

# Hablemos de git

Introducción al uso de git

# ¿Qué es Git?\*

## Git es un sistema de control de versiones

- Es open source desarrollado por Linus Torvalds en 2005
- El sistema de control de versiones más usado actualmente

# ¿Qué es un sistema de control de versiones?

**Un software de utilería que ayuda a rastrear y manejar la historia de cambios de un equipo de colaboradores trabajando en un mismo proyecto de desarrollo de software**

En la historia de cambios se puede encontrar:

- Qué cambios se hicieron
- Quién hizo el cambio
- Cuándo se hicieron los cambios
- Por qué se necesitan los cambios

# DVCS you say

- A diferencia de CVS y Subversion, Git mantiene el historial de versiones en todas las copias del sistema en desarrollo (Distributed Version Control System -DVCS)
- DVCSs no necesitan una conexión constante al repositorio central
- Las colaboraciones pueden ser asíncronas

# Beneficios de usar control de versiones

- Corrección de errores sin interrumpir al equipo de desarrollo
  - Si se comete un error, el desarrollador puede volver a la versión anterior (o ante-anterior) y corregir el error cometido
- Permite el trabajo concurrente
  - Mediante la utilización de ramas (branches)
  - Manejo de conflictos
- Genera transparencia

# Let's start

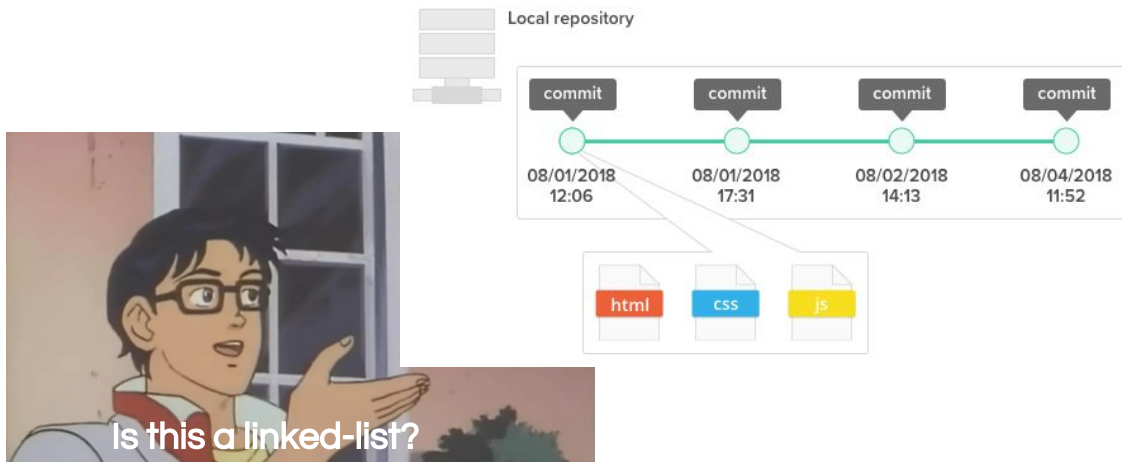
- Crear un repositorio
- Rastrear cambios
- Guardar el historial de cambios
- Hacer ramas
- Unir ramas
- Repositorios remotos
- Trabajo colaborativo

---

# Repositorio

Un repositorio (o proyecto Git) contiene todos los archivos y carpetas asociadas con ese proyecto, así como la historia de cambios de CADA archivo.

La historia de cada archivo puede verse como instantáneas (snapshots) en el tiempo: commits

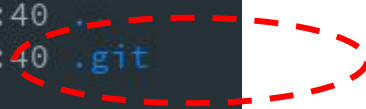


# Hacer un repositorio local

1. Hacer una carpeta (la que contendrá el repositorio)
2. Moverse a la carpeta de trabajo
3. Con el comando `git init`
  - a. Se escribe una sola vez, para indicar que se inicializará el repositorio
  - b. Esto crea una rama maestra (master branch)

```
[gabriela@oppy holamundo]$ git init
Initialized empty Git repository in /home/gabriela/Documents/Docencia/TallerGit/holamundo/.git/
```

```
[gabriela@oppy holamundo]$ ls -all
total 12
drwxrwxr-x. 3 gabriela gabriela 4096 Mar  9 15:40 .
drwxrwxr-x. 3 gabriela gabriela 4096 Mar  9 15:40 ..
drwxrwxr-x. 7 gabriela gabriela 4096 Mar  9 15:40 .git
[gabriela@oppy holamundo]$
```





