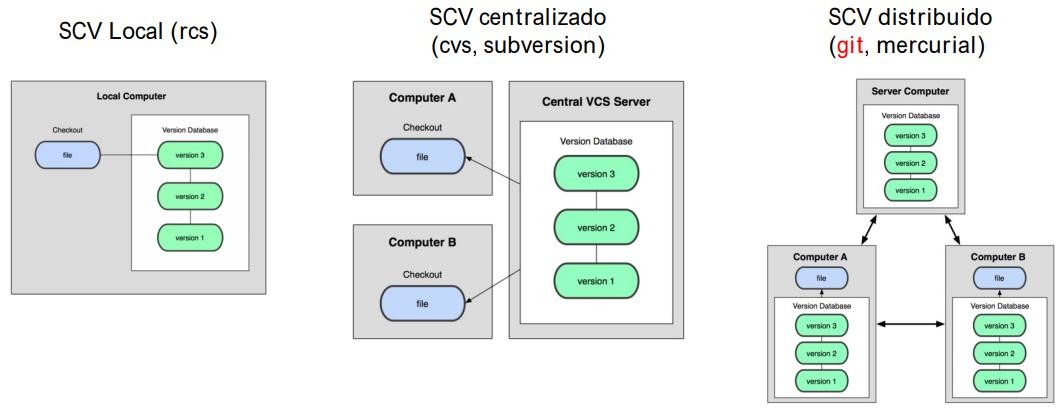
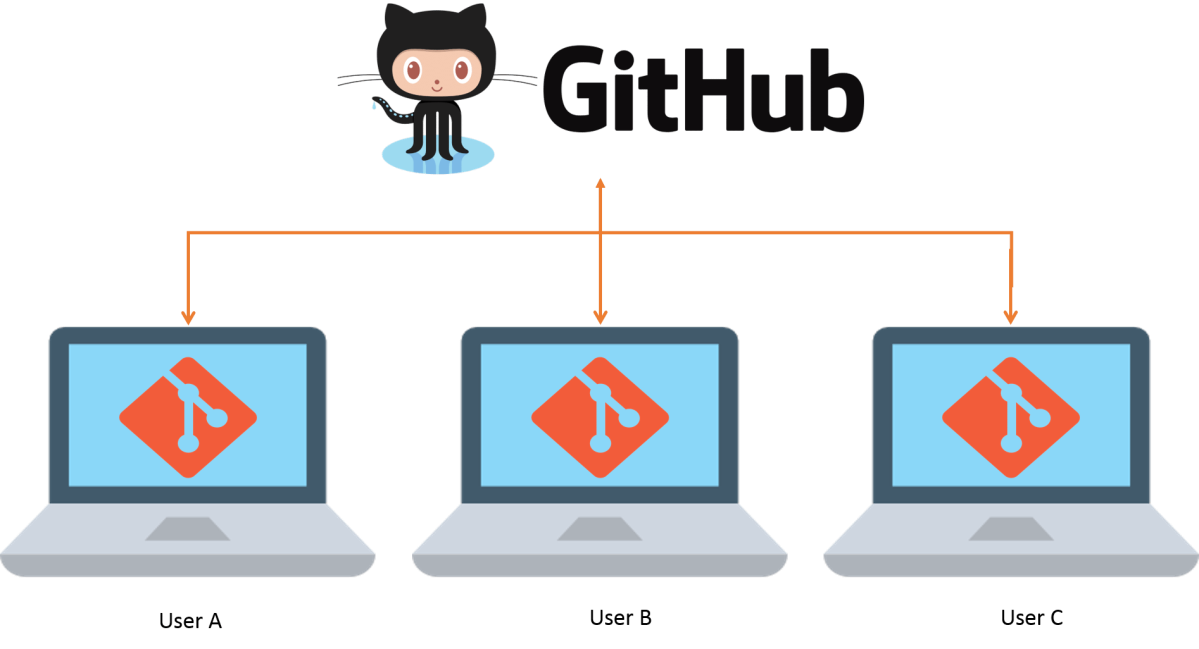
GIT -- Github

ING. JAIDER OSPINA NAVAS.



Control de Versiones. “Gestionar el cambio” “Creando Inteligencia”



[Fuente: https://aprendiendoarduino.wordpress.com/tag/git/](https://aprendiendoarduino.wordpress.com/tag/git/)

**Git** es un **sistema de control de versiones**



**<https://github.com/>**

**distribuido**.

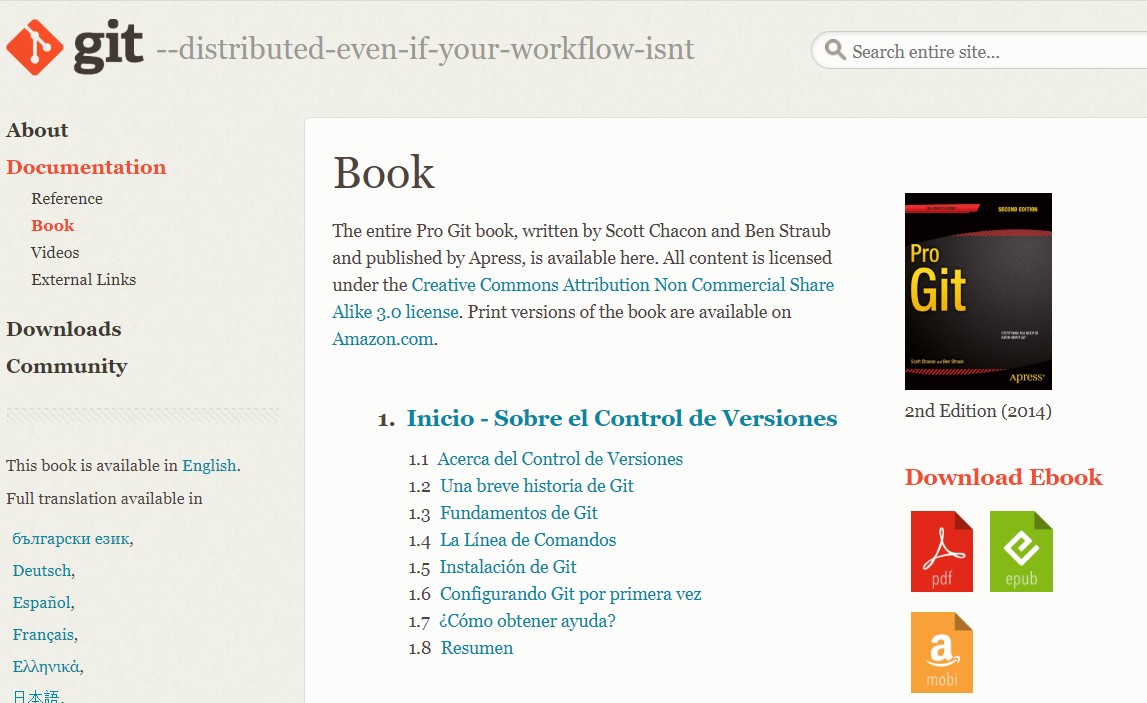
[GitHub](http://github.com/)

“hosting” de los proyectos. Una

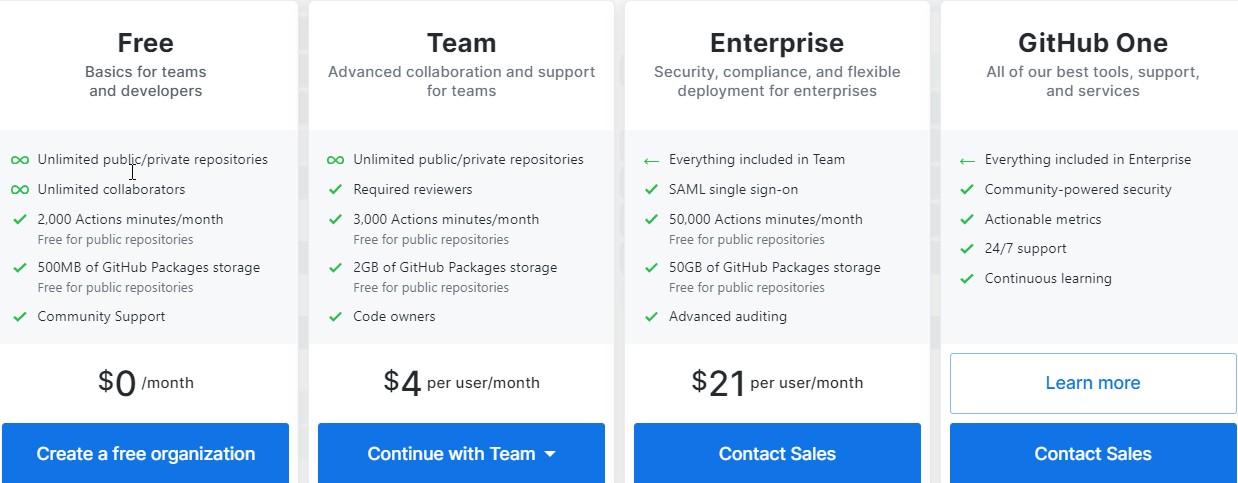
es la plataforma de

comunidad llena de personas que desarrollan y comparten, usando GIT.

