



Sistema de Control de Versiones GIT



www.sena.edu.co

GIT



¿Qué es un Sistema Control de Versiones?

Registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo.

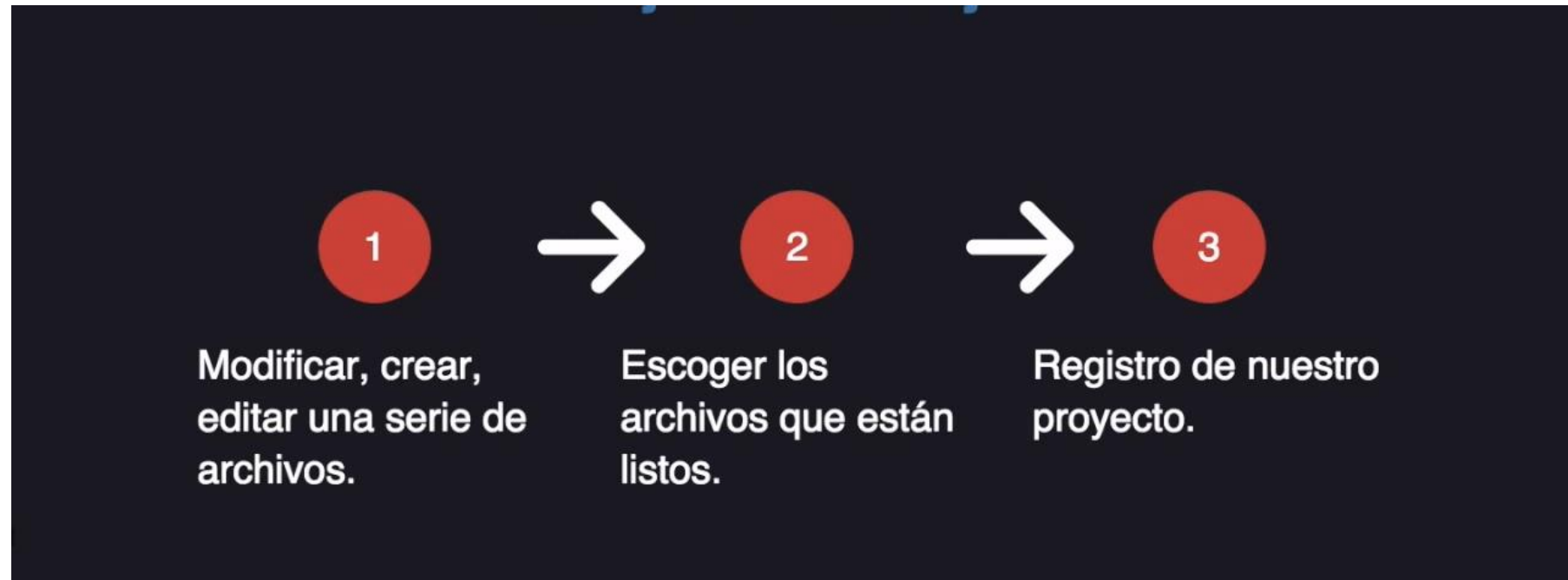
¿Qué es Git?

- Sistema de control de versiones
- Open Source
- Creado por Linus Torvalds

Qué permite GIT

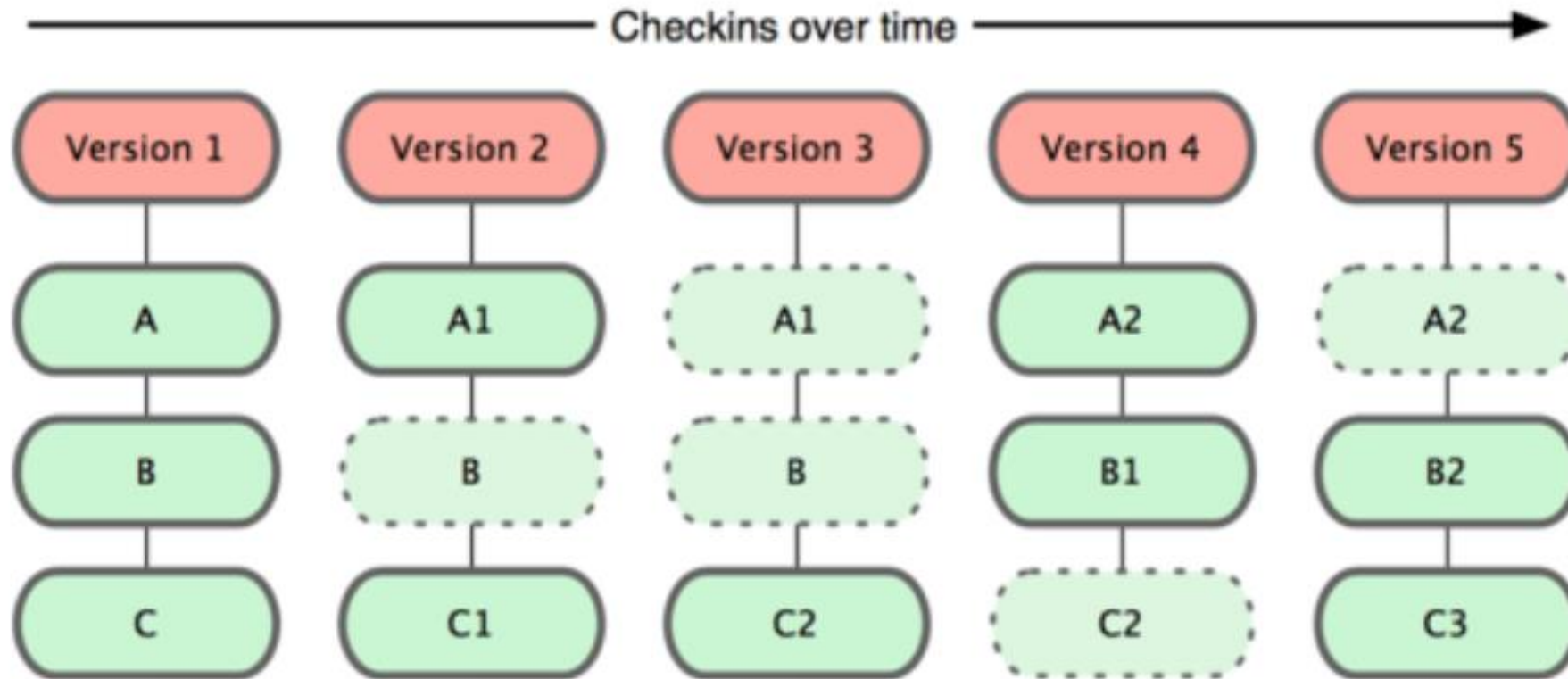
- Crear copias de seguridad y restaurarlas
- Sincronizar (mantener al día) a los desarrolladores respecto a la última versión de desarrollo
- Deshacer cambios
- Gestionar la autoría del código
- Realizar pruebas (aisladas)

GIT Flujo de Trabajo



Conceptos

- Git gestiona el repositorio como instantáneas de su estado.
- El sistema de versiones gestiona el repositorio llevando la cuenta de los cambios incrementales que ha habido.





Instalación Git

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

Instalación Git

staging areas, and multiple workflows.



About

The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.



Downloads

GUI clients and binary releases for all major platforms.



Community

Get involved! Bug reporting, mailing list, chat, development and more.



Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).



[Windows GUIs](#)



[Tarballs](#)



[Mac Build](#)



[Source Code](#)



Instalación Git

About

Documentation

Downloads

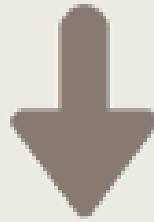
GUI Clients

Logos

Community

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Downloading Git



You are downloading the latest (2.29.2) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **27 days ago**, on 2020-11-04.

[Click here to download manually](#)

Other Git for Windows downloads

Git for Windows Setup

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)

