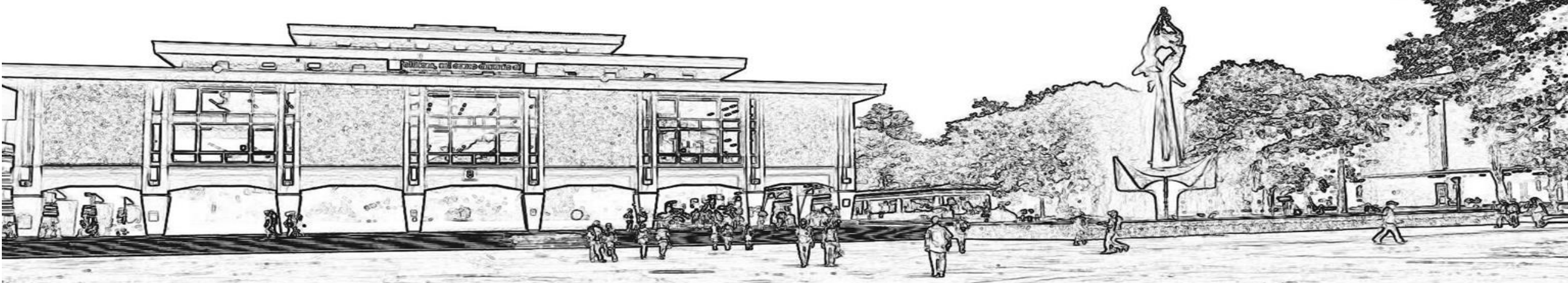




UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

Ingeniería & Sociedad



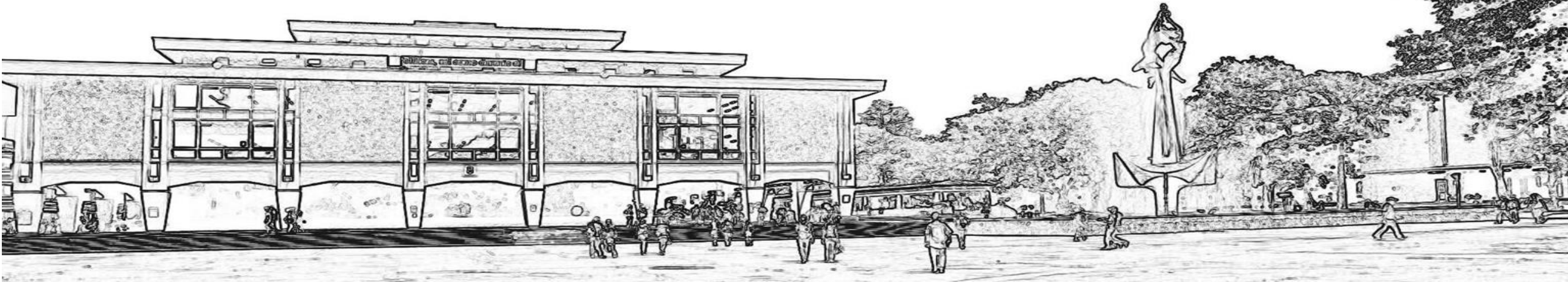
Julián Andrés Castillo Grisales
Ingeniero de Sistemas
Magister en Ingeniería
jandres.castillo@udea.edu.co



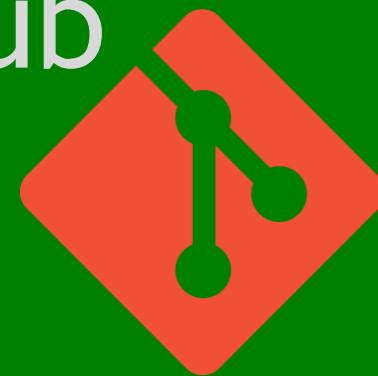
UNIVERSIDAD DE ANTIOQUIA

Facultad de Ingeniería

Ingeniería & *Sociedad*



Introducción a Git con GitHub

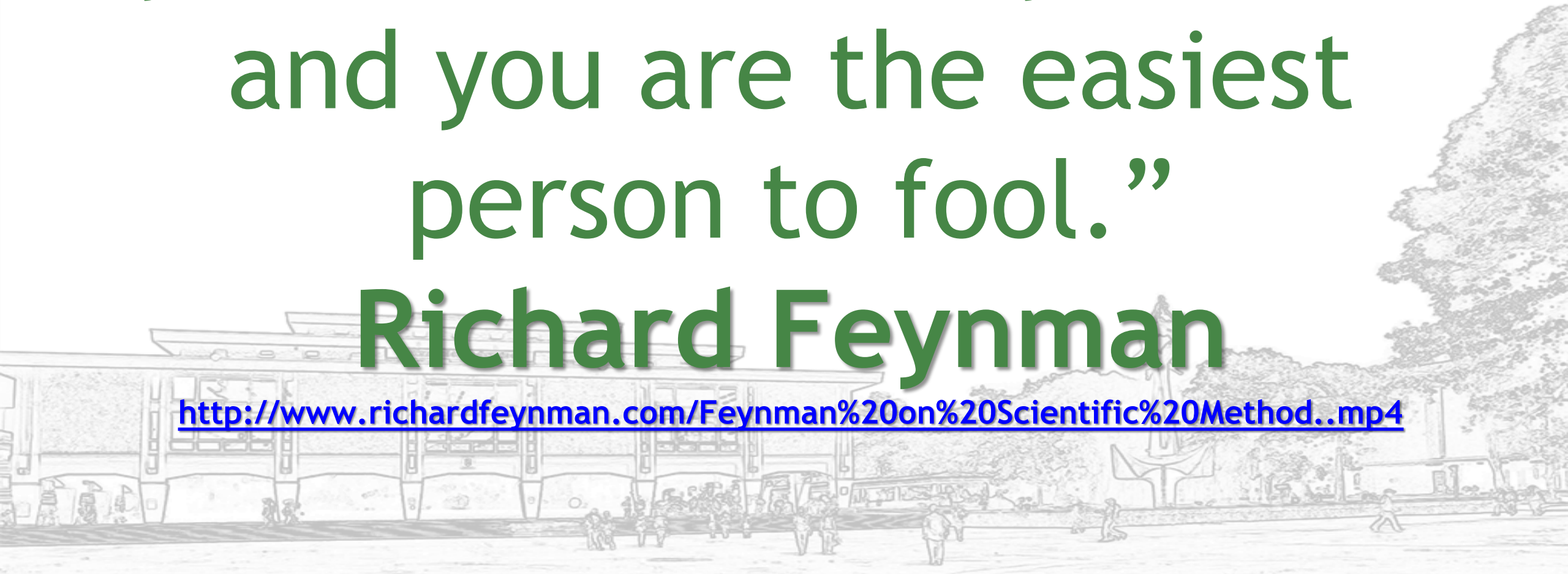


git

“The first principle is that
you must not fool yourself,
and you are the easiest
person to fool.”

Richard Feynman

<http://www.richardfeynman.com/Feynman%20on%20Scientific%20Method..mp4>



Agenda

- 1.VCS
- 2.Git
- 3.Instalación de Git
- 4.Cuenta en GitHub
- 5.Crear un repositorio

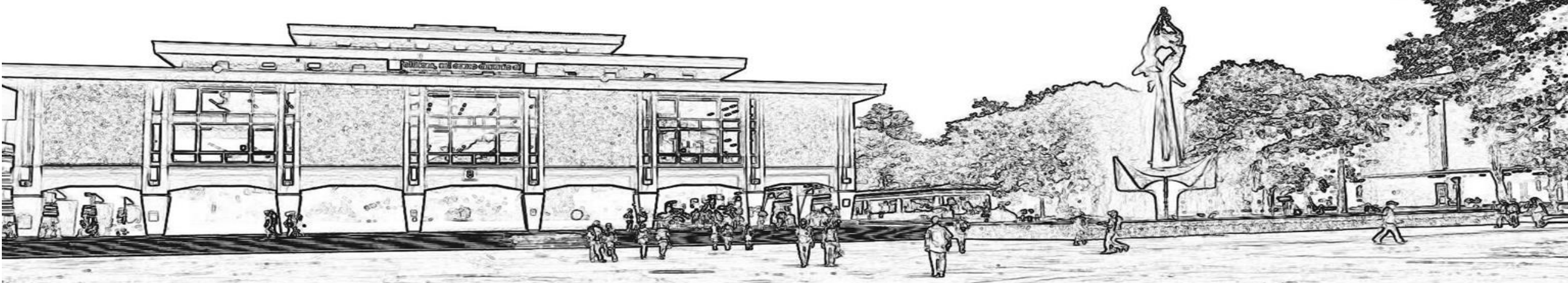




UNIVERSIDAD DE ANTIOQUIA

Ingeniería & *Sociedad*

Facultad de Ingeniería



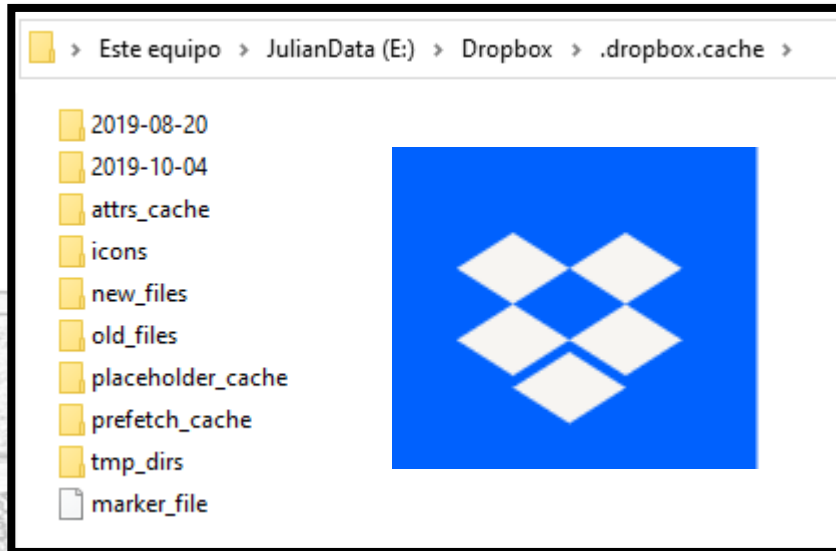
1. VCS - Version Control System

Sistema de control de versiones

VCS - Version Control System

¿Qué es el "control de versiones" y por qué debería importarme?

El control de versiones es un sistema que registra los cambios en un archivo o conjunto de archivos a lo largo del tiempo para poder recuperar versiones específicas en un futuro o a necesidad. Los ejemplos detallados aquí se realizan orientados a código o software, pero pueden aplicarse a cualquier tipo de archivo en una computadora. Ejemplo de control de versiones para archivos:



Google Drive

Dropbox



OneDrive

amazon drive

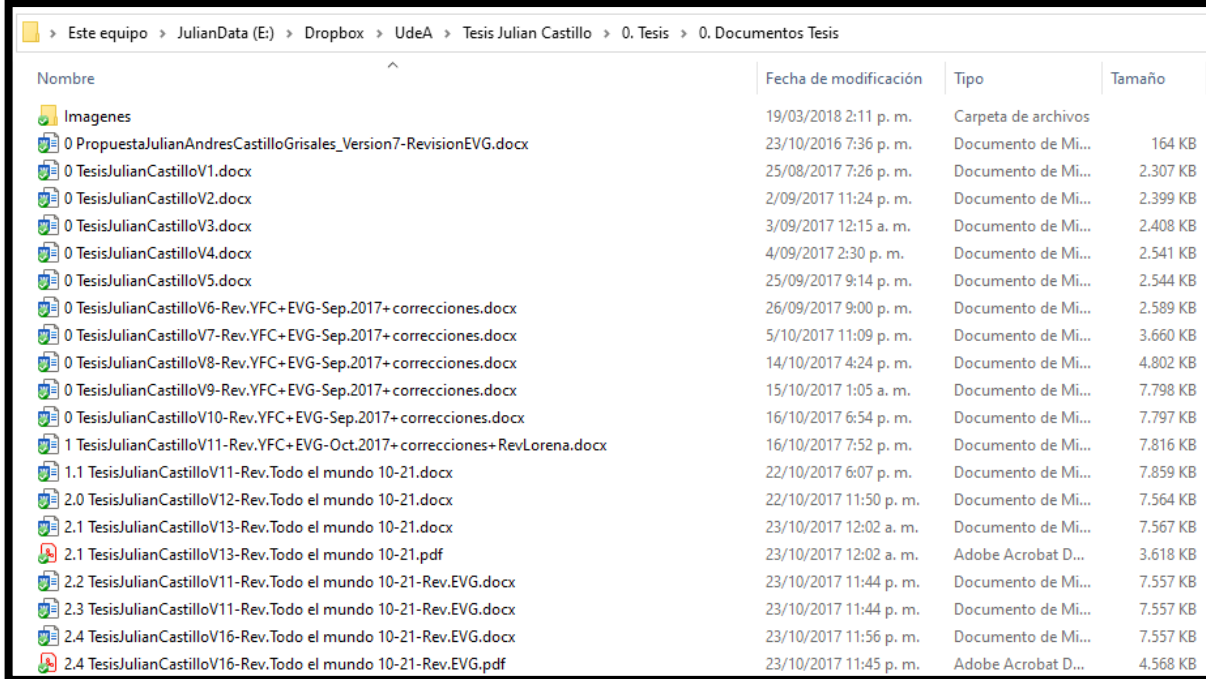


VCS - Version Control System

¿Qué es el "control de versiones" y por qué debería importarme?

Supongamos que estamos realizando nuestra tesis de maestría, lo normal a realizar por parte de nosotros como autores es tener un versionado personalizado, basado en iteraciones.

El uso de un VCS también generalmente significa que si arruinas las cosas o pierdes archivos, puedes recuperarlos fácilmente. Además, obtienes todo esto por muy pocos gastos computacionales.



Nombre	Fecha de modificación	Tipo	Tamaño
Imágenes	19/03/2018 2:11 p. m.	Carpeta de archivos	
0 PropuestaJulianAndresCastilloGrisales_Version7-RevisionEVG.docx	23/10/2016 7:36 p. m.	Documento de Mi...	164 KB
0 TesisJulianCastilloV1.docx	25/08/2017 7:26 p. m.	Documento de Mi...	2.307 KB
0 TesisJulianCastilloV2.docx	2/09/2017 11:24 p. m.	Documento de Mi...	2.399 KB
0 TesisJulianCastilloV3.docx	3/09/2017 12:15 a. m.	Documento de Mi...	2.408 KB
0 TesisJulianCastilloV4.docx	4/09/2017 2:30 p. m.	Documento de Mi...	2.541 KB
0 TesisJulianCastilloV5.docx	25/09/2017 9:14 p. m.	Documento de Mi...	2.544 KB
0 TesisJulianCastilloV6-Rev.YFC+EVG-Sep.2017+correcciones.docx	26/09/2017 9:00 p. m.	Documento de Mi...	2.589 KB
0 TesisJulianCastilloV7-Rev.YFC+EVG-Sep.2017+correcciones.docx	5/10/2017 11:09 p. m.	Documento de Mi...	3.660 KB
0 TesisJulianCastilloV8-Rev.YFC+EVG-Sep.2017+correcciones.docx	14/10/2017 4:24 p. m.	Documento de Mi...	4.802 KB
0 TesisJulianCastilloV9-Rev.YFC+EVG-Sep.2017+correcciones.docx	15/10/2017 1:05 a. m.	Documento de Mi...	7.798 KB
0 TesisJulianCastilloV10-Rev.YFC+EVG-Sep.2017+correcciones.docx	16/10/2017 6:54 p. m.	Documento de Mi...	7.797 KB
1 TesisJulianCastilloV11-Rev.YFC+EVG-Oct.2017+correcciones+RevLorena.docx	16/10/2017 7:52 p. m.	Documento de Mi...	7.816 KB
1.1 TesisJulianCastilloV11-Rev.Todo el mundo 10-21.docx	22/10/2017 6:07 p. m.	Documento de Mi...	7.859 KB
2.0 TesisJulianCastilloV12-Rev.Todo el mundo 10-21.docx	22/10/2017 11:50 p. m.	Documento de Mi...	7.564 KB
2.1 TesisJulianCastilloV13-Rev.Todo el mundo 10-21.docx	23/10/2017 12:02 a. m.	Documento de Mi...	7.567 KB
2.1 TesisJulianCastilloV13-Rev.Todo el mundo 10-21.pdf	23/10/2017 12:02 a. m.	Adobe Acrobat D...	3.618 KB
2.2 TesisJulianCastilloV11-Rev.Todo el mundo 10-21-Rev.EVG.docx	23/10/2017 11:44 p. m.	Documento de Mi...	7.557 KB
2.3 TesisJulianCastilloV11-Rev.Todo el mundo 10-21-Rev.EVG.docx	23/10/2017 11:44 p. m.	Documento de Mi...	7.557 KB
2.4 TesisJulianCastilloV16-Rev.Todo el mundo 10-21-Rev.EVG.docx	23/10/2017 11:56 p. m.	Documento de Mi...	7.557 KB
2.4 TesisJulianCastilloV16-Rev.Todo el mundo 10-21-Rev.EVG.pdf	23/10/2017 11:45 p. m.	Adobe Acrobat D...	4.568 KB



Dropbox

Tu Dropbox está casi lleno. [Subir de categoría](#)

Buscar

Página principal

> Todos los archivos

Recientes

Destacados

Compartido

Solicitudes de archivos

Archivos eliminados

2.4 TesisJulianCastilloV16-Rev.Todo el... Historial de versiones

Puedes restaurar cualquier versión a continuación para convertirla en el archivo actual. De todos modos, se guardarán todas las demás versiones.

16 de febrero de 2018

Julian Castillo movió el archivo a /0. tesis/0. Documentos Tesis/2.4 TesisJulianCastilloV1... 7,38 MB [Versión actual](#)

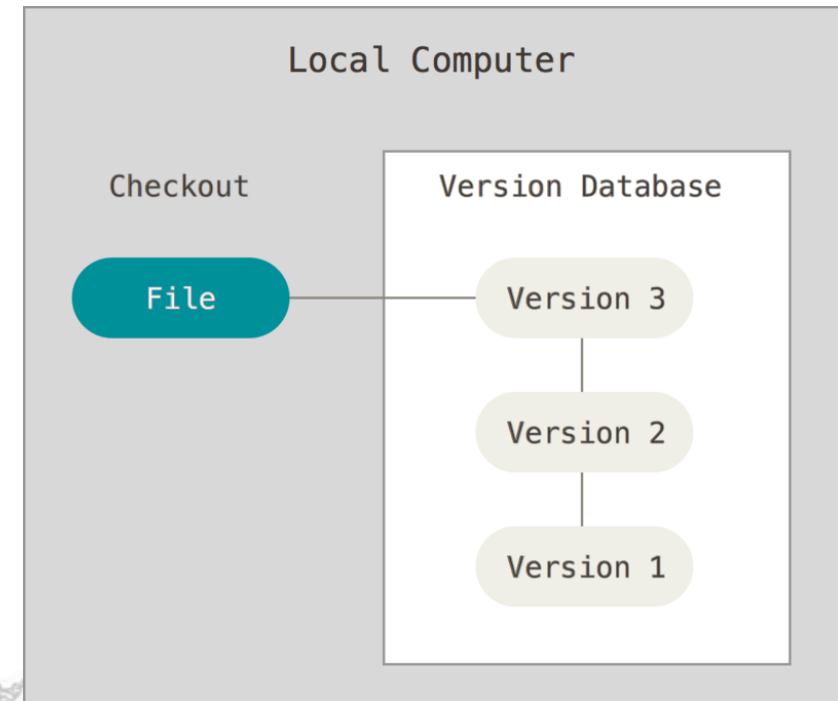
20:33 • Escritorio

VCS - Version Control System

VCS - Locales

Para solucionar el problema anterior de la tesis de grado, hace mucho tiempo que los programadores desarrollaron VCS locales que tenían una base de datos simple que mantuvo todos los cambios en los archivos bajo control de revisión.