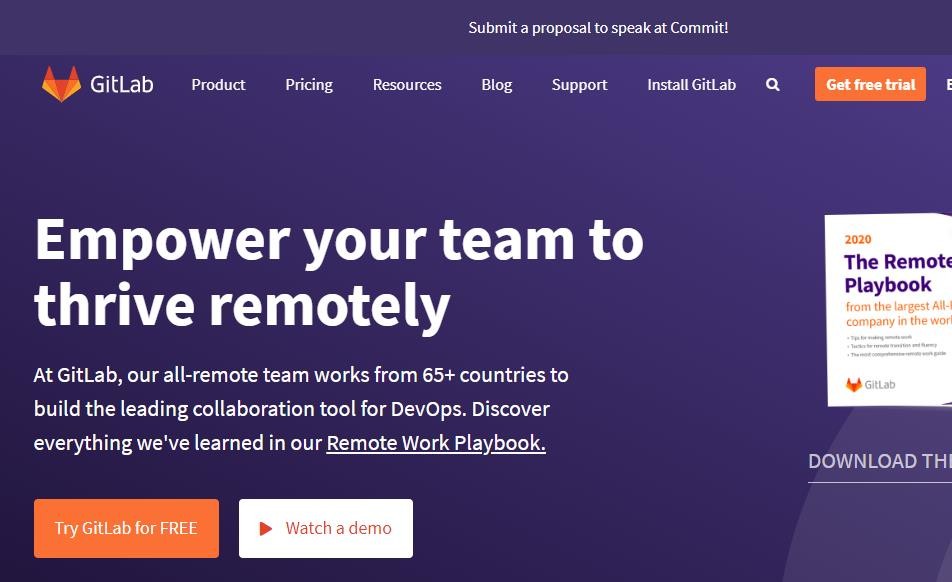
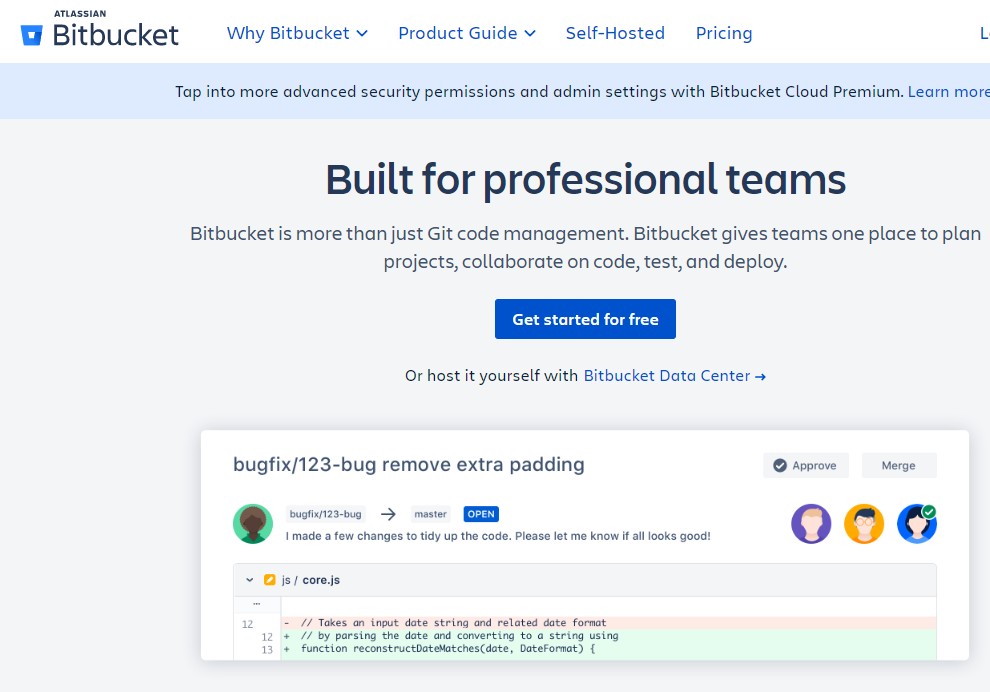
**GitHub** es una página que **ofrece un grupo de servicios que facilitan el uso de Git**, como por ejemplo hosting de proyectos, facilidades de colaboración, reviews de código, perfiles personales, pull requests, issues, etc.



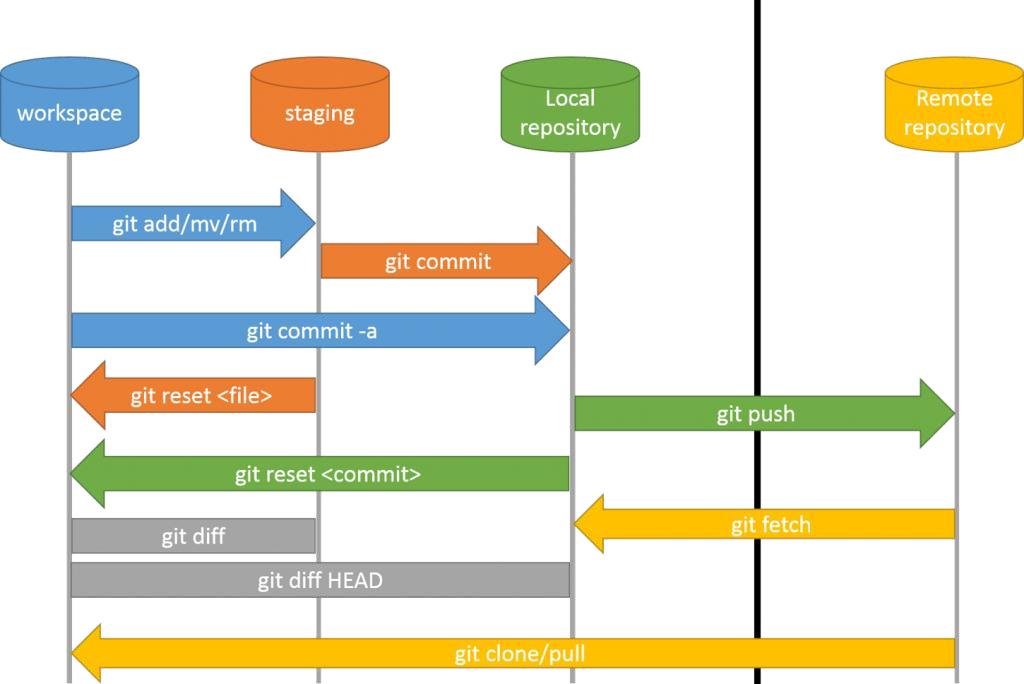
**<https://git-scm.com/book/es/v2>**



**Bitbucket /// Gitlab**



[**https://bitbucket.org/**](https://bitbucket.org/)[**https://about.gitlab.com/**](https://about.gitlab.com/)