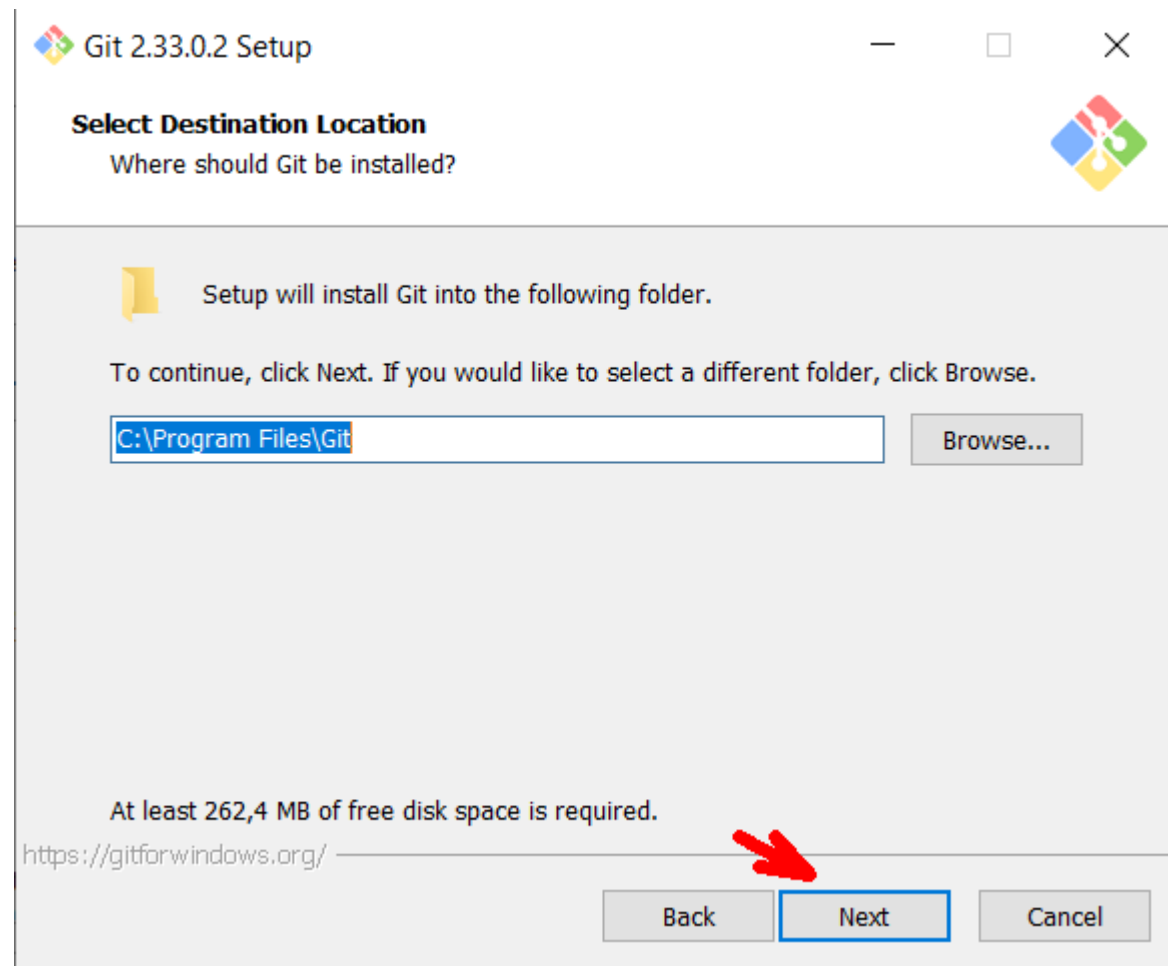
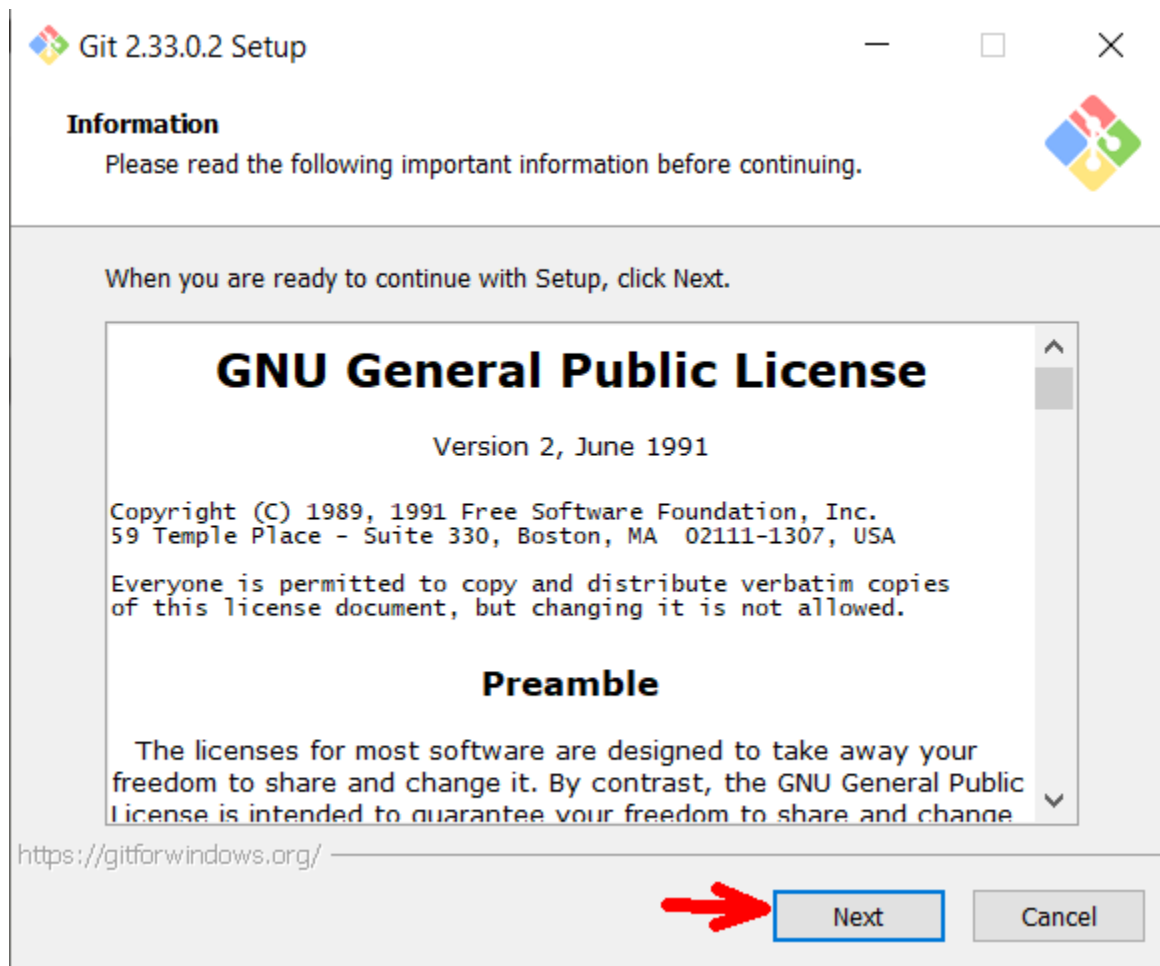


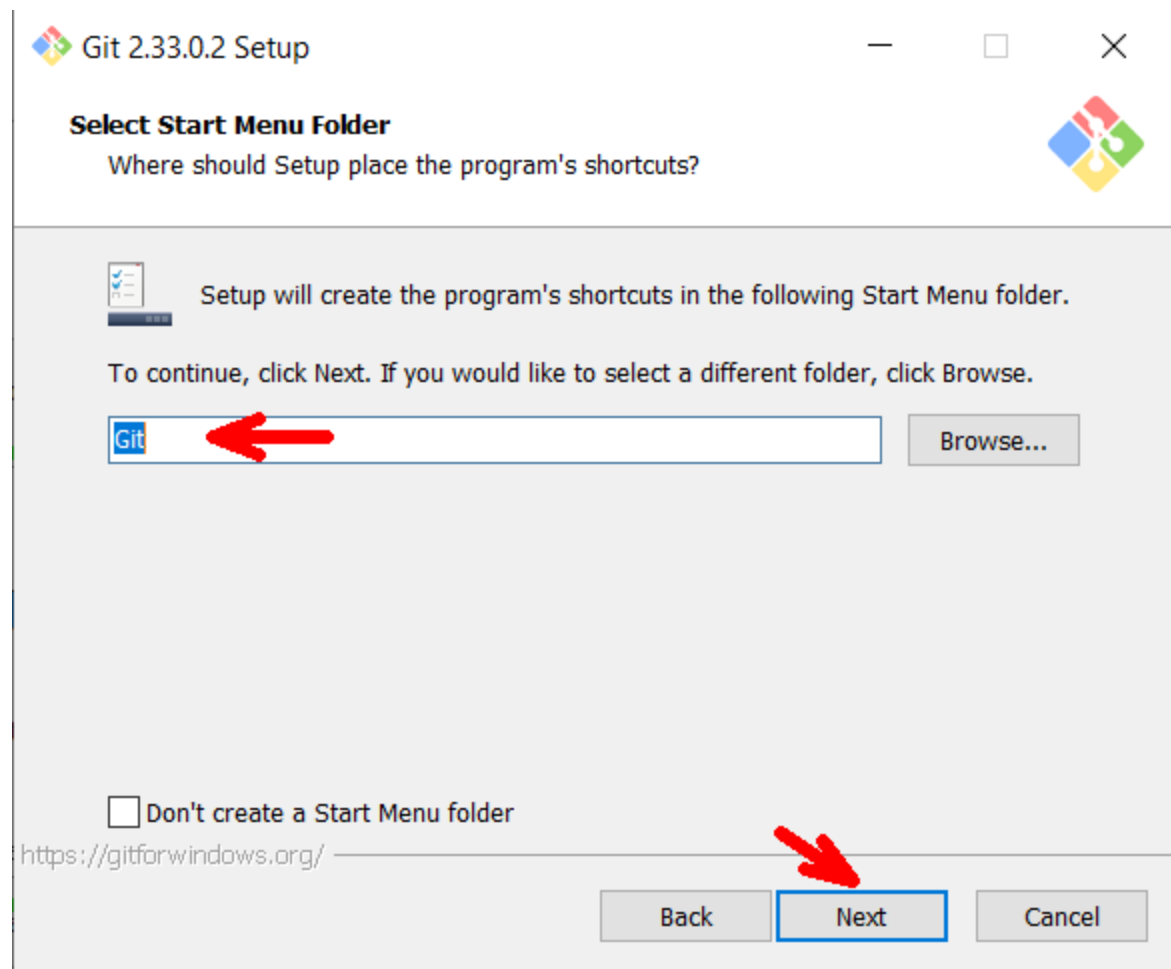
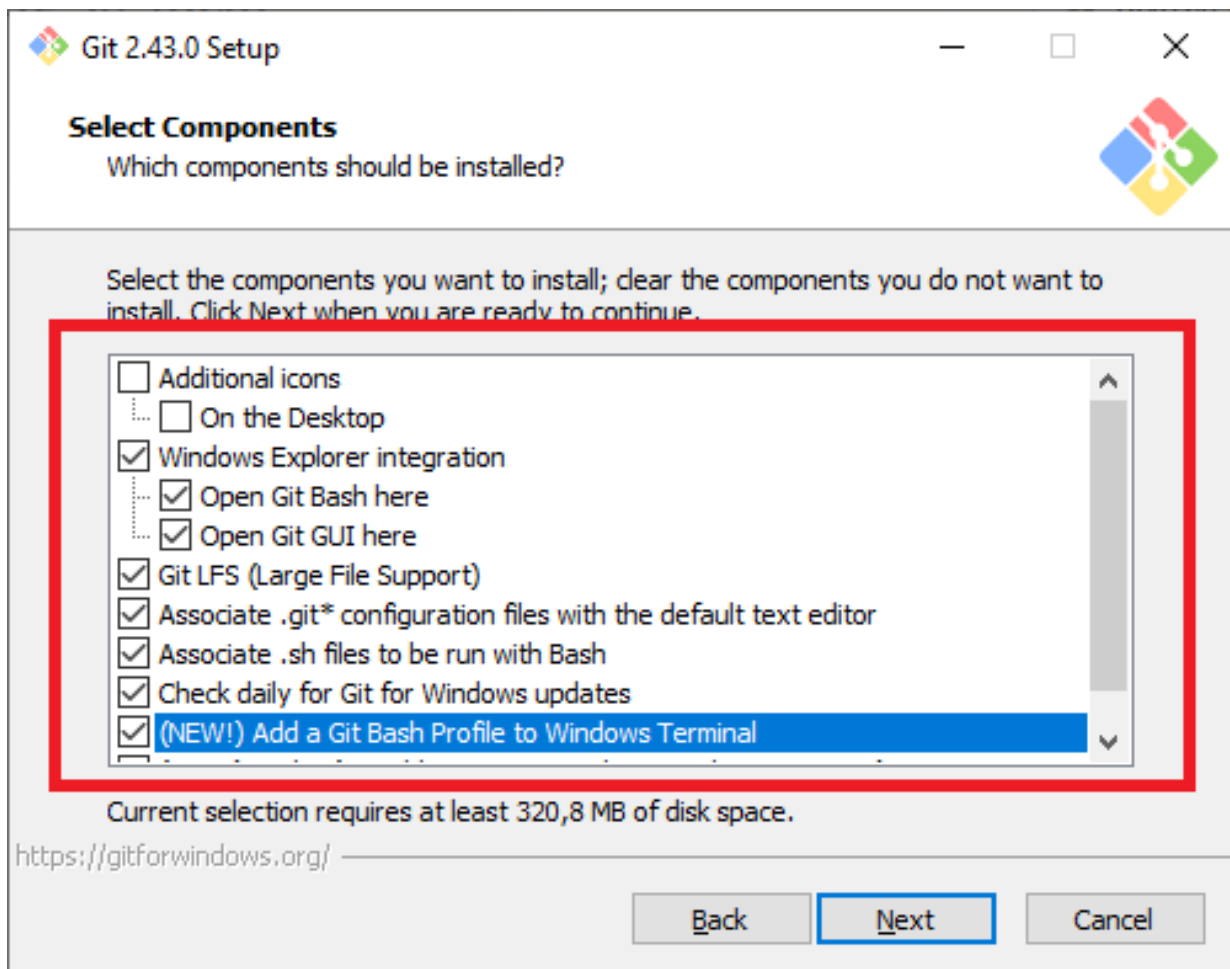
Git for Windows Portable ("thumbdrive edition")