Una de las herramientas de VCS más populares fue un sistema llamado “Revision Control System” ([RCS](https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control)), que todavía se distribuye en muchas computadoras en la actualidad.



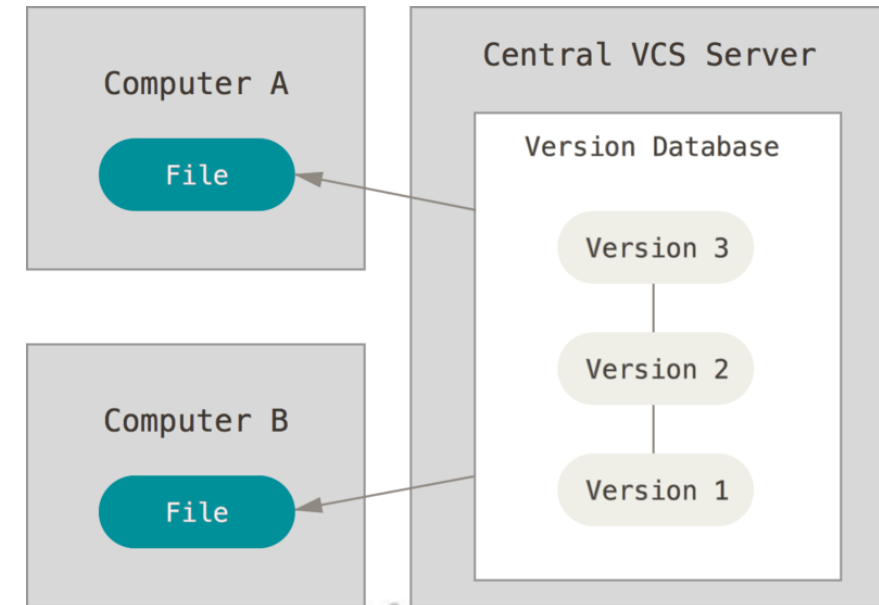
VCS - Version Control System

CVCS - Centralized Version Control Systems

El siguiente gran problema fue cuando varias personas necesitan colaborar archivos con otros sistemas.

Para solucionar este problema, se desarrollaron sistemas de control de versiones centralizados (CVCS).

Estos sistemas (como CVS, Subversion y Perforce) tienen un solo servidor que contiene todos los archivos versionados y una cantidad de clientes que extraen archivos desde ese lugar central. Durante muchos años, este ha sido el estándar para el control de versiones.



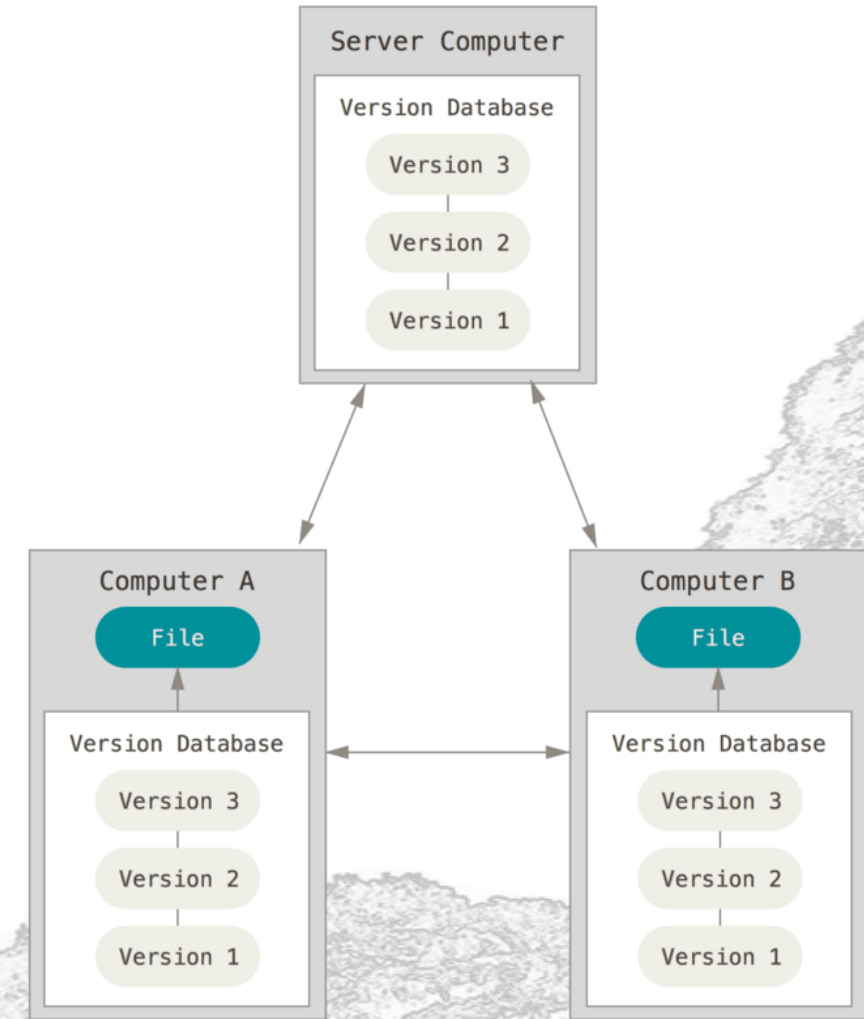
VCS - Version Control System

DVCS - Distributed Version Control Systems

Aquí es donde intervienen los sistemas de control de versiones distribuidos (DVCS).

En un DVCS (como Git, Mercurial, Bazaar o Darcs), los clientes reflejan completamente el repositorio, incluido su historial completo.

Por lo tanto, si algún servidor muere y estos sistemas colaboran a través de ese servidor, cualquiera de los repositorios del cliente se puede copiar en el servidor para restaurarlo. Cada clon es realmente una copia de seguridad completa de todos los datos.

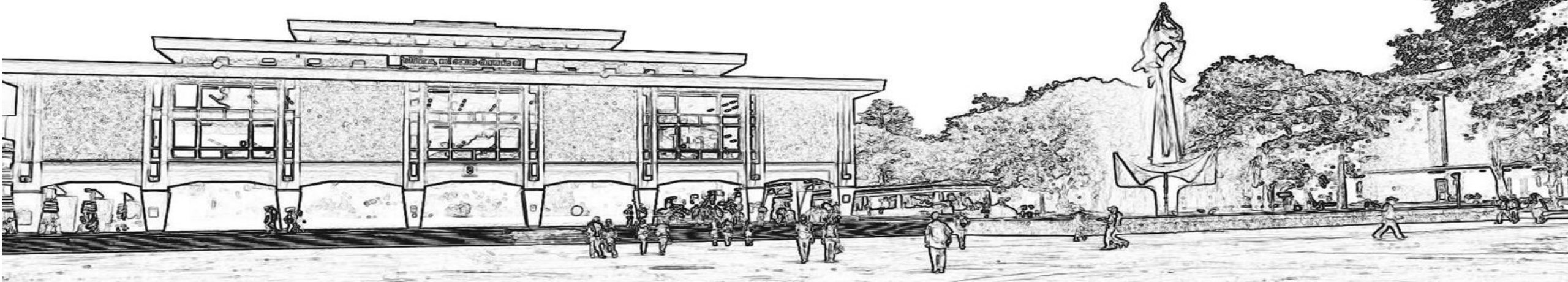




UNIVERSIDAD DE ANTIOQUIA

Ingeniería & Sociedad

Facultad de Ingeniería



1. Git

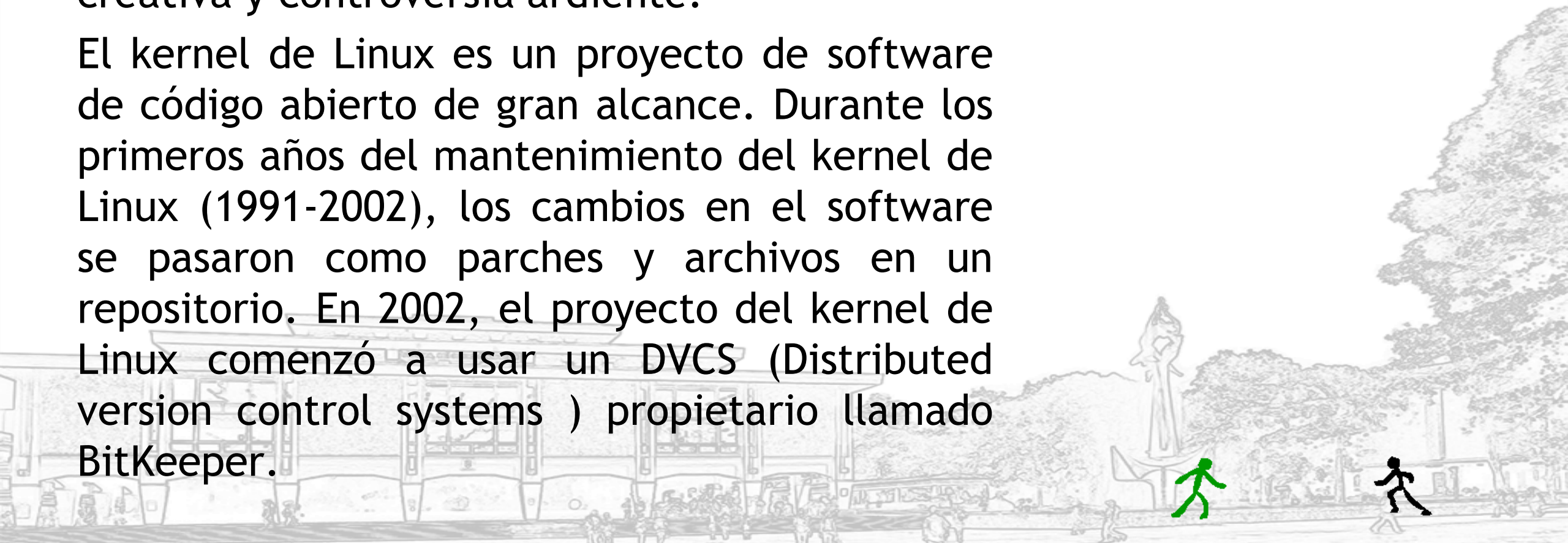


Al principio existía BitKeeper



Al igual que con muchas grandes cosas en la vida, Git comenzó con un poco de destrucción creativa y controversia ardiente.

El kernel de Linux es un proyecto de software de código abierto de gran alcance. Durante los primeros años del mantenimiento del kernel de Linux (1991-2002), los cambios en el software se pasaron como parches y archivos en un repositorio. En 2002, el proyecto del kernel de Linux comenzó a usar un DVCS (Distributed version control systems) propietario llamado BitKeeper.



En 2005, la relación entre la comunidad que desarrolló el núcleo Linux y la compañía comercial que desarrolló BitKeeper se rompió cuando unilateralmente BitKeeper decidió revocar la licencia de uso gratuito del kernel de Linux.


Esto llevó a la comunidad de desarrollo de Linux y Linus Torvalds, el creador de Linux, a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron mientras usaban BitKeeper. Algunos de los objetivos del nuevo sistema eran los siguientes:

- Velocidad
- Diseño simple
- Fuerte apoyo al desarrollo no lineal (miles de ramas paralelas)
- Totalmente distribuido
- Capaz de manejar grandes proyectos como el kernel de Linux

Git



No solo existe Git

-  **GitHub** • [GitHub](#): GitHub es un proveedor de alojamiento para el desarrollo de software y el control de versiones mediante Git. Ofrece el control de versiones distribuido y la funcionalidad de administración de código fuente de Git.
-  **GitLab** • [GitLab](#): GitLab es software para crear código colaborativo, administrar repositorios Git con excelente nivel de detalle para mantener tu código Seguro.
-  **Beanstalk** • [Beanstalk](#): Beanstalk es una opción ideal para trabajar desde lugares remotos. Este software es orientado al navegador y la nube.
-  **PERFORCE** • [Perforce](#): Perforce ofrece las capacidades de control de versiones a través de HelixCore. Nota: Gratis hasta 5 personas.
-  **SUBVERSION** • [Apache Subversion](#): Apache Subversion tanto el campo de código abierto como las empresas lo consideran una opción confiable para datos valiosos.
-  **AWS CodeCommit** • [AWS CodeCommit](#): AWS CodeCommit aloja repositorios de Git privados. Se conecta sin problemas con productos de Amazon Web Services (AWS).
-  **Azure DevOps** • [Azure DevOps Server](#): Anteriormente conocido como Team Foundation Server (TFS), Azure DevOps Server es un conjunto de herramientas de desarrollo de software colaborativo, alojadas localmente el cual se integra con su IDE o editor existente.

Otros:



mercurial

CVS - Concurrent Versions System



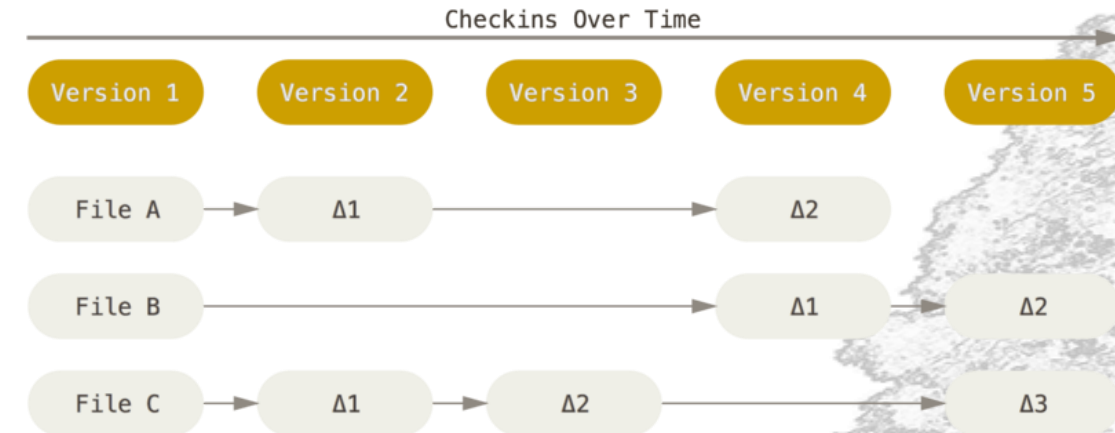
¿Qué es Git?

Snapshots, Not Differences

↓ Diferencias ↓

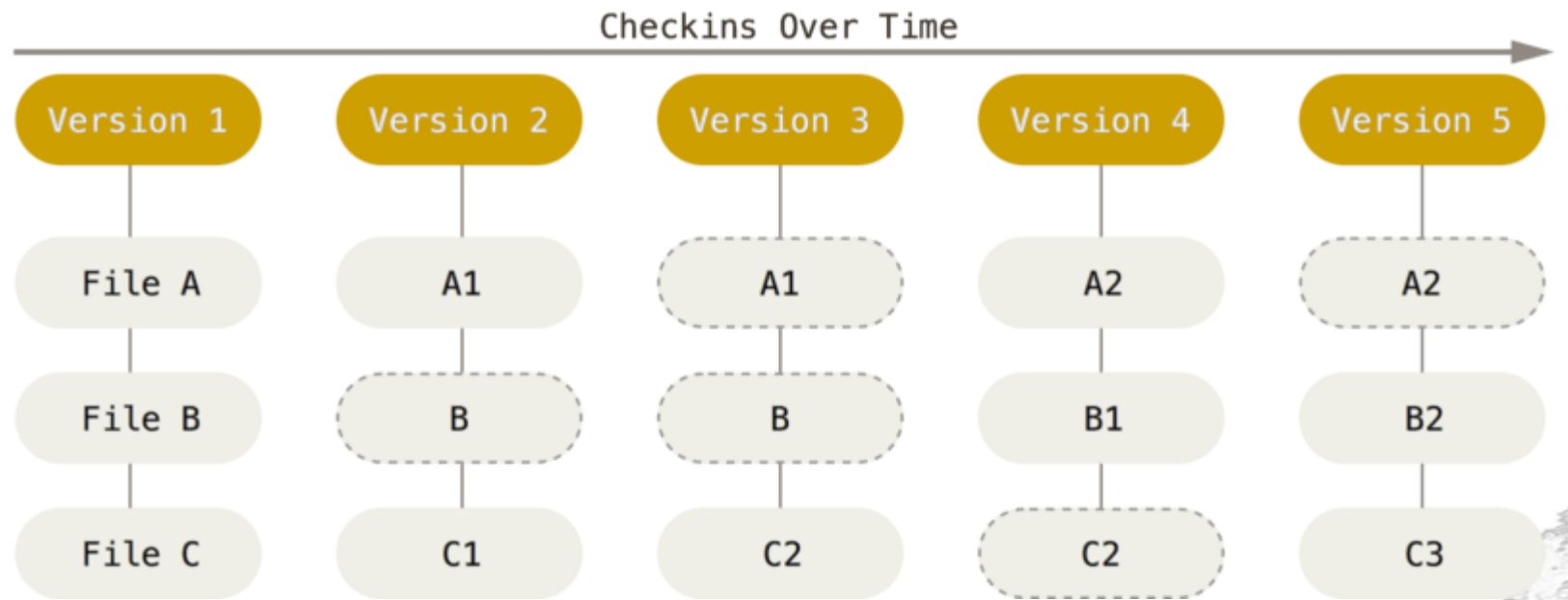
Conceptualmente, la mayoría de los demás sistemas almacenan información como una lista de cambios basados en archivos.

Estos sistemas (CVS, Subversion, Perforce, Bazaar, etc.) piensan en la información que almacenan como un conjunto de archivos y los cambios realizados en cada archivo a lo largo del tiempo (esto se describe comúnmente como control de versiones delta-based).



¿Qué es Git?

Snapshots



En cambio, Git piensa en sus datos más como una serie de fotos de un sistema de archivos en miniatura.

Con Git, cada vez que realiza “`commit`” o guarda el estado de su proyecto, Git básicamente toma una foto de cómo se ven todos sus archivos en ese momento y almacena una referencia a esa foto, es decir, si los archivos no han cambiado, Git no almacena el archivo nuevamente, solo un enlace al archivo idéntico anterior que ya ha almacenado. Git piensa en sus datos más como un flujo de fotos.

