

**MAKE** **it** **REAL**  
           .camp

# Classroom Questions - de ayer

---

# Temas Dia 3

---

*1- Git y Github*

# Descargar

---

Para descargar e instalar Git:

- Si estás en Mac o Linux ingresa a <https://git-scm.com/download>.
- Si estás en Windows ingresa a <https://desktop.github.com/>.

## *Primera Advertencia*

*Git*



*Github*

## *Control de Versiones:*

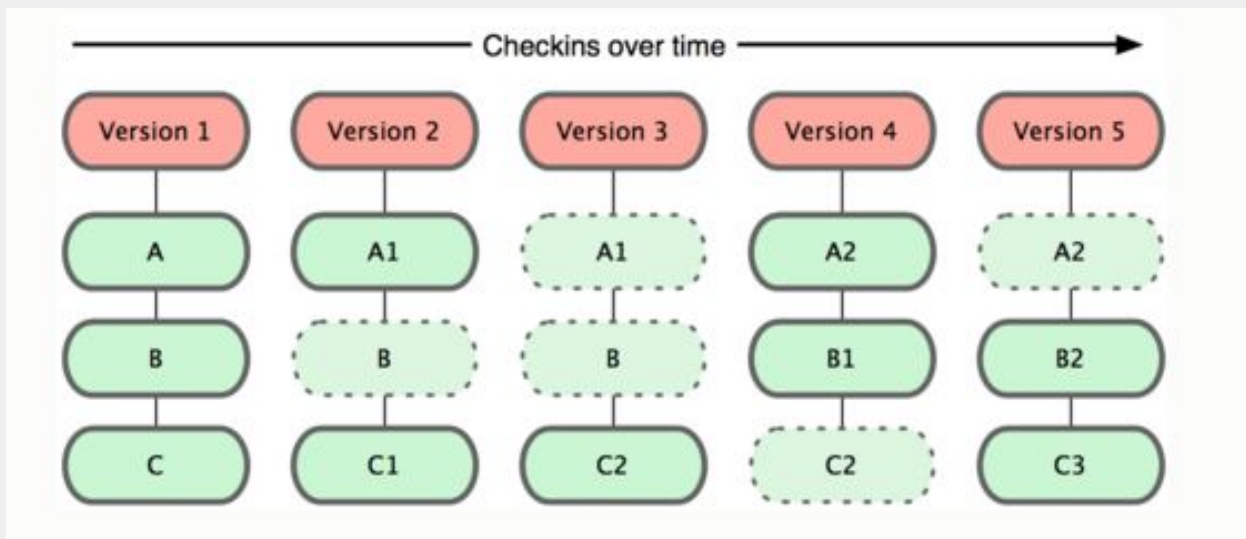
*Es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo*

- google drive example vs word (Microsoft)*

## Una breve historia de Git

Como muchas de las grandes cosas en esta vida, Git comenzó con un poco de destrucción creativa y encendida polémica. El núcleo de Linux es un proyecto de software de código abierto con un alcance bastante grande. Durante la mayor parte del mantenimiento del núcleo de Linux (1991-2002), los cambios en el software se pasaron en forma de parches y archivos. En 2002, el proyecto del núcleo de Linux empezó a usar un DVCS propietario llamado BitKeeper.

# Instantâneas no Diferencias





*Casi cualquier operación es local.*

# *Tiene integridad:*

*imposible cambiar contenido sin que git lo sepa*

40 caracteres hexadecimales (0-9 y a-f), y se calcula en base a los contenidos del archivo o estructura de directorios

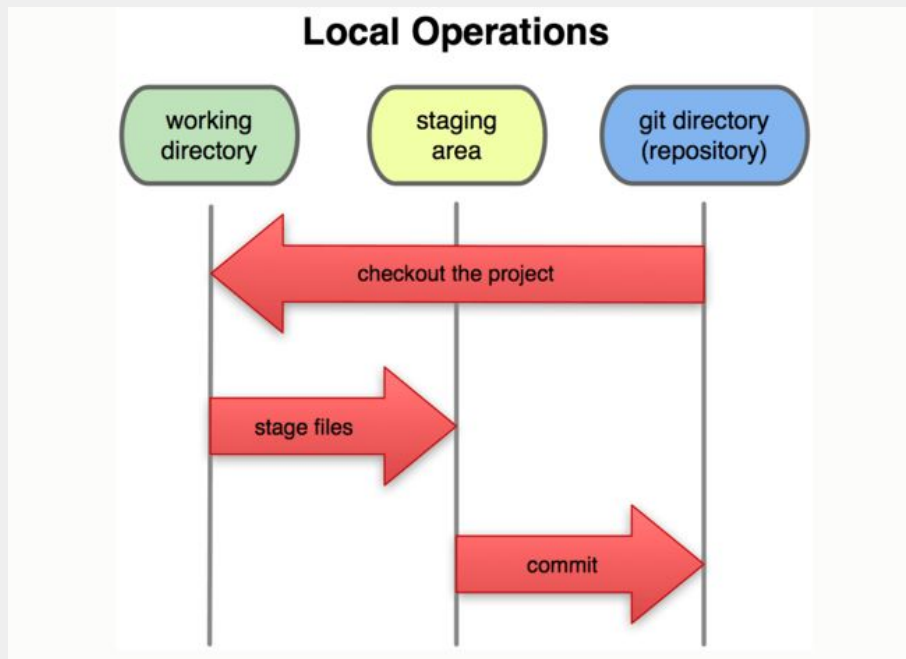
```
24b9da6552252987aa493b52f8696cd6d3b00373
```