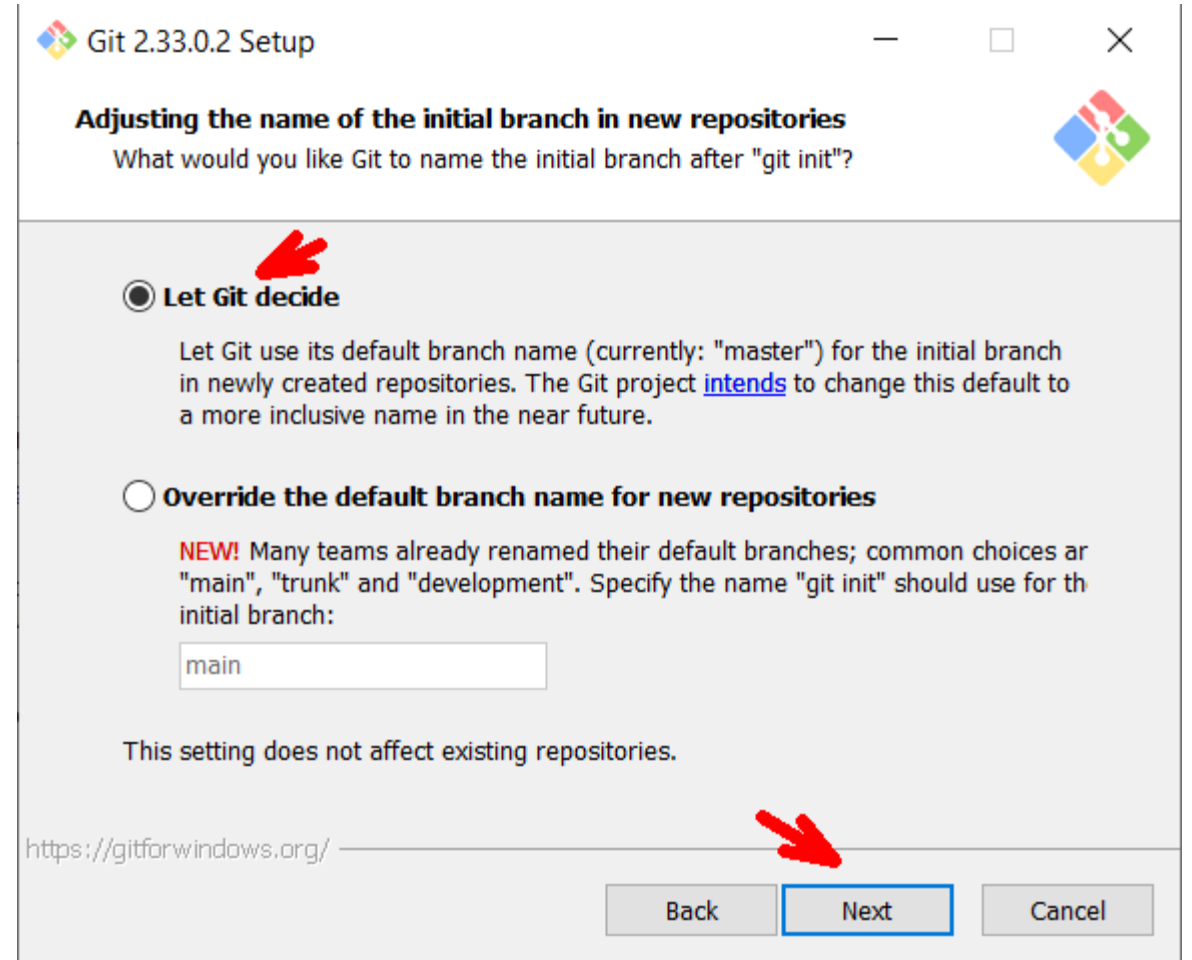
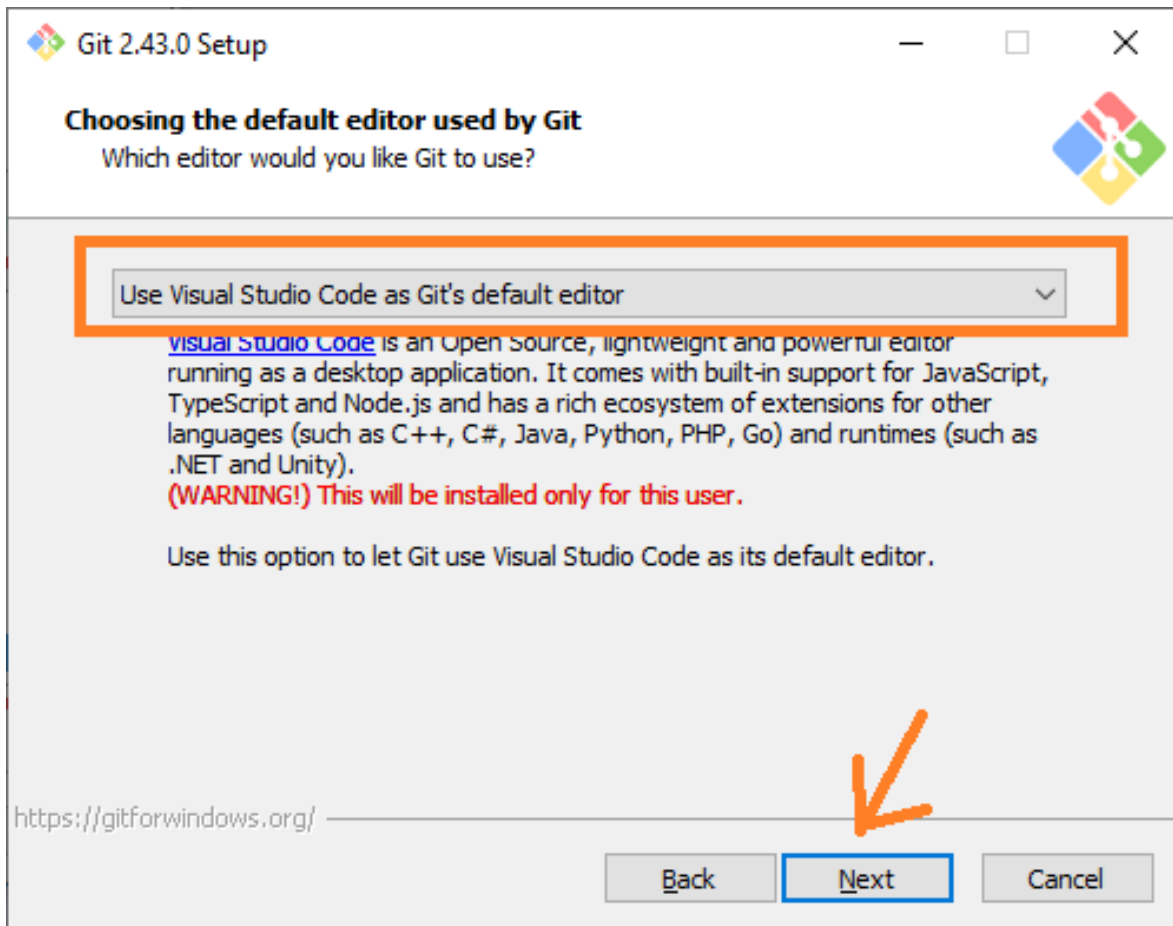
[32-bit Git for Windows Portable.](#)