# Flujo interno de Git

-

**• • 1 • ' • .** !

**Remete repository**

pull fetch ush

**Local repository**

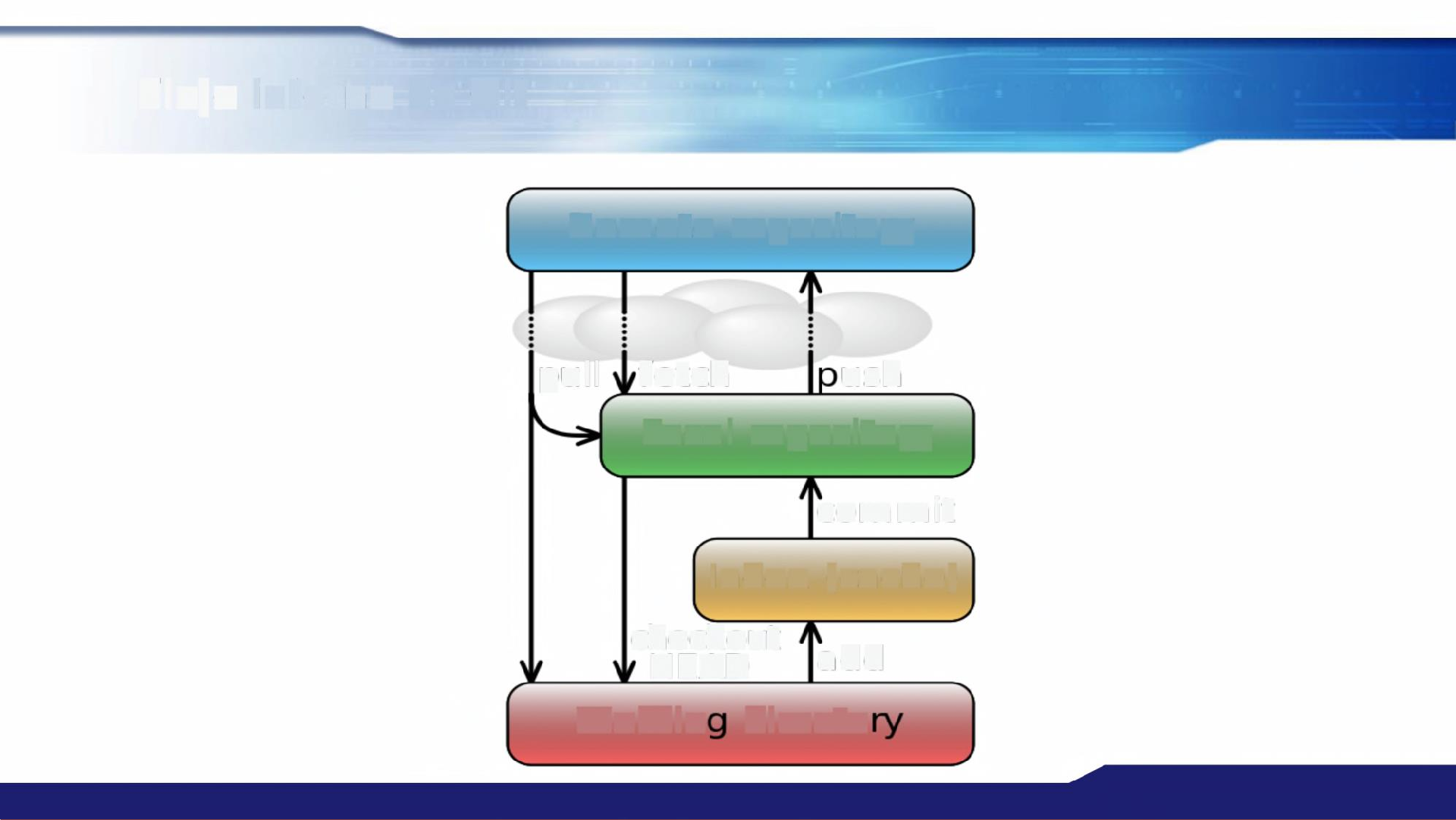
**lndex**

checkout

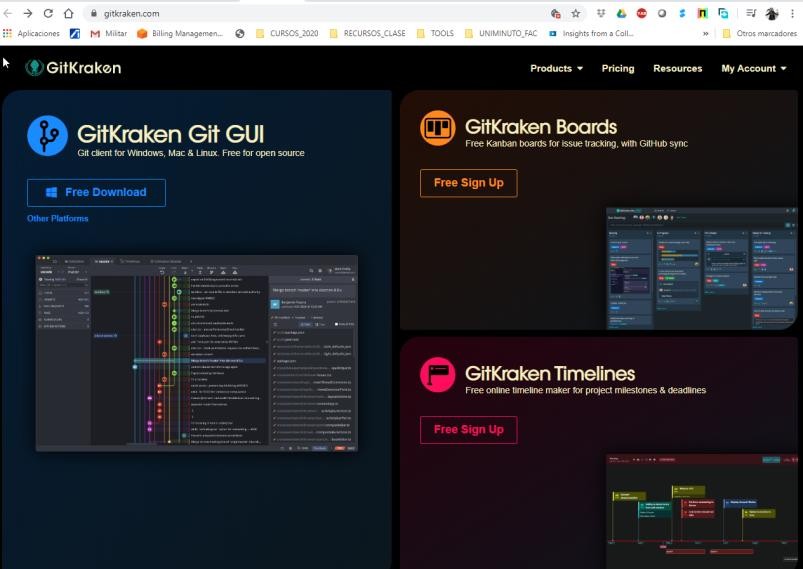
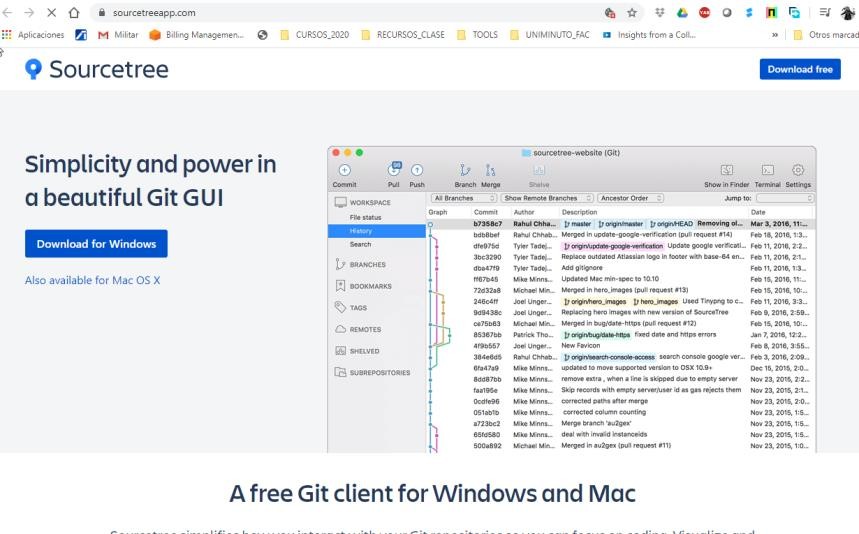
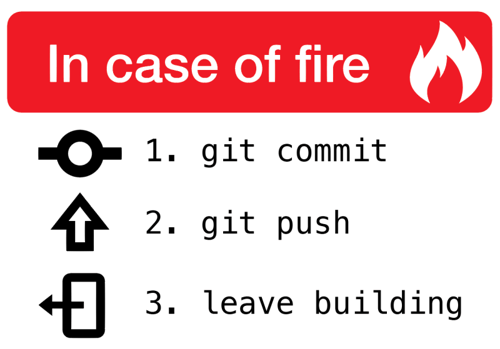
c o mmi t

(cache)