# Los 3 estados:

---

- 1- Modificados (modified): significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos*
- 2- Preparado (staged): significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación*
- 3- Confirmado (committed): Los datos están almacenados de forma segura en tu base de datos local.*

# *Las 3 secciones de un proyecto git:*



# *Flujo de Trabajo == Iteración Básica*

---

- 1- Modificas una serie de archivos en tu directorio de trabajo  
(modified)*
- 2- Preparas los archivos, añadiéndolos a tu area de preparacion  
(staged)*
- 3- Confirmas los cambios, lo que toma los archivos tal y como  
están en el area de preparacion, y almacena esas instantaneas  
de manera permanente en tu directorio de Git (committed)*

# Verificando Git

---

*git --version*

# Configurando Git

---

- `git config --global user.name <name>` : define el nombre que se va a utilizar en los commits de forma global (para el usuario actual).
- `git config --global user.email <email>` : define el email que se va a utilizar en los commits de forma global (para el usuario actual).

# Obteniendo ayuda

---

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

Por ejemplo, puedes ver la página del manual para el comando config ejecutando:

```
$ git help config
```



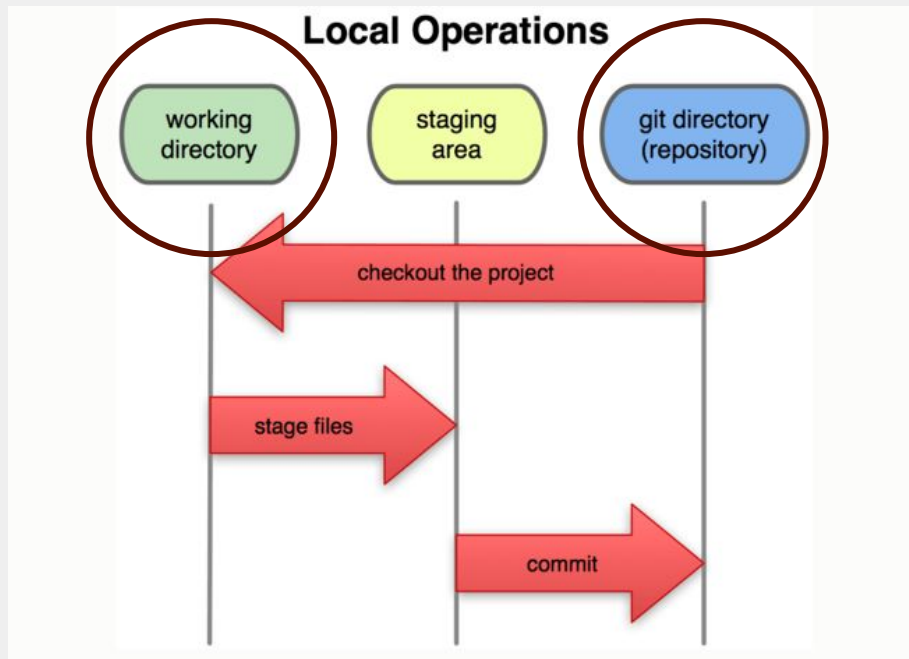
# Inicializando Repositorio

---

- 1- Creamos una carpeta desde Linea de Comandos, llamada: *dia\_3*
- 2- Ingresamos a esa carpeta
- 3-

- `git init`: este comando inicializa el repositorio (esto va a crear una carpeta oculta `.git` en la carpeta donde ejecutes este comando).