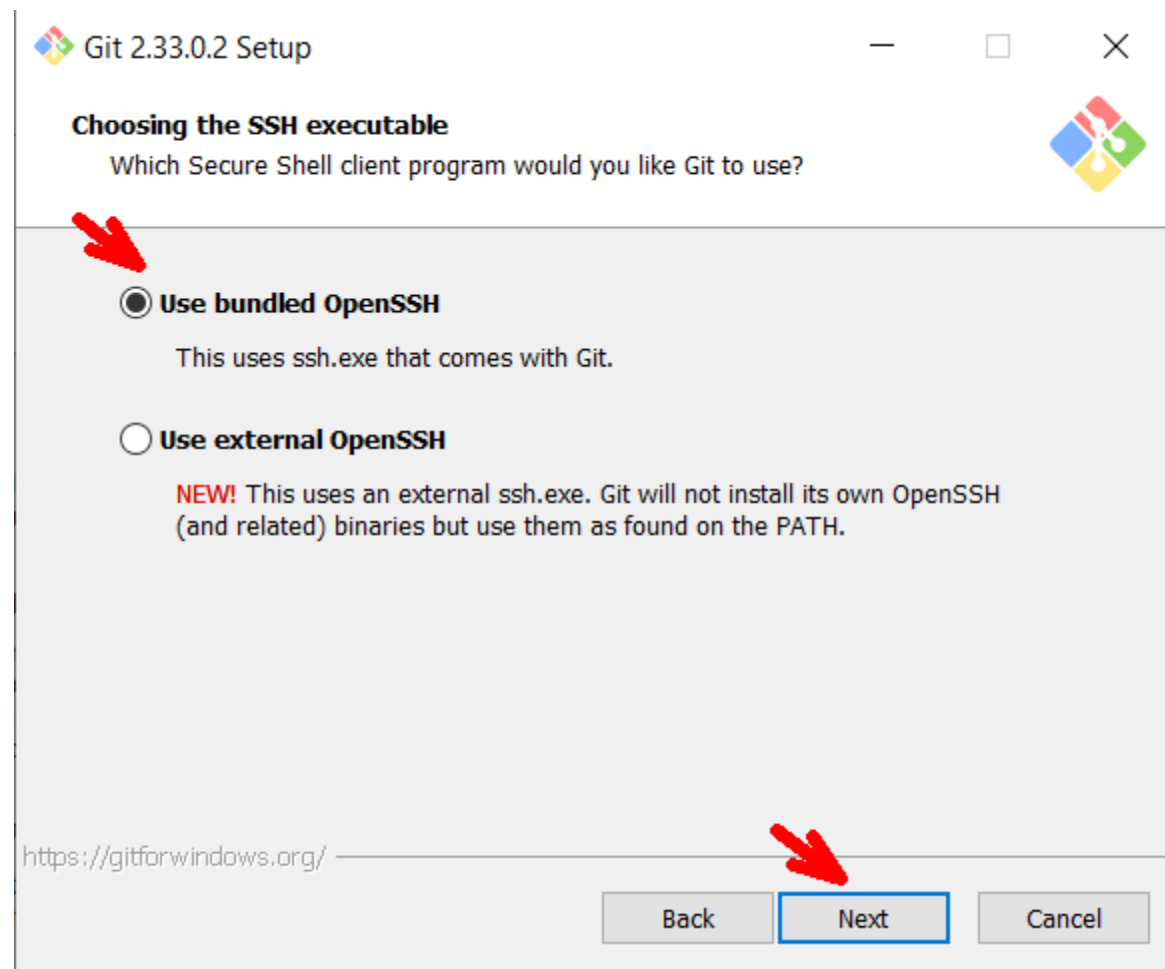
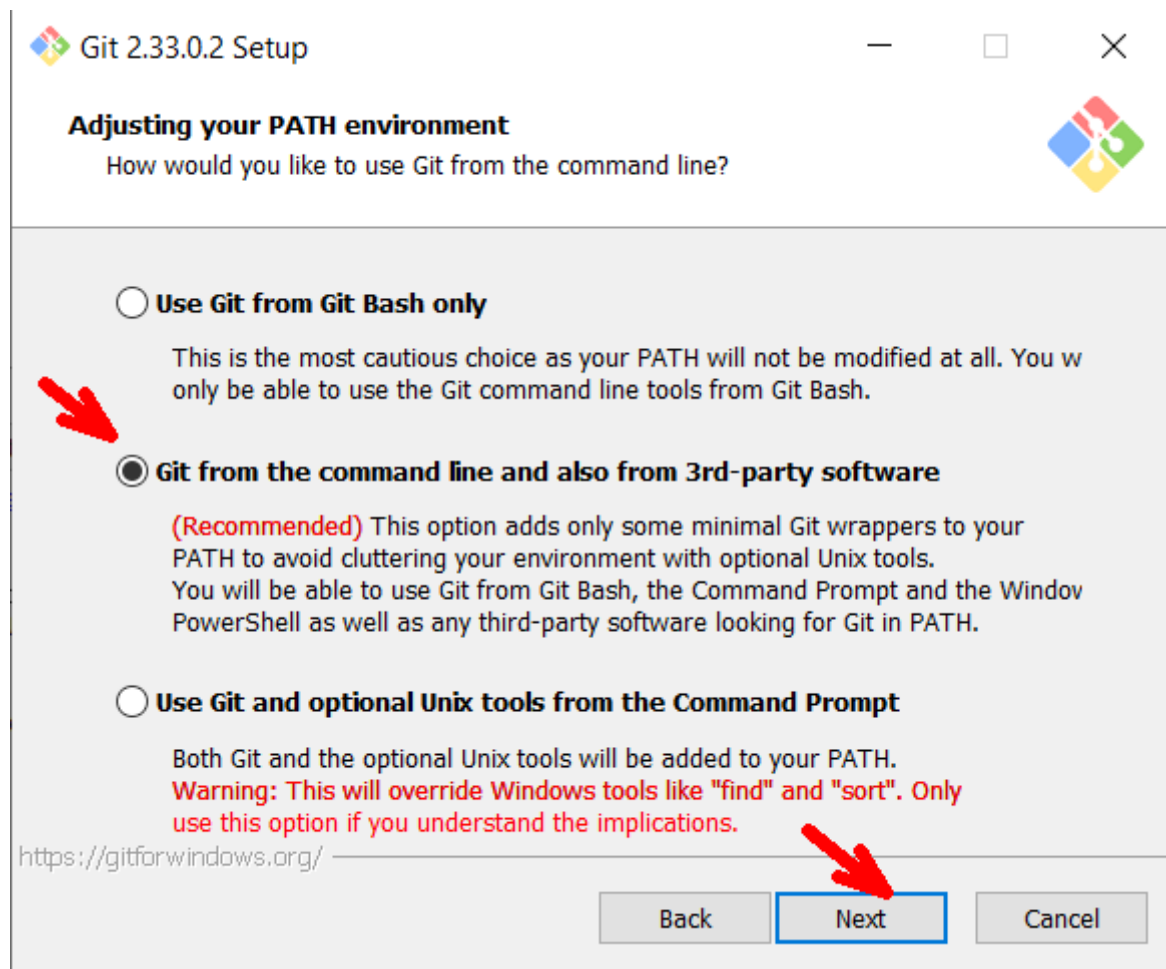
[64-bit Git for Windows Portable.](#)

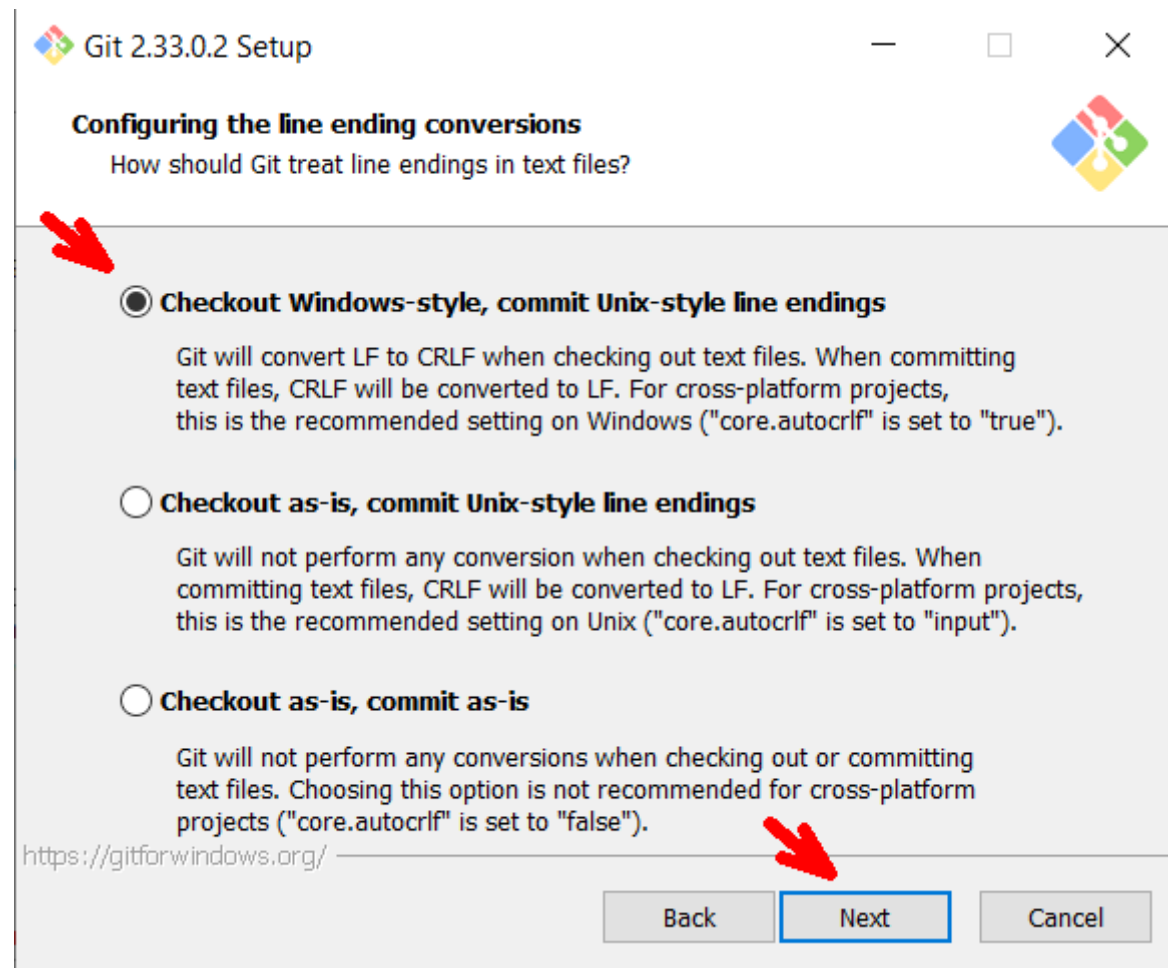
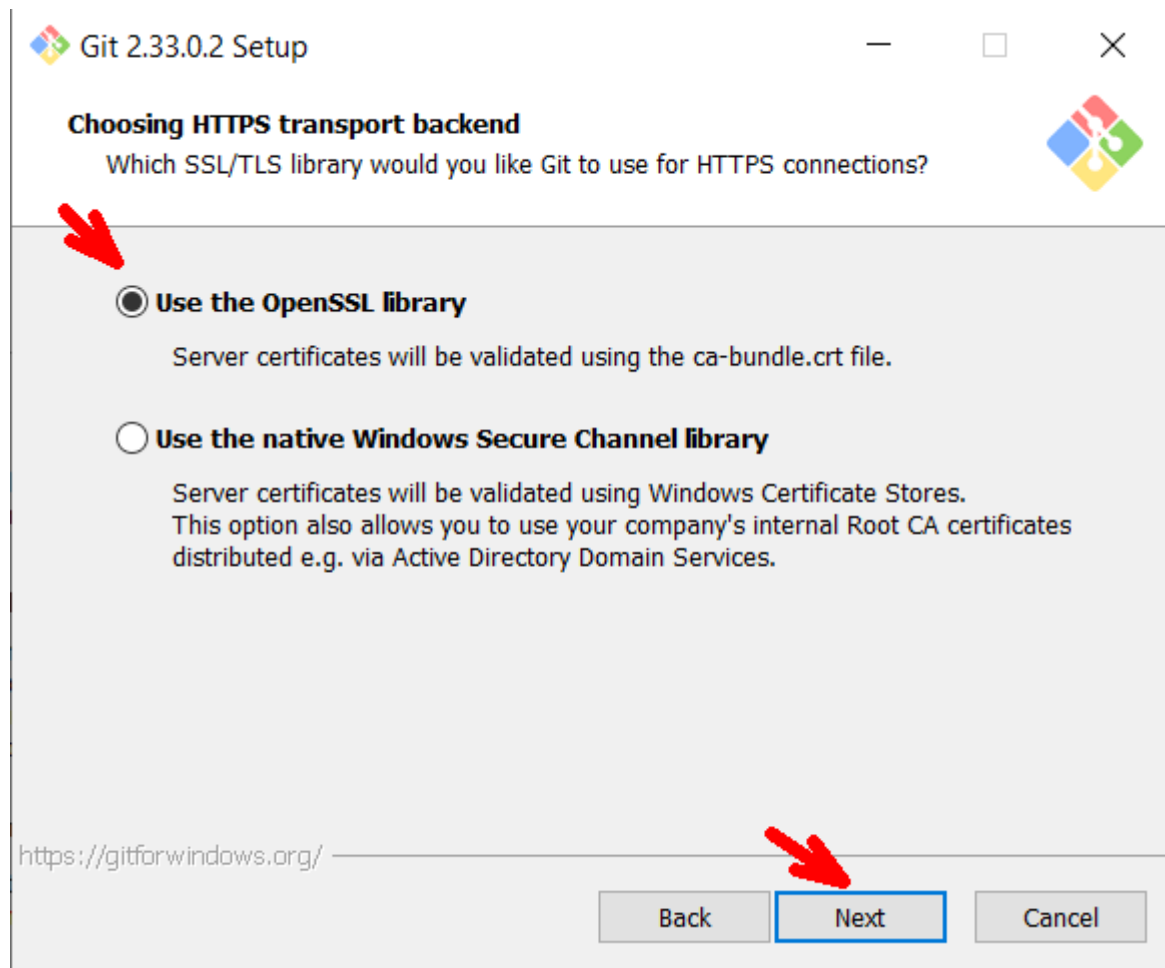
The current source code release is version 2.29.2. If you want the newer version, you can build it from [the source code](#).