# Add and commit - ejemplo

1. Crear un archivo de texto llamado *holamundo.txt*
2. Agregar la linea “Hola mundo, mi nombre es <Gabriela>.”
3. Verificar el estatus del repositorio con el comando **git status**

```
[gabriela@oppy holamundo]$ ls
HolaMundo.txt
[gabriela@oppy holamundo]$ git status
On branch master

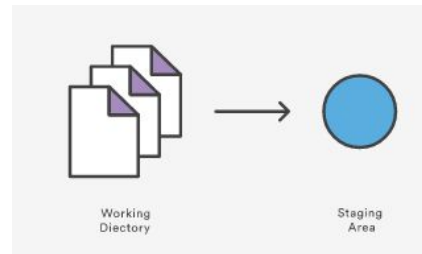
No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    HolaMundo.txt

nothing added to commit but untracked files present (use "git add" to track)
[gabriela@oppy holamundo]$
```

# Add and commit

4. Para indicar a git que queremos rastrear (track) los cambios del nuevo archivo hacemos: `git add HolaMundo.txt`
  - a. Esto lleva los cambios del directorio actual al **área de ensayo** de Git (staging area)
  - b. El **área de ensayo** es donde se prepara lo que será el snapshot del conjunto de cambios antes de hacer un commit
5. Vemos el estado del repositorio
  - a. Los cambios ahora están “staged”



```
[gabriela@oppy holamundo]$ git add HolaMundo.txt
[gabriela@oppy holamundo]$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   HolaMundo.txt

[gabriela@oppy holamundo]$
```

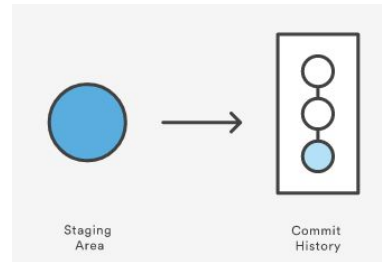
# Add and commit

6. Ahora vamos a *commit* los cambios hechos en el repositorio: `git commit -m`

## “Commit inicial”

- Cada commit debe llevar un mensaje que describa lo que se está agregando al repositorio (algunas ideas de cómo escribir buenos mensajes <https://chris.beams.io/posts/git-commit/>)

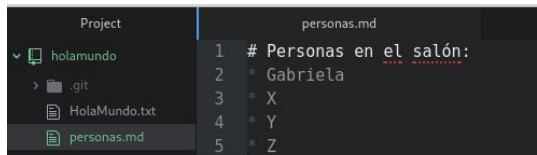
7. Vemos el estado del repositorio



```
[gabriela@oppy holamundo]$ git commit -m "Commit inicial"
[master (root-commit) cac546f] Commit inicial
1 file changed, 1 insertion(+)
 create mode 100644 HolaMundo.txt
[gabriela@oppy holamundo]$ git status
On branch master
nothing to commit, working tree clean
[gabriela@oppy holamundo]$
```

# Add and commit

8. Agregamos otro archivo al repositorio y lo llamamos `personas.md` con una lista de las personas que asistieron hoy
9. Staged and commit
10. Ver es estado del repositorio
11. Usar `git log` para ver el historial de cambios realizados



```
[gabriela@oppy holamundo]$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  personas.md

nothing added to commit but untracked files present (use "git add" to track)
[gabriela@oppy holamundo]$ git add personas.md
[gabriela@oppy holamundo]$ git commit -m "Agrega lista de personas presentes"
[master 5eb3ab1] Agrega lista de personas presentes
1 file changed, 5 insertions(+)
create mode 100644 personas.md
[gabriela@oppy holamundo]$ git status
On branch master
nothing to commit, working tree clean
[gabriela@oppy holamundo]$ git log
commit 5eb3ab1a4460f50e720d66599fd345712e416370 (HEAD -> master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 19:57:26 2020 -0600

    Agrega lista de personas presentes

commit cac546f558563e1f3b8134ad960f5fe2c7606edc
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 16:43:09 2020 -0600

    Commit inicial
[gabriela@oppy holamundo]$
```