# *Las 3 secciones de un proyecto git:*



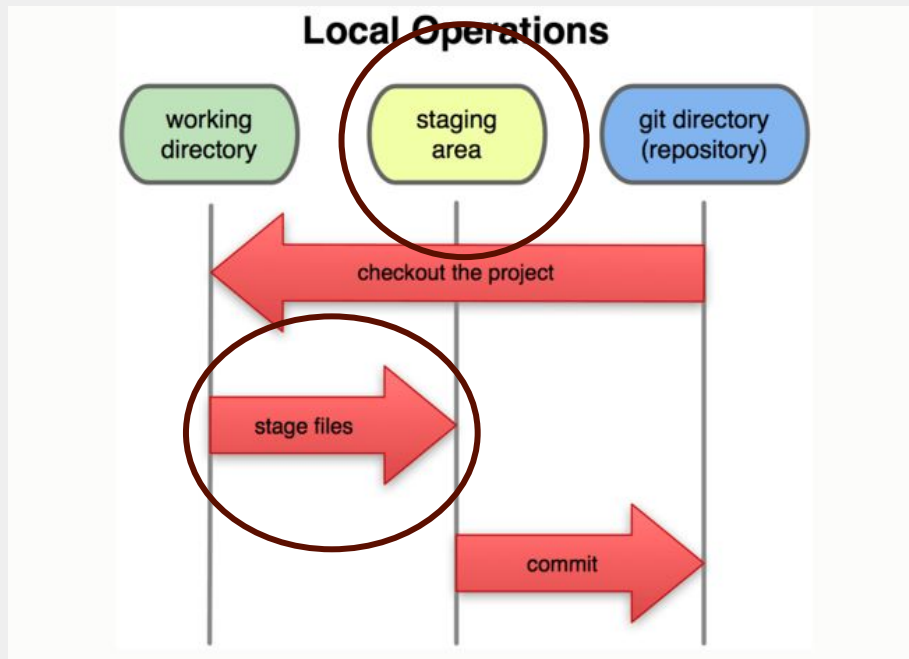
# Inicializando Repositorio

---

- 4- Ejecutamos “git status” - Muestra el estado actual de nuestro espacio de trabajo
- 5- Creamos un archivo llamado “empiezo-con-git.txt”
- 6- Ejecutamos “git status”
- 7- Abrimos archivo y lo modificamos “texto”
- 8-

- `git add .` : prepara los archivos para el `commit` .

# *Las 3 secciones de un proyecto git:*



# Inicializando Repositorio

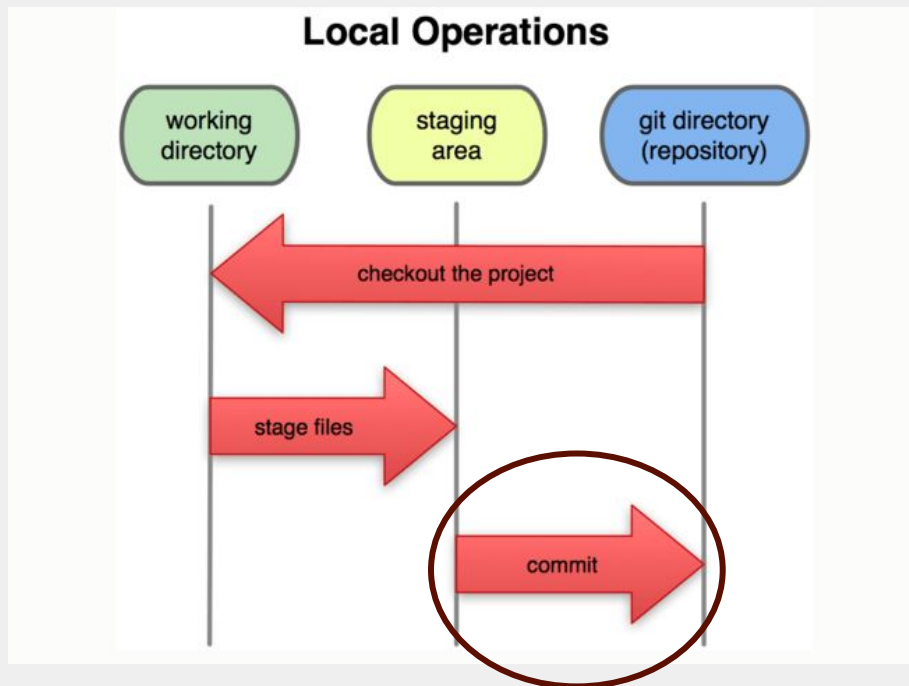
---

## 8- Ejecutamos

- `git commit -m <descripción de los cambios>` : crea un commit a partir de los cambios que están en el index con el mensaje que se le pase a la opción `-m`.