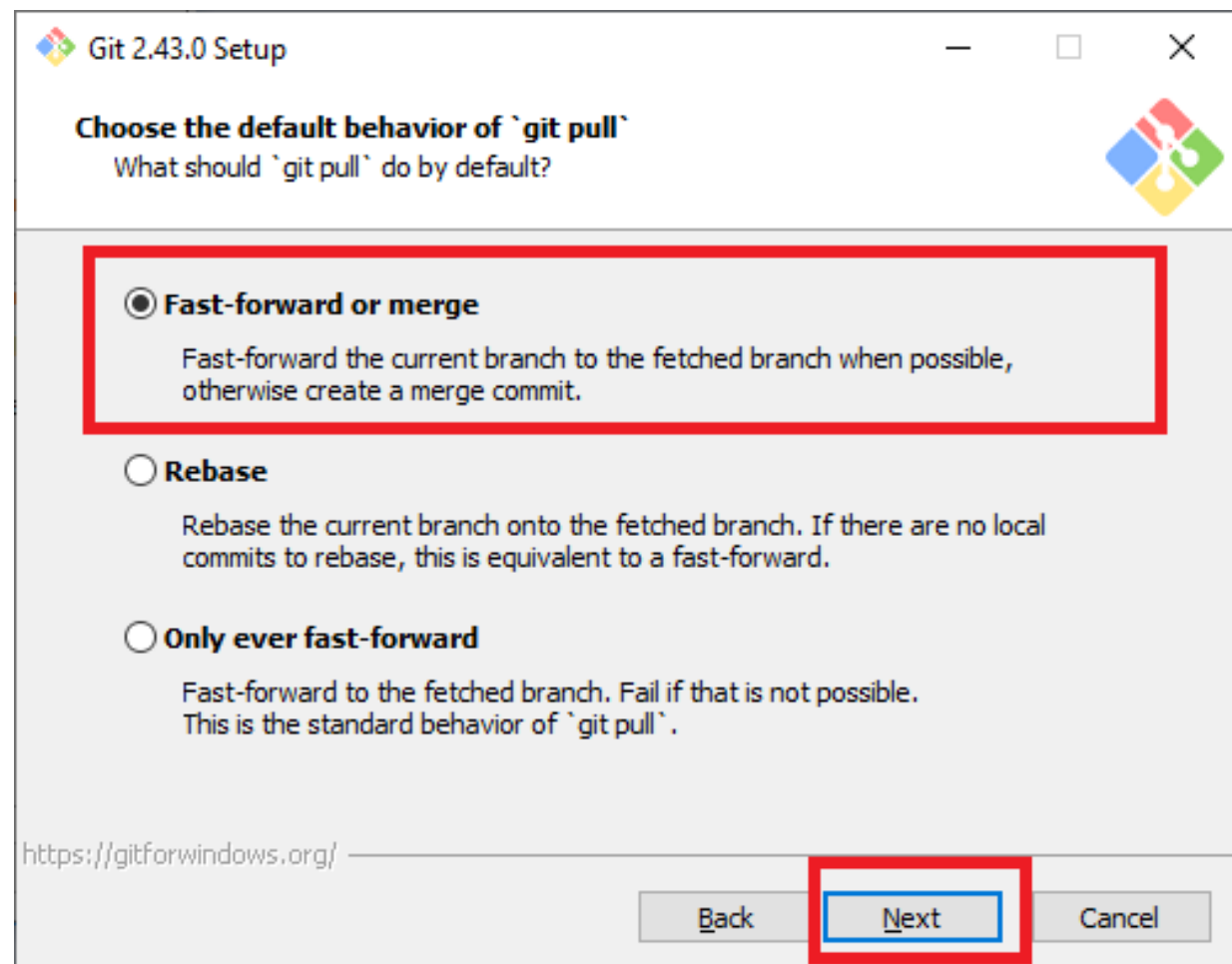
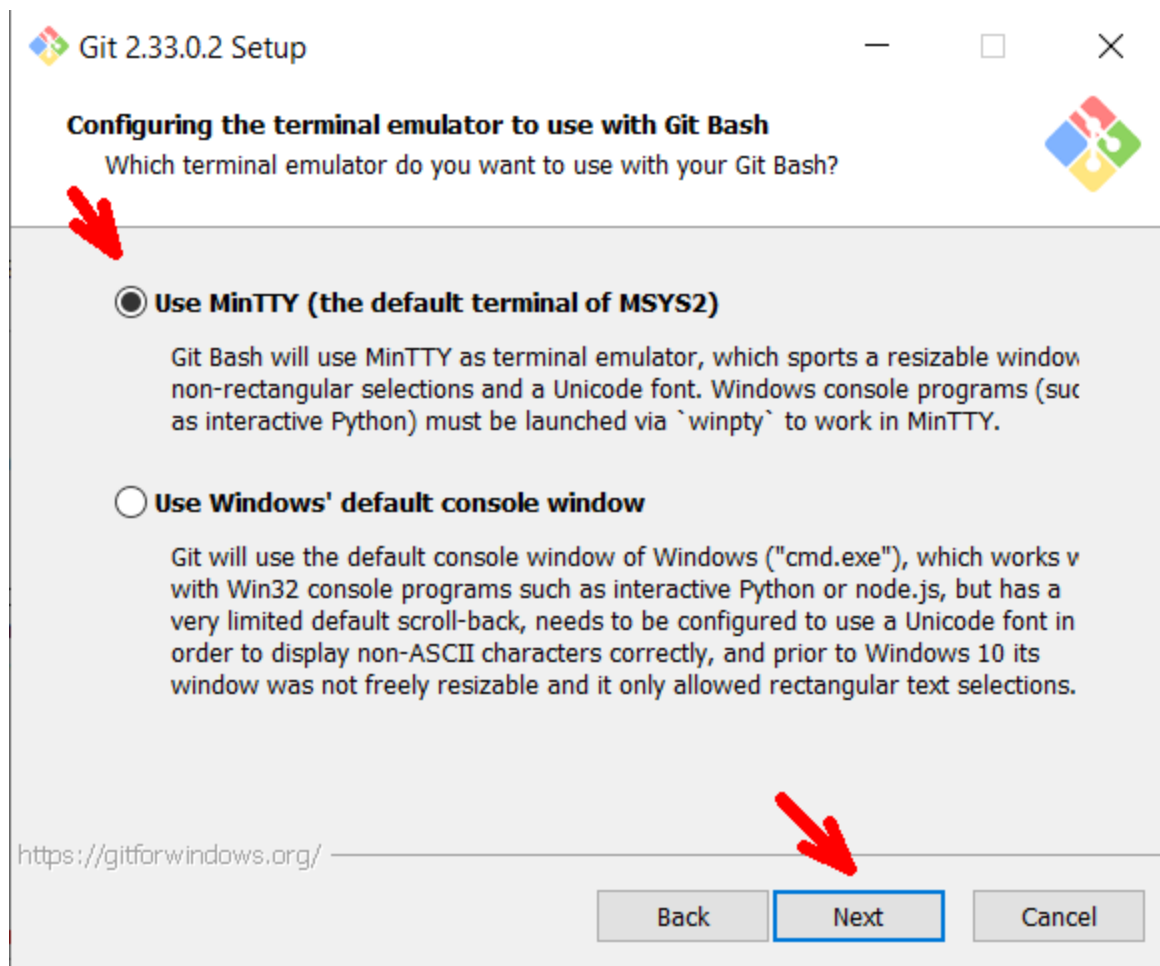