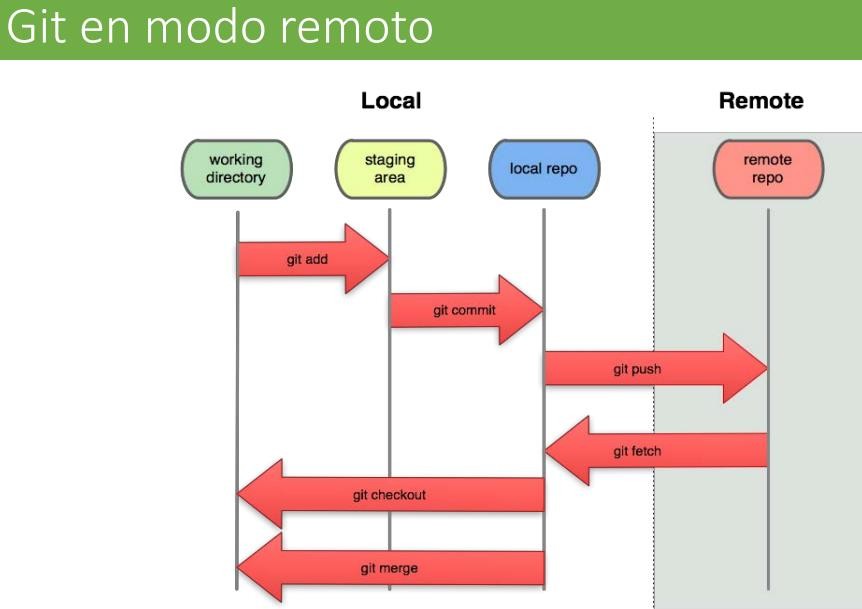
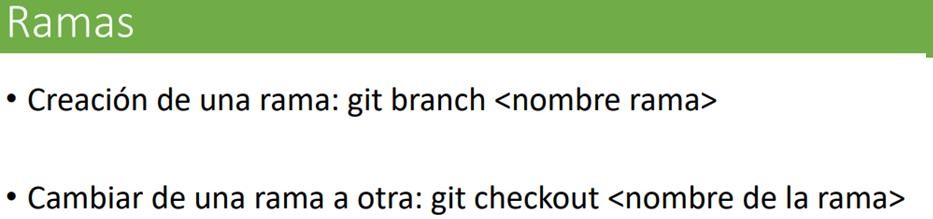
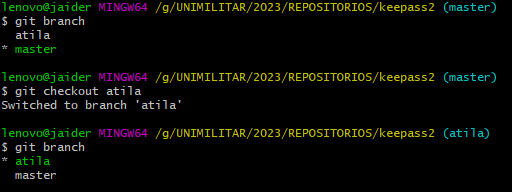
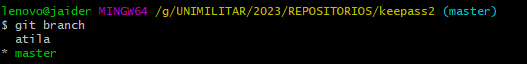
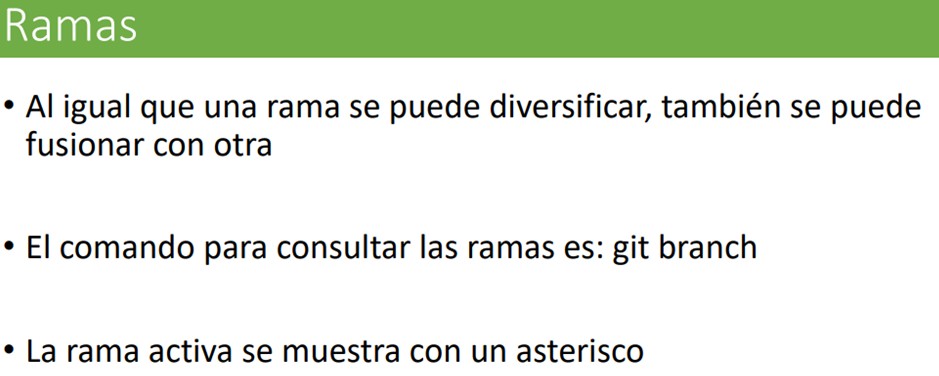
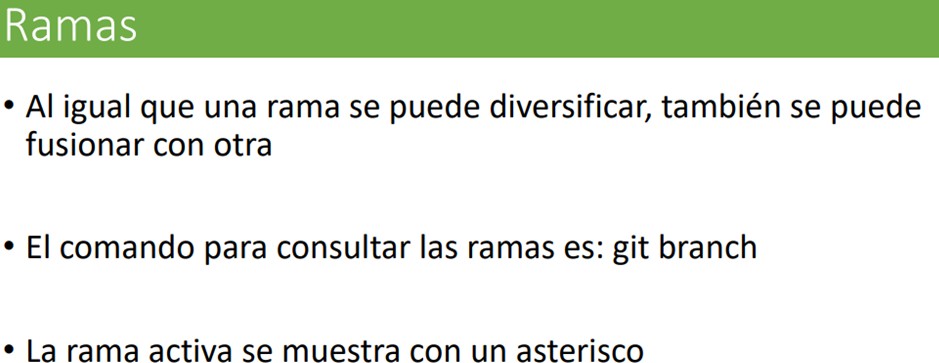
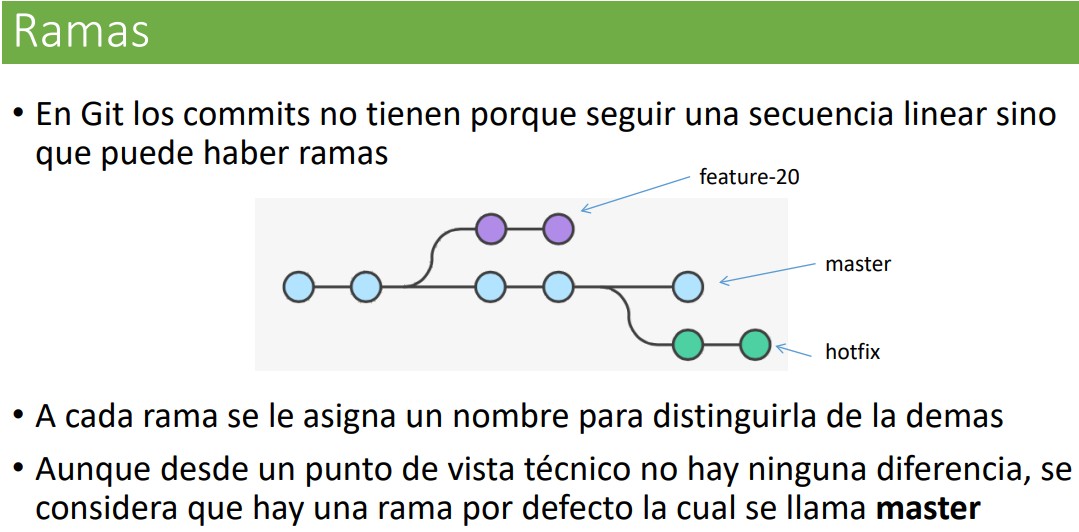
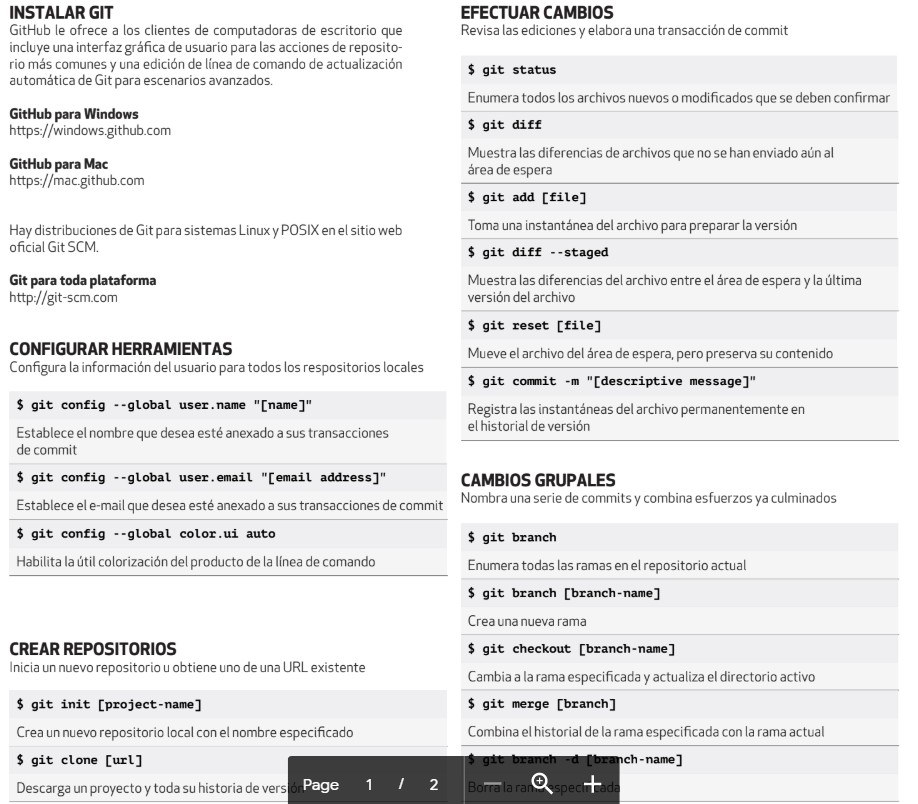
dd