## 9- Ejecutamos “git status”

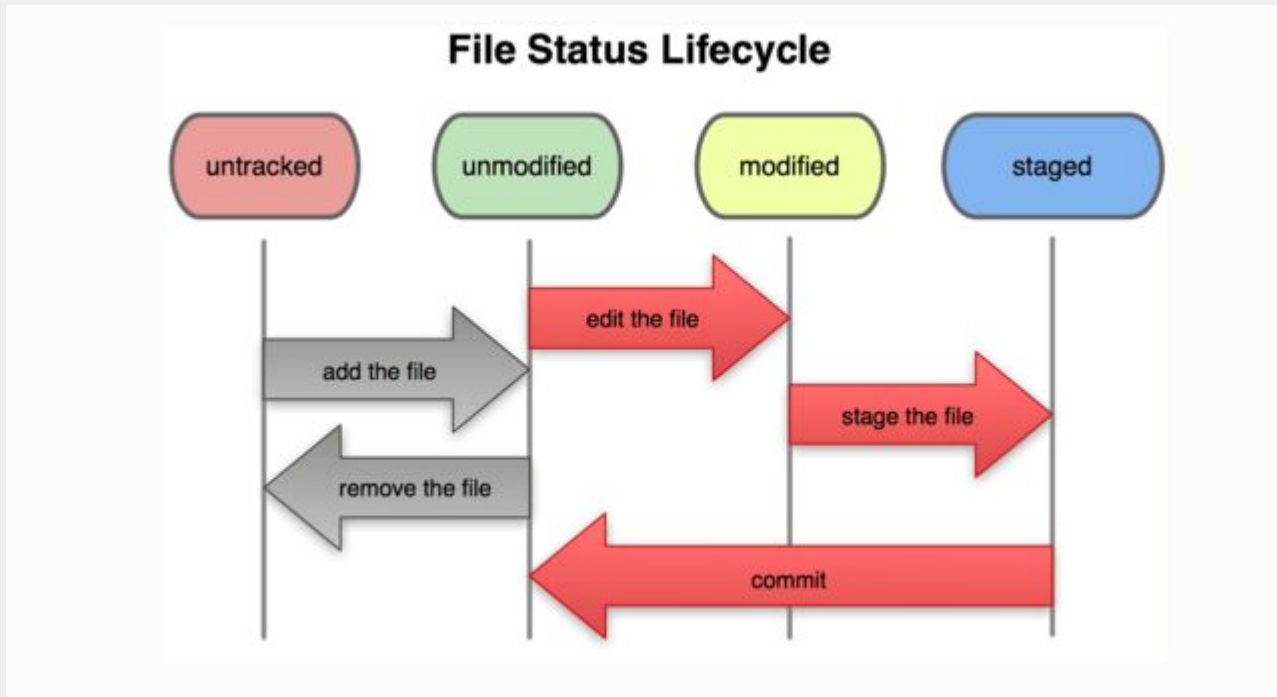
# *Las 3 secciones de un proyecto git:*



## *Ejemplo de Flujo de Trabajo == Iteración Básica*

- 1- Salir de la carpeta anterior, crear una nueva carpeta llamada: *mi-segundo-repo*
- 2- Entrar a esta nueva carpeta y ejecutar el comando *"git init"*
- 3- Crear un archivo llamado: *archivo1.txt*
- 4- Abrir el archivo y escribir: *"Hola mundo con git"* y darle guardar
- 5- Ejecutar el comando *"git status"*
- 6- Ejecutar el comando *"git add ."*
- 7- Ejecutar el comando *"git status"*
- 9- Ejecutar el comando *"git commit -m 'commit inicial hecho por mi'"*
- 10 - Ejecutar el comando *"git status"*

# *Ciclo de Vida del Archivo (status)*





# Classroom Questions

---

# Otros Comandos

---

1- *git diff*

2- *git log*

*Trabajo con Ramificaciones*

3- *git branch*

4- *git checkout*

*Repositorios Remotos*

5- *git fetch, git pull*

6- *git push*

7- *git clone*

*Otros*

- *Trabajar en proyectos*

*Open Source*

- *Trabajar con equipos remotos*

# Viendo cambios

*git diff: muestra exactamente las líneas añadidas y eliminadas. Trabaja muy bien con git status*

Para ver lo que has modificado pero aún no has preparado, escribe `git diff`:

```
$ git diff
diff --git a/benchmarks.rb b/benchmarks.rb
index 3cb747f..da65585 100644
--- a/benchmarks.rb
+++ b/benchmarks.rb
@@ -36,6 +36,10 @@ def main
     @commit.parents[0].parents[0].parents[0]
   end

+  run_code(x, 'commits 1') do
+    git.commits.size
+  end
+
   run_code(x, 'commits 2') do
     log = git.commits('master', 15)
     log.size
```

# Viendo histórico de “commits”

*git log: para mirar atrás y ver qué modificaciones se han llevado a cabo*

Cuando ejecutes `git log` sobre este proyecto, deberías ver una salida similar a esta:

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

# Viendo histórico de “commits”

*git log: para mirar atrás y ver qué modificaciones se han llevado a cabo*

Cuando ejecutes `git log` sobre este proyecto, deberías ver una salida similar a esta:

```
$ git log
commit ca82a6dff817ec66f44342007202690a93763949
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Mon Mar 17 21:52:11 2008 -0700

    changed the version number

commit 085bb3bcb608e1e8451d4b2432f8ecbe6306e7e7
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 16:40:33 2008 -0700

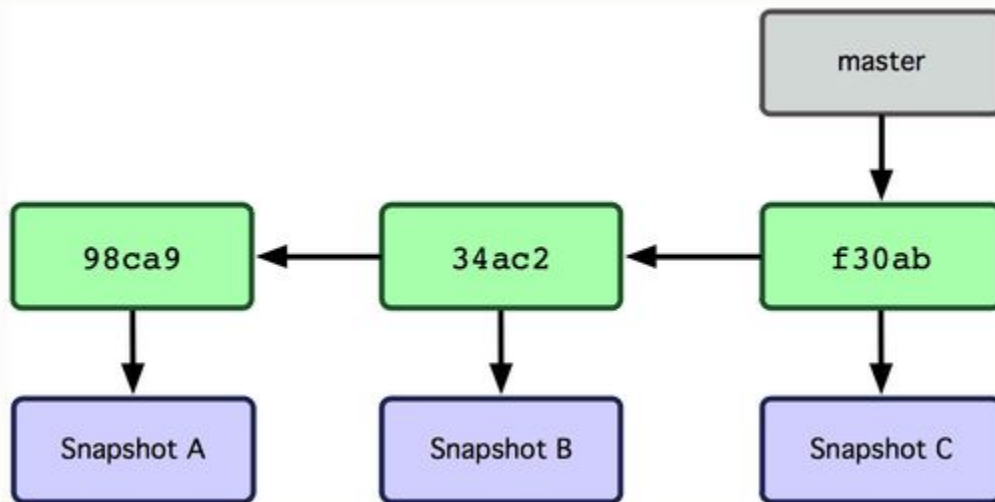
    removed unnecessary test code

commit a11bef06a3f659402fe7563abf99ad00de2209e6
Author: Scott Chacon <schacon@gee-mail.com>
Date:   Sat Mar 15 10:31:28 2008 -0700

    first commit
```