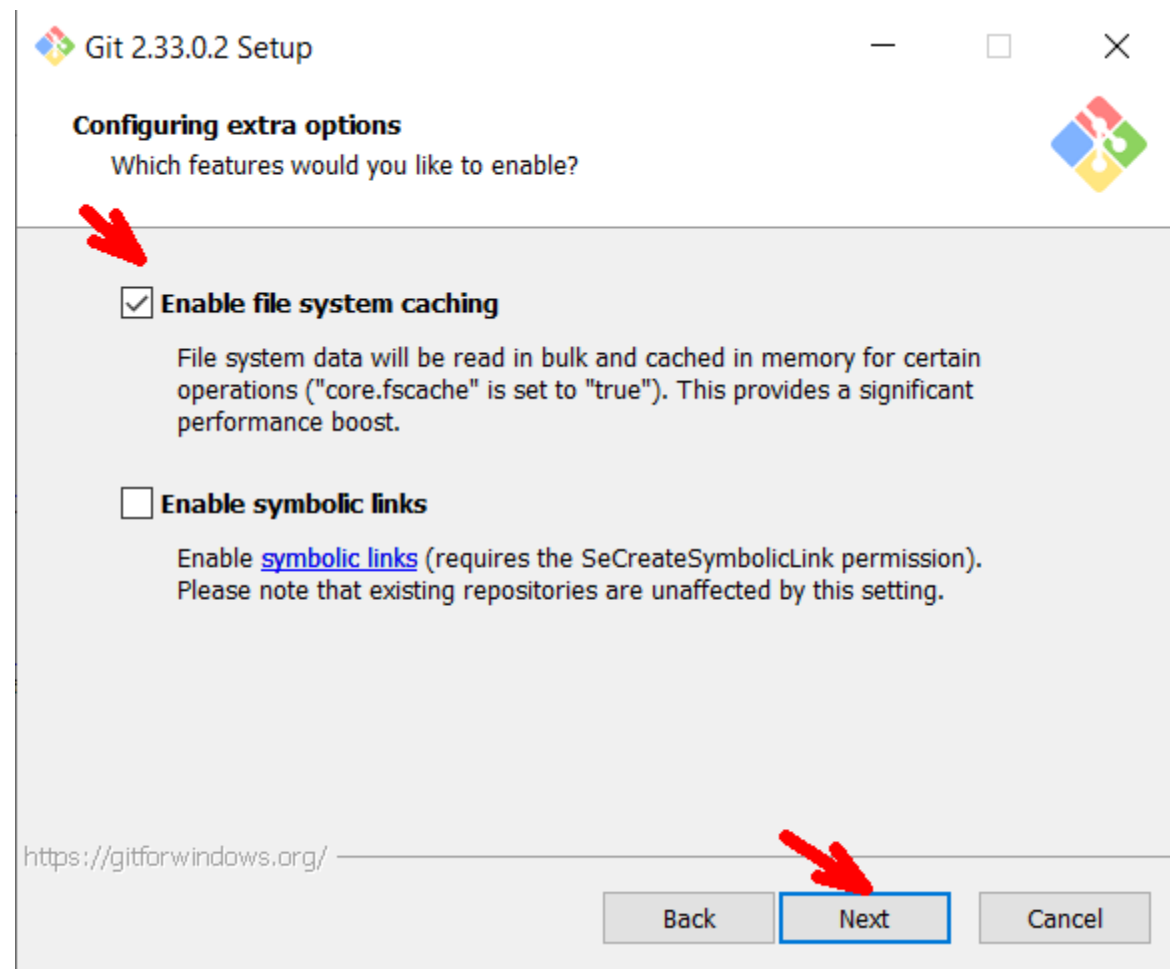
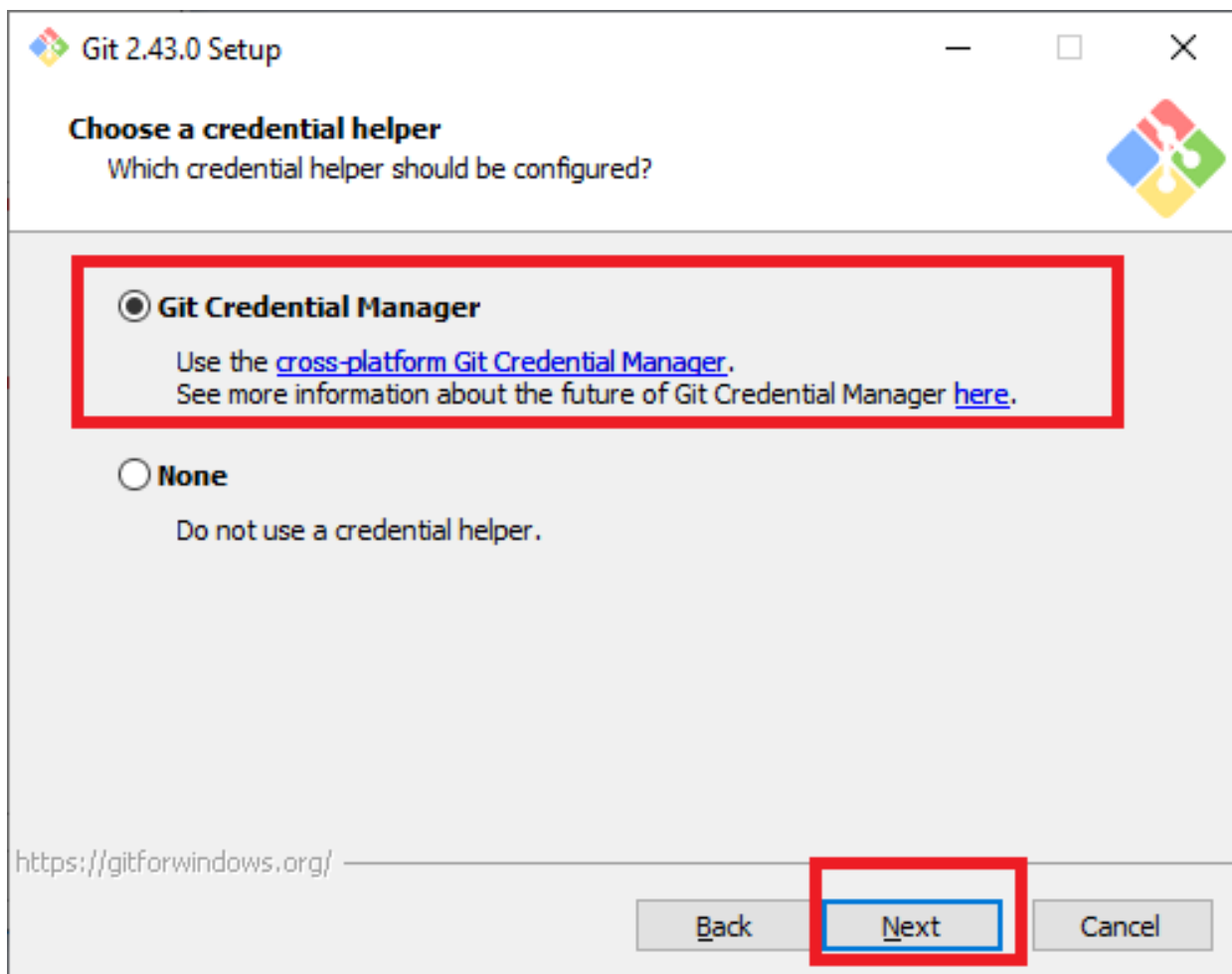


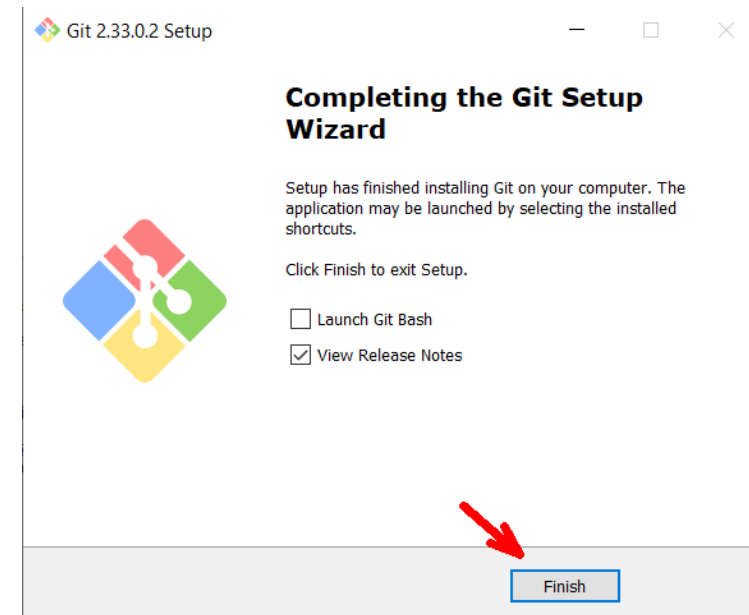
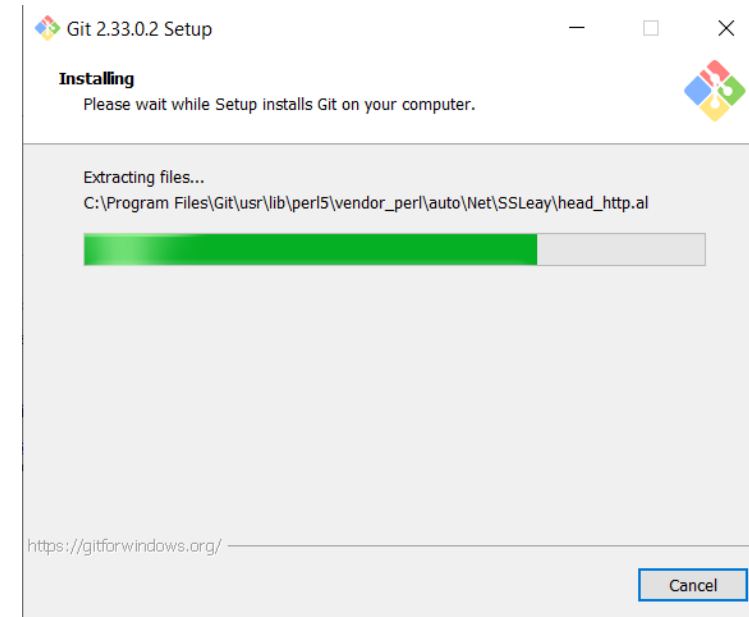
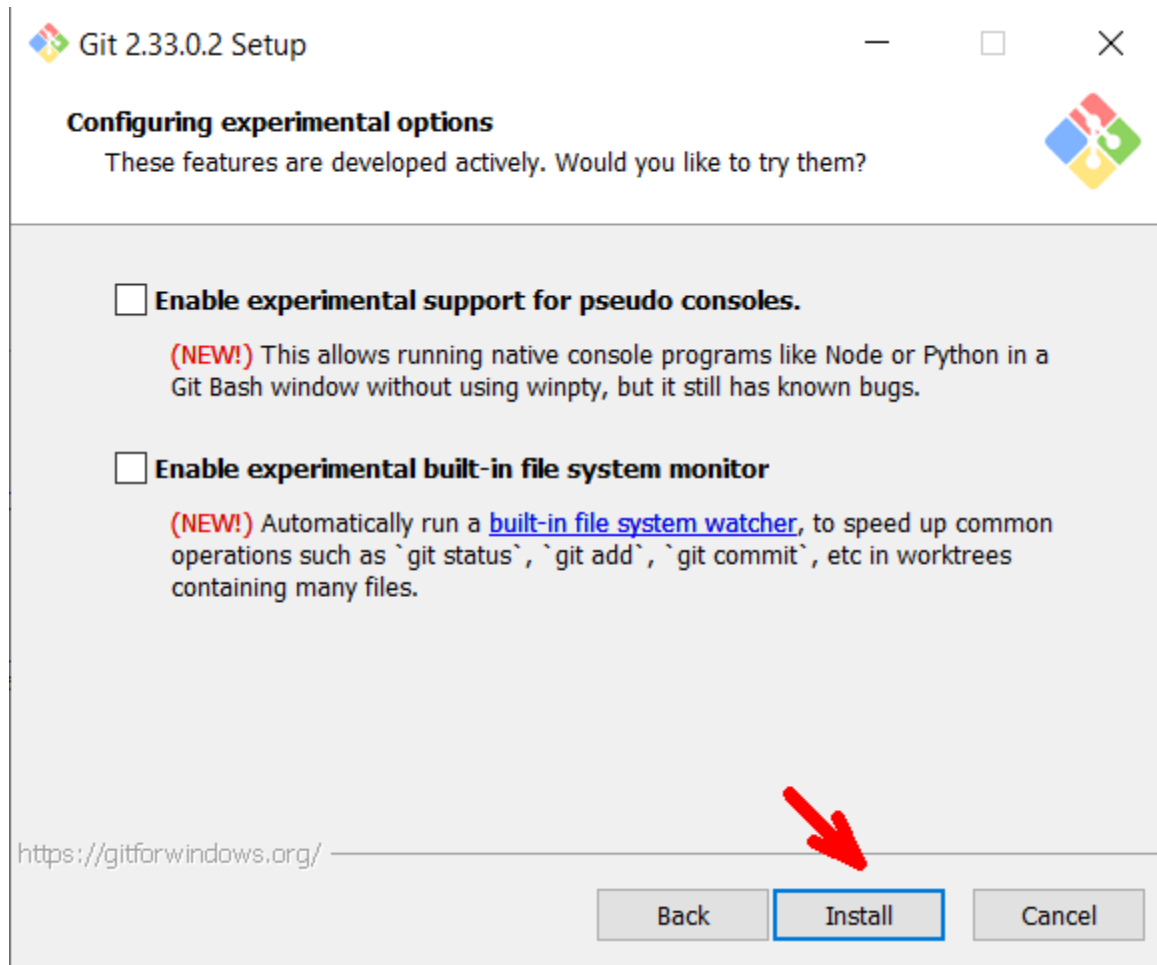


