

Esta clase va a ser

- grabada



# Git

¿Recuerdas que la clase pasada te pedimos que instales Git en tu computadora?

¡Comenzaremos a poner manos al código! 🧐

Psst... en caso de que no lo hayas hecho, puedes hacerlo ahora mientras esperamos a que lleguen todos los estudiantes

[Ver tutorial](#)

Clase 16. PYTHON

# Git y Github

# Objetivos de la clase

- **Analizar** el control de versiones líder (GIT).
- **Crear** un proyecto y versiones con GIT.
- **Utilizar** el repositorio GITHUB.

# Temario

15

## Scripts, módulos y paquetes

- ✓ Script
- ✓ Módulos
- ✓ Paquetes

16

## Git – GitHub

- ✓ [Git](#)
- ✓ [GitHub](#)

17

## Django – Portfolio Parte I

- ✓ Django
- ✓ Plantillas Django

**GIT**

# ¿Qué es GIT?

# ¿Qué es GIT?

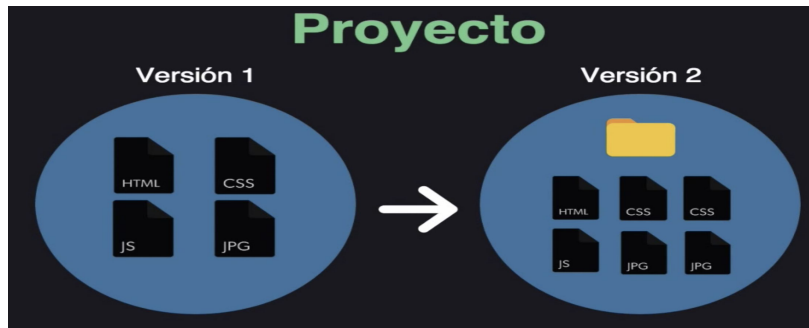
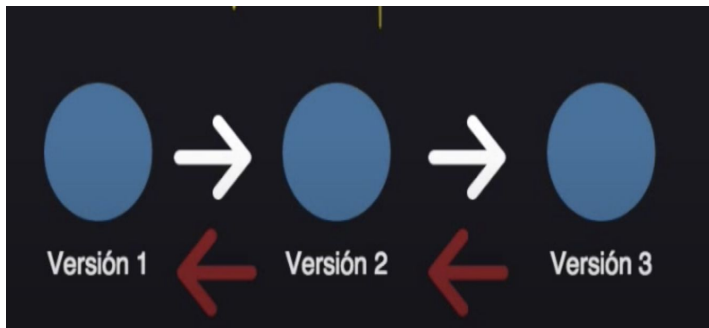
Git es un sistema de control de versiones gratuito y de código abierto, diseñado para manejar desde pequeños a grandes proyectos de manera rápida y eficaz. Se entiende como control de versiones a todas las herramientas que nos permiten hacer modificaciones en nuestro proyecto. Un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo.





# ¿Qué es GIT?

Con GIT, podemos ir a versiones anteriores, muy útil para errores y para la organización.



# GIT – Los 3 estados

**1er estado** (comienzo del trabajo)

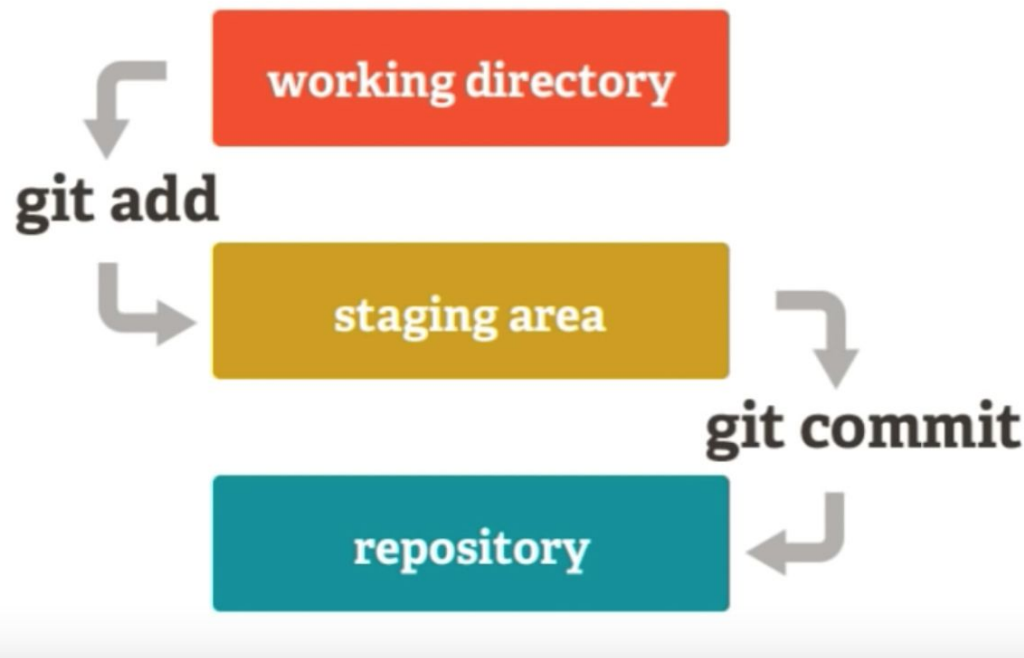
"preparamos las cajas"