# Ramificaciones en Git (hacemos ejercicio)

*Datos en el repo tras una serie de confirmaciones sencillas*  
***nombre rama = master***



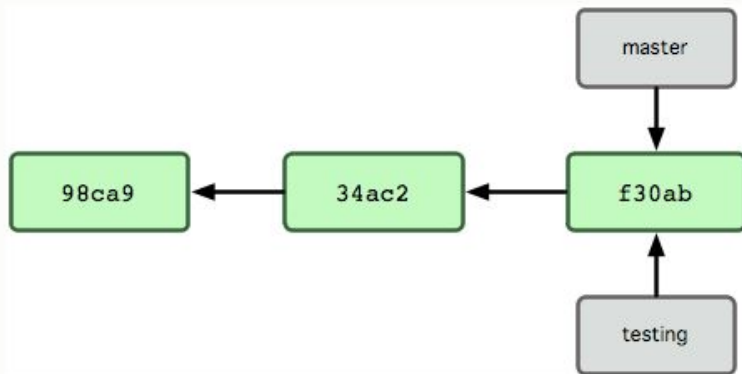
# Ramificaciones en Git

*Creamos nueva rama*

***git branch <nombre nueva rama>***

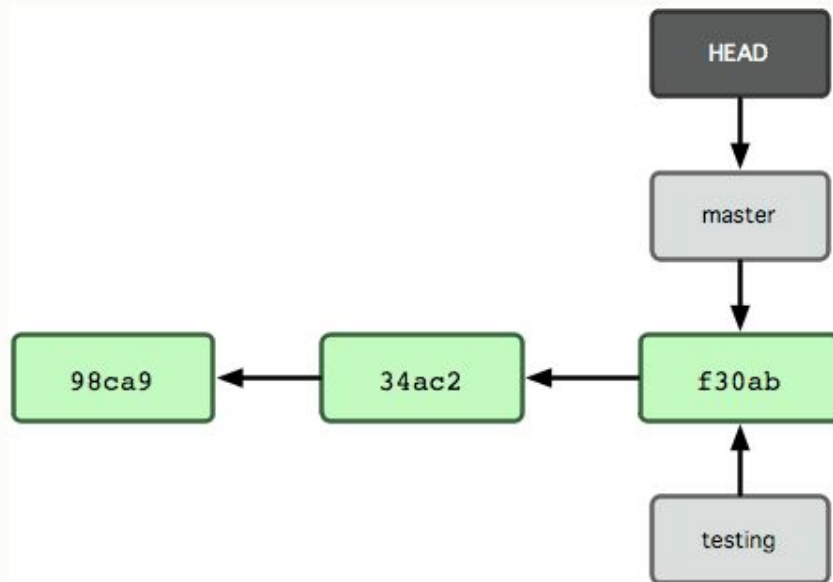
```
$ git branch testing
```

Esto creará un nuevo apuntador apuntando a la misma confirmación donde estás actualmente (ver Figura 3-4).



# Ramificaciones en Git

*Apuntadores de varias ramas en el registro de confirmaciones de cambio. **apuntador HEAD***