HEAD a

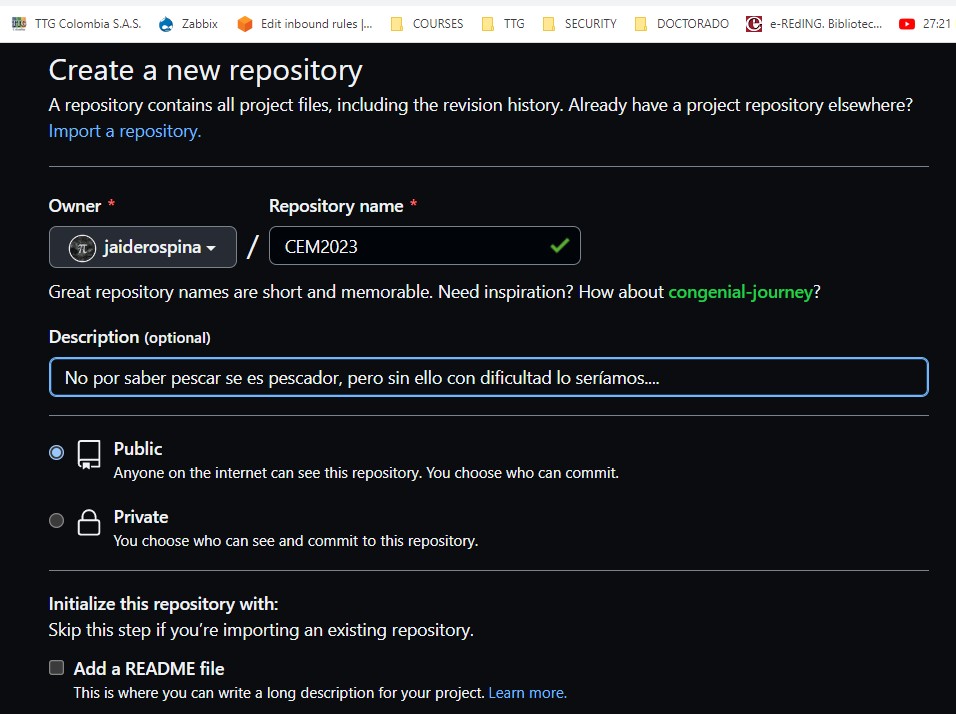
**Workin directo**



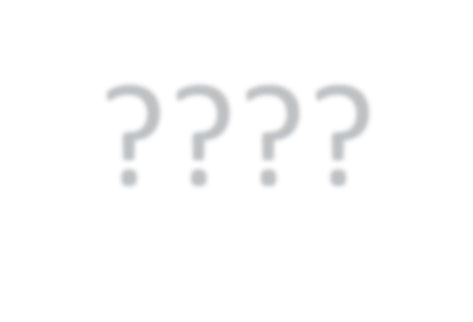
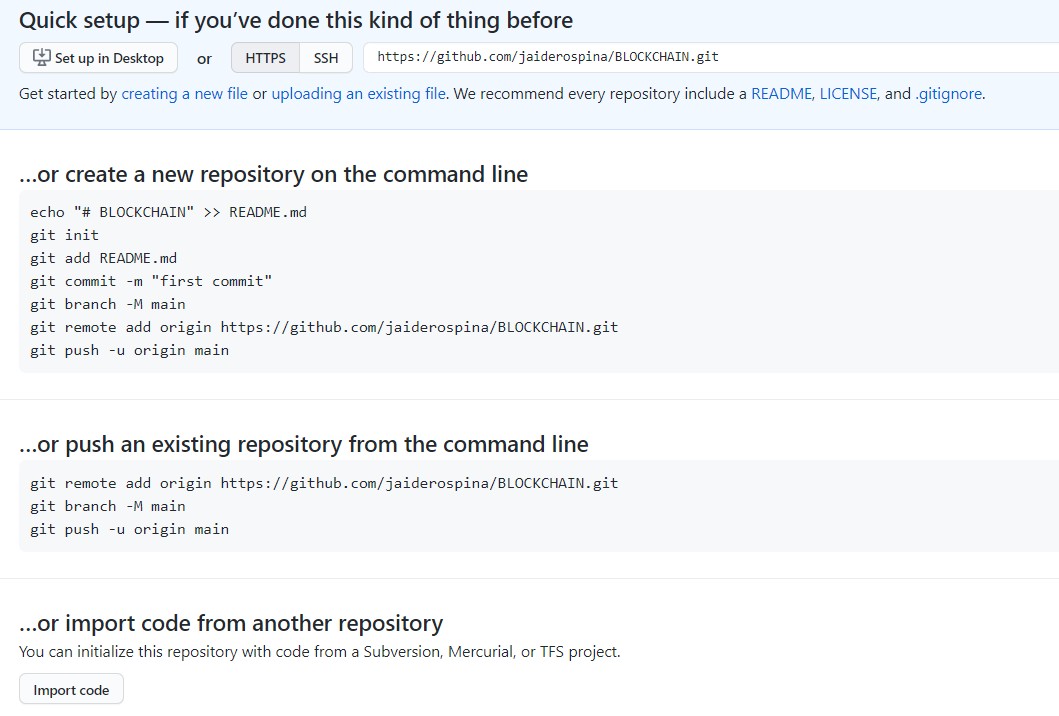
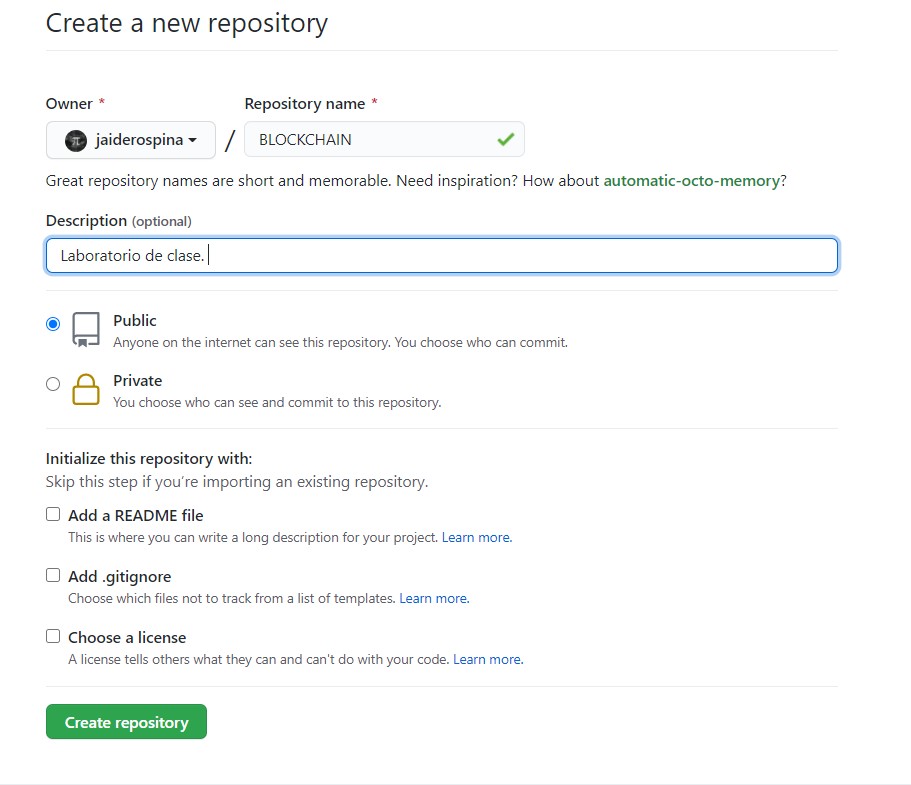
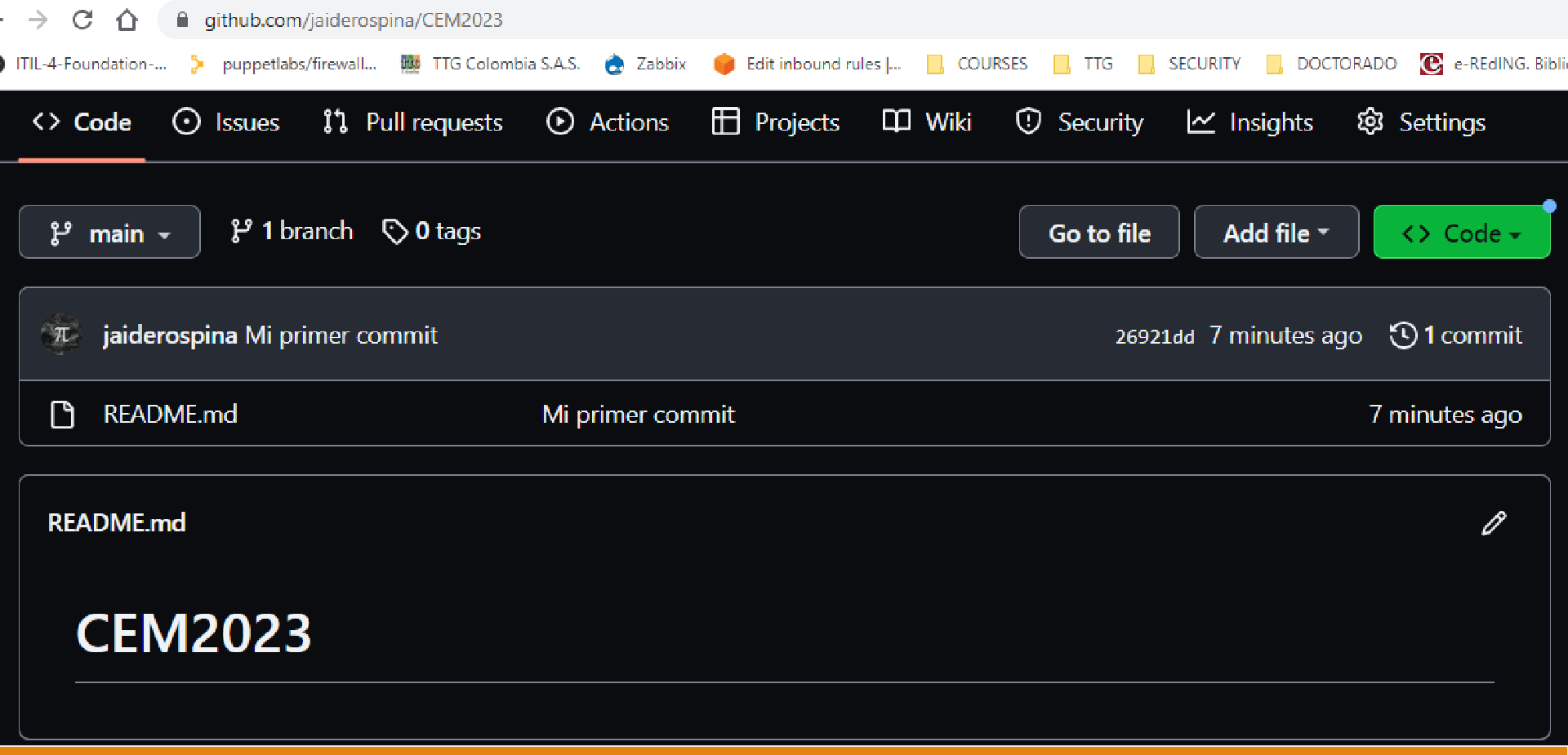
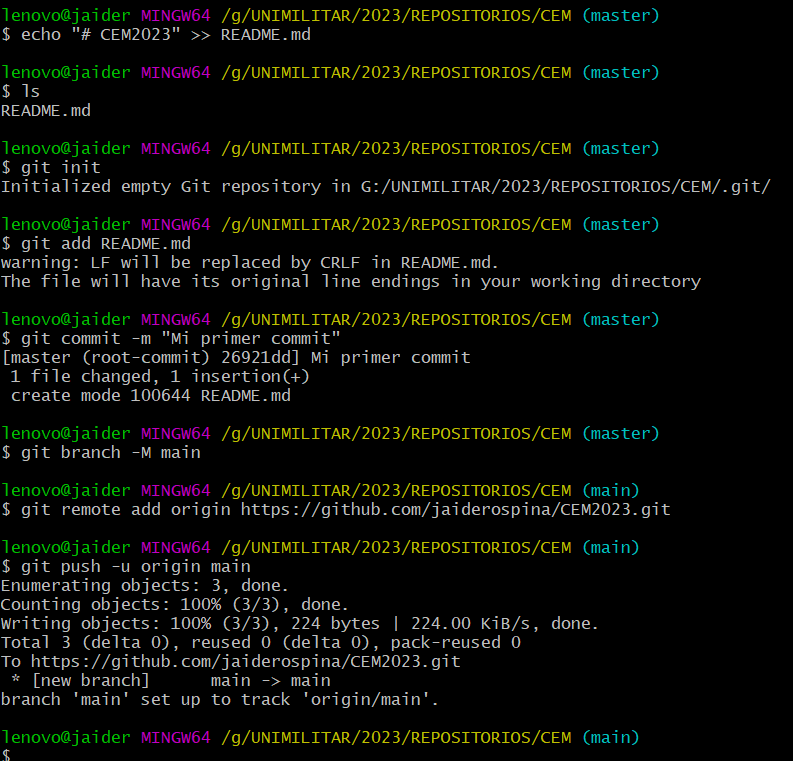
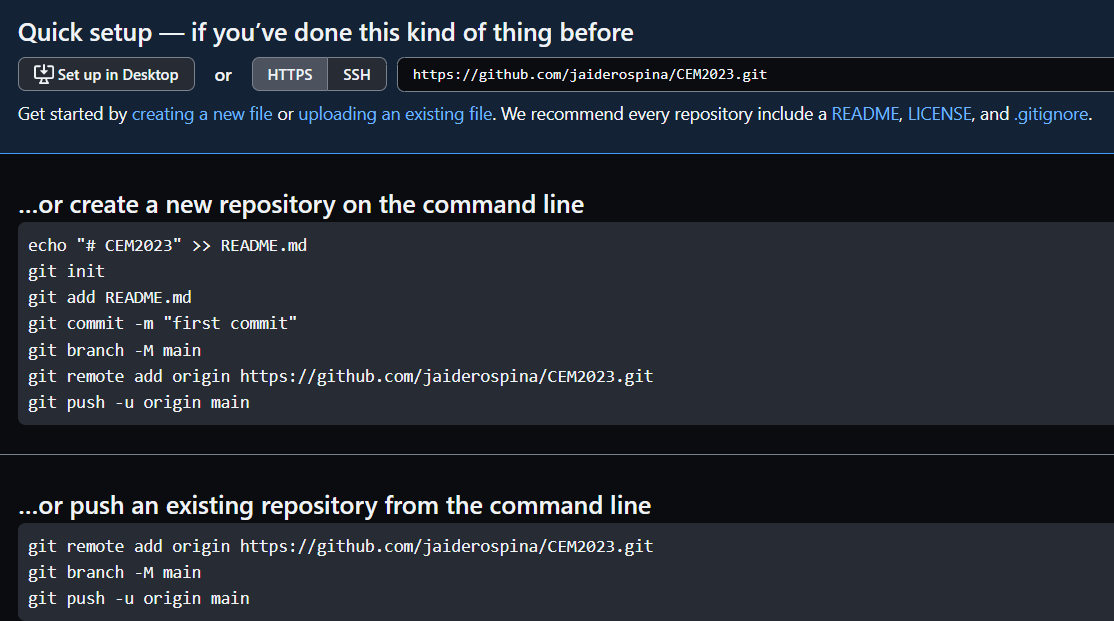
**<https://www.sourcetreeapp.com/>** [**https://www.gitkraken.com/**](https://www.gitkraken.com/)