**2do estado** (archivos listos)

"agregamos las cajas listas"

**3er estado** (registro de todos los archivos)

"Lote listo"



# Relación entre GIT y GITHUB



git



github  
SOCIAL CODING

Git es uno de los sistemas de control de versiones más populares entre los desarrolladores. Y parte de su popularidad se la debe a GitHub, un excelente **servicio de alojamiento de repositorios de software con este sistema.**

# Instalación y configuración de GIT



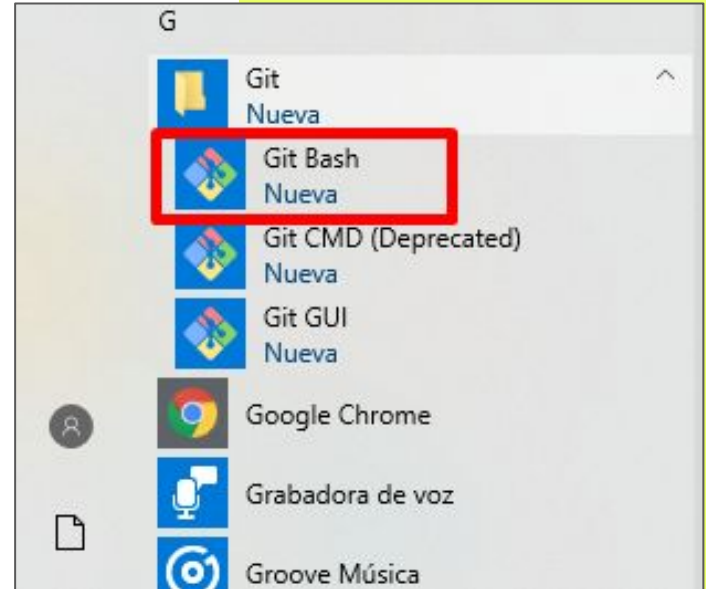
Lo primero es lo primero: tienes que instalarlo.

Puedes obtenerlo de varias maneras; las dos principales son instalarlo desde código fuente, o instalar un paquete existente para tu plataforma.

# Empecemos con GIT

# Empecemos con Git

Buscar en su menú el **Git Bash** para abrir la terminal e iniciar con los comandos.



# Verificando versión Git

Escribir `git --version` y presionar "Enter".

```
john@MyShopSolutions: ~$ git --version
git version 2.17.1
john@MyShopSolutions: ~$
```



# Configurando Git por primera vez: tu identidad

Lo primero que debes hacer cuando instalas Git es establecer tu **nombre de usuario** y **dirección de correo electrónico**. Esto es importante porque las confirmaciones de cambios (commits) en Git usan esta información, y es introducida de manera inmutable en los commits que envías.

# Configurando Git por primera vez



Elegir un nombre de usuario que recuerdes fácil, y el email que en la próxima clase se usará en Github.



Establecer el nombre con el comando: `git config --global user.name "Nombre Apellido"`



Establecer el correo a usar con el comando. `git config --global user.email johndoe@example.com`

# Configurando Git por primera vez

Comprobamos los pasos en nuestra consola.

```
/* Paso 2 */
```

```
john@MyShopSolutions: ~$ git config --global user.name "John Doe"
```

```
/* Paso 3 */
```

```
john@MyShopSolutions:~$ git config --global user.email johndoe@example.com
```

# Configurando Git por primera vez

Vamos a comprobar si guardamos bien el usuario usando el comando:  
`git config --list.`

```
john@MyShopSolutions: ~$ git config --list
/* Se puede ver el usuario, el email y otros parámetros que
dependerán de cada sistema operativo */
user.name=John Doe
user.email=johndoe@example.com
color.status=auto
color.branch=auto
color.interactive=auto
color.diff=auto
```

# Comprobando tu configuración

Puedes también comprobar qué valor tiene la clave nombre en Git ejecutando: `git config user.name`.

Puedes consultar de la misma manera `user.email`.

```
john@MyShopSolutions: ~$ git config user.name  
John Doe
```

# Obteniendo ayuda

Si alguna vez necesitas ayuda usando Git, hay tres formas de ver la página del manual (manpage) para cualquier comando de Git:

```
/* Los tres comandos que disparan la ayuda de Git */  
john@MyShopSolutions: ~$ git help config  
john@MyShopSolutions: ~$ git config --help  
john@MyShopSolutions: ~$ man git-config
```

Hasta el momento hemos aprendido los **primeros pasos en GIT**. Tenemos funcionando en el sistema una versión de Git configurada con tu identidad. Es el momento de aprender algunos **fundamentos de Git**.



## Para pensar

Con lo visto en clase hasta ahora  
¿cuál podrían decir que es la diferencia  
principal entre GIT y GitHub?