# El historial

```
[gabriela@oppy holamundo]$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    personas.md

nothing added to commit but untracked files present (use "git add" to track)
[gabriela@oppy holamundo]$ git add personas.md
[gabriela@oppy holamundo]$ git commit -m "Agrega lista de personas presentes"
[master 5eb3ab1] Agrega lista de personas presentes
 1 file changed, 5 insertions(+)
 create mode 100644 personas.md
[gabriela@oppy holamundo]$ git status
On branch master
nothing to commit, working tree clean
[gabriela@oppy holamundo]$ git log
commit 5eb3ab1a4460f50e720d66599fd345712e416370 (HEAD -> master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date:   Mon Mar 9 19:57:26 2020 -0600

    Agrega lista de personas presentes

commit cac546f558563e1f3b8134ad960f5fe2c7606edc
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date:   Mon Mar 9 16:43:09 2020 -0600

    Commit inicial
[gabriela@oppy holamundo]$
```

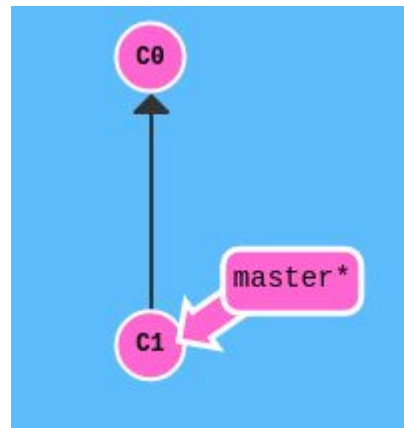
# Errores

# Master branch

En general, las ramas (branches) son **punteros** a commit específicos

- Cada commit es un nodo en la lista ligada de la historia de cambios
- El branch master siempre apunta al commit actual

Dos commits, c0, el inicial el actual, c1:



# Branching locally

**Escenario:** Creamos el proyecto, agregamos archivos al área de ensayo, y luego a la historia de cambios, repetimos esto varias veces hasta que el proyecto hace lo que debe hacer.

Ahora necesitamos hacer una *mejora* a nuestro proyecto que ya funciona. Hay dos opciones:

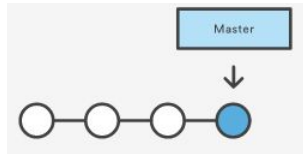
- a) Modificar el proyecto que ya funciona y arriesgarnos a que ya no funcione al hacer modificaciones
- b) Hacer una rama independiente que parte del nodo actual en la historia de cambios



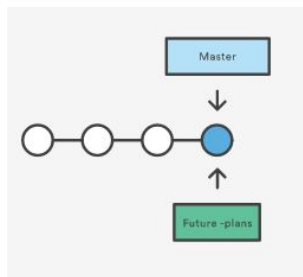
# Branching locally

Recuerden que una **rama** es un **puntero** a un commit

0. El repositorio luce así después de 4 commits



1. En el directorio de trabajo, crear una nueva rama con el comando **git branch future-plan** que contendrá el trabajo futuro que se le hará al repositorio

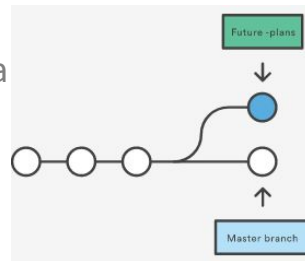


- Esto creará el puntero Future plants pero no nos hemos cambiado a esa área de trabajo
- Esto NO copia archivos o directorios

2. Vemos las ramas que tenemos actualmente desde línea de comandos con **git branch**

```
-1
[gabriela@oppy holamundo]$ git branch -l
future-plans
* master
[gabriela@oppy holamundo]$
```

3. Para cambiarse a la rama que acabamos de crear y empezar a usarla, hacemos **git checkout future-plans**:



```
[gabriela@oppy holamundo]$ git checkout future-plans
Switched to branch 'future-plans'
[gabriela@oppy holamundo]$
```