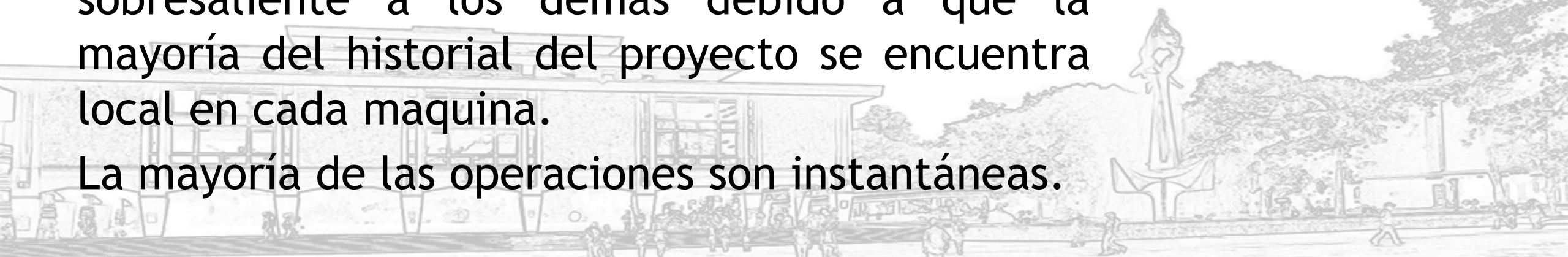
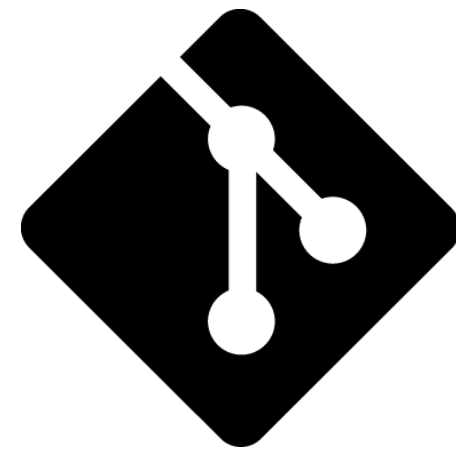
Esto hace que Git sea más como un mini sistema de archivos con algunas herramientas increíblemente poderosas construidas sobre él, en lugar de simplemente un VCS.

Jugando de Local

La gran mayoría de operaciones de Git son locales aplicando recursos locales (requiere un computador de regular en adelante) a diferencia de CVCS (Centralized Version Control Systems) donde la mayoría de las operaciones tienen esa sobrecarga de latencia de red.

La localía de Git le otorga una velocidad sobresaliente a los demás debido a que la mayoría del historial del proyecto se encuentra local en cada maquina.

La mayoría de las operaciones son instantáneas.



Jugando sin Red

Sin Conexión, se estanca la situación hasta que llegues a una conexión de red para cargar los cambios.

En otros sistemas, hacerlo es imposible o doloroso. En Subversion y CVS, puede editar archivos, pero no puede realizar cambios en su base de datos (la base de datos está fuera de línea).

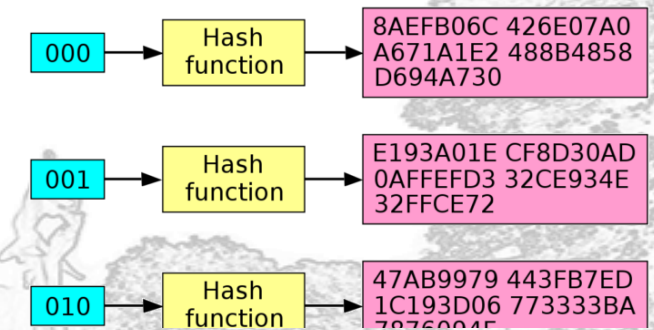
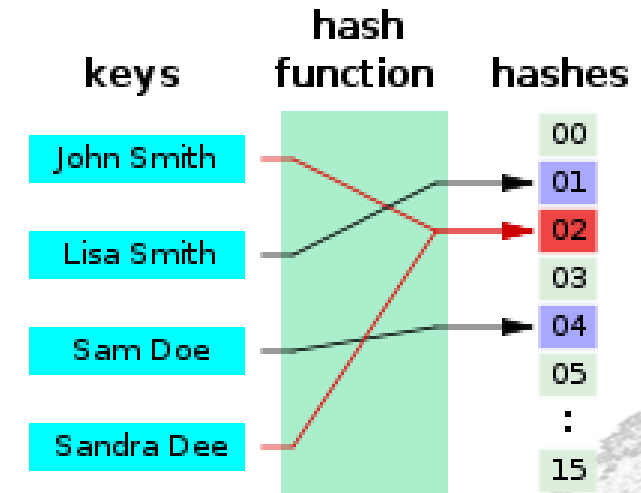


SHA-1

Usado en criptografía, SHA-1 (Secure Hash Algorithm 1) es una función de hash criptográfica que toma una entrada y produce un valor hash de 160 bits (20 bytes) conocido como un resumen de mensaje, que normalmente se representa como un número hexadecimal de 40 dígitos.

Fue diseñado por la Agencia de Seguridad Nacional de los Estados Unidos y es un Estándar Federal de Procesamiento de Información de los Estados Unidos. Se recomienda usar SHA-2 o 3.

Una función hash es cualquier función que se pueda utilizar para asignar datos de tamaño arbitrario a valores de tamaño fijo.



Integridad de Git con SHA-1

Todo en Git se procesa bajo verificación de acumulación y es almacenado con estos criterios.

Esto significa que Git siempre sabrá los cambios realizados gracias a los procesos de acumulación o “Checksum”.

Esto permite que no se pierda o se cambie información sin que Git lo sepa.

Git no usa SHA-1 como algoritmo de cifrado o para seguridad, sino para identificar revisiones y garantizar que los datos no hayan cambiado debido a daños accidentales.

Otros paquetes como el muy conocido XAMPP para desarrollo web usa SHA-1 como validadores de paquetes de instalación.

XAMPP para Windows 7.3.29, 7.4.22 & 8.0.9

Versión	Suma de comprobación	Tamaño
7.3.29 / PHP 7.3.29	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	158 Mb
7.4.22 / PHP 7.4.22	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	159 Mb
8.0.9 / PHP 8.0.9	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	160 Mb

[Requisitos](#) [Complementos](#) [Más Descargas »](#)

Windows XP or 2003 are not supported. You can download a compatible version of XAMPP for these platforms [here](#).

XAMPP para Linux 7.3.29, 7.4.22 & 8.0.9

Versión	Suma de comprobación	Tamaño
7.3.29 / PHP 7.3.29	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	151 Mb
7.4.22 / PHP 7.4.22	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	151 Mb
8.0.9 / PHP 8.0.9	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	154 Mb

[Requisitos](#) [Complementos](#) [Más Descargas »](#)

XAMPP para OS X 7.3.29, 7.4.22, 8.0.9, 7.3.29, 7.4.22 & 8.0.9

Versión	Suma de comprobación	Tamaño
7.3.29 / PHP 7.3.29	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	163 Mb
7.4.22 / PHP 7.4.22	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	163 Mb
8.0.9 / PHP 8.0.9	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	164 Mb
7.3.29 / PHP 7.3.29	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	361 Mb
7.4.22 / PHP 7.4.22	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	360 Mb
8.0.9 / PHP 8.0.9	¿Qué está incluido?. md5 sha1 Descargar (64 bit)	360 Mb

Integridad de Git con SHA-1

Git usa SHA-1 como algoritmo de validación de acumulación o “Checksum”.

SHA-1 es una función que retorna una cadena de 40 caracteres hexadecimales (0-9 y a-f) y calculada con base al contenido de una estructura de directorio o archivo de Git. Git almacena todo con valores Hash del contenido de los archivos.

Por ejemplo el texto “Grupo Ingenieria & Sociedad” retorna el HASH SHA-1

0f29abd045161522750679d24c9fd12c832bf80e

```
1
2 #- coding: utf-8 -*-
3 """
4 Created on Mon Aug 9 14:45:58 2021
5
6 @author: Julian Castillo
7 @script: Implementar SHA-1
8 """
9
10 import hashlib
11
12 texto = "Grupo Ingenieria & Sociedad" #@param {type:"string"}
13
14 # Pasamos a Unicode el texto
15 # Luego se pasa a la funcion SHA1() de la libreria hashlib
16 resultado = hashlib.sha1(texto.encode())
17 # Imprimir resultado solo me retorna la posición de memoria
18 print("Posición en Memoria -> ", resultado)
19 resultado = resultado.hexdigest()
20 # printing the equivalent hexadecimal value.
21 print("El valor hexadecimal de SHA1 es ->", resultado)
22 print("El largo de la cadena Hash es de", len(resultado), "caracteres")
```

Posición en Memoria -> <sha1 HASH object @ 0x7fb46ecb8420>
El valor hexadecimal de SHA1 es -> 0f29abd045161522750679d24c9fd12c832bf80e
El largo de la cadena Hash es de 40 caracteres

texto: " Grupo Ingenieria & Sociedad

How Hashing Works



Gollum's Riddle
(Input)



Hash Function
(Hashing Algorithm)



Hash Value
(Output)

Integridad de Git con SHA-1

Pequeños cambios en una cadena de texto producen un cambio significativo en una cadena HASH SHA-1, Ejemplo:

Cadena de entrada: “Ingenieria y Sociedad”

Hash Salida:

8588b58bd1c2b1eaf09a7b3ff654da34f91d5830

Una Tilde de diferencia

Cadena de entrada: “Ingeniería y Sociedad”

Hash Salida:

862603d2331383690354d232f16fc073ab1d17e7

Ruta Google Colab: <https://colab.research.google.com/drive/1ocRuLEiMXZ6lr9aRNzAZzvf8qEayy3Mr>

Ruta Jupyter Notebook -> Google Colab --> Script Python en GitHub

<https://github.com/juliancastillo-udea/GitHubIntro/blob/main/PublicHashSHA1.ipynb>

<https://github.com/juliancastillo-udea/GitHubIntro/blob/main/publichashsha1.py>

```
1 cadena1 = '8588b58bd1c2b1eaf09a7b3ff654da34f91d5830' # --> "Ingenieria y Sociedad"
2 cadena2 = '862603d2331383690354d232f16fc073ab1d17e7' # --> "Ingeniería y Sociedad"
3 #Comparemos las cadenas una a una luego de ver su tipo de datos
4 print(type(cadena1))
5 print(type(cadena2))
6 igualdadposicional = 0
7 for i in range(len(cadena1)):
8     if cadena1[i] == cadena2[i]:
9         igualdadposicional +=1
10 print('Total de posiciones iguales: {} de un total de 40'.format(igualdadposicional))
11 print('\tEsto equivale a un {:.2f}%'.format(igualdadposicional/len(cadena1)*100))

<class 'str'>
<class 'str'>
Total de posiciones iguales: 5 de un total de 40
Esto equivale a un 12.50%
```

Git solo agrega información

Cuando realiza acciones en Git, casi todas solo agregan datos a la base de datos de Git.

Es difícil lograr que el sistema haga algo que no se pueda deshacer o que borre datos de alguna manera.

Al igual que con cualquier VCS, puede perder o estropear los cambios que aún no ha realizado, pero después de enviar una foto de los documentos o código a Git, es muy difícil de perder, especialmente si envía regularmente su base de datos a otro repositorio.

Esto permite experimentar sin peligro de perder información.



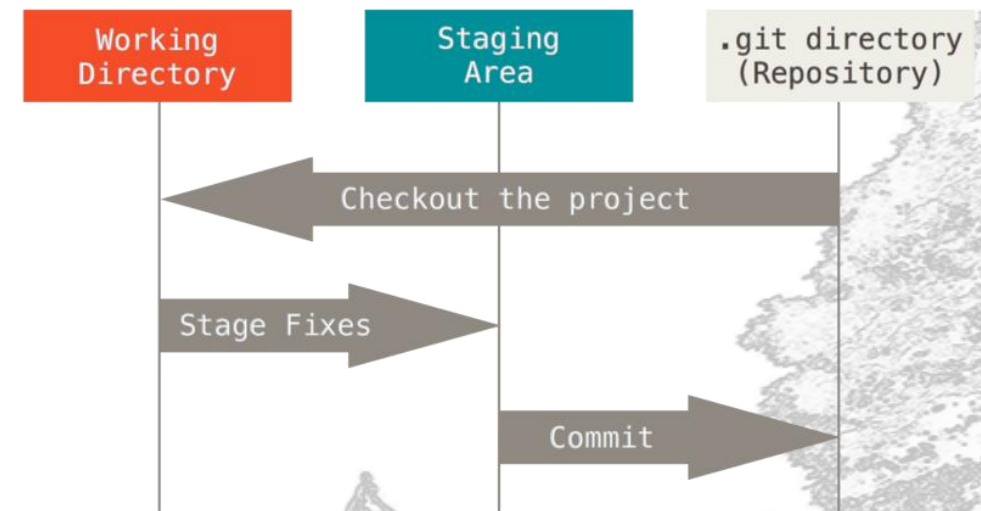
Los tres estados de Git

Git tiene tres estados principales en los que pueden existir sus archivos: modified, staged y committed:

Modified significa que ha cambiado el archivo pero aún no lo ha enviado a su base de datos.

Staged significa que ha marcado un archivo modificado en su versión actual para ir a su próxima carga de confirmación.

Committed significa que los datos se almacenan de forma segura en su base de datos local.



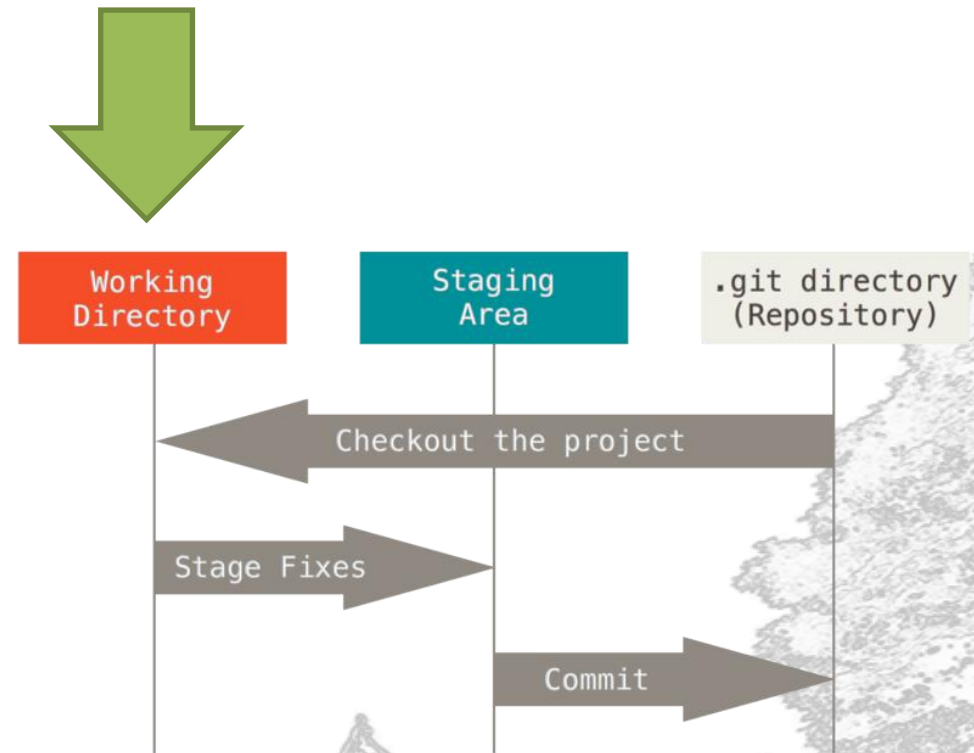
Los tres estados de Git

Working
Directory

Working Directory

El árbol de trabajo es un proceso de salida de archivos único de una versión del proyecto.

Estos archivos se extraen de la base de datos comprimida en el directorio Git y se colocan en el disco para que los use o modifique.



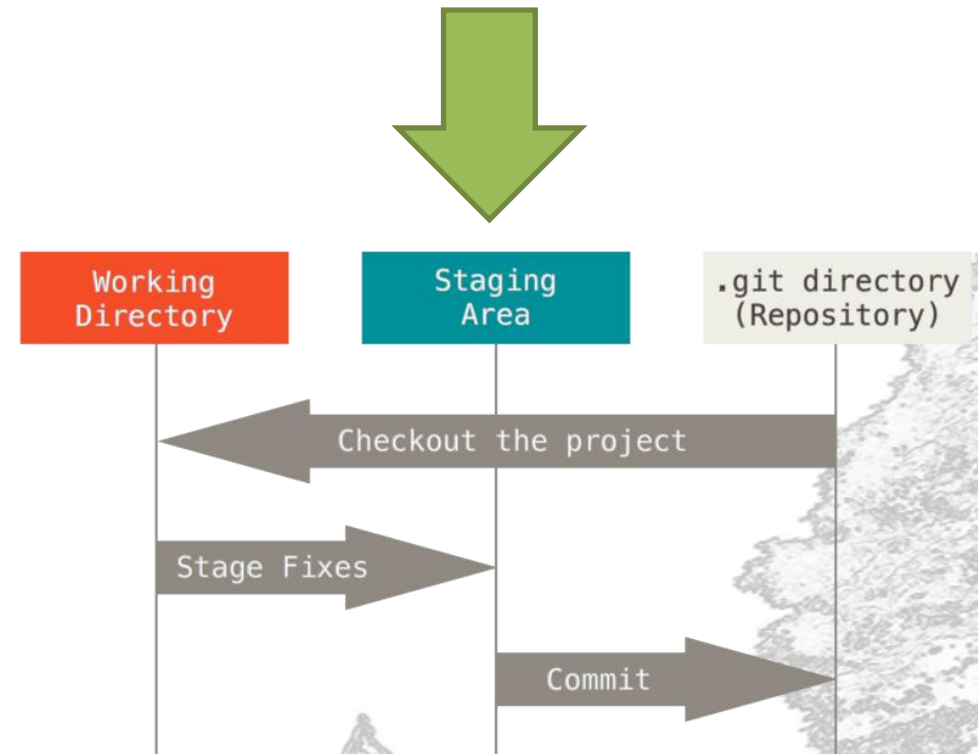
Los tres estados de Git

Staging
Area

Staging Area

El área de preparación es un archivo en su directorio de Git local que almacena información sobre lo que se incluirá en su próximo Commit (confirmación).

Su nombre técnico en el lenguaje de Git es “index” (índice), pero la frase “área de preparación” es más común en los entornos de desarrollo.



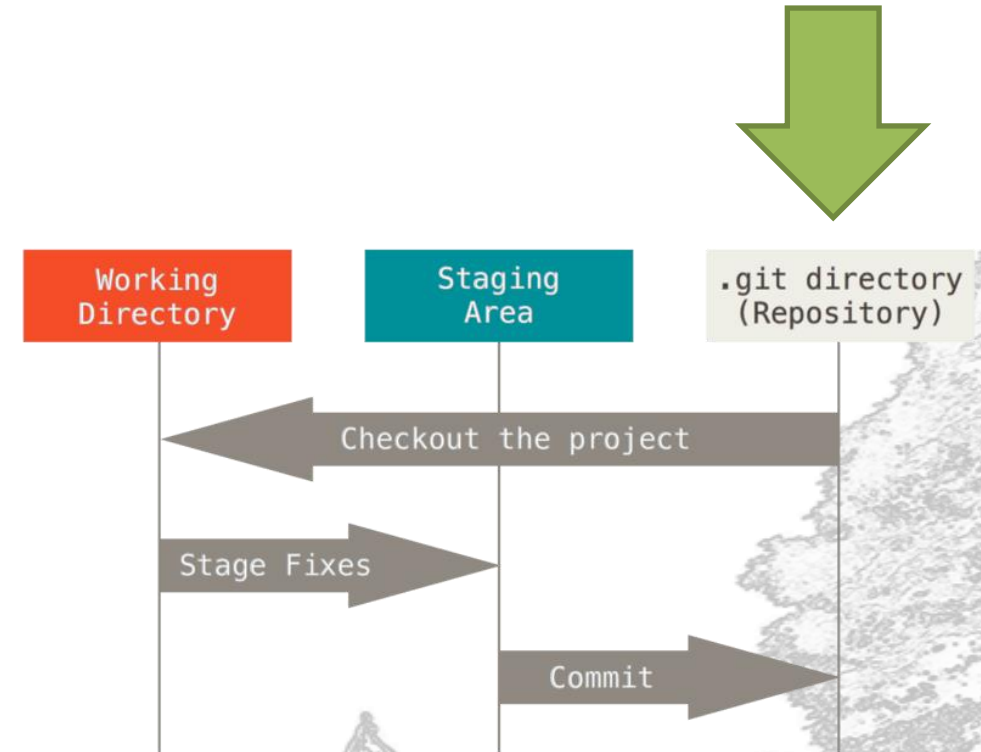
Los tres estados de Git

`.git directory
(Repository)`

`.git directory (Repository)`

El directorio de Git es donde se almacenan los metadatos y la base de datos de objetos para su proyecto (usando los Hash).