Contesta mediante el chat de Zoom





# Break

¡10 minutos y volvemos!

# Creando repositorios



PARA RECORDAR

# ¿Qué es un repositorio?

Un repositorio es un **espacio centralizado** donde se almacena, organiza, mantiene y difunde información.

Será **“la carpeta”** donde guardaremos nuestro proyecto para más adelante compartirlo con el equipo a través de un repositorio en la nube (en internet, por ejemplo en Github).

# Git Int

Este comando se usa para crear un nuevo repositorio en Git. Nos crea un repositorio de manera local y lo hará en la carpeta donde estamos posicionados. También se le puede pasar `[nombre_de_la_carpeta]` y creará una con ese nombre.

A continuación vemos el ejemplo:

# Git Int

/\* Paso 1: Me ubico en la carpeta donde quiero crear mi proyecto \*/

```
john@MyShopSolutions :~$ cd Documents/Proyectos_Coder/
```

/\* Paso 2: Ya dentro de la carpeta inicio el proyecto con el nombre que le asigne a mi repositorio\*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/$ git init mi_repositorio
```

/\* Arrojará el siguiente mensaje \*/

```
Initialized empty Git repository in /home/usuario/Documents/Proyectos_Coder/mi_repositorio/.git/
```

/\* Paso 3: Comprobamos que el repositorio se creó \*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/$ dir  
mi_repositorio
```

/\* Paso 4: Me ubico en mi repositorio \*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/$ cd mi_repositorio
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

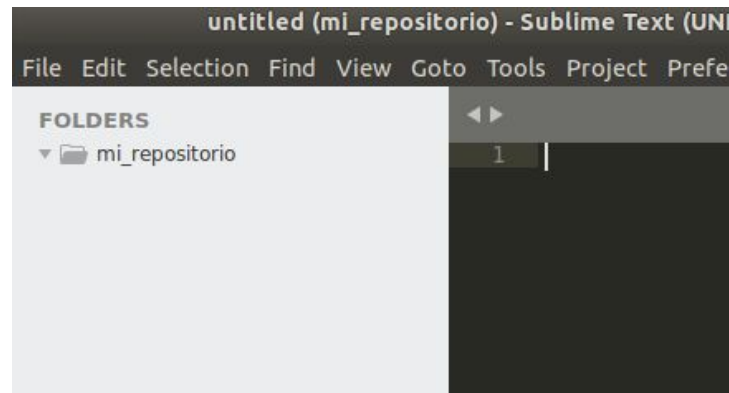
# Git Status

1

Ya hemos visto cómo inicializar un repositorio localmente utilizando *git init*. Ahora nos toca crear los archivos que vamos a usar en este repositorio.

## Vamos a Sublime Text:

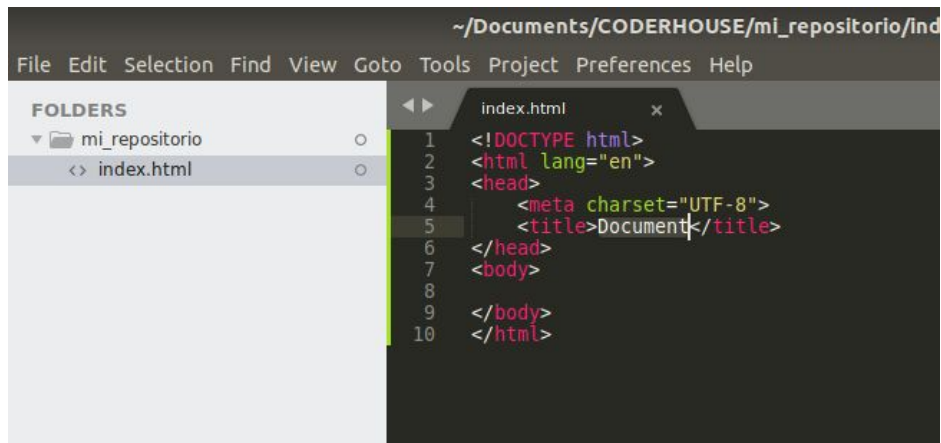
Buscamos el repositorio creado



# Git Status

2

Creamos un archivo index.html que se guardará en el repositorio.



The screenshot shows a code editor window with the title bar indicating the file path: `~/Documents/CODERHOUSE/mi_repositorio/index.html`. The menu bar includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. On the left, a 'FOLDERS' sidebar shows a tree view with 'mi\_repositorio' expanded, containing 'index.html'. The main editor area displays the content of 'index.html' with line numbers 1 through 10. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6 </head>
7 <body>
8
9 </body>
10</html>
```

# Git Status

3

Vamos a la terminal y con git status chequeamos el estado de nuestro repositorio

```
john@MyShopSolutions
~/Documents/Proyectos_Coder/mi_reposito
rio$ git status
On branch master