# Branching locally

- Al hacer una rama, se copia todo el historial de commits hasta este punto, y el puntero HEAD cambiará al branch actual. Usa `git log` para ver más información:
- Al archivo `personas.md` vamos a actualizarlo con los nombre reales de las personas presentes, en lugar de X, Y y Z pondremos Ángeles, Brenda, Gabriela, Tonantzin.
- Guardar, staged & commit (“Lista personas presentes en el salón”)

```
[gabriela@oppy holamundo]$ git log
commit 5eb3ab1a4460f50e720d66599fd345712e416370 (HEAD -> future-plans, master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 19:57:26 2020 -0600

    Agrega lista de personas presentes

commit cac546f558563e1f3b8134ad960f5fe2c7606edc
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 16:43:09 2020 -0600

    Commit inicial

[gabriela@oppy holamundo]$
```

```
[gabriela@oppy holamundo]$ git commit -m "Lista personas presentes en el salón"
[future-plans 1943871] Lista personas presentes en el salón
1 file changed, 3 insertions(+), 3 deletions(-)
[gabriela@oppy holamundo]$ git log
commit 1943871f9fcf05521977ec2cd2d5c6752eaa272 (HEAD -> future-plans)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 20:12:20 2020 -0600

    Lista personas presentes en el salón

commit 5eb3ab1a4460f50e720d66599fd345712e416370 (master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 19:57:26 2020 -0600

    Agrega lista de personas presentes

commit cac546f558563e1f3b8134ad960f5fe2c7606edc
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date: Mon Mar 9 16:43:09 2020 -0600

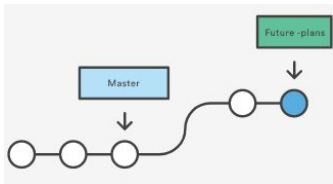
    Commit inicial

[gabriela@oppy holamundo]$
```

# Branching locally

Ya hicimos modificaciones al proyecto, ahora queremos agregar esta funcionalidad al proyecto principal (master)

Los cambios que hemos hechos generan una árbol lineal

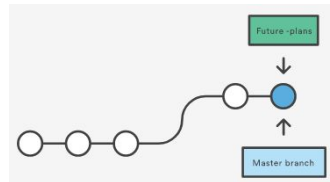


La unión es simple, sólo hay que cambiar el *master* al commit de *future-plans*. Esto se conoce como *fast-forward merging*

7. Nos cambiamos a la rama master (git checkout master)
  - a. Para verificar que lo que hicimos en future-plans no está en master, ver el estatus ahora
8. Unimos los cambios que hicimos en future-plans con el comando git merge future-plans

```
[gabriela@oppy holamundo]$ git merge future-plans
Updating 5eb3ab1..1943871
Fast-forward
 personas.md | 6 +++---
 1 file changed, 3 insertions(+), 3 deletions(-)
[gabriela@oppy holamundo]$
```

9. Borramos future-plans con el comando **git branch -d future-plans**



# GitHub

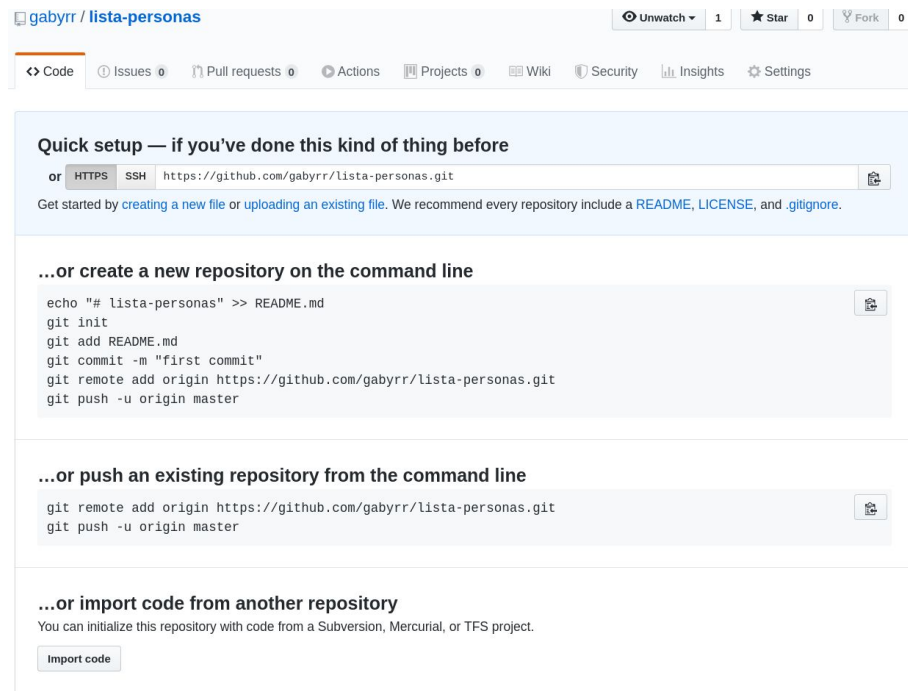
Usaremos GitHub, pero existen otras plataformas como bitbucket (<https://bitbucket.org/account/signup/>) que utilizan git.