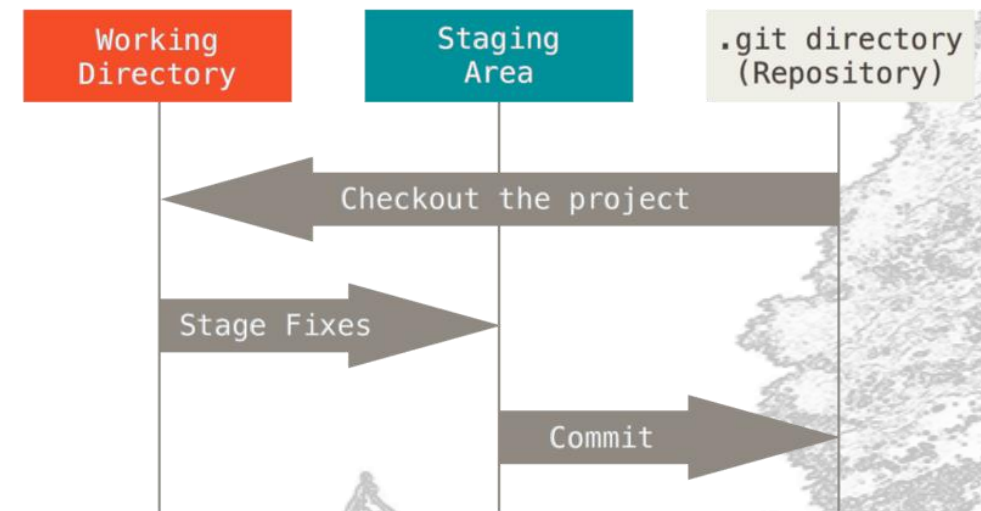
Esta es la parte más importante de Git, y es lo que se copia cuando clona un repositorio desde otra computadora.



Flujo de trabajo de Git

El flujo de trabajo de Git se representa de la siguiente manera:

1. Modifica archivos en su árbol de trabajo.
2. Organiza selectivamente solo aquellos cambios que desea que formen parte de su próximo **commit**, que agrega solo esos cambios al staging area.
3. Realiza un commit, que toma los archivos tal como están en el área de preparación y almacena esa copia instantánea de forma permanente en su directorio de Git.



Branching and Merging

La característica de Git que realmente lo distingue de casi todos los demás SCM (source-control management) es su modelo de ramificación (branching).

Git permite tener múltiples ramificaciones (branches) locales que pueden ser completamente independientes entre sí.

La creación (creation), fusión (merging) y eliminación (deletion) de esas líneas de desarrollo lleva solo segundos.



Branching and Merging

Branching and Merging permite:

Frictionless Context Switching: Cree una rama, experimente y acepte o rechace.

Role-Based Codelines: Ramas con lineamientos específicos, por ejemplo una rama orientada a producción.

Feature Based Workflow: Ramas para probar nuevas funcionalidades y eliminarlas cuando se fusionen con la rama principal.

Disposable Experimentation: Ramas para realizar pruebas desechables.



Branching and Merging

Branching and Merging permite:

En particular, no se tiene que mover o fusionar todas sus ramas. Puede optar por compartir solo una, algunas de ellas o todas.

Esto tiende a liberar a las personas para que prueben nuevas ideas sin preocuparse por tener que planificar cómo y cuándo van a fusionarlas o compartirlas con otros.

Git hace que este proceso sea increíblemente fácil y cambia la forma en que la mayoría de los desarrolladores trabajan cuando lo aprenden.

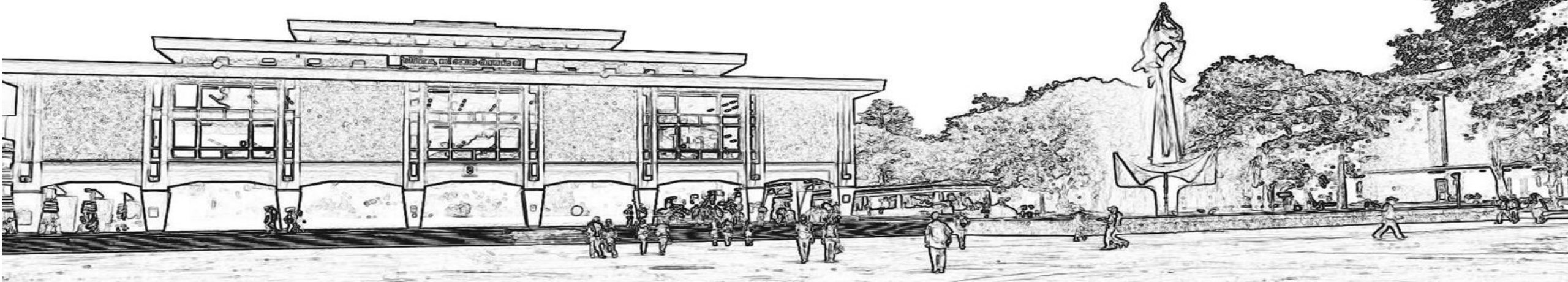




UNIVERSIDAD DE ANTIOQUIA

Ingeniería & *Sociedad*

Facultad de Ingeniería



3. Instalación de Git

Dos Formas de usar Git

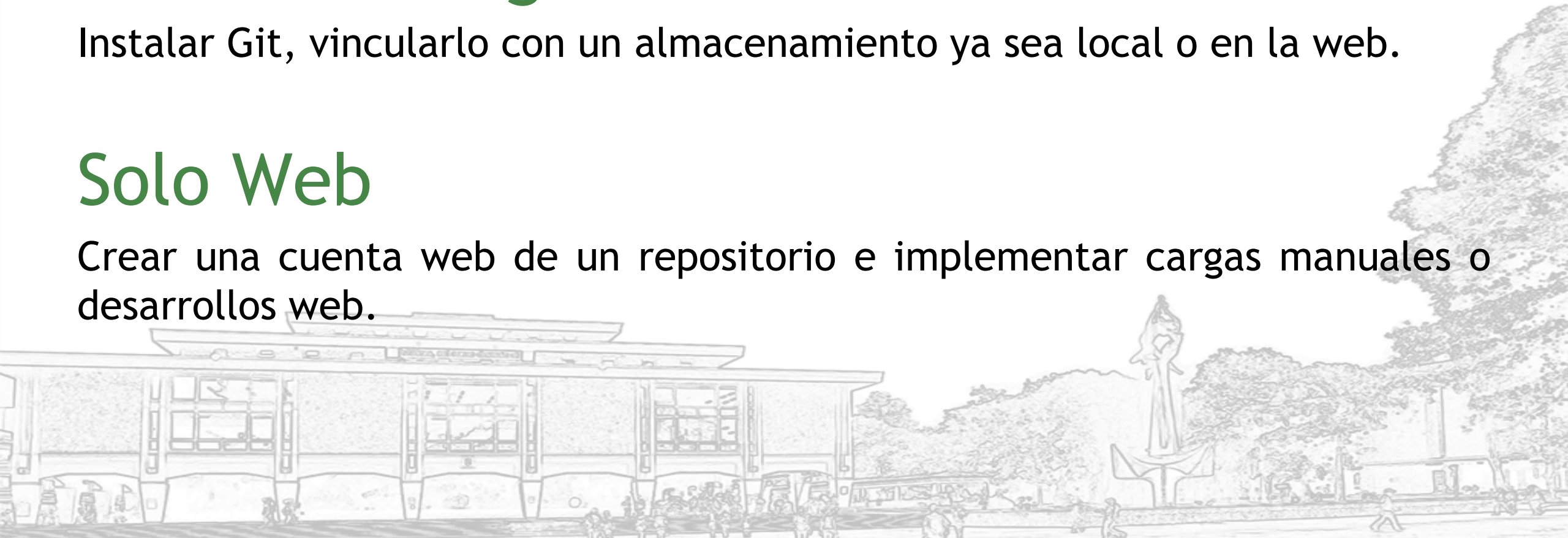
Se pueden implementar principalmente dos formas de usar Git.

Local con carga web

Instalar Git, vincularlo con un almacenamiento ya sea local o en la web.

Solo Web

Crear una cuenta web de un repositorio e implementar cargas manuales o desarrollos web.



Local con carga web

Para realizar la instalación de Git accedemos a la ruta:

<https://git-scm.com/>

The image shows a screenshot of the Git website (git-scm.com) with a focus on the Linux download path. A blue arrow points from the 'Linux' text to the 'Download for Linux' button on the 'Latest source Release' section. The website header includes the Git logo and the tagline '--everything-is-local'. The main content area describes Git as a free and open source distributed version control system. The footer contains links for 'About', 'Documentation', and 'Downloads'. The 'Downloads' section highlights the latest source release (2.32.0) and provides a button to download for Linux.

git --everything-is-local

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference page
Git book content, videos

Downloads
GUI clients and binary releases for all major platforms.

Latest source Release
2.32.0
Release Notes (2021-06-06)
[Download for Linux](#)

Linux

La Web Git identifica el sistema operativo y sugiere la descarga para Windows en el presente caso.

Descarga Git

La descarga inicia automáticamente, en caso contrario clic en: [Click here to download manually,](#)

The image is a composite of three screenshots illustrating the Git download process on Windows. The top-right screenshot shows the 'Downloading Git' page on the Git website, with a large blue arrow pointing to the 'Click here to download manually,' link. The bottom-left screenshot shows a 'Save As' dialog box with the file name 'Git-2.32.0.2-64-bit.exe' and the save type 'Application (*.exe)', with a blue arrow pointing to the 'Save' button. The bottom-right screenshot shows a Windows notification area with a notification for 'Git-2.32.0.2-64-bit.exe' and a link to 'Show in folder'.

Downloading Git

Your download is starting...

You are downloading the latest (**2.32.0**) **64-bit** version of **Git for Windows**. This is the most recent **maintained build**. It was released **about 1 month ago**, on 2021-07-06.

[Click here to download manually](#), if your download hasn't started.

Other Git for Windows downloads

- Git for Windows Setup
- 32-bit Git for Windows Setup.**
- 64-bit Git for Windows Setup.**
- Git for Windows Portable ("thumbdrive edition")
- 32-bit Git for Windows Portable.**
- 64-bit Git for Windows Portable.**

The current source code release is version **2.32.0**. If you want the newer version, you can build it from [the source code](#).

Save As

File name: **Git-2.32.0.2-64-bit.exe**

Save as type: **Application (*.exe)**

Save **Cancel**


Today

Git-2.32.0.2-64-bit.exe

<https://github-releases.githubusercontent.com/23216272/d1c7a880-deb2-11eb-...>

[Show in folder](#)

Instalación Git

Name	Type	Size
 Git-2.32.0.2-64-bit.exe	Application	48,790 KB

Doble clic en el archivo descargado e inicia el proceso de instalación con el cual debes seguir los siguientes pasos:

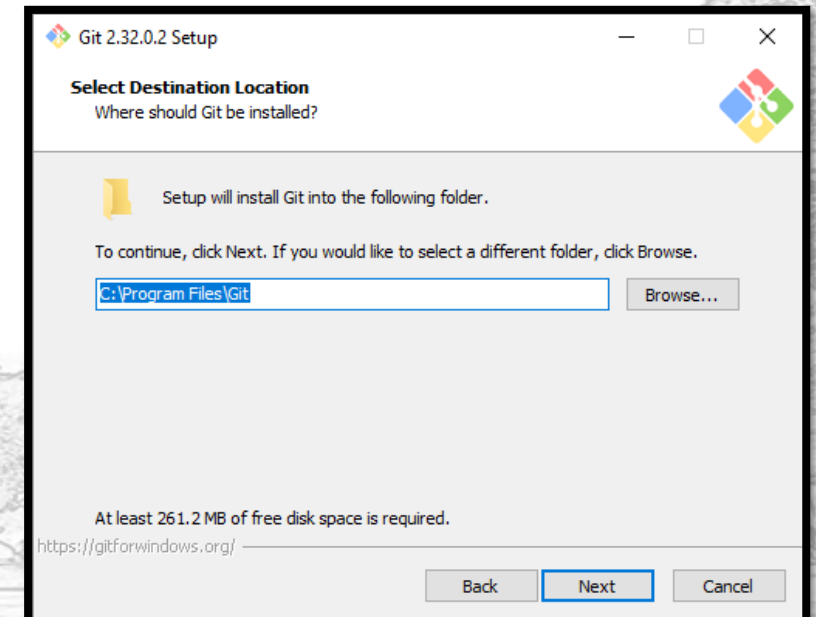
1. Permitir la ejecución del instalador, esto se debe a que el programa Git debe tener permisos para administrar almacenamiento. Clic en si (Yes)



Instalación Git

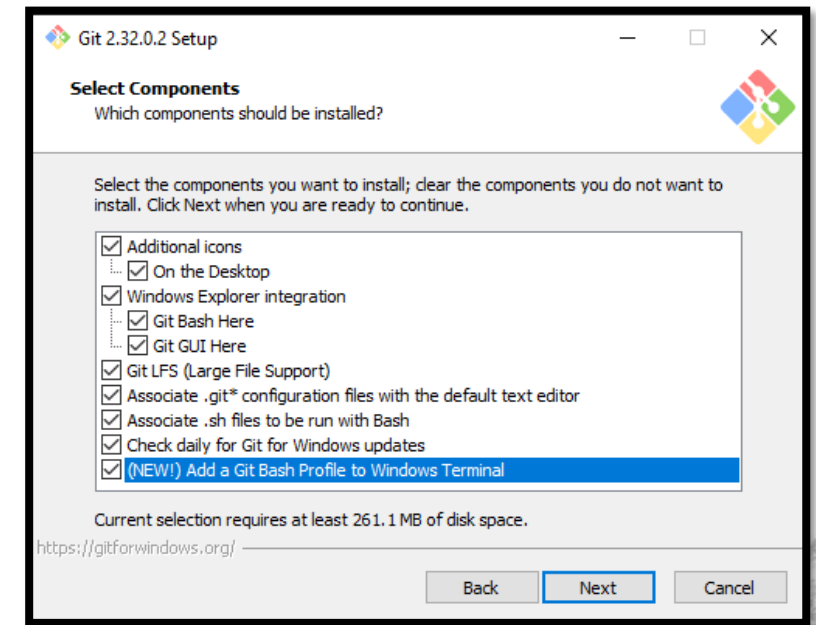
2. Recuerde leer la licencia. Clic en siguiente (Next)

3. Seleccione la ruta de instalación si desea cambiarla, de lo contrario conserve la ubicación sugerida. Clic en siguiente (Next)

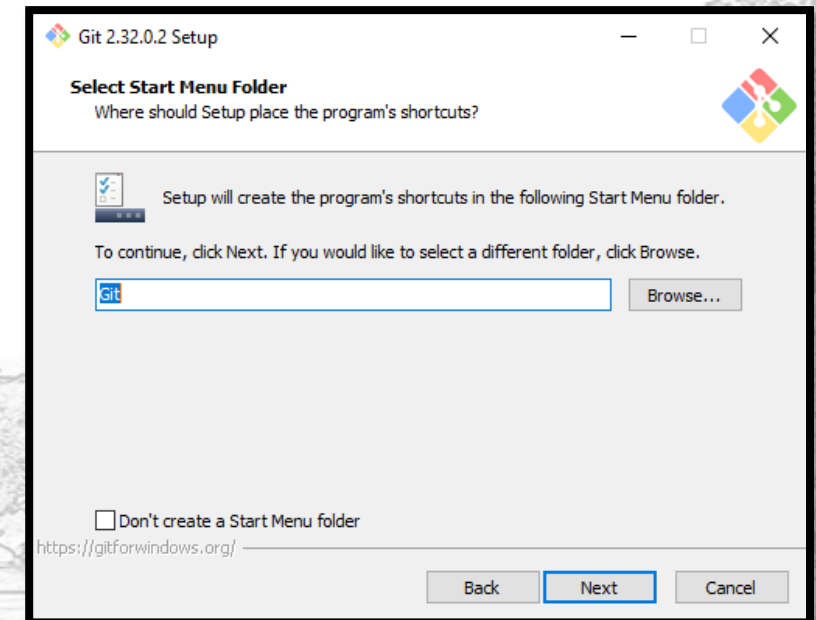


Instalación Git

4. Se recomienda seleccionar todas las opciones de instalación. Clic en siguiente (Next)

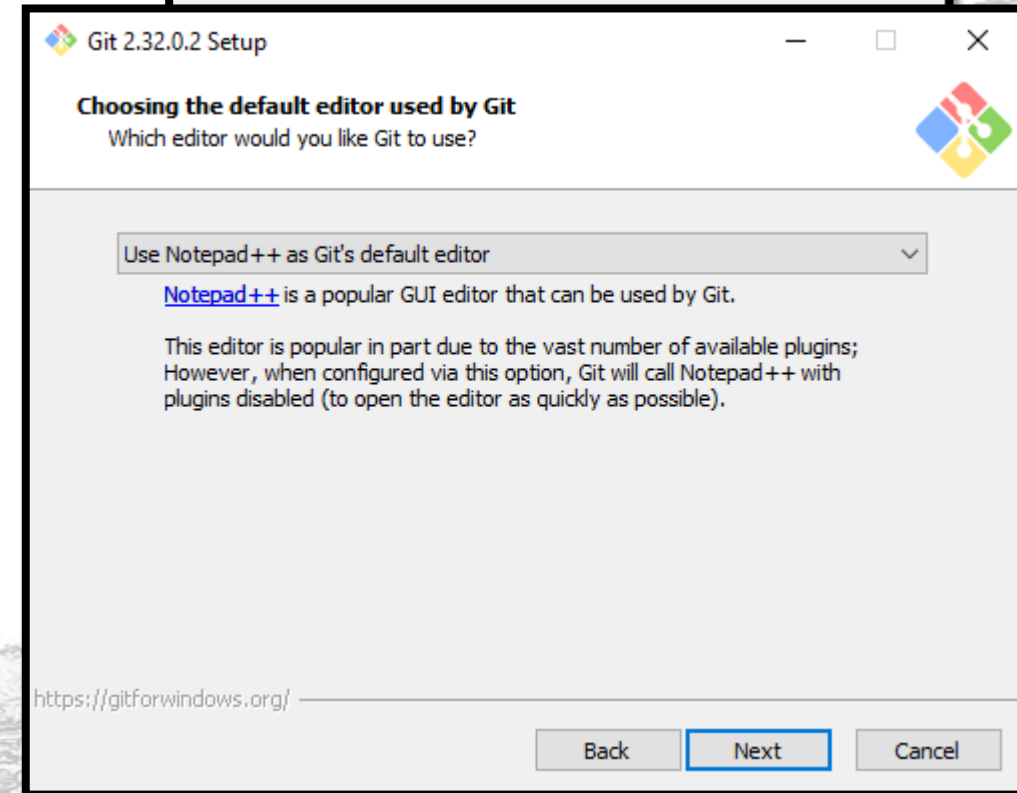
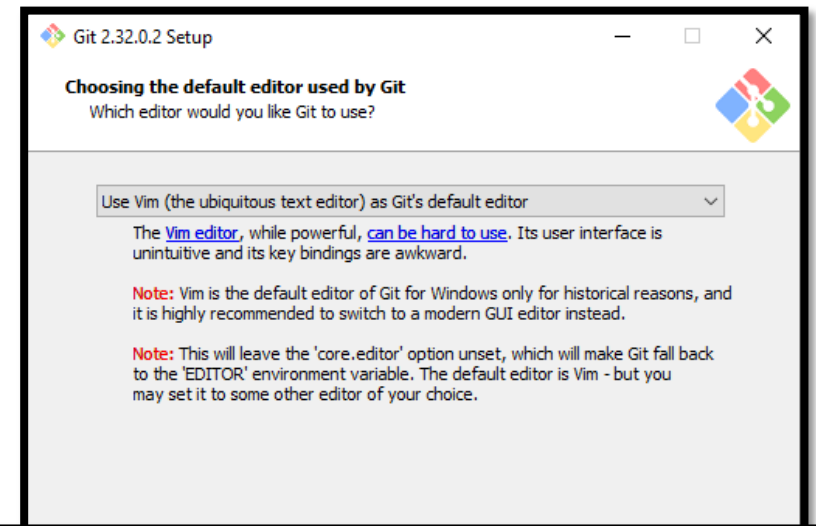