No commits yet

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

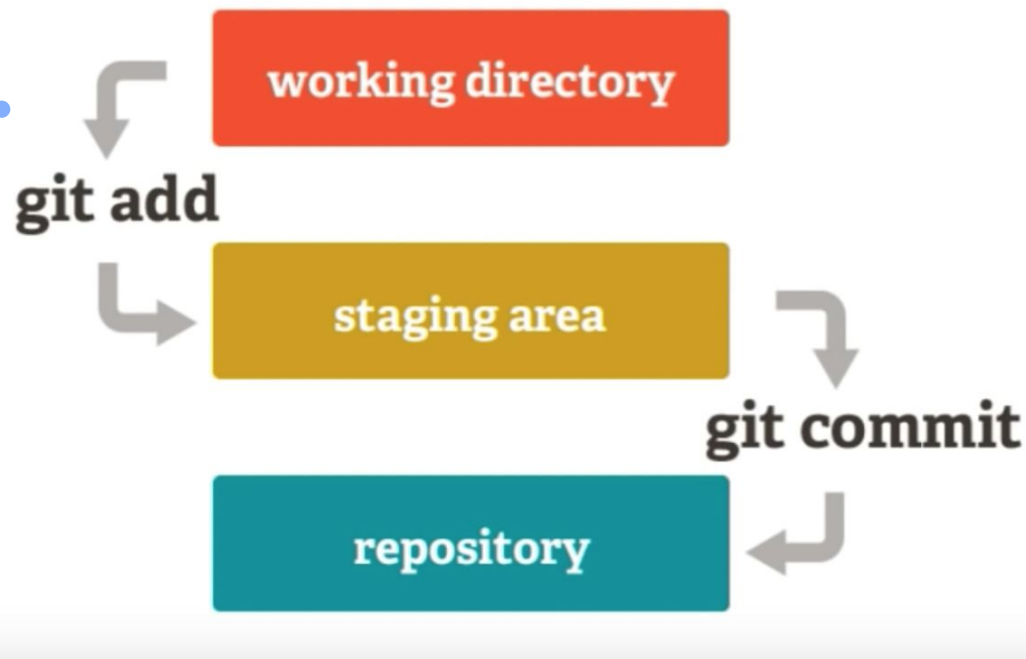
    index.html

nothing added to commit but untracked files
present (use "git add" to track)
```



# Git: ¿recuerdan los 3 estados?

Estamos aquí con el `index.html` creado



# Git Add

Ahora se necesita agregar el o los archivos al **Staging Area**. En nuestro caso, para el index.html vamos a usar el comando `git add + el nombre del archivo`, lo cual permite adherir el archivo para subirlo luego al repositorio. También se puede usar `git add .` que adhiere todos los archivos nuevos.

Para verificar si funciona, nuevamente utilizamos **git status**.

# Git Add

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git add index.html
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git status
```

On branch master

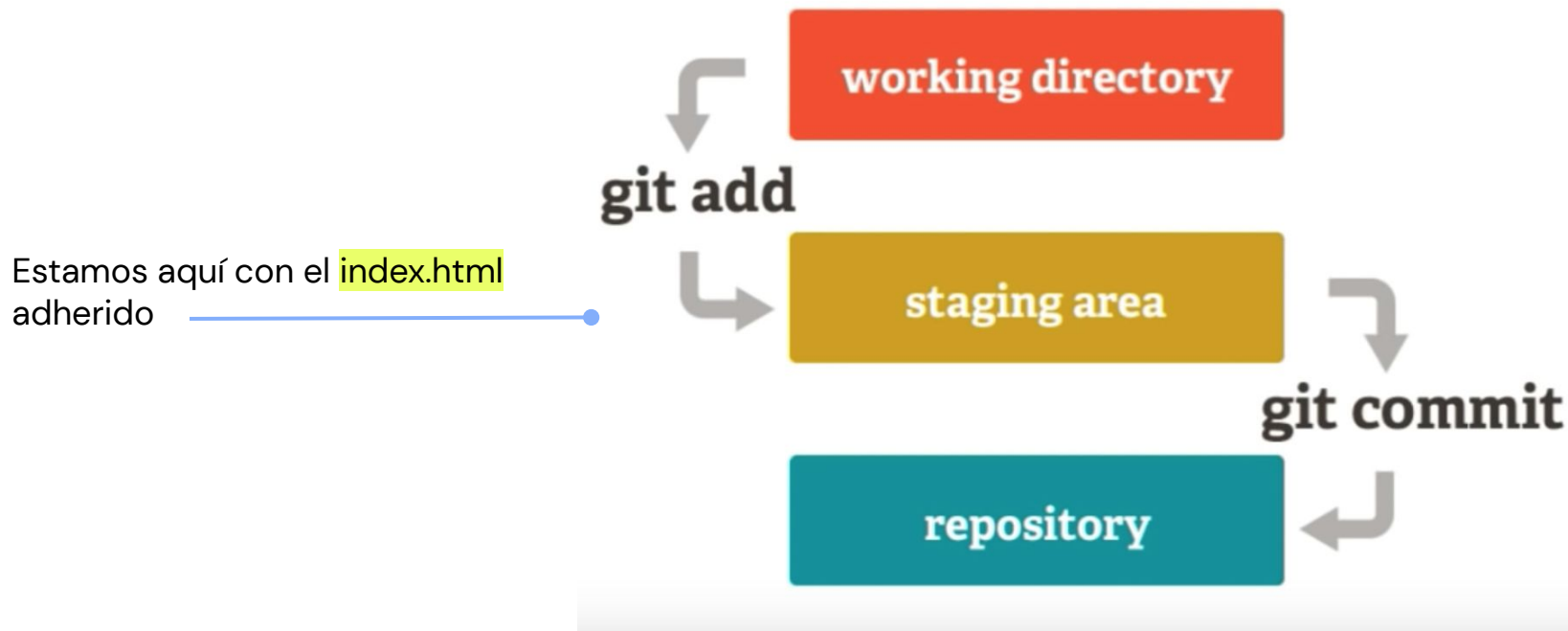
No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