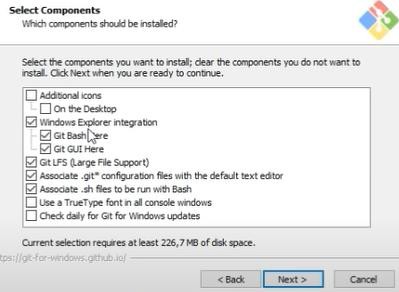
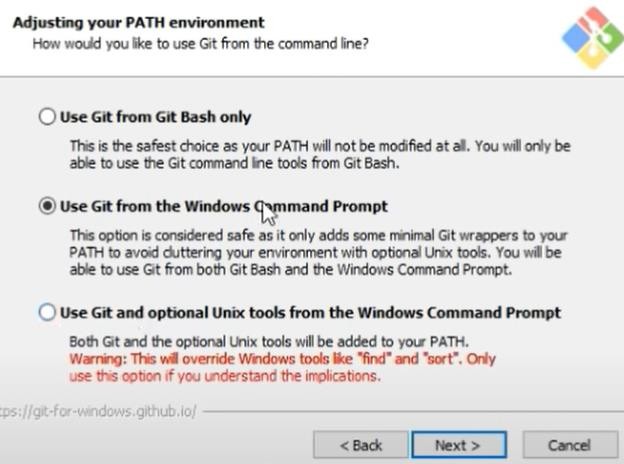
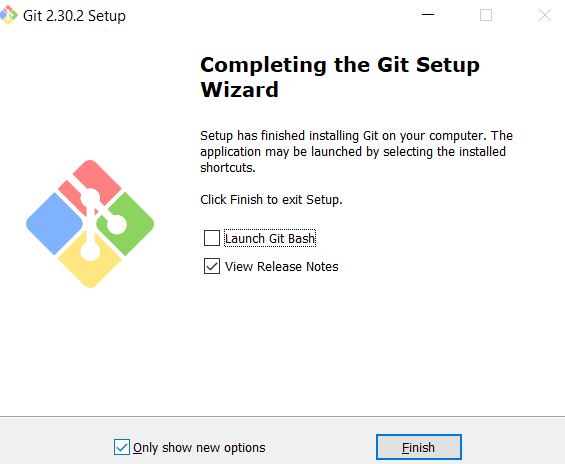
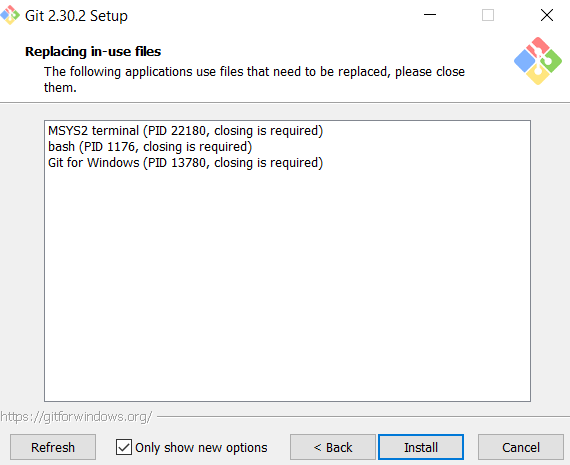
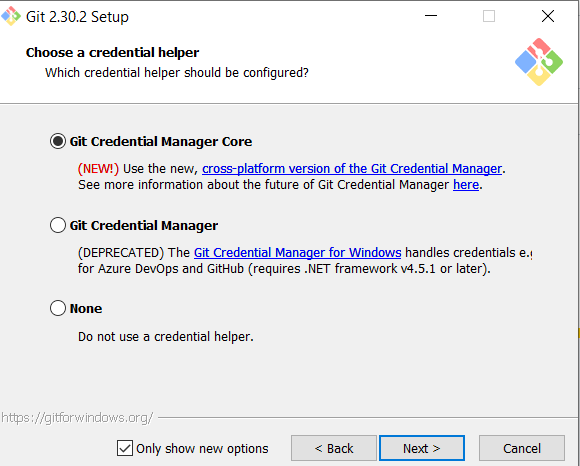
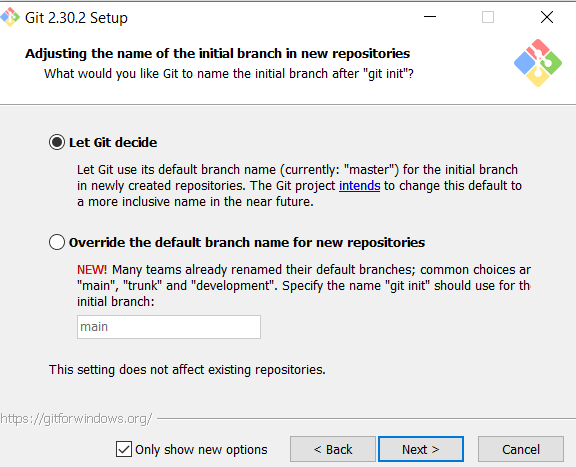




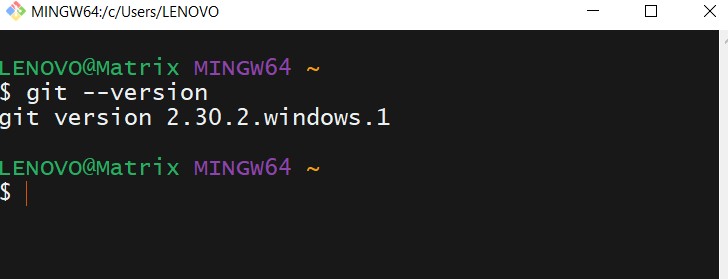
y….porqué git????