5. Conserve el texto del menú de instalación. Clic en siguiente (Next)



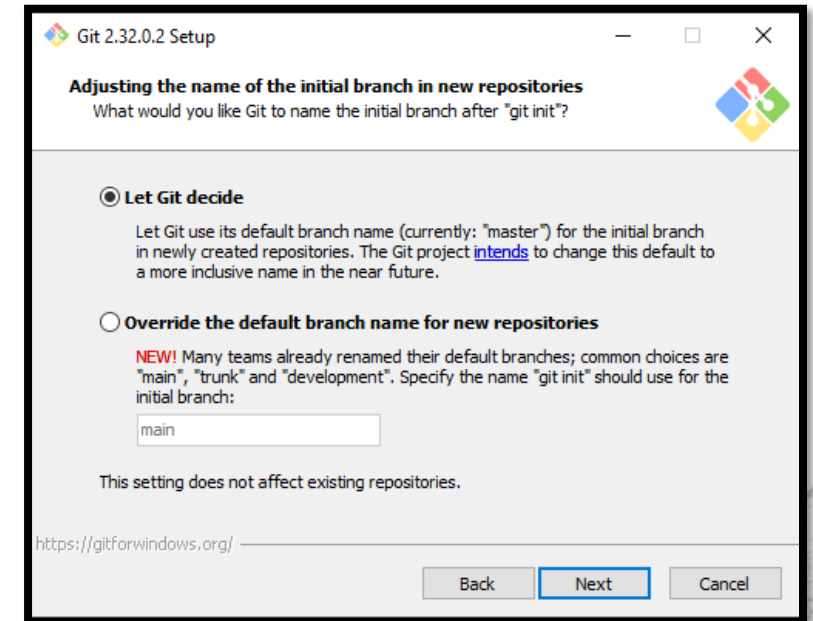
Instalación Git

6. El editor por defecto de Git es Vim, un editor complejo, por facilidad se recomienda [Notepad++](#), selecciona Notepad++ (luego de instalarlo). Clic en siguiente (Next)



Instalación Git

7. Por criterios de compatibilidad se recomienda conservar la opción de permitir que Git decida la nomenclatura de las ramas. Clic en siguiente (Next)



Desde su inicio, el nombre de rama principal por Git se estableció en master. Cada repositorio de Git tiene una rama master y esta juega un papel integral en el mundo del desarrollo de software y hardware.

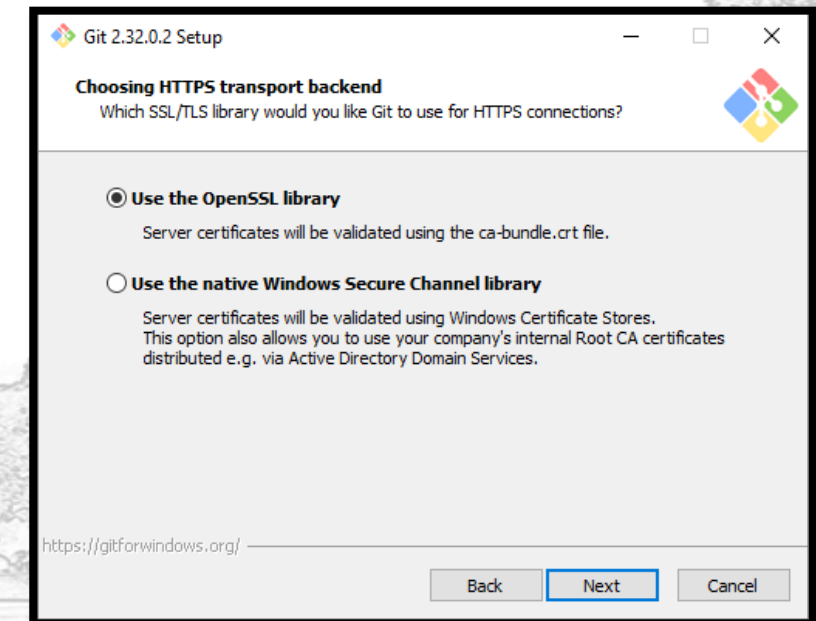
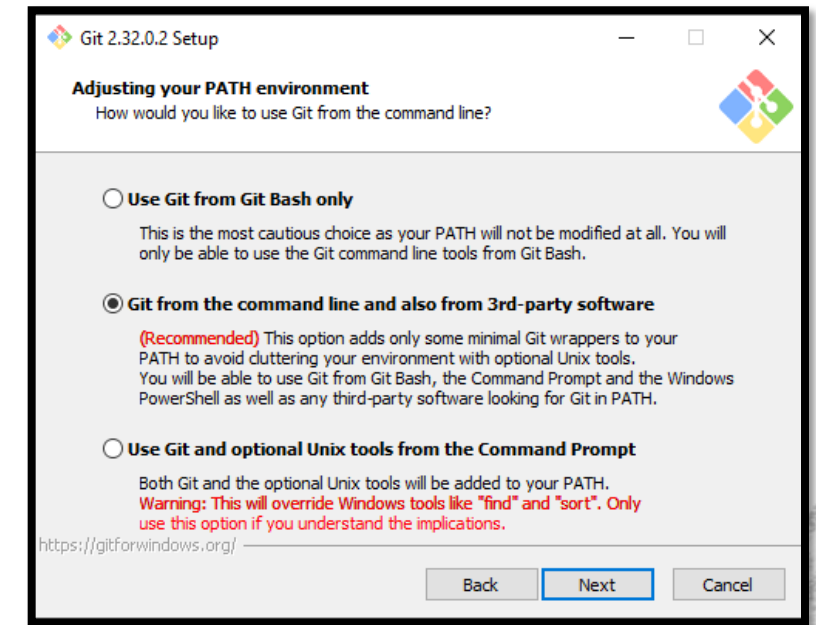
El uso de los términos master y slave por parte de la industria de la computación llamó la atención de todos en el verano de 2020. En medio de las muchas protestas y el creciente malestar social, estos términos dañinos y anticuados ya no se consideraban apropiados.

"Tanto Conservancy como el proyecto Git son conscientes de que el nombre de la rama inicial, 'master', es ofensivo para algunas personas y simpatizamos con los afectados por el uso de ese término", dijo Software Freedom Conservancy. Nota: El cambio no es retroactivo.

Instalación Git

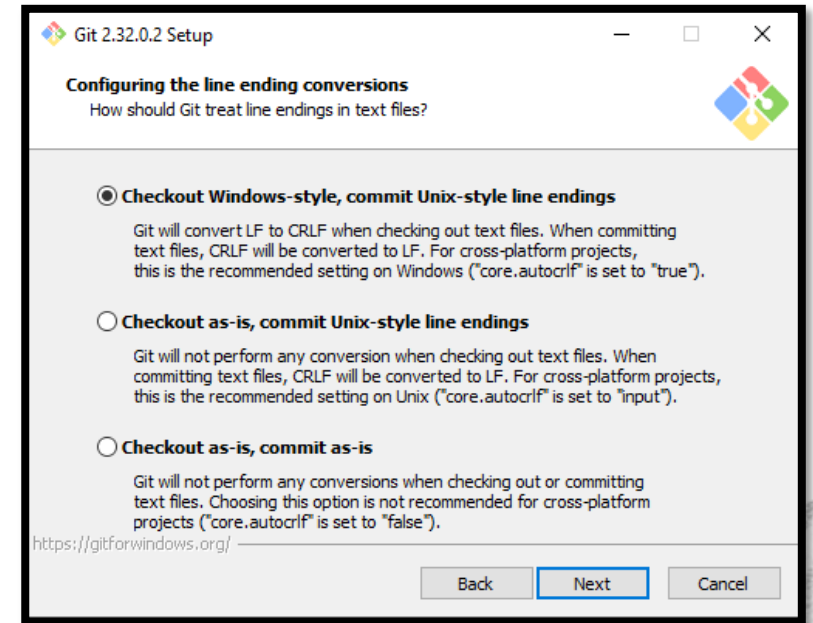
8. La ruta y administración de variables de entorno en Windows se recomienda utilizar en ambos casos, para la línea de comandos de Git y para terceros. Clic en siguiente (Next)

9. Se recomienda el uso de la librería OpenSSL para el transporte de datos por la red. Clic en siguiente (Next)

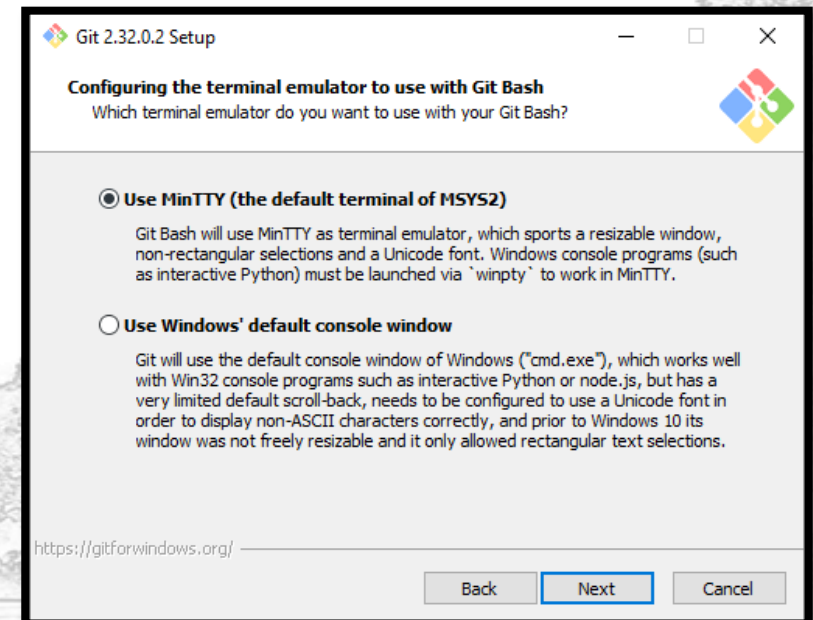


Instalación Git

10. Se recomienda a Git conservar el estilo de Windows y Unix para el formato de terminación de líneas en el código (conocido como EOL y EOF). Clic en siguiente (Next)

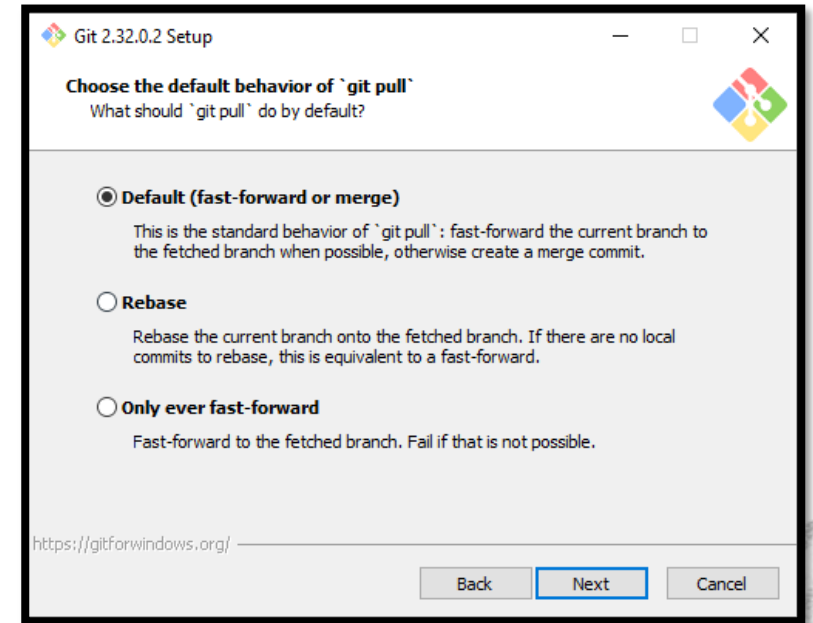


11. Se recomienda el uso de la terminal emulada predeterminada por defecto y no usar la consola (CMD) de Windows. Clic en siguiente (Next)

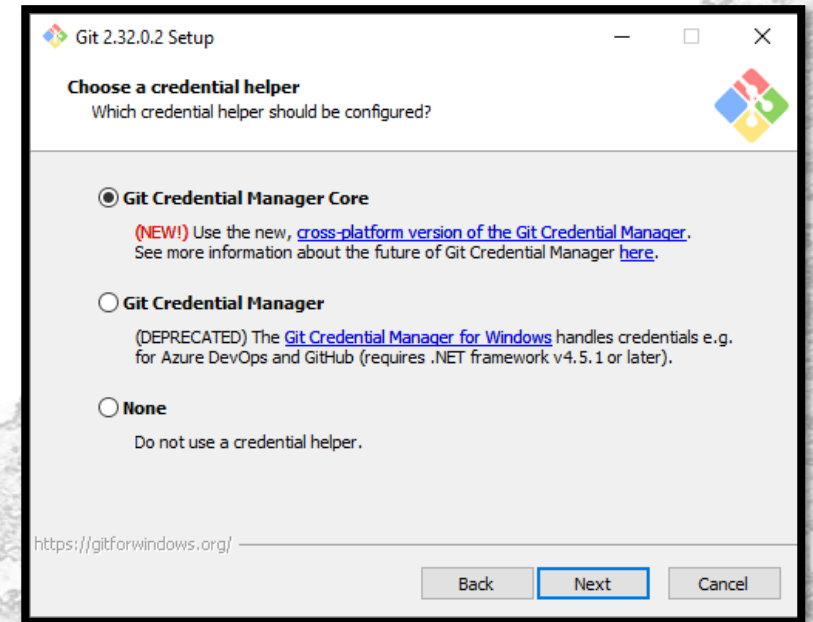


Instalación Git

12. Se recomienda el uso por defecto para el 'git pull' en cuanto a la unificación de ramas. Clic en siguiente (Next)



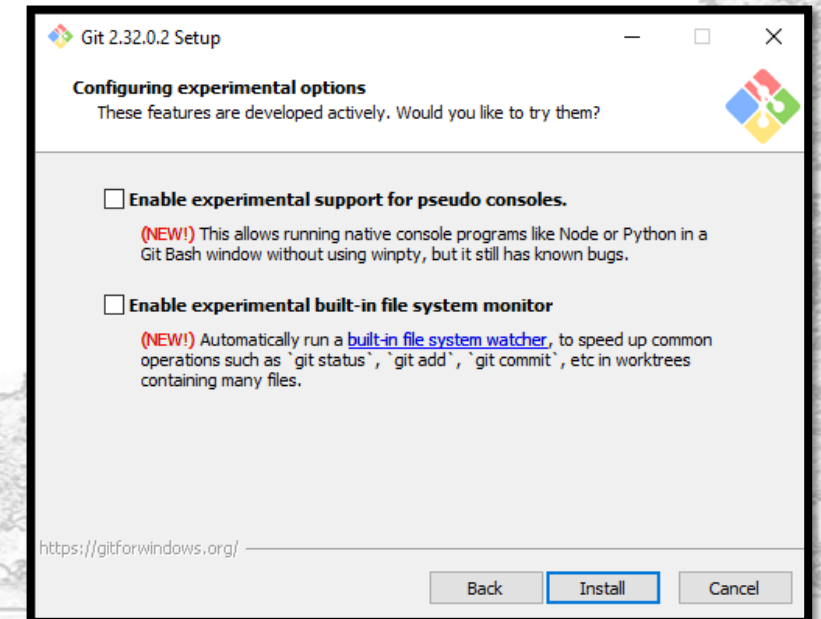
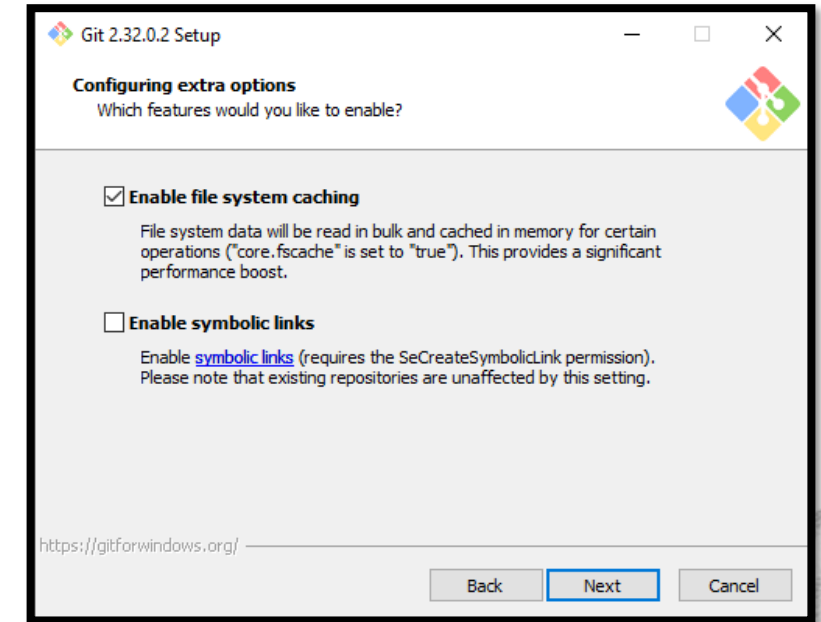
13. Se recomienda el uso de la administración de credenciales por defecto. Clic en siguiente (Next)