# Git: ¿recuerdan los 3 estados?



# Git Commit

Una vez que nuestros archivos están en el **Staging Area** debemos pasarlos a nuestro repositorio local y para eso debemos usar el **git commit** que es el comando que nos va a permitir comprometer nuestros archivos.

Es decir, que lo subirá al repositorio que se ha creado.

# Git Commit

El comando es el siguiente:

```
git commit -m "Comentario de qué se trata el commit que se está realizando"
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git commit -m "Primer archivo del repositorio"
```

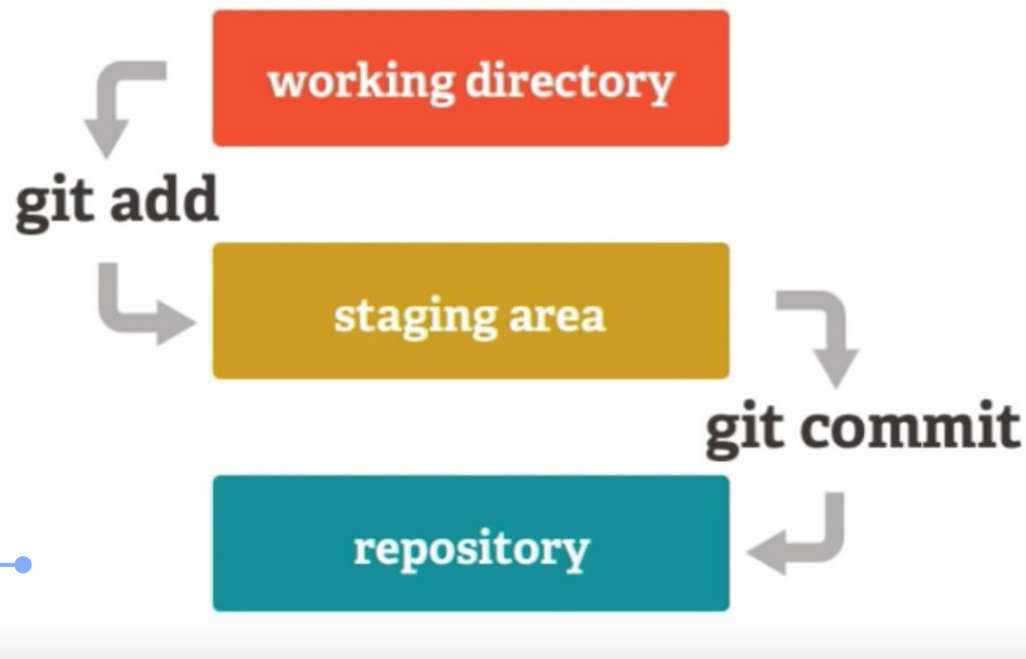
```
/* Esta sería el resultado del comando */
```

```
[master (root-commit) 1734915] nuevo archivo
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 index.html
```

# GIT: ¿recuerdan los 3 estados?



Estamos aquí con el `index.html` comprometido para el repositorio

# Git Log

Los primeros pasos a seguir:

```
/* Con git log podemos ver los logs (historial) de lo que ha pasado en el repositorio */  
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$ git log  
commit 1734915470ce9983f703b77807a68e42166b47dd (HEAD -> master)  
Author: John Doe <johndoe@example.com>  
Date: Sat May 22 18:53:24 2020 -0300
```

Primer archivo del repositorio



# Git Log

La documentación de git log es superextensa, por eso puedes indagar en el siguiente link

[Git-Scm](#)