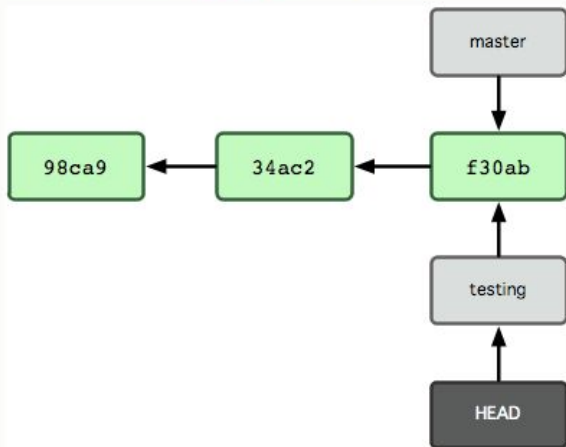
# Ramificaciones en Git

*Movemos el apuntador*

***git checkout <nombre nueva rama>***

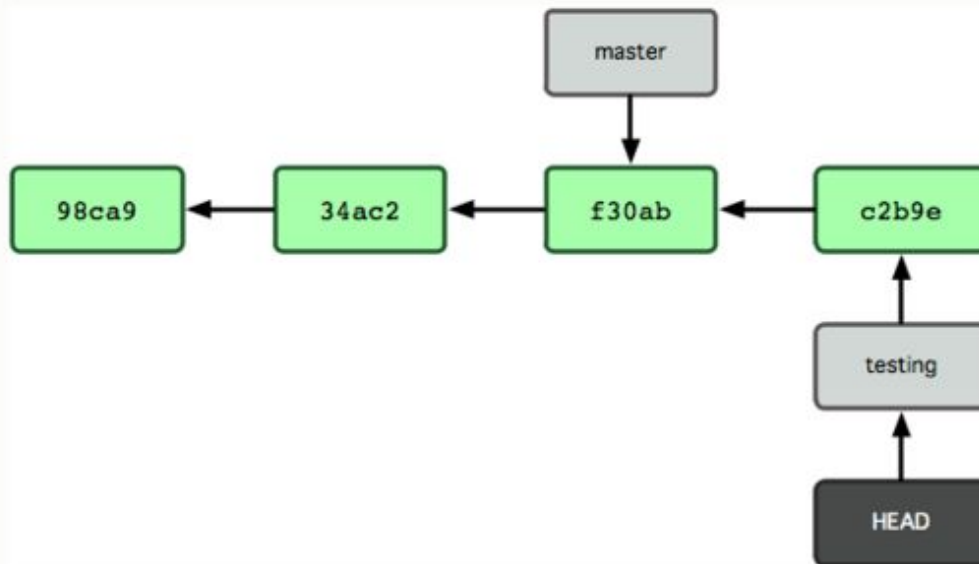
```
$ git checkout testing
```

Esto mueve el apuntador HEAD a la rama **testing** (ver Figura 3-6).



# Ramificaciones en Git

*Trabajo sobre la rama*  
***iteración básica***

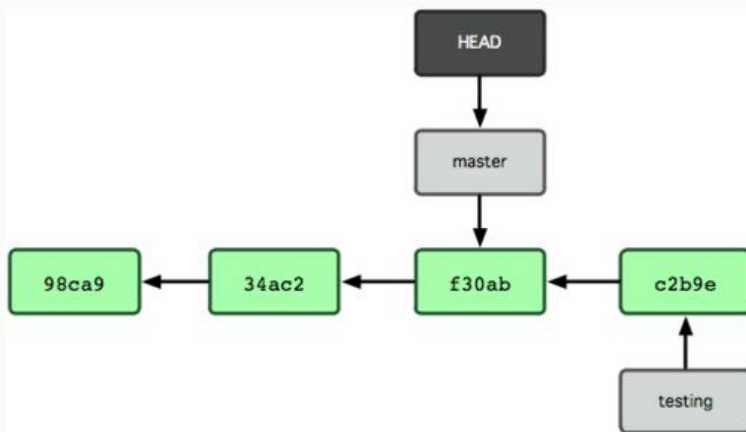


# Ramificaciones en Git

*Volvemos a master*  
***git checkout master***

```
$ git checkout master
```

La Figura 3-8 muestra el resultado.



# Ramificaciones en Git

---

*Fusionamos*

***git merge <nombre nueva rama>***

# Repositorios remotos

*Clonar repositorio remoto*

**git clone**

<https://github.com/germanescobar/courses-students-a-p-p.git>

```
$ git clone git://github.com/schacon/ticgit.git
Initialized empty Git repository in /private/tmp/ticgit/.git/
remote: Counting objects: 595, done.
remote: Compressing objects: 100% (269/269), done.
remote: Total 595 (delta 255), reused 589 (delta 253)
Receiving objects: 100% (595/595), 73.31 KiB | 1 KiB/s, done.
Resolving deltas: 100% (255/255), done.
$ cd ticgit
$ git remote
origin
```

# Repositorios remotos

---

*Clonar repositorio remoto*

**1- Abre proyecto en atom**

**2- Iteración básica**

**3- `git push origin master`** *(solo funciona con todo configurado)*

# Repositorios remotos

---

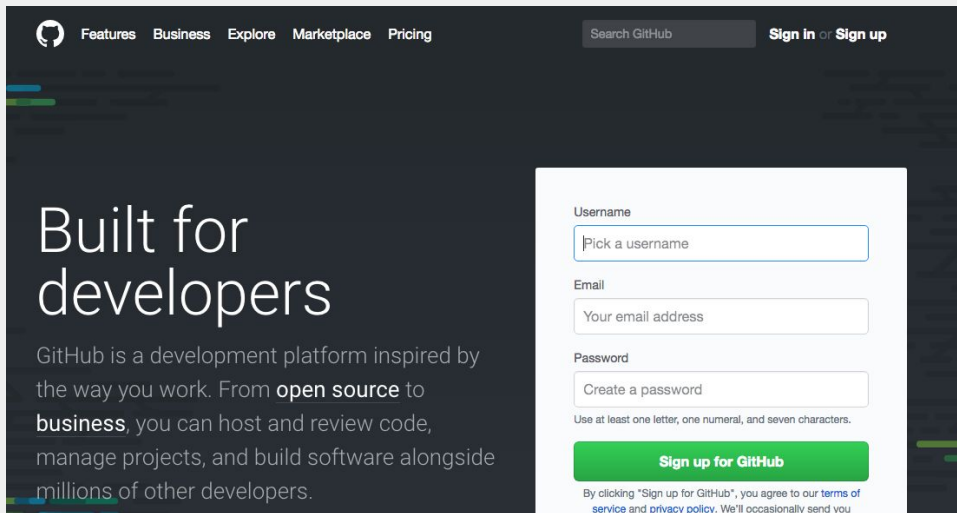
- *Trabajar en proyectos Open Source*
- *Trabajar con equipos remotos*

# Github

Github es un servicio que nos brinda la posibilidad de crear una réplica de nuestro repositorio local en la nube.