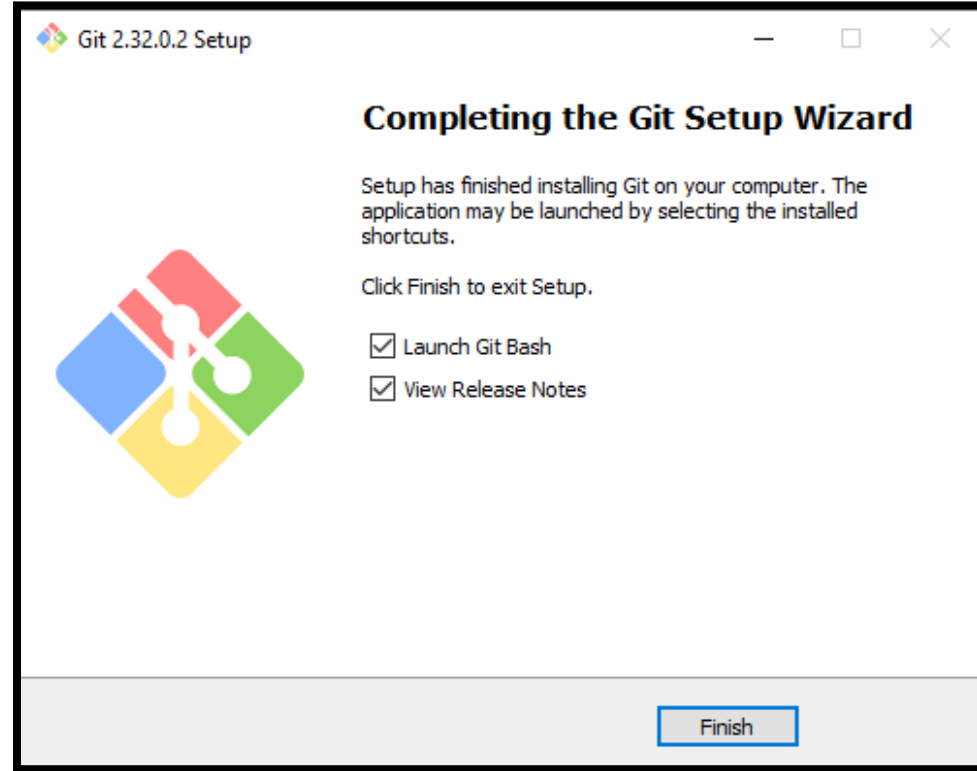
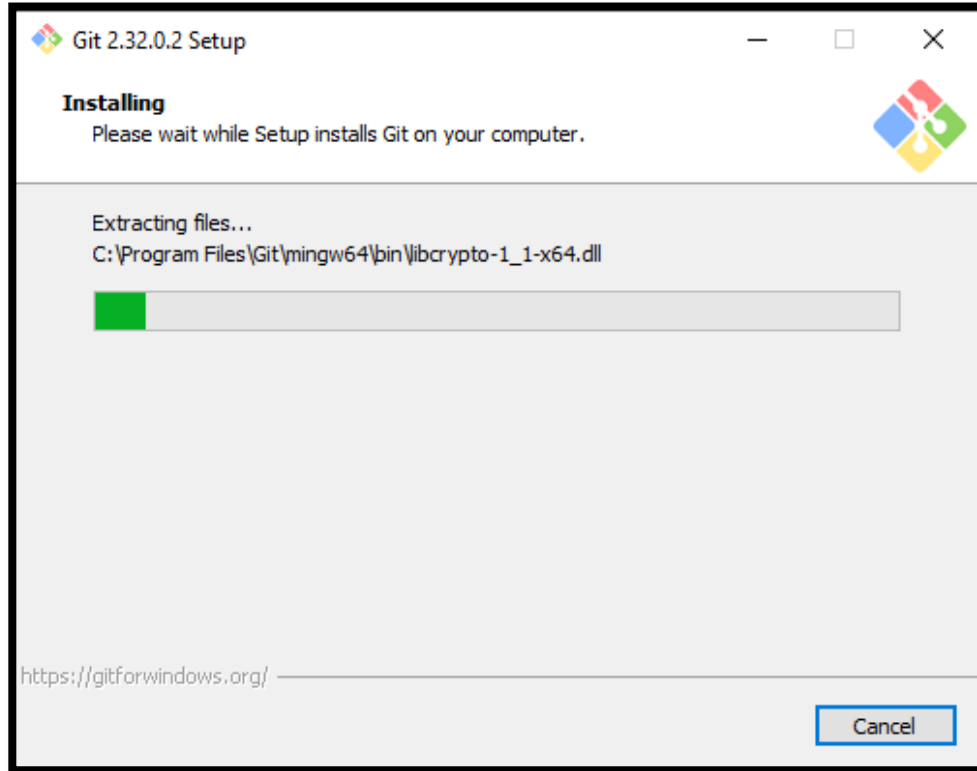
Instalación Git

14. Se recomienda el uso en cache de datos masivos. Clic en siguiente (Next)

15. No se recomienda el uso de componentes experimentales. Clic en Instalar (Install)



Instalación Git

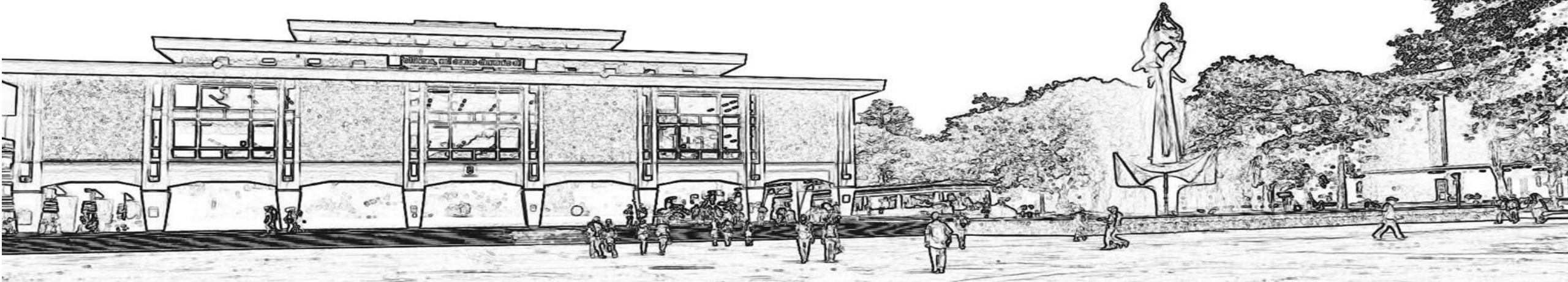




UNIVERSIDAD DE ANTIOQUIA

Ingeniería & Sociedad

Facultad de Ingeniería



4. Cuenta en GitHub

¿Qué es GitHub? - Compartir Código

Es posible que hayas escuchado de GitHub en una cafetería, en un pasillo hablando con colegas Ingenieros de Sistemas o con amigos desarrolladores de software o los llamadores “coders”.

GitHub se puede relacionar con un servicio de publicación y uso compartido de código, o con una red social para programadores para compartir código o librerías.



¿Qué es GitHub? - Núcleo Git + Goodies

GitHub es un servicio de alojamiento de repositorios Git, pero agrega muchas de sus propias características.

Si bien Git es una herramienta de línea de comandos, GitHub proporciona una interfaz gráfica basada en la web.

También proporciona control de acceso y varias funciones de colaboración, como wikis y herramientas básicas de gestión de tareas para cada proyecto.



¿Qué es GitHub? - “forking”

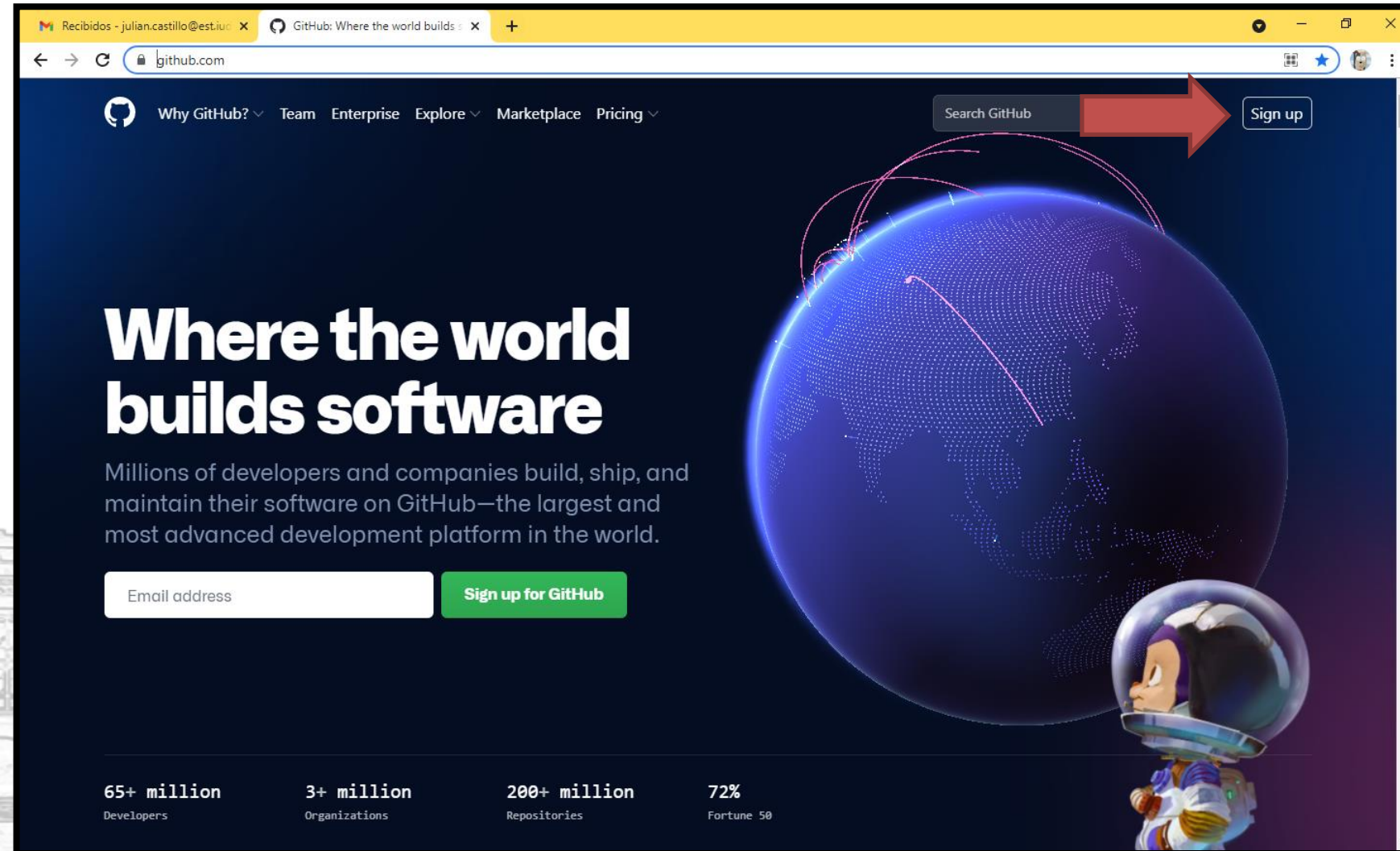
La funcionalidad insignia de GitHub es “forking”: copiar un repositorio de la cuenta de un usuario a otro. Esto le permite tomar un proyecto para el que no tiene acceso de escritura y modificarlo en su propia cuenta.

Si realiza cambios que le gustaría compartir, puede enviar una notificación llamada “pull request” al propietario original. Luego, ese usuario puede, con un clic, fusionar los cambios encontrados en su repositorio con el repositorio original.



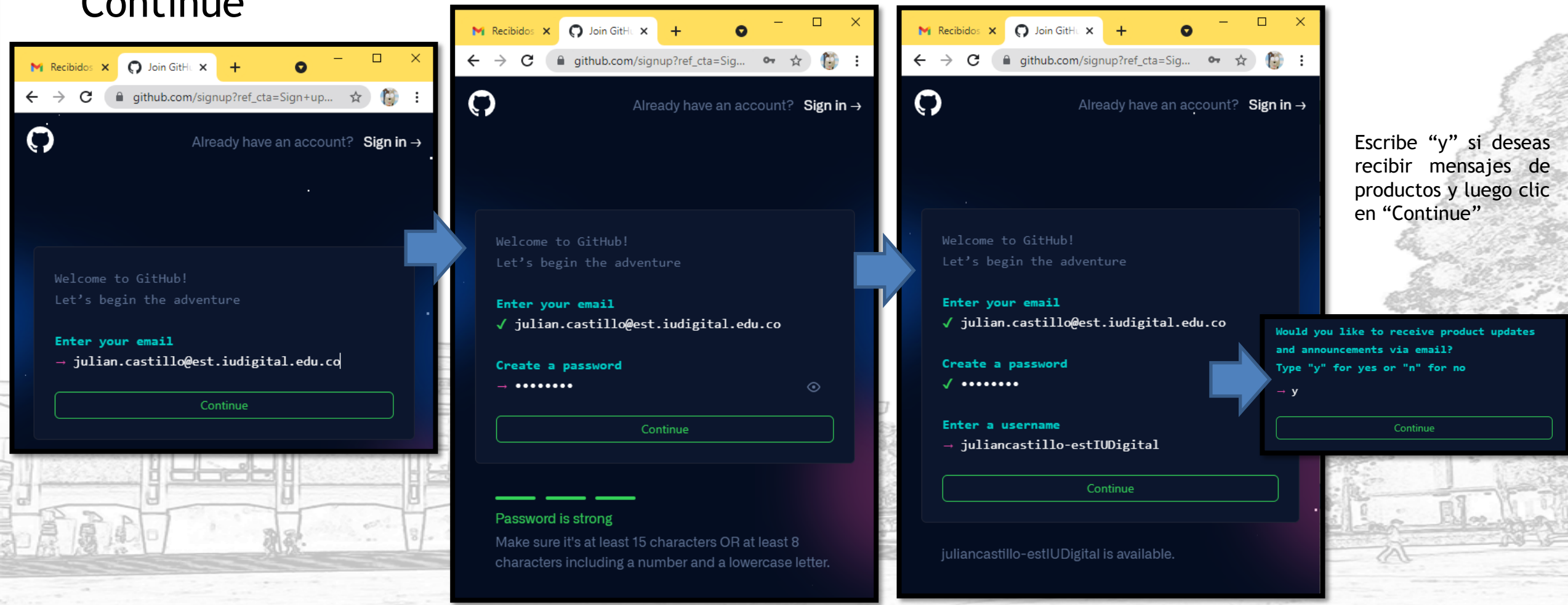
Crear una cuenta en GitHub

Accedemos a <https://github.com/> Clic en “Sign Up” (Suscribirse)



Crear una cuenta en GitHub

Crearemos una cuenta con un correo estudiantil (presente ejemplo), ingresamos el correo y clic en “Continue” y escribimos una contraseña y clic en “Continue”, luego seleccionamos un usuario no registrado y clic en “Continue”



The image shows three sequential screenshots of the GitHub sign-up process, connected by blue arrows indicating the flow.

Screenshot 1: The GitHub sign-up page. The user has entered the email `julian.castillo@est.iudigital.edu.co` in the "Enter your email" field. The "Continue" button is highlighted.

Screenshot 2: The user has entered a password in the "Create a password" field. The password is masked with dots. The "Continue" button is highlighted. Below the password field, a message states: "Password is strong. Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter."

Screenshot 3: The user has entered the username `juliancastillo-estIUDigital` in the "Enter a username" field. The "Continue" button is highlighted. Below the username field, a message states: "juliancastillo-estIUDigital is available."

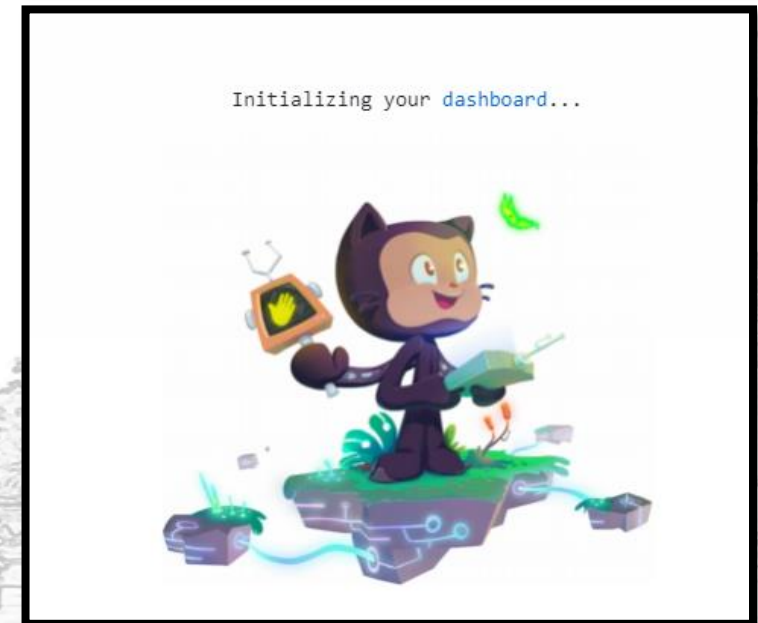
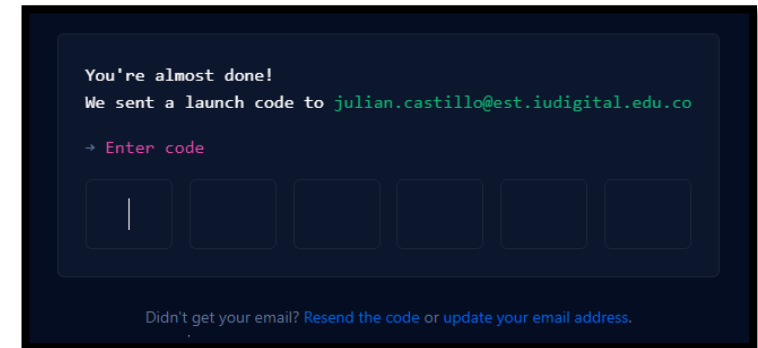
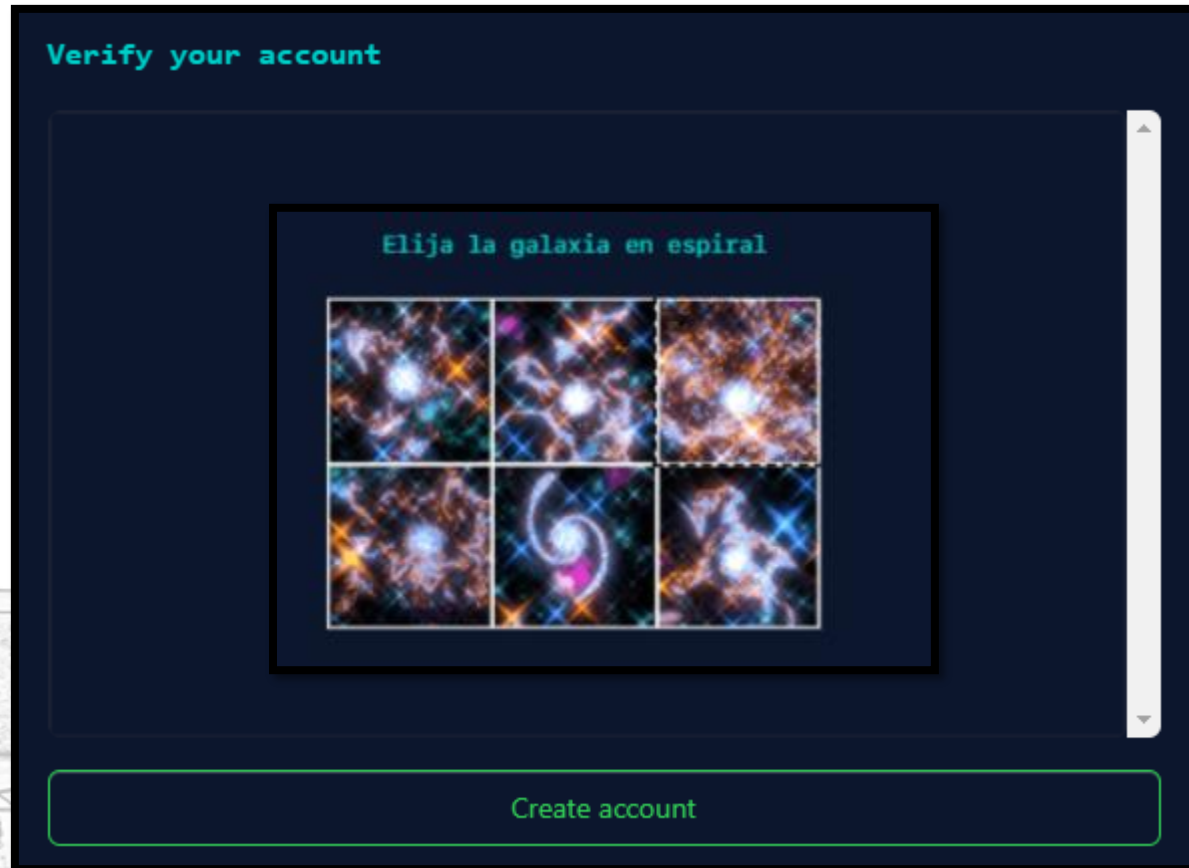
Additional Information:

- At the top of each screenshot, there is a link: "Already have an account? Sign in →".
- On the right side of the third screenshot, there is a text box: "Escribe 'y' si deseas recibir mensajes de productos y luego clic en 'Continue'".
- Below the text box, there is a small form asking: "Would you like to receive product updates and announcements via email? Type 'y' for yes or 'n' for no". The user has entered "y".