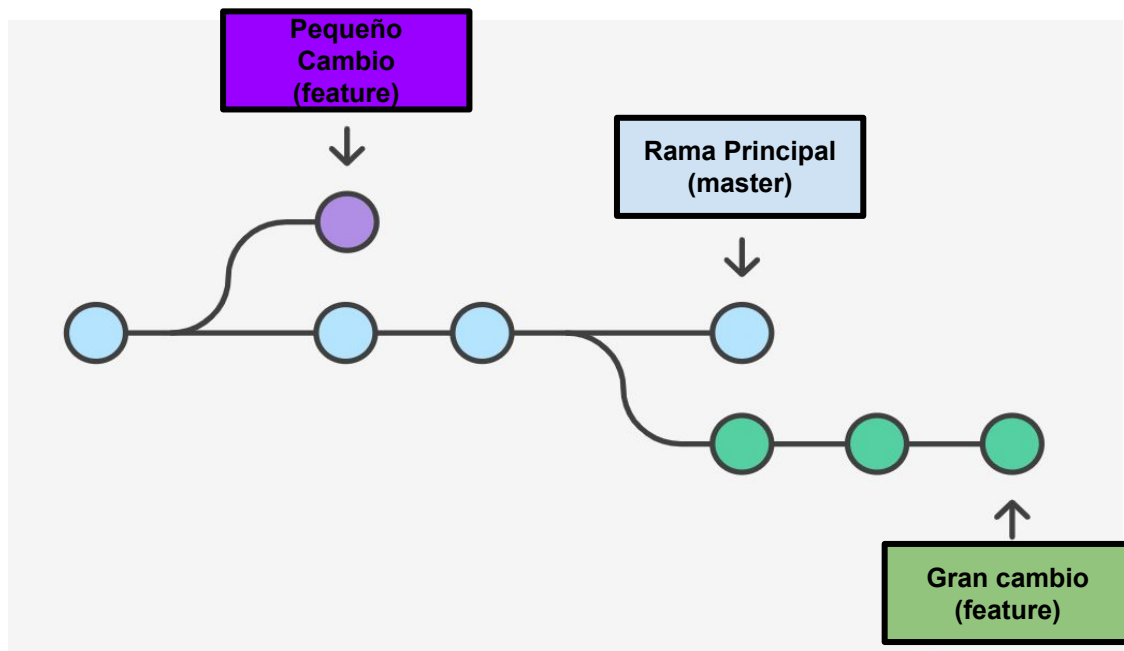
# Ramas

# Ramas

Para añadir una nueva función o solucionar un error (sin importar su tamaño), generas una nueva rama para alojar estos cambios. Esto te da la oportunidad de organizarte mejor con los cambios o correcciones experimentales.

👉 Podemos crear una rama escribiendo  
`"git branch mi-rama"`

# Ramas



Así funciona

# Git Branch: creando ramas

Veamos cómo crear una rama.

```
/* Paso 1: Verifico en cuál rama estoy */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master  
/* Paso 2. Creo la rama que voy a usar para el cambio */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch mi_rama  
/* Paso 3: Verifico que se creó la rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -l  
*master  
mi_rama  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

# Git Branch: movernos entre ramas

¿Será muy complicado hacerlo?

```
/* Para moverme a la rama que cree uso el comando de git checkout */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout mi_rama  
Switched to branch 'mi_rama'  
/* Verifico nuevamente que me movi de rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -l  
master  
*mi_rama  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

# Git Branch D: borrando ramas

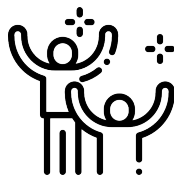
Penúltimo paso 😊

```
/* Paso 1: Me muevo a la rama principal "master" */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout master  
/* Paso 2: Verificar que se está en la rama de master */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master  
mi_rama  
/* Paso 3: Procedo a borrar la rama que ya no voy a usar */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch -D mi_rama  
Deleted branch mi_rama (was 6d6c28c)  
/* Paso 4: Verificar que se borró la rama */  
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch  
*master
```

# Git checkouts: listar commits

Así como nos movemos entre ramas, nos podemos mover entre commits. Recuerden que al hacer cambios, adherirlos y comitearlos, se crea un historial de dichos cambios, los logs.

La posibilidad de volver a un commit en específico es una ventaja de los controladores de versiones, que permiten volver a un estado anterior si se presenta un problema, error o cambio inesperado.





# Git checkouts listar commits

Comenzamos listando.

```
/* Para ver los commits realizados, los listamos con el comando git log --oneline para verlos en una sola línea*/
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git log --oneline
```

```
/* Se listan todos los cambios que se han realizado sobre el index.html */
```

```
fc59b88 (HEAD -> nueva_rama) Ahora agregamos un título
```

```
6bcff19 Agregar un texto al index.html
```

```
41e6121 (master) Primer archivo del repositorio
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

# Git checkout: mover a un commit

/\* Supongamos que me equivoqué en agregar el título, quiero volver al punto anterior del texto, busco el número de commit y muevo hacia ese punto \*/

```
john@MyShopSolutions: ~/Documents/Proyectos_Coder/mi_repositorio$ git checkout 6bcff19
```

Note: checking out 6bcff19.

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -b with the checkout command again. Example:

```
git checkout -b <new-branch-name>
```

HEAD is now at 6bcff19... Agregar un texto al index.html

# Git checkout: movernos a un commit

```
/* Si verifico donde estoy parado co git branch se puede observar que se está en el  
commit y el index.html ha cambiado*/
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch
```