1. Crear cuenta en github.com
2. Seguir el tutorial: Hello world (<https://guides.github.com/activities/hello-world/>)
  - a. Esto hace todo en la plataforma, en linea.

# Crear un repositorio remoto a partir de un repositorio local

1. Ir a github.com
2. Agregar repositorio “lista-personas”
3. Copiar y pegar instrucciones para agregar un repositorio existente
4. Done!

Ahora el repositorio local (helloworld) y el repositorio remoto (lista-personas) contienen la misma información



The screenshot shows the GitHub interface for a repository named 'lista-personas' by user 'gabyrr'. The repository has 1 unwatch, 0 stars, and 0 forks. The 'Code' tab is selected, showing the 'Quick setup' section. The setup options are:

- Quick setup — if you've done this kind of thing before**
  - or **HTTPS** **SSH** `https://github.com/gabyrr/lista-personas.git`
  - Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).
- ...or create a new repository on the command line**

```
echo "# lista-personas" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/gabyrr/lista-personas.git
git push -u origin master
```
- ...or push an existing repository from the command line**

```
git remote add origin https://github.com/gabyrr/lista-personas.git
git push -u origin master
```
- ...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

# Change locally and push changes

1. Localmente, borrar el archivo HolaMundo.txt
2. Para mantener el cambio del área de trabajo, hay que actualizar el área de ensayo (stage area), hacemos git rm HolaMundo.txt
3. Commit
4. Ver el log
5. Ahora nuestro repositorio local tiene un nodo adicional al repositorio remoto

```
[gabriela@oppy holamundo]$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
[gabriela@oppy holamundo]$
```

```
[gabriela@oppy holamundo]$ rm HolaMundo.txt
[gabriela@oppy holamundo]$ ls
personas.md
[gabriela@oppy holamundo]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    HolaMundo.txt

no changes added to commit (use "git add" and/or "git commit -a")
[gabriela@oppy holamundo]$ git rm HolaMundo.txt
rm 'HolaMundo.txt'
[gabriela@oppy holamundo]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    HolaMundo.txt

[gabriela@oppy holamundo]$ git commit -m "Delete HolaMundo.txt"
[master 14568eb] Delete HolaMundo.txt
1 file changed, 1 deletion(-)
delete mode 100644 HolaMundo.txt
[gabriela@oppy holamundo]$ git log
commit 14568eb1f1cd78b4376cd3d3254fc53b39814b69 (HEAD -> master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date:   Mon Mar 9 21:53:24 2020 -0600

    Delete HolaMundo.txt

commit 1943871f9fcf05521977ec2cd2d5c6752eeaa272 (origin/master)
Author: Gabriela Ramirez <gabriela@oppy.mars>
Date:   Mon Mar 9 20:12:20 2020 -0600
```

# Change locally and push changes

1. Hay que “empujar” los cambios al repositorio remoto con el comando `git push`
2. Ahora todo está en la misma versión

# Proyecto colaborativo

Seguiremos este tutorial juntos: [The Ultimate Github Collaboration Guide - Jonathan Mines](#)

---



# Clone repositorio

1. Clonar mi repositorio  
<https://github.com/gabyrr/lista-personas>
2. Cada uno de ustedes, hará una nueva rama para agregar nuevos elementos al proyecto
  - a. Ejemplo de nombre de rama puede ser: rama-gabriela
  - b. El objetivo de sus ramas, será llenar la información faltante sobre ustedes en el archivo correos.md
3. En su rama, agregar la información sobre ustedes que falte, stage & commit
4. Push su rama al origen
  - a. Esto generará varias ramas en mi repositorio
5. Comparar y pull request
6. Asignar revisor
7. Merge

# Git everywhere