Github es gratis para proyectos de código abierto, pero si quieres tener repositorios privados debes pagar una mensualidad (los planes empiezan en 7 dólares al mes).

Si aún no tienes una cuenta en Github es hora de crear una antes de continuar.

A screenshot of the GitHub website's sign-up page. The background is dark with the GitHub logo and navigation links (Features, Business, Explore, Marketplace, Pricing) at the top. A search bar and 'Sign in or Sign up' links are also visible. The main heading 'Built for developers' is prominently displayed. Below it, a paragraph describes GitHub as a development platform. On the right, a white sign-up form is overlaid, containing fields for Username, Email, and Password, each with a placeholder text. A green 'Sign up for GitHub' button is at the bottom of the form, followed by a small disclaimer about terms of service and privacy policy.

Features Business Explore Marketplace Pricing

Search GitHub Sign in or Sign up

## Built for developers

GitHub is a development platform inspired by the way you work. From **open source** to **business**, you can host and review code, manage projects, and build software alongside millions of other developers.

Username  
Pick a username

Email  
Your email address

Password  
Create a password

Use at least one letter, one numeral, and seven characters.

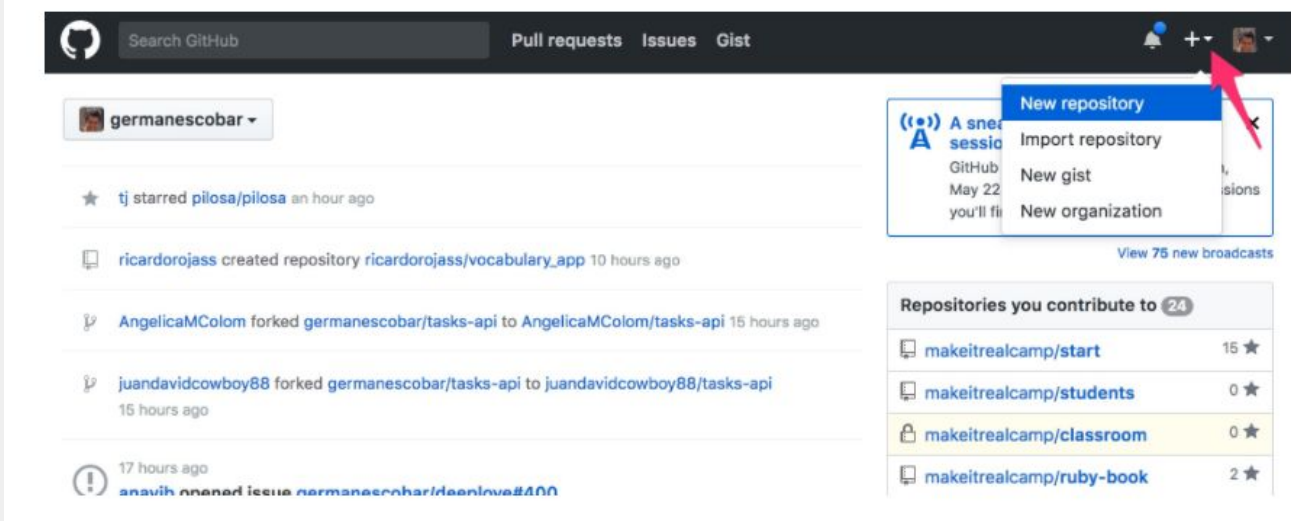
**Sign up for GitHub**

By clicking "Sign up for GitHub", you agree to our [terms of service](#) and [privacy policy](#). We'll occasionally send you




## *Nuestro primer repositorio (repo):*




Una vez que hayas creado tu cuenta, para crear un repositorio en [Github](#) haz click sobre la opción "New Repository" como se muestra en la siguiente imagen:



En la siguiente pantalla debes darle un nombre al repositorio, asegurarte que esté público y oprimir el botón "Create repository\*":

 Search GitHub

Pull requests Issues Gist


  

### Create a new repository


A repository contains all the files for your project, including the revision history.

Owner

Repository name

 germanescobar


/


mi-repo 

Great repository names are short and memorable. Need inspiration? How about [special-octo-waddle](#).