```
* (HEAD detached at 6bcff19)
```

```
master
```

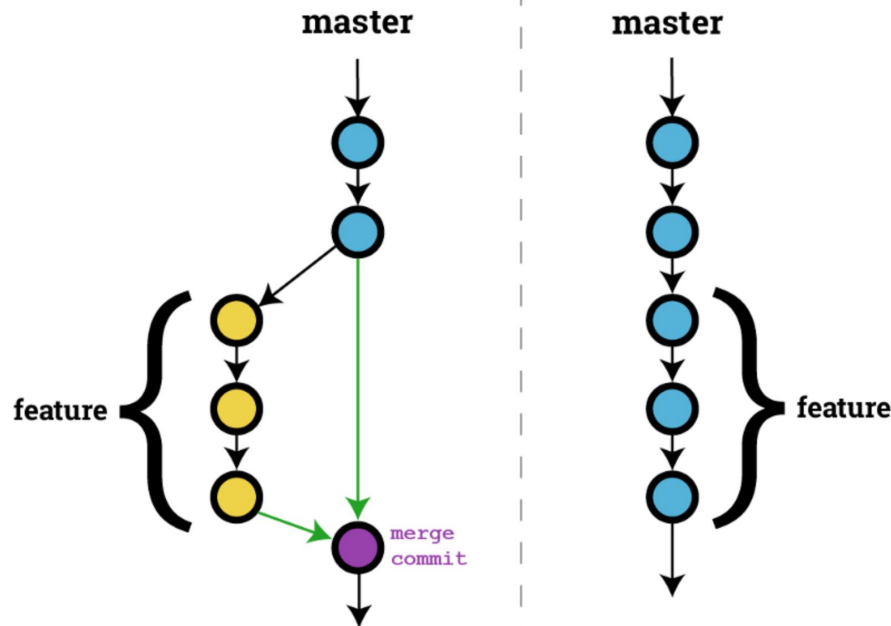
```
nueva_rama
```

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$
```

# Git: fusionar (merge)

Una vez que tenemos una rama (o más), podemos experimentar características nuevas.  
Para luego **FUSIONARLAS** con la rama **MASTER**.

A continuación veamos cómo hacerlo...



# Git Merge

/\* Paso 1: Ubicarse en la rama master, que es a donde quiero fusionar los cambios usando el comando de git checkout master. \*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git checkout master
```

/\* Paso 2: Verificar que estoy en master con git branch. Se puede observar en el archivo de index.html que no tiene ni título ni texto. \*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git branch
*master
```

Nueva\_rama

/\* Paso 3: Realizar la fusión. Hacer el merge con el comando **git merge nueva\_rama**\*/

```
john@MyShopSolutions :~/Documents/Proyectos_Coder/mi_repositorio$ git merge nueva_rama
```

Updating 41e6121..fc59b88

Fast-forward

index.html | 2 ++

1 file changed, 2 insertions(+)



## Ejemplo en vivo

Vamos a crear una rama para listar commits en GIT.

# Listar commits

1

Crear una rama con git  
branch **nueva\_rama**

2

Cambiar de rama con git  
checkout **nueva\_rama**

3

Verificar que cambie de  
rama con git branch -l

# Paso a paso

4

Agregar al index.html un  
texto nuevo

5

Verificar que hubo un  
cambio en el index.html  
con git status

6

Adherir el cambio con  
git add



# Listar commits

7

Comitear el cambio con  
`git commit -m`  
"Agregando texto al htm

8

Agregar un título al  
`index.html` y repetir los  
pasos para poder  
comitear el cambio.



# Repaso

- ✓ **Git Init:** indicarle que en ese directorio, donde ejecutamos este comando, será usado con GIT.
- ✓ **Git Add:** Agregar todos los archivos creados, modificados, eliminados al estado 2 (stage).
- ✓ **Git Commit - m "mensaje":** mensaje obligatorio para mostrar que hemos cambiado, por ejemplo al estado 3.
- ✓ **Git log -- online:** para conocer los códigos de los commits realizados.



# Repaso

- ✓ **Git checkout rama:** para cambiar de rama e ir a un commit específico (debemos conocer su código anteriormente).
- ✓ **Git merge rama:** Debemos estar en un MASTER para funcionar.
- ✓ **Git branch rama:** creación de una rama (si queremos eliminar una rama ponemos `git branch -D nombre-rama`).

GitHub

***CODERHOUSE***

# ¿Qué es GitHub?

# Definición

Por ahora todo lo que venía ocurriendo en Git era de manera local, no necesitábamos nada de internet para guardar nuestros commits y nuestro repositorio. Ahora queremos compartir nuestro trabajo con otros (compañeros de proyecto, clientes, etc.). ¡Para eso utilizamos GitHub!

[Github](#) es una especie de “red social” de programadores. Con este sitio podemos subir nuestros proyectos y lograr que otras personas colaboren.

# Crear nuestro repositorio en GitHub



# Creando un repositorio

Luego de hacer clic en el enlace de verificación, aparecerá una pantalla así, que indica que tu e-mail ha sido verificado, y permite que hagas tu primer repositorio.

Your email was verified. Would you like to create your first repository?

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner

Repository name \*

 tutorial-hash ▾

/ Nombre repositorio

Great repository names are short and memorable. Need inspiration? How about **super-duper-octo-computing-machine?**

Description (optional)



**Public**

Anyone can see this repository. You choose who can commit.