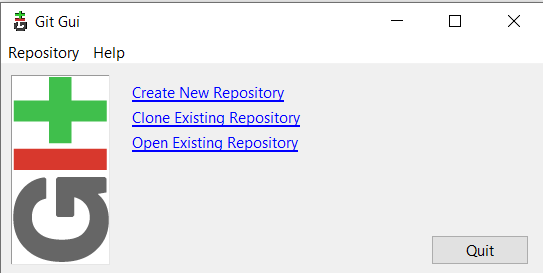


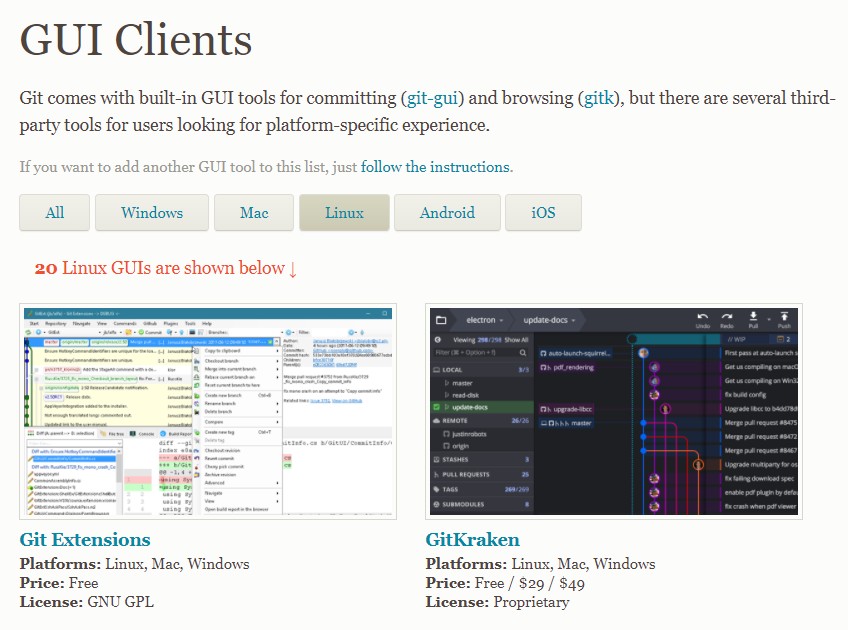
Instalación de git https://git-scm.com/











# Configuración Básica

-

**• • 1 • ' • .** !

* Abrimos la aplicación Git Bash y en ella ejecutamos:
* $ git config
* $ git config

--global user.name

--global user.email

"Mi nombre"

["mi@correo.com](mailto:mi@correo.com)"

# Git lnit

-

**• • 1 • ' • .** !



* Luego Solo hay que ubicarnos en la carpeta raíz de nuestro

proyecto y ejecutar git init para crear el repositorio:

* $ cd /proyectos/mi-proyecto
* $ git init

# Git status

-

**• • 1 • ' • .** !



* ejecutando git status podemos verificar los cambios hechos en el estado actual:
* $ g it status

# Git add

-

**• • 1 •**

**' • .** !

* Para registrar

ejecutamos:

* $ git add .

todos los archivos contenidos en nuestro proyecto

* y verificamos:
* $ git status
* Desde luego es solo especificar

posible agregar archivos individualmente con la ruta relativa con relación a la raíz del

repositorio, por ejemplo:

* $ git add mi-proyecto/archivo.php

# Git commit

-

**• • 1 • ' • .** !



* Finalmente debemos dar por hechos los cambios que llevamos en el repositorio, entonces ejecutamos git commit para confirmarlo:
* $ git commit -m "Primer commit; agregué la estructura de archivos"

# Git branch

-

**• • 1 •**

**' • .** !



* Lo ideal es que siempre trabajemos la versión estable de

nuestros proyectos en la rama master.

* para mostrar las ramas existentes localmente ejecutamos:
* $ git branch

# Git branch

-

**• • 1 • ' • .** !

* Es recomendable

desarrollo:

crear la rama develop para trabajar en el

* $ git branch develop
* para
* $ git

movernos entre ramas debemos usar checkout: checkout develop

# Git branch

-

**• • 1 •**

**' • .** !



* a partir de la rama en la que nos encontremos podemos crear ramas inferiores para separar nuestros cambios de acuerdo con la estructura de archivos que usemos en nuestro proyecto. Esto se hace mediante:
* $ git checkout -b subrama

# Git branch

-

**• • 1 •**

**' • .** !



* suponiendo que hemos trabajado sobre la subrama creada y no queremos perder los cambios realizados entonces debemos hacer commit:
* $ git commit -am "Comentario de ejemplo"
* el atributo -am nos permite agregar automáticamente los

archivos modificados y establecer un mensaje para el commit.

# Git merge

-

**• • 1 •**

**' • .** !



* Para integrar los cambios usamos la instruccion merge:
* nos ubicamos en la rama de desarrollo:
* $ git checkout develop
* y despues integramos los cambios
* $ git merge subrama
* Y ahí tenemos, nuestros cambios realizados en la rama auxiliar los hemos integrado a la rama oficial de desarrollo.

# Eliminando subramas

-

**• • 1 •**

**' • .** !



* Suponiendo que ya no vamos a usar la subrama creada pues nuestro código se encuentra estable y necesitamos eliminar referencias innecesarias. Hacemos lo siguiente para eliminarla:
* $ git branch -d subrama

# Clonando un repositorio existente

-

**• • 1 •**

**' • .** !

* Si deseas obtener una copia de un repositorio

comando que necesitas es git clone

Git existente el



* $ git clone git://git.diens.com.co/usuario/proyecto.git
* con esto crearemos una copia local de nuestro repositorio.

# Trabajando con repositorios Remotos

-

**• • 1 • ' • .** !

* Para ver qué repositorios remotos tienes configurados,

ejecutar el comando remate.

* $ git remate -v

puedes

* con la opción -v podemos ver remoto.

la URL asociada a cada repositorio

# Añadiendo repositorios remotos

-

**• • 1 •**

**' • .** !

* Para añadir

nombre con

* $ git remate

un nuevo repositorio Git remoto, asignándole un

el que referenciarlo fácilmente. add [nombre] [url]:

* $ git remate add us git://git.diens.com.co/usuario/proyecto.git
* Por ejemplo, para recuperar toda la información de 'usuario'

todavía no tienes en tu repositorio, puedes ejecutar:

que

* $ git fetch us

# Recibiendo de tus repositorios remotos

-

**• • 1 •**

**' • .** !

* Como acabas de ver, para recuperar datos

remotos puedes ejecutar:

* $ git fetch [remote-name]

de tus repositorios

* Este comando recupera todos no tengas todavía.

los datos del proyecto remoto que

# Enviando a tus repositorios remotos

-

**• • 1 •**

**' • .** !

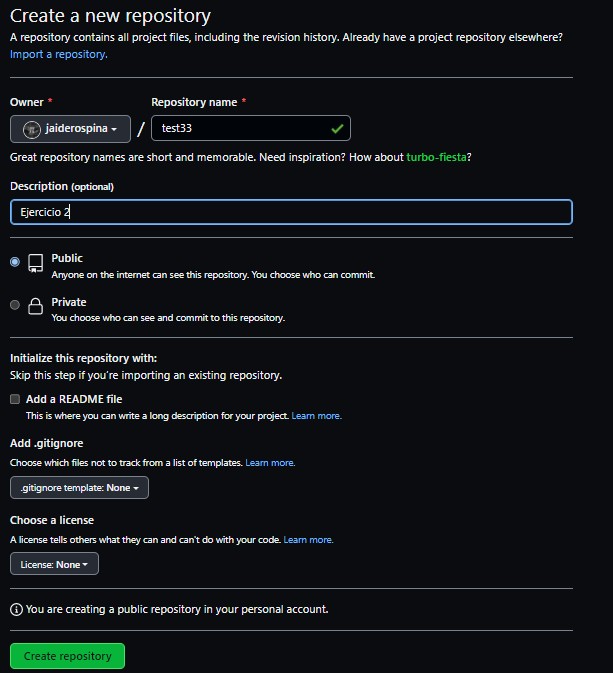


* Cuando tu proyecto se encuentra en un estado que quieres

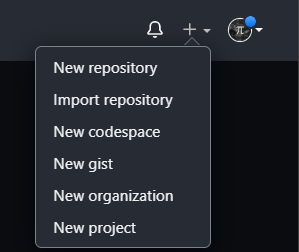
compartir el comando que te permite hacer esto es sencillo:

