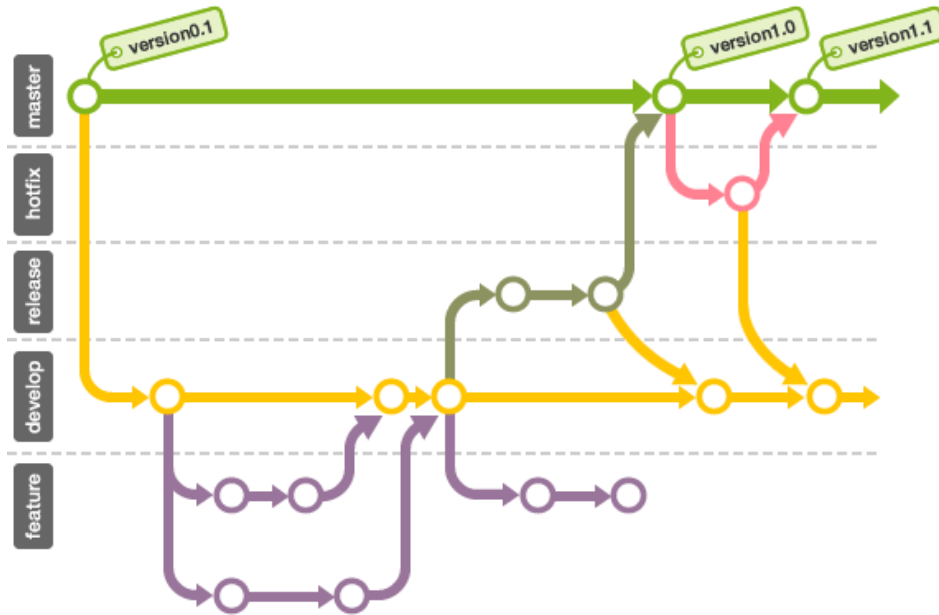
# Ramas (branches)



# Ramas (branches)



# Ramas (branches)



# Git Beginner's Guide for Dummies



<https://backlogtool.com/git-guide/>



Git hands on it!

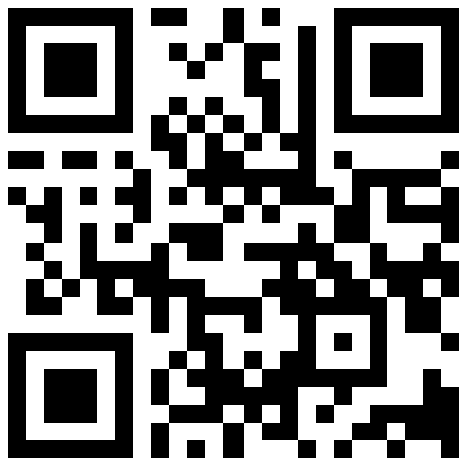


echo “¡Gracias!”;

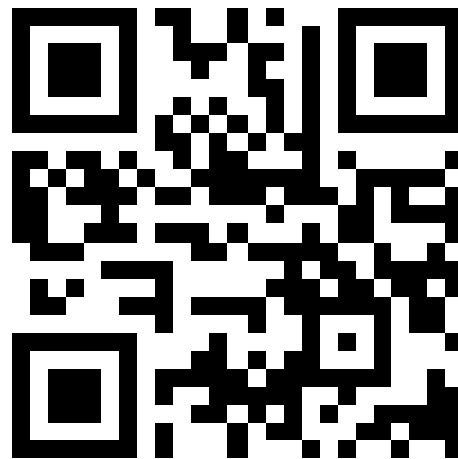


**Contacto:**

@cubosensei & info@manalco.co

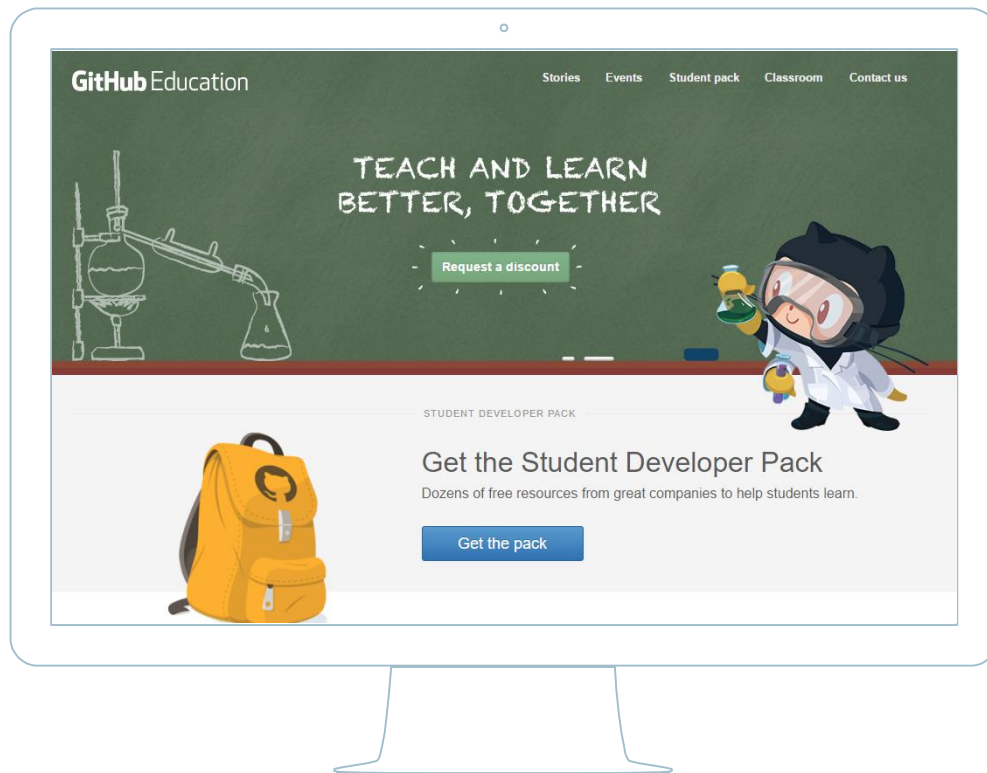


Pro Git V1 Español



Pro Git V2 Inglés

<http://git-scm.com/book>



GitHub Student Pack  
<https://education.github.com>



GitHub Student Pack  
<https://education.github.com>



Referencia

\* **Pro Git, Second Edition**

Scott Chacon & Ben Straub