Crear una cuenta en GitHub

Verifica la cuenta y clic en “Create account”.

Escribe el código enviado al correo para validar el e-mail.

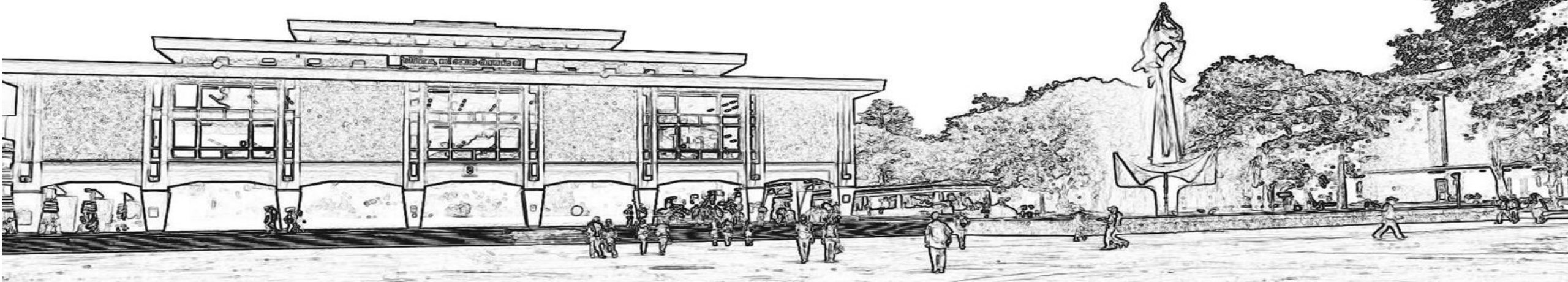




UNIVERSIDAD DE ANTIOQUIA

Ingeniería & *Sociedad*

Facultad de Ingeniería



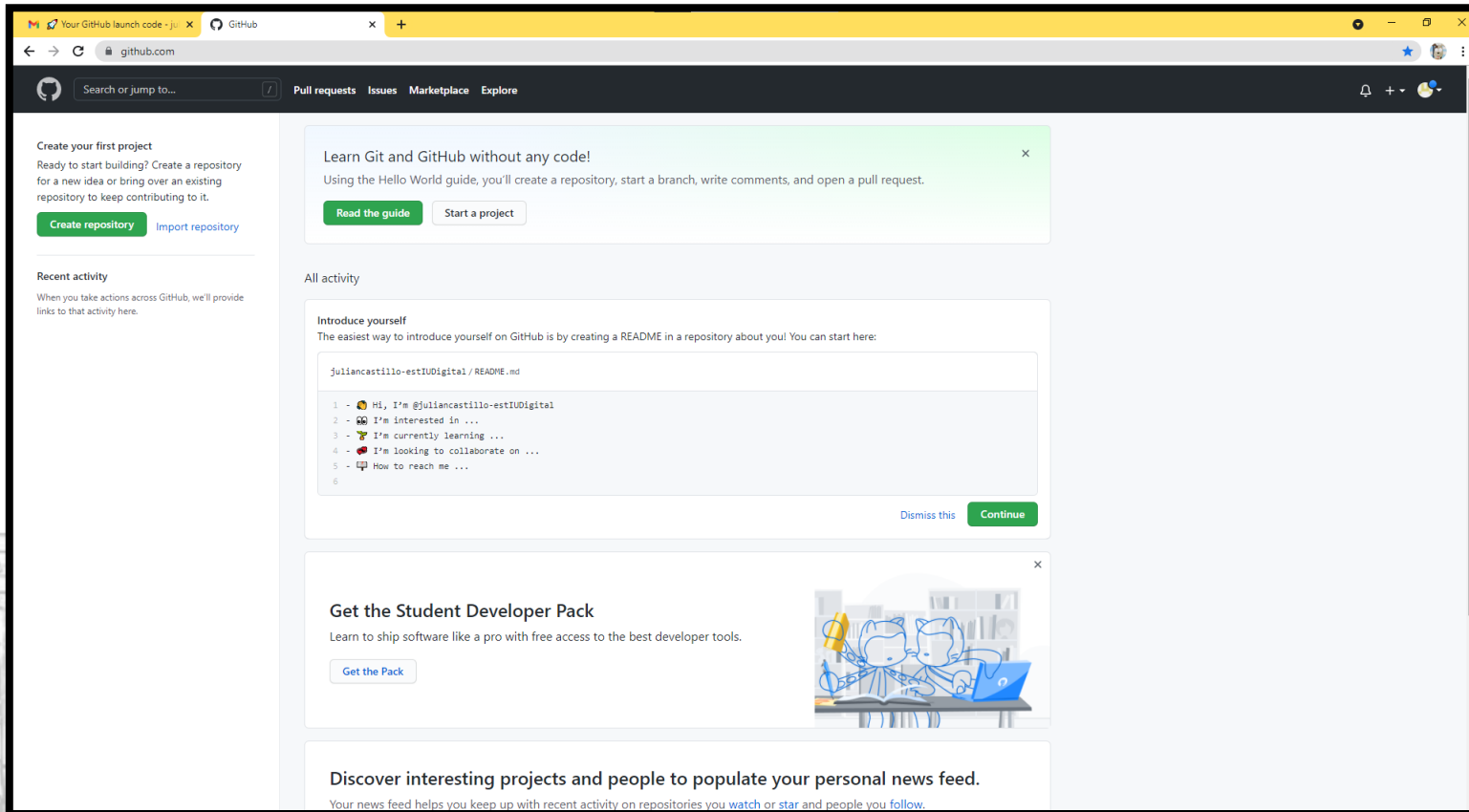
5. Crear un repositorio

Crear un repositorio en GitHub

Ahora en nuestro tablero de control (dashboard) procedemos a crear un nuevo repositorio.

Clic en “Create Repository”

Create repository



Crear un repositorio en GitHub


Seleccionamos el nombre del repositorio y una descripción.

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 *

 juliancastillo-estIUDigital ▾

/


IntroduccionGit ✓


Great repository names are short and memorable. Need inspiration? How about [legendary-memory](#)?

Description (optional)

Repo para describir Git en Ingeniería & Sociedad

Seleccionamos el tipo de repositorio, ya sea publico o privado.

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

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

Crear un repositorio en GitHub

Seleccionamos el estado inicial del repositorio, podemos agregar un README, un pequeño manual del repositorios, un gitignore que determina que archivos ignorar y luego seleccionar una licencia, se recomienda MIT para compartir con restricciones mínimas en Internet.


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

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☒ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▼

This will set  main as the default branch. Change the default name in your [settings](#).

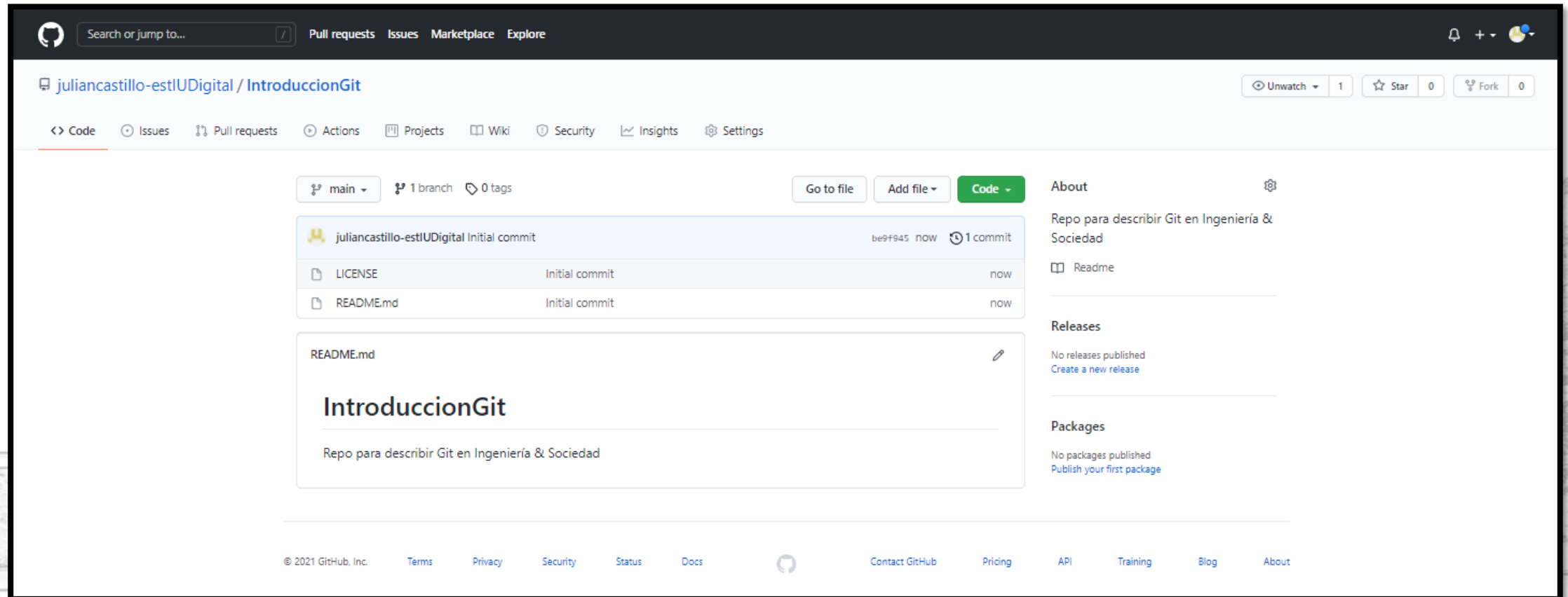
[Create repository](#)

Apache License 2.0
GNU General Public License v3.0
✓ MIT License
BSD 2-Clause "Simplified" License
BSD 3-Clause "New" or "Revised" License
Boost Software License 1.0
Creative Commons Zero v1.0 Universal
Eclipse Public License 2.0
GNU Affero General Public License v3.0
GNU General Public License v2.0
GNU Lesser General Public License v2.1
Mozilla Public License 2.0

La página <https://choosealicense.com/> facilita la operación de escoger una licencia de acuerdo a tus necesidades.

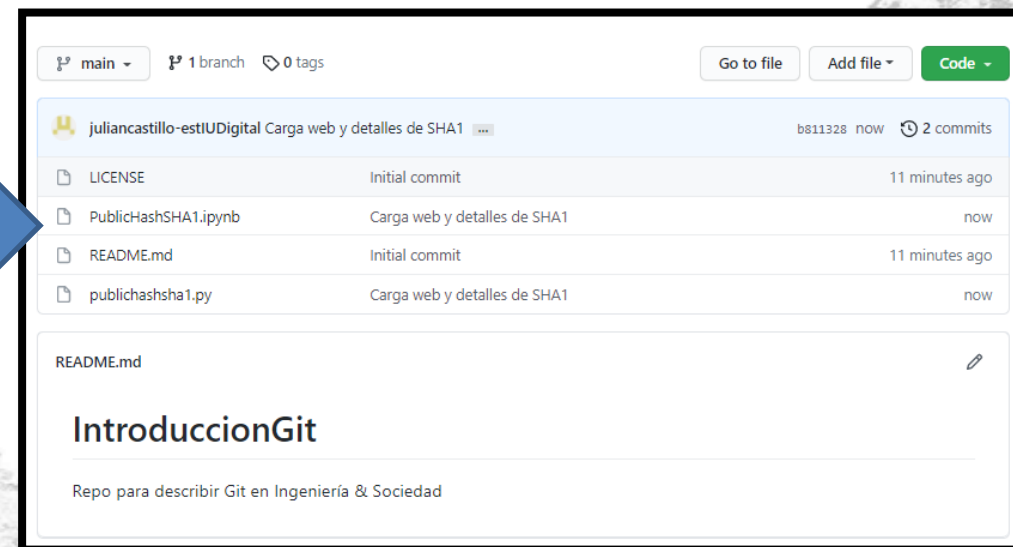
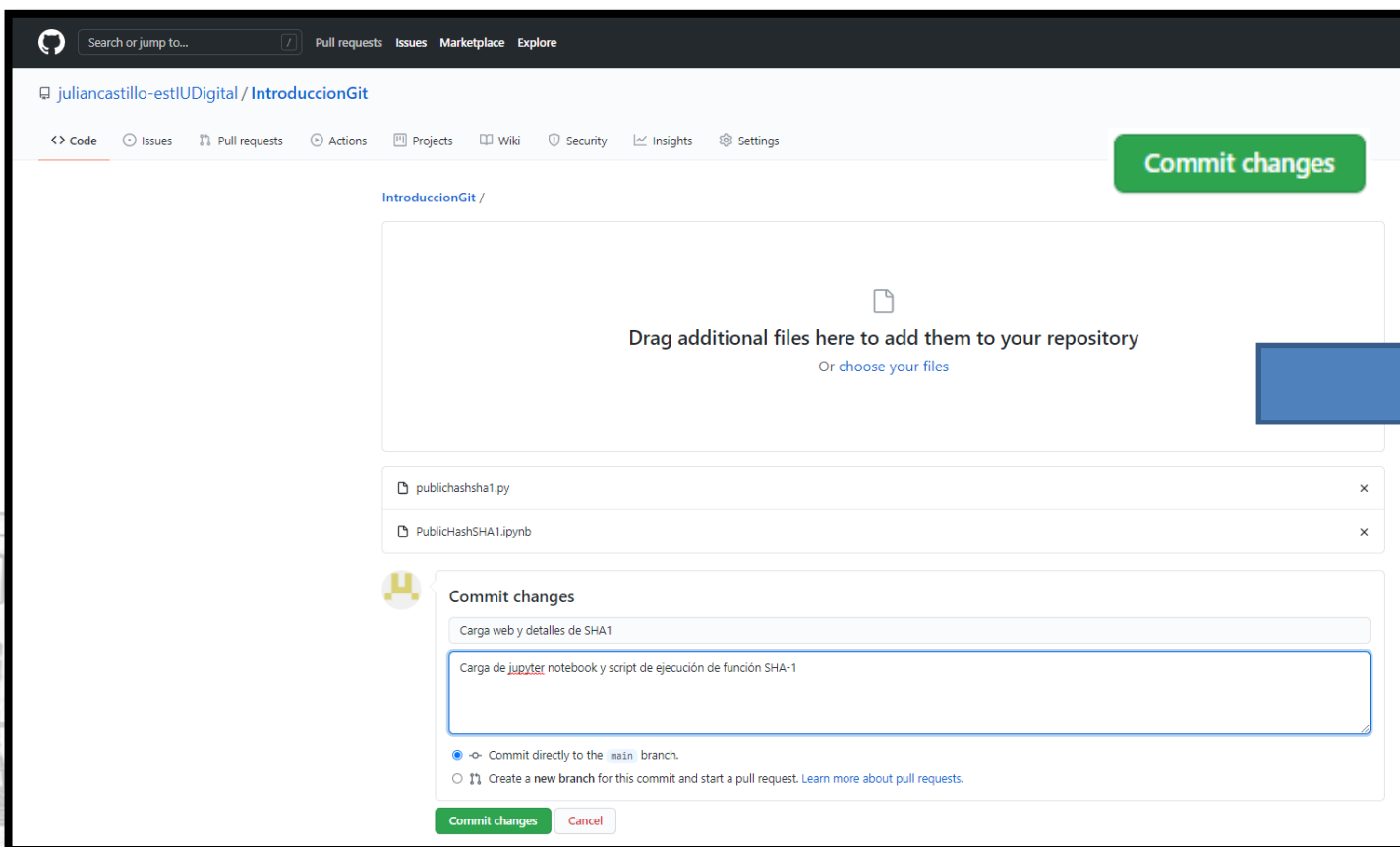
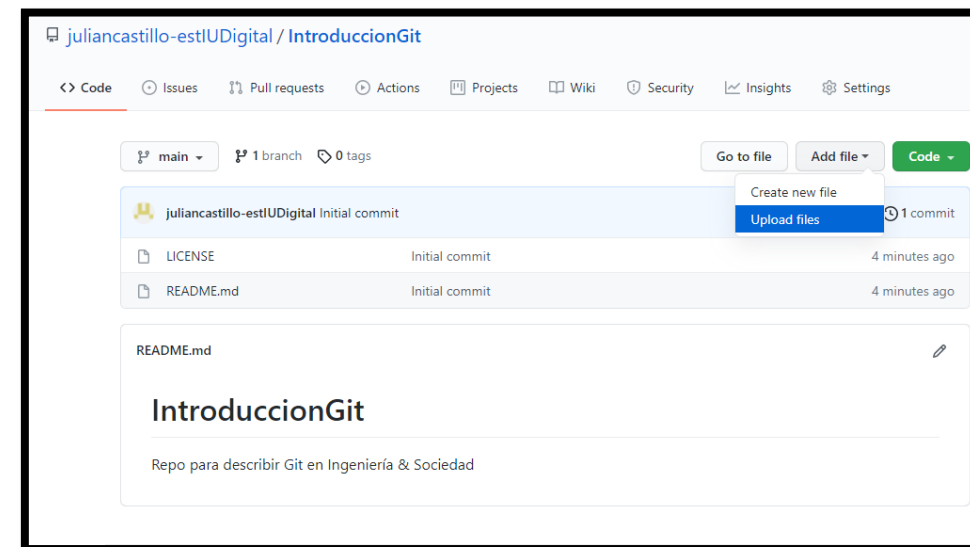
Creado el repo en GitHub

Ahora con el repositorio creado podemos agregar o vincular documentos, incluso crear ramas.