* $ git push [nombre-remoto][nombre-rama]
* Por ejemplo para enviar a la rama mestra del servidor de origen:
* $ git push origin master

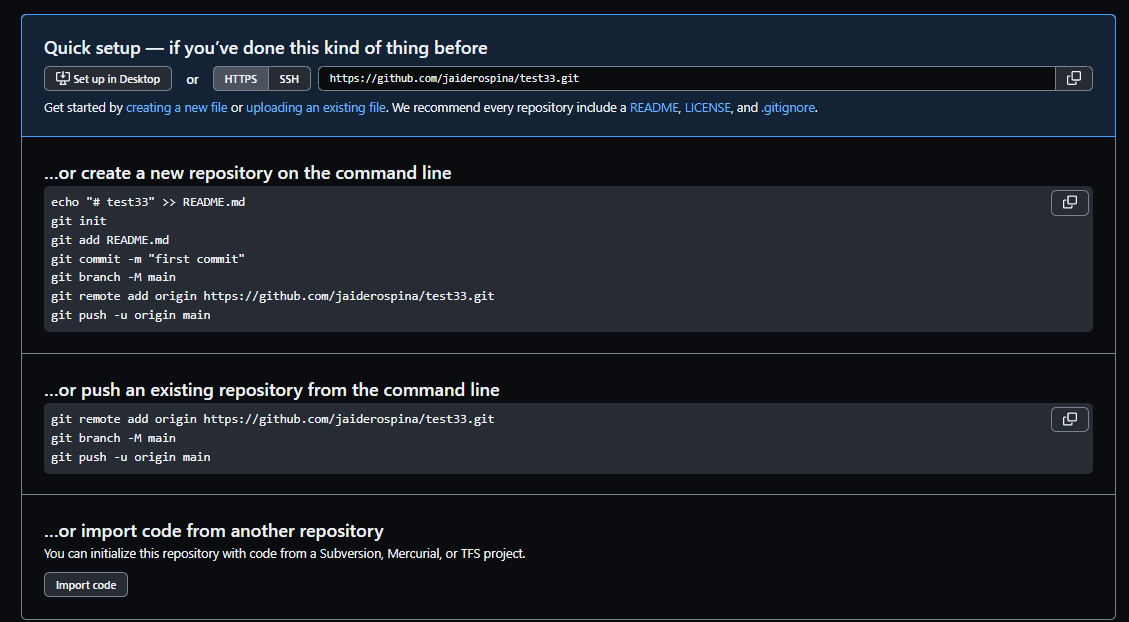
Ejemplo práctico 2



Repositorio desde Github



**Ejemplo práctico 2 Repositorio desde Github**



Terminar el ejercicio, realizando un repo local y enviándolo a github.

**Ejemplo práctico 2 Repositorio desde Github**

-

**• • 1 • ' • .** !

Continuar estos pasos en su repositorio

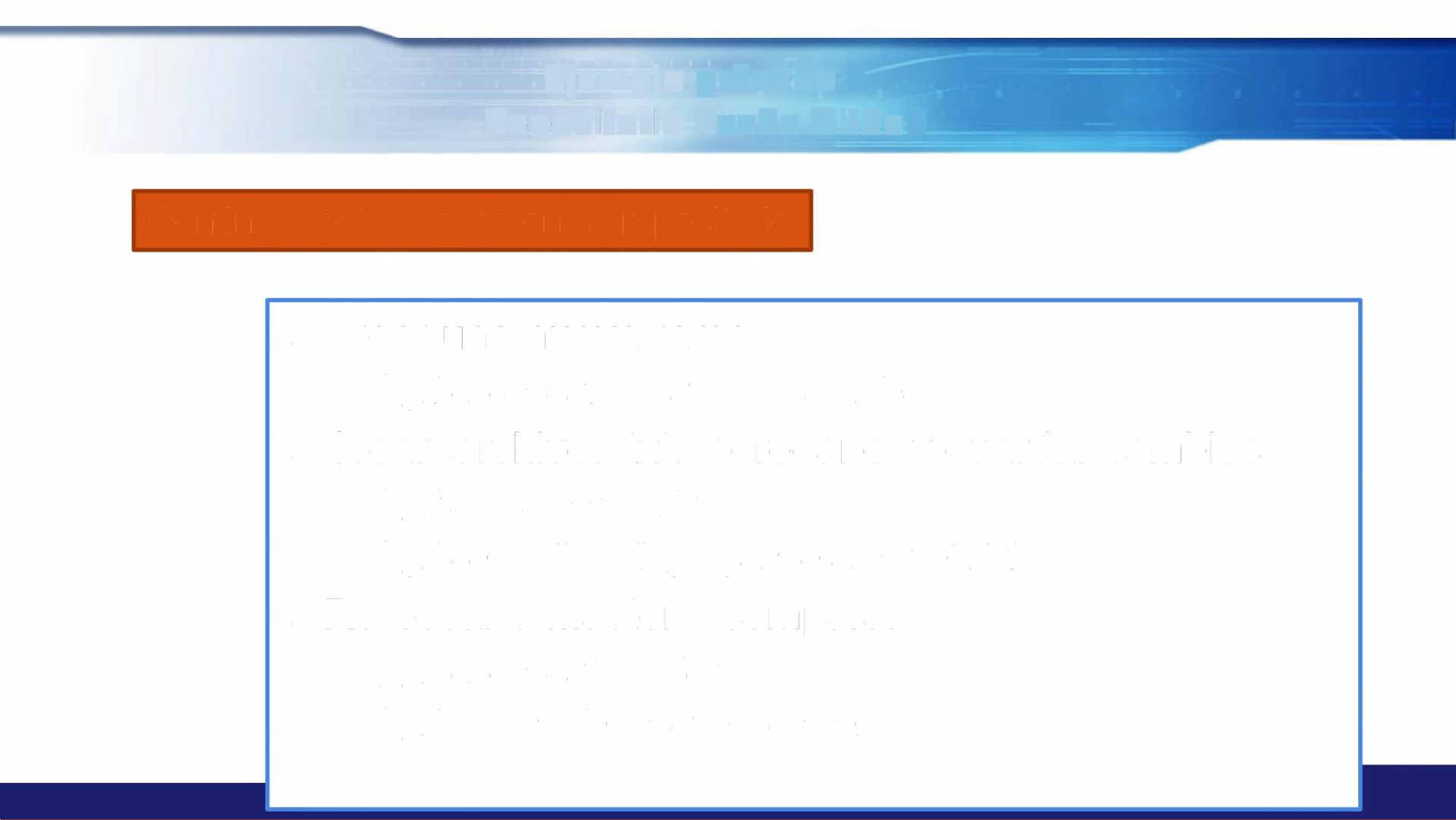
* + Crear una nueva rama
    - *$ git checkout -b nuevos\_cambios*
  + Crear archivo7.txt,

agregarlo

y guardar cambios

* + - *$ git*

*add archivo7.txt*

* + - *$ git commit* -m *"Agregando archivo* 7. *txt"*
  + Regresar a master y comparar
    - *$ git checkout master*
    - *$ git diff master nuevos\_cambios*

# Ejemplo práctico

-

**• • 1 • ' • .** !

* Realizar merge y enviar cambios
  + *$ git merge nuevos\_cambios*
  + *$ git push origin master*
* Eliminar rama que ya no se empleará

*•*

* Ir

*•*

*git branch -d nuevos\_cambios*

a newproject y traer los nuevos cambios

*$ git pul/ origin master*

# Github / Bitbucket

-

**• • 1 •**

**' • .** !

# . Github

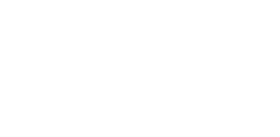
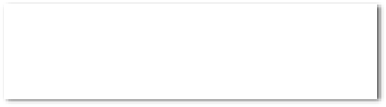
* Trabaja con repositorios de pago para repositorio

# . Bitbucket

netamente de tipo git, tiene una privados .

versión

* A diferencia del anterior permite trabajar con repositorios privados y trabaja con mercurial y git.



TALLER



Referencias

https://aprendiendoarduino.wordpress.com/tag/git/ https://bitbucket.org/ https:/[/www.sourcetreeapp.com/](http://www.sourcetreeapp.com/)

-

• • 1 • ' • . !