Description (optional)


Este es un repositorio de prueba

☒  **Public**  
Anyone can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

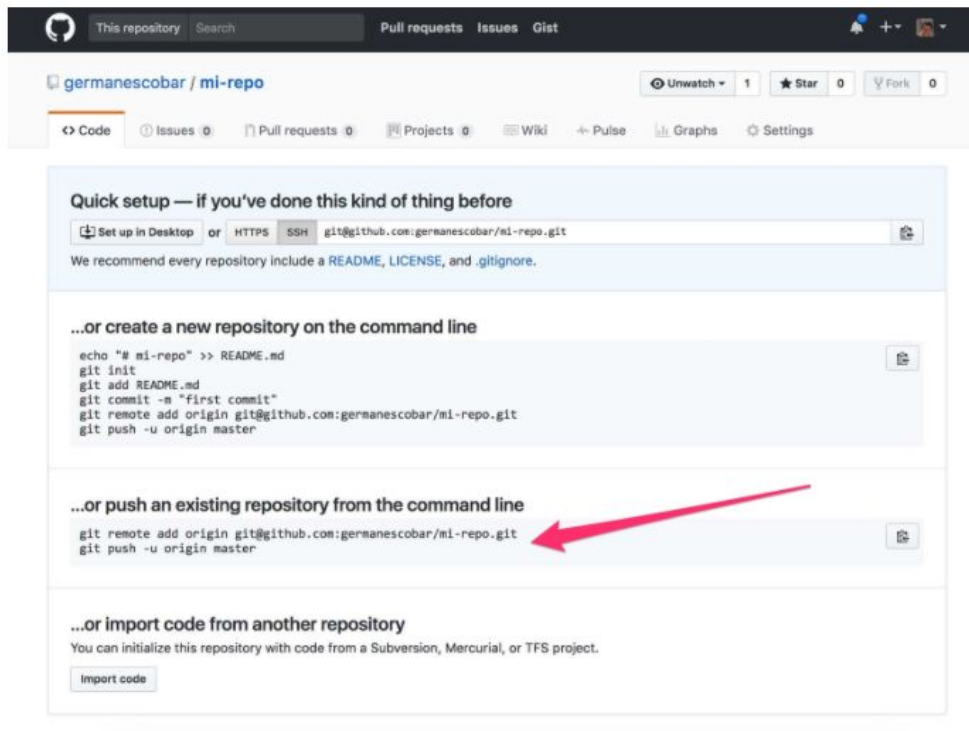
Add a license: **None** 

Create repository

- *Creamos una carpeta en local, llamada repo-en-local*
- *Entramos a esa carpeta*

# Github

Por último te va a aparecer una pantalla como la que ves a continuación. Copia y pega en la línea de comandos las líneas que se indican con la flecha roja:



The screenshot shows the GitHub interface for a repository named 'mi-repo' by 'germanescobar'. The repository has 1 Unwatch, 0 Stars, and 0 Forks. Below the repository name, there are tabs for Code, Issues, Pull requests, Projects, Wiki, Pulse, Graphs, and Settings. The 'Code' tab is selected, showing a 'Quick setup' section with options to 'Set up in Desktop', 'HTTPS', or 'SSH'. The 'SSH' option is selected, showing the URL 'git@github.com:germanescobar/mi-repo.git'. Below this, there is a section titled '...or create a new repository on the command line' with a list of commands: 'echo "# mi-repo" >> README.md', 'git init', 'git add README.md', 'git commit -m "first commit"', 'git remote add origin git@github.com:germanescobar/mi-repo.git', and 'git push -u origin master'. A red arrow points to the 'git push -u origin master' command. Below this, there is a section titled '...or push an existing repository from the command line' with the same list of commands. A red arrow points to the 'git push -u origin master' command. At the bottom, there is a section titled '...or import code from another repository' with a note that you can initialize the repository with code from a Subversion, Mercurial, or TFS project, and an 'Import code' button.

germanescobar / mi-repo

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Pulse Graphs Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH git@github.com:germanescobar/mi-repo.git

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# mi-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:germanescobar/mi-repo.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:germanescobar/mi-repo.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

## *Ejemplo de Flujo de Trabajo == Iteración Básica*

- 1- Salir de la carpeta anterior, crear una nueva carpeta llamada: mi-tercer-repo*
- 2- Crear un archivo llamado: archivo2.txt*
- 3- Abrir el archivo y escribir: "Ya estoy aprendiendo a realizar una iteración básica con git"*
- 4- Ejecutar todos los comandos hasta que al darle "git status" salga "nothing to commit, working directory clean"*

# Ejercicio Completo Git Clone

---

- 1- clonar flow overstack: <https://github.com/danielmoralesp/flow-overstack>
- 2- modificar `app/views/questions/index.html.erb`
- 3- hacer iteracion completa

# Ejercicio Completo

---

- 1- Salir de la carpeta anterior, crear una nueva carpeta llamada: *mi-cuarto-repo*
- 2- Crear un archivo llamado: *archivo4.txt*
- 3- Abrir el archivo y escribir: *“Voy a tener la habilidad para crear un repo en git y github”*
- 4- Ejecutar todos los comandos hasta que al darle *“git status”* salga *“nothing to commit, working directory clean”*
- 5- Crear un repo en github con el nombre *“primer repo completo en github”*
- 6- Conectar repositorio remoto con repositorio local
- 7- Hacer *“git push -u origin master”*