**Público:** Cualquiera puede ver el repositorio. Podes elegir quien puede commitear a tu repositorio.



**Private**

You choose who can see and commit to this repository.



**Privado:** Podes elegir quien puede ver y commitear a tu repositorio





# Creando un repositorio

Por ejemplo, podría ser llamado **“mi\_repositorio”**, para que pruebes con los archivos que trabajaste en el desafío de GIT.

- 
- ☐  **Public**  
Anyone can see this repository. You choose who can commit.
- ☒  **Private**  
You choose who can see and commit to this repository.
- 

Skip this step if you're importing an existing repository.

- ☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **None** ▼





---

Create repository



# Creando un repositorio

Elegimos “**público**” o “**privado**”. Si bien con privado limitamos el acceso a cualquier persona, no nos permitirá mostrar nuestro código como página web, por lo que elegimos “público”.

- 
- ☐  **Public**  
Anyone can see this repository. You choose who can commit.
- ☒  **Private**  
You choose who can see and commit to this repository.
- 

Skip this step if you're importing an existing repository.

- ☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **None** ▼





---

Create repository



# Creando un repositorio

Luego hacemos clic en “**Create repository**”

- 
- ☐  **Public**  
Anyone can see this repository. You choose who can commit.
- ☒  **Private**  
You choose who can see and commit to this repository.
- 

Skip this step if you're importing an existing repository.

- ☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **None** ▼



---

**Create repository**



# Creando un repositorio

Repositorio creado 😎

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

or   `https://github.com/Isaine/mi_repositorio.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 "# mi_repositorio" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Isaine/mi_repositorio.git
git push -u origin master
```

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

```
git remote add origin https://github.com/Isaine/mi_repositorio.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.



# GitHub

Subiremos nuestro primer repositorio.

Duración: **10 minutos**



ACTIVIDAD EN CLASE

# GitHub

## Descripción de la actividad.

Crea el primer repositorio en GitHub con el material del proyecto de la clase anterior: Primer Módulo y Feliz Cumpleaños.



ACTIVIDAD EN CLASE

# GitHub: vamos a subir nuestro repositorio

Vamos a nuestra terminal y nos ubicamos en el proyecto creado en la clase pasada. Copiaremos las siguientes líneas para realizar el “push” de los archivos a nuestro servidor en GitHub.

```
/* Paso 1: Me ubico en mi repositorio */  
john@MyShopSolutions :~$ cd Documents/Proyectos_Coder/mi_repositorio  
/* Paso 2: Indico cuál será mi nuevo repositorio remoto */  
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$ git remote add origin  
https://github.com/miuser/mi_repositorio.git
```



## ACTIVIDAD EN CLASE

GitHub está trabajando...



```
/* Paso 3: Pusheamos todos nuestros archivos al repositorio de github */  
john@MyShopSolutions:~/Documents/Proyectos_Coder/mi_repositorio$ git  
push -u origin master  
Username for 'https://github.com': miuser /* Pedirá el usuario de github */  
Password for 'https://isaine@github.com': /* Pedirá el la clave de github */  
Counting objects: 9, done.  
Delta compression using up to 4 threads.  
Compressing objects: 100% (6/6), done.  
Writing objects: 100% (9/9), 869 bytes | 217.00 KiB/s, done.  
Total 9 (delta 2), reused 0 (delta 0)  
remote: Resolving deltas: 100% (2/2), done.  
To https://github.com/miuser/mi_repositorio.git  
* [new branch] master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.
```





## ACTIVIDAD EN CLASE

Los archivos ya están  
en GitHub 🎉🎉

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Projects 0](#) [Wiki](#) [Security 0](#) [Insights](#) [Settings](#)

*No description, website, or topics provided.* [Edit](#)

[Manage topics](#)

3 commits

1 branch

0 packages

0 releases

1 environment

0 contributors

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾

isaneduque

Ahora agregamos un titulo

✓ Latest commit fc59b88 7 hours ago

index.html

Ahora agregamos un titulo

7 hours ago

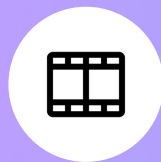
Help people interested in this repository understand your project by adding a README.

Add a README

# ¡Atención!

Recuerda instalar **Django** para el próximo encuentro.  
Encontrarás un tutorial en la carpeta de la clase.





**¿Quieres saber más?**  
**Te dejamos material  
ampliado de la clase**



MATERIAL AMPLIADO

# Recursos multimedia

- ✓ [Git Cheat Sheet](#) | GitHub Education
- ✓ [Tutoriales de uso GitHub](#) | GitHub
- ✓ [Cómo usar la integración Git en Visual Studio Code](#) | Digital Ocean

¿Preguntas?

# Resumen de la clase hoy

- ✓ Git: instalación, configuración, repositorio y ramas.
- ✓ GitHub: definición, creación de repositorio, suba de proyecto.

**Muchas gracias.**

**Opina y valora**  
esta clase



**#DemocratizandoLaEducación**