Creado el repo en GitHub

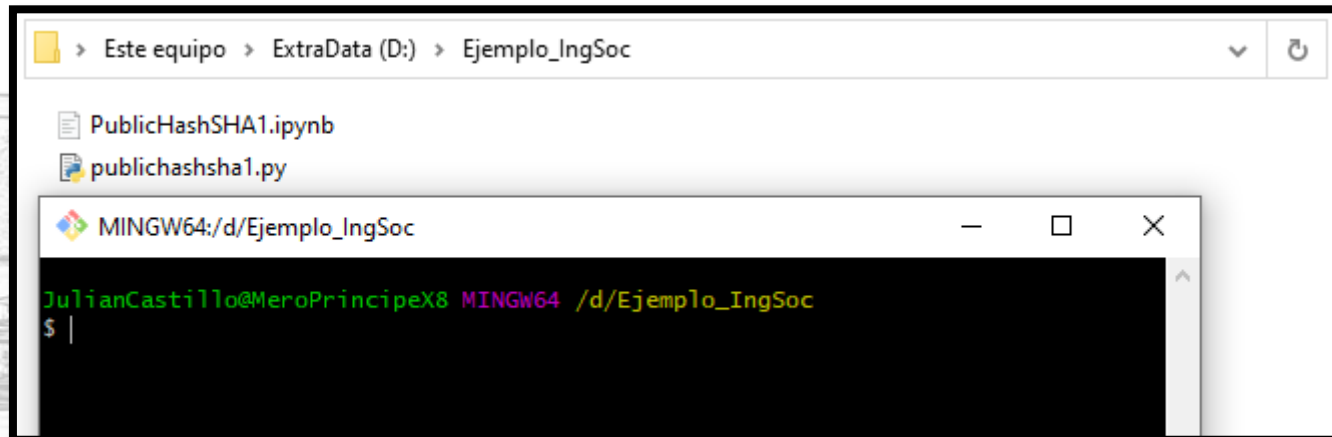
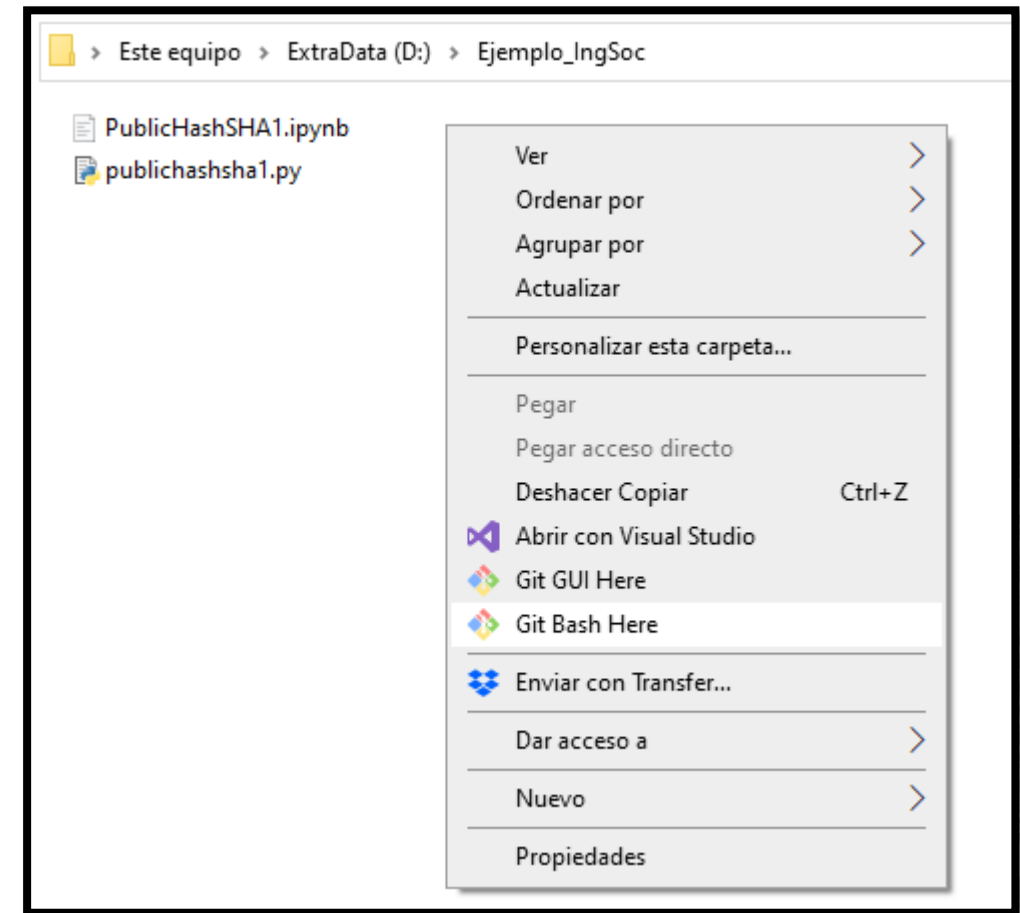
Para cargar documentos seleccionamos la opción “Add file” y clic en “Upload files” y detallar las observaciones del Commit y luego clic en **Commit changes**



Usando Git Bash

Para usar GitHub con Git Bash desde una carpeta desde mi computador, accedemos a la carpeta especifica y luego de tener instalado Git, realizamos clic derecho en ella y seleccionamos “Git Bash Here”.

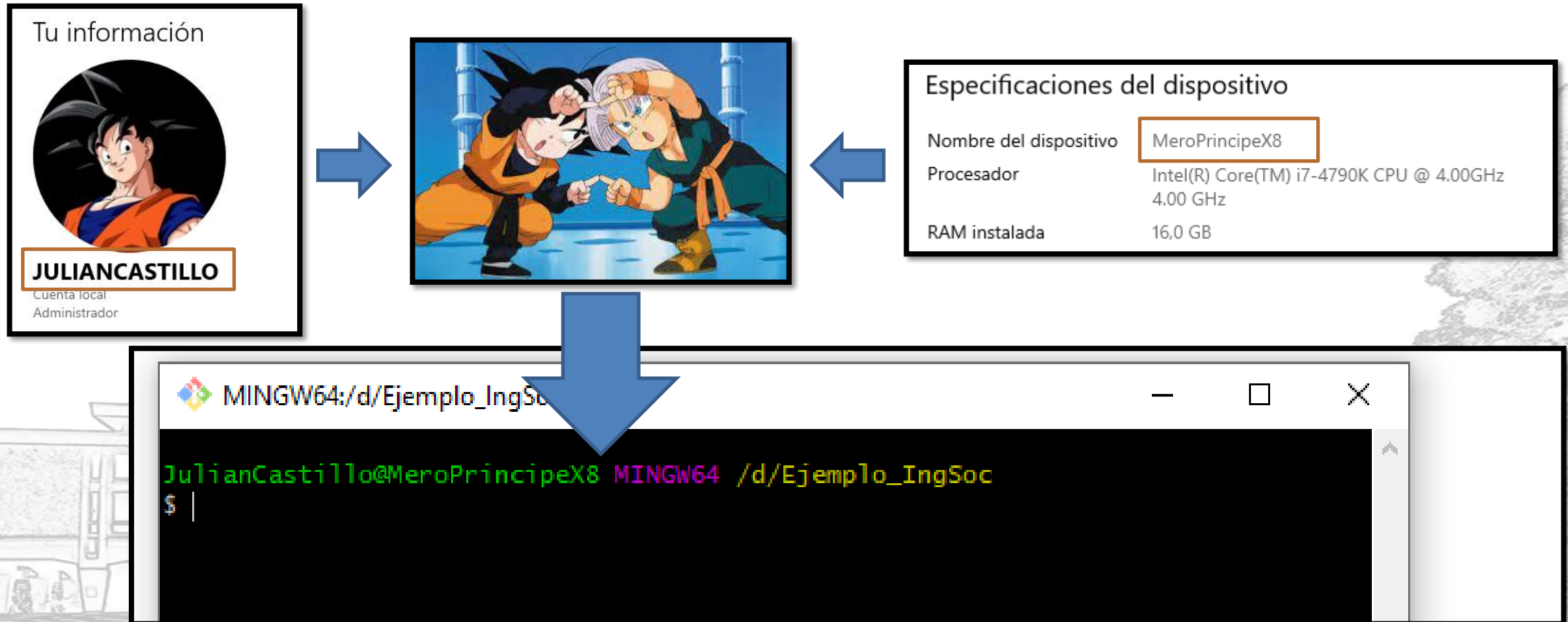
Esto nos abre una CLI (command-line interface) en donde podemos utilizar una serie de comandos para crear, agregar o vincular documentos en GitHub con Git.



El símbolo \$ nos detalla una línea de escritura en donde podemos realizar nuestras instrucciones.

Usando Git Bash

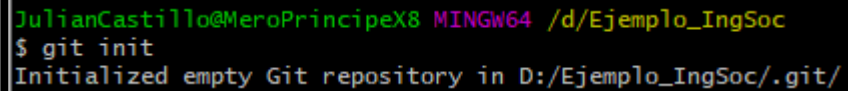
El usuario para Git Bash se crea de la combinación del usuario de sistema actual y el nombre del dispositivo (computador, por defecto es un código).



Usando Git Bash

Inicializar el directorio local como un repositorio de Git.

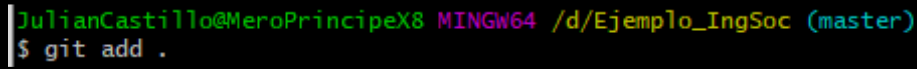
```
$ git init
```



JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc
\$ git init
Initialized empty Git repository in D:/Ejemplo_IngSoc/.git/

Agregar los archivos a tu nuevo repositorio local. Esto representa la primera confirmación. Agrega el archivo en el repositorio local y lo presenta para la confirmación. “.” detalla todos los archivos.

```
$ git add .
```



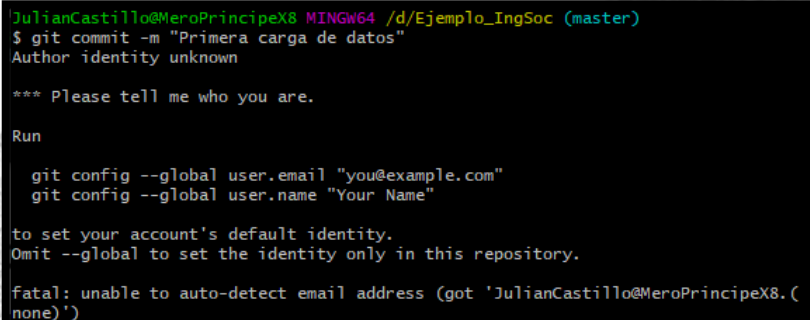
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)
\$ git add .

La alerta detalla que los finales de línea usados por Unix (LF) serán reemplazados por los saltos de línea de Windows (CRLF). [\(¿?LF¿?CRLF¿?\)](#)

No hemos detallado nuestras credenciales de GitHub por lo cual al realizar el primer commit nos solicitará ingresarlas. Para confirmar los archivos que has preparado en tu repositorio local.

```
$ git commit -m "Primer Commit"
```

-m detalla el ingreso de un mensaje



```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)  
$ git commit -m "Primera carga de datos"  
Author identity unknown  
  
*** Please tell me who you are.  
  
Run  
  
    git config --global user.email "you@example.com"  
    git config --global user.name "Your Name"  
  
to set your account's default identity.  
Omit --global to set the identity only in this repository.  
  
fatal: unable to auto-detect email address (got 'JulianCastillo@MeroPrincipeX8.(none)')
```

Usando Git Bash

Procedemos a definir el correo y el nombre de nuestro usuario de Git.

```
$ git config --global user.email "julian.castillo@est.iudgital.edu.co"
```

```
$ git config --global user.name "Julian Castillo"
```

```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)
$ git config --global user.email "julian.castillo@est.iudigital.edu.co"


JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)
$ git config --global user.name "Julian Castillo"
```

Para confirmar los archivos hacemos Commit.

```
$ git commit -m "Primer Commit"
```

```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)
$ git commit -m "primer commit"
[master (root-commit) fea4a8b] primer commit
2 files changed, 235 insertions(+)
create mode 100644 PublicHashSHA1.ipynb
create mode 100644 publichashsha1.py
```


Usando Git Bash

Procedemos a definir el repositorio, para acceder al vinculo del repositorio clic en el botón Code  y copiamos el vinculo HTTPS del repositorio.

`https://github.com/juliancastillo-estIUDigital/IntroduccionGit.git`

Debemos agregar la URL para el repositorio remoto donde se subirá tu repositorio local.

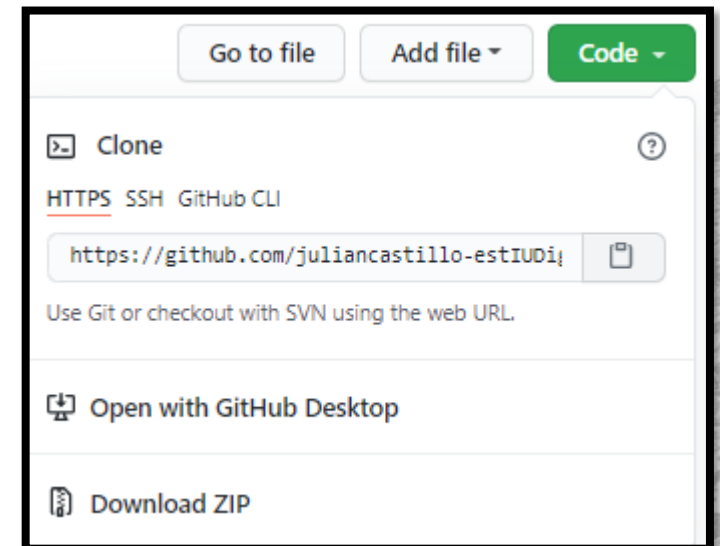
```
$ git remote add origin  
https://github.com/juliancastillo-  
estIUDigital/IntroduccionGit.git
```

```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)  
$ git remote add origin https://github.com/juliancastillo-estIUDigital/IntroduccionGit.git
```

Definimos el nuevo documento remoto y nos muestra la ruta del directorio asignada con la instrucción -v

```
$ git remote -v
```

```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)  
$ git remote -v  
origin https://github.com/juliancastillo-estIUDigital/IntroduccionGit.git (fetch)  
origin https://github.com/juliancastillo-estIUDigital/IntroduccionGit.git (push)
```



Usando Git Bash

Procedemos a cargar en la web los documentos con Commit.

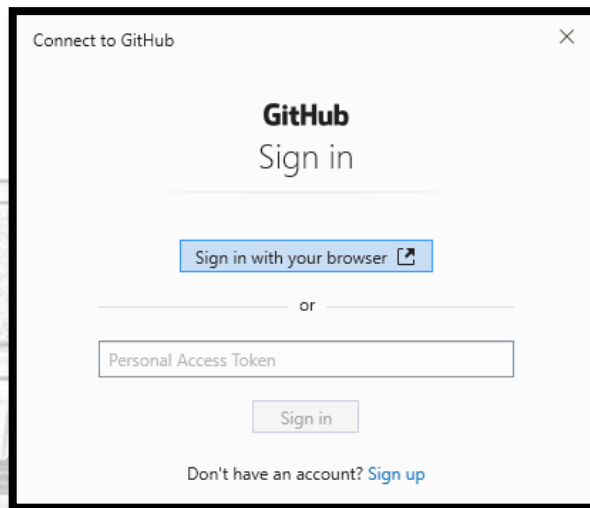
```
$ git push origin master
```

Al realizar esta acción nos solicita las credenciales de acceso a GitHub, diligenciamos los campos requeridos.

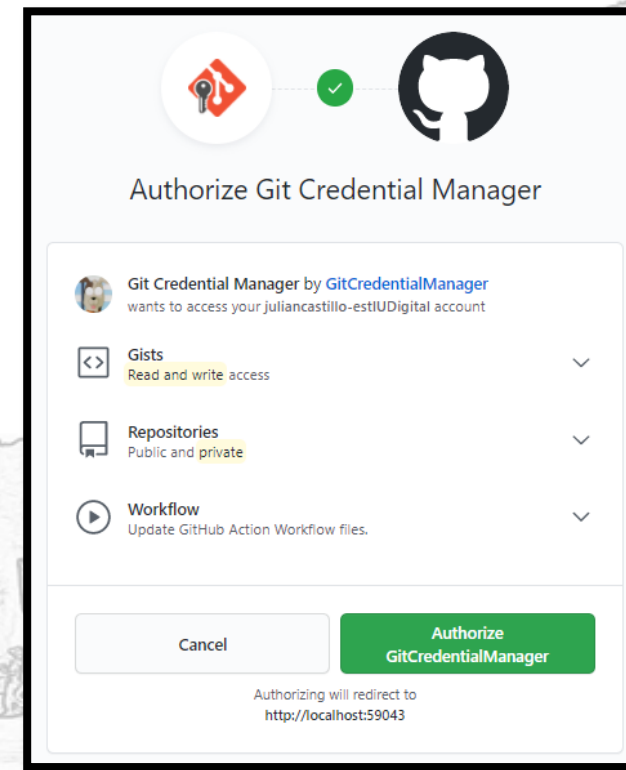
Podemos ingresar con el explorador o con un token especial.

Para crear un token podemos acceder al siguiente sitio.

<https://github.com/settings/tokens/new>




Luego de aceptar o detallar las credenciales podemos visualizar el avance de carga de los documentos en el CLI



Usando Git Bash

Los datos se cargan exitosamente, a lo cual CLI nos informa los detalles.

```
JulianCastillo@MeroPrincipeX8 MINGW64 /d/Ejemplo_IngSoc (master)
$ git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 3.60 KiB | 1.80 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/juliancastillo-estIUDigital/IntroduccionGit/pull/new/master
remote:
To https://github.com/juliancastillo-estIUDigital/IntroduccionGit.git
 * [new branch]      master -> master
```

 master had recent pushes 5 minutes ago

[Compare & pull request](#)

Algunas licencias recomendadas

MIT License

A short and simple permissive license with conditions only requiring preservation of copyright and license notices. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- Commercial use
- Distribution
- Modification
- Private use

Conditions

- License and copyright notice

Limitations

- Liability
- Warranty

GNU General Public License v3.0

GNU GPLv3

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

Permissions

- Commercial use
- Distribution
- Modification
- Patent use
- Private use

Conditions

- Disclose source
- License and copyright notice
- Same license
- State changes

Limitations

- Liability
- Warranty

No License

When you make a creative work (which includes code), the work is under exclusive copyright by default. Unless you include a license that specifies otherwise, nobody else can copy, distribute, or modify your work without being at risk of take-downs, shake-downs, or litigation. Once the work has other contributors (each a copyright holder), “nobody” starts including you.

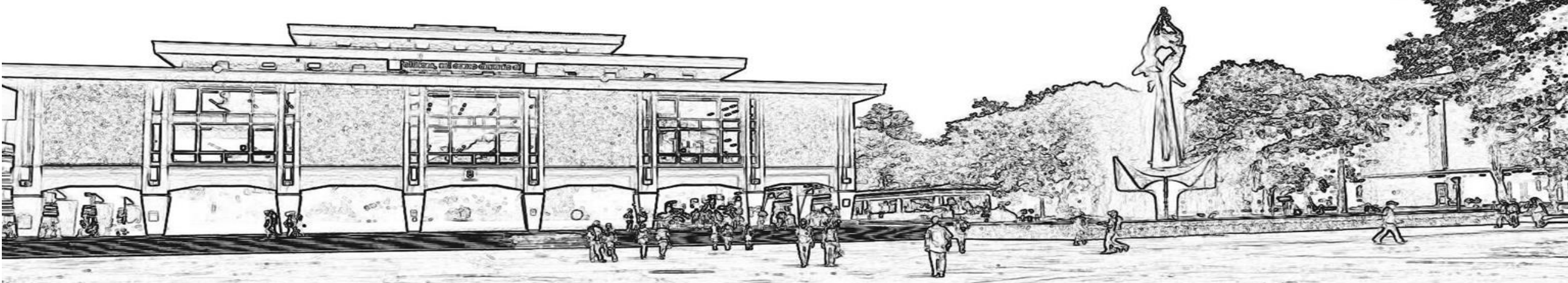
Even in the absence of a license file, you may grant some rights in cases where you publish your source code to a site that requires accepting terms of service. For example, if you publish your source code in a public repository on GitHub, you have accepted the [Terms of Service](#), by which you allow others to view and fork your repository. Others may not need your permission if [limitations and exceptions to copyright](#) apply to their particular situation. Neither site terms nor jurisdiction-specific copyright limitations are sufficient for the kinds of collaboration that people usually seek on a public code host, such as experimentation, modification, and sharing as fostered by an open source license.



UNIVERSIDAD DE ANTIOQUIA

Ingeniería & Sociedad

Facultad de Ingeniería



Muchas gracias por su atención

Julián Andrés Castillo G.
Jandres.castillo@udea.edu.co