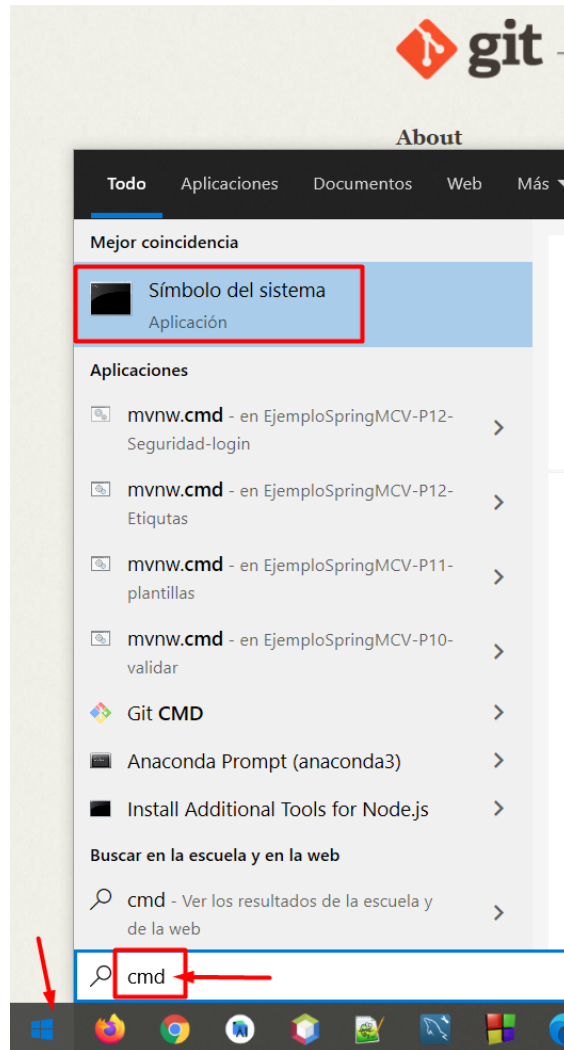












```
Command Prompt

Microsoft Windows [Version 10.0.19045.3803]
(c) Microsoft Corporation. All rights reserved.

C:\Users\SENA>git --version
git version 2.43.0.windows.1

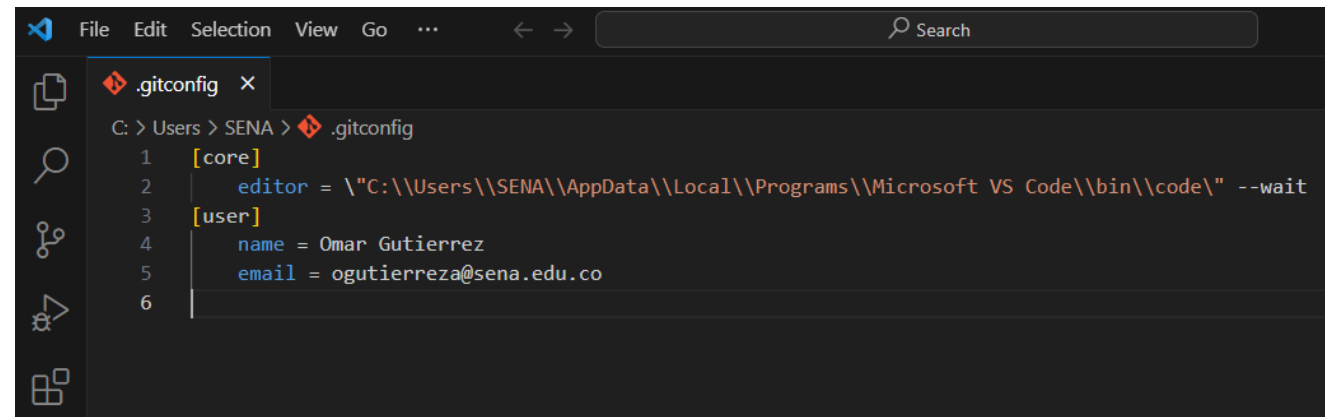
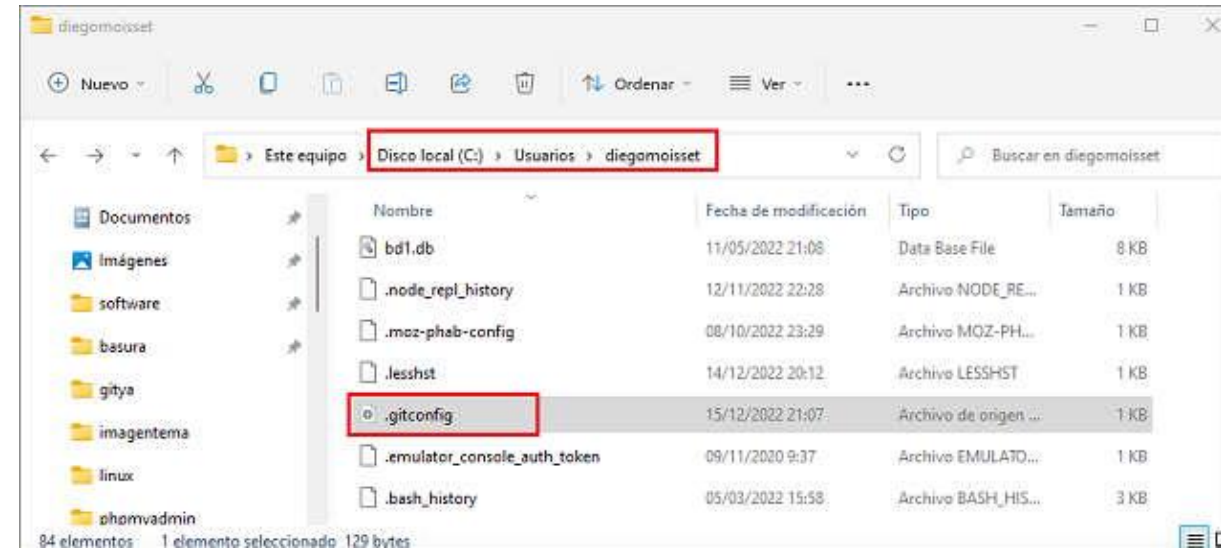
C:\Users\SENA>
```

Configuración Global Git



Command Prompt

```
C:\Users\SENA>git config --global user.name "Omar Gutierrez"  
C:\Users\SENA>git config --global user.email ogutierrez@sena.edu.co  
C:\Users\SENA>
```





Si queremos ver los datos de configuración que modificamos y otros que se inicializaron por defecto podemos nuevamente ejecutar el comando 'config' pasando el parámetro --list:

git config --list

CA Select Command Prompt

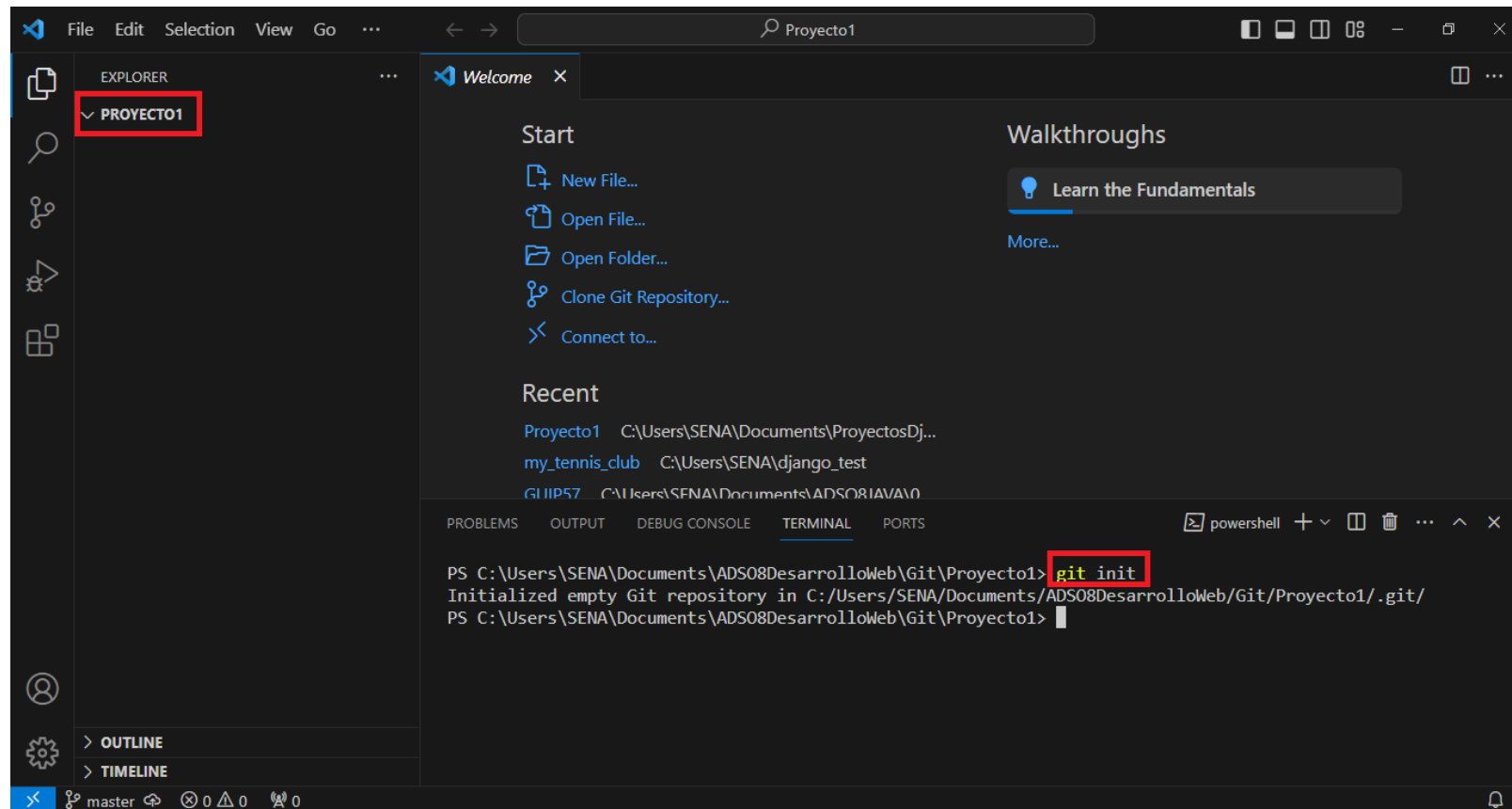
```
C:\Users\SENA>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
core.editor="C:\Users\SENA\AppData\Local\Programs\Microsoft VS Code\bin\code" --wait
user.name=Omar Gutierrez
user.email=ogutierrez@sena.edu.co

C:\Users\SENA>
```

Creando un proyecto desde cero y su respectivo repositorio

- Crearemos un directorio llamado 'proyecto1' y luego desde la línea de comando posicionados en dicho directorio ejecutaremos el comando 'init' de Git:

git init

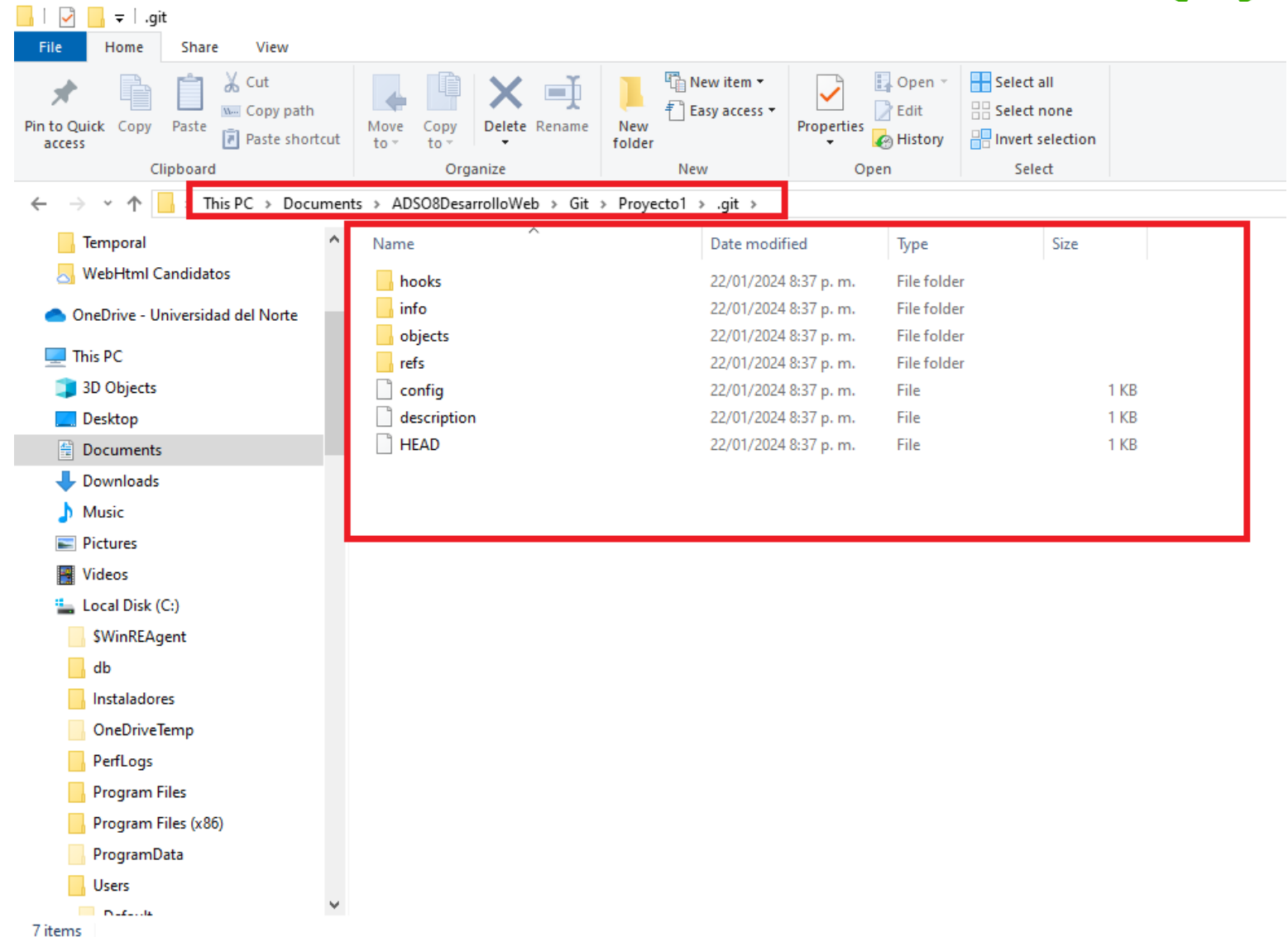


The screenshot shows the Visual Studio Code interface with a dark theme. In the Explorer sidebar on the left, a folder named 'PROYECTO1' is highlighted with a red box. The main editor area displays the 'Welcome' page with options like 'New File...', 'Open File...', and 'Clone Git Repository...'. Below this, the 'Recent' section lists several projects. At the bottom, the 'TERMINAL' panel is active, showing a PowerShell session. The command 'git init' is entered and highlighted with a red box. The output of the command is displayed below it: 'Initialized empty Git repository in C:/Users/SENA/Documents/ADS08DesarrolloWeb/Git/Proyecto1/.git/'.

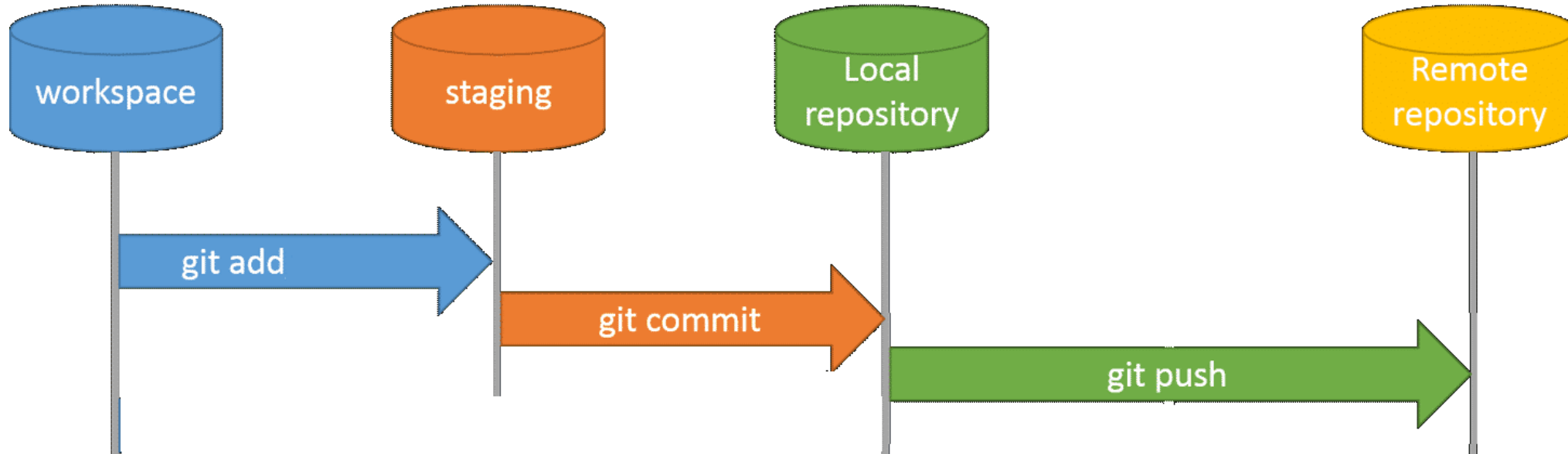
```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git init
Initialized empty Git repository in C:/Users/SENA/Documents/ADS08DesarrolloWeb/Git/Proyecto1/.git/
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Creando un proyecto desde cero y su respectivo repositorio

- Luego de ejecutar este comando en apariencia no se muestra nada en Visual Studio Code (pero si nos informa que se ha creado el repositorio: Initialized empty Git repository in C:/Users/SENA/Documents/ADSO8DesarrolloWeb/Git/Proyecto1/.git/), pero se ha creado una carpeta oculta llamada '.git' donde el programa Git va a ir guardando todos los cambios de nuestro proyecto



Flujo de trabajo



Tu repositorio local está compuesto por tres “áreas” administrados por git. El primero es WorkSpace, el segundo es el Staging y el último es el Local repository.

La metodología más común de trabajar con Git es la siguiente:

- Creamos y modificamos una serie de archivos en tu directorio de trabajo.
- Preparamos los archivos, añadiéndolos al área de preparación.
- Confirmamos los cambios, lo que toma los archivos tal y como están en el área de preparación y almacenamos esa copia instantánea de manera permanente en el directorio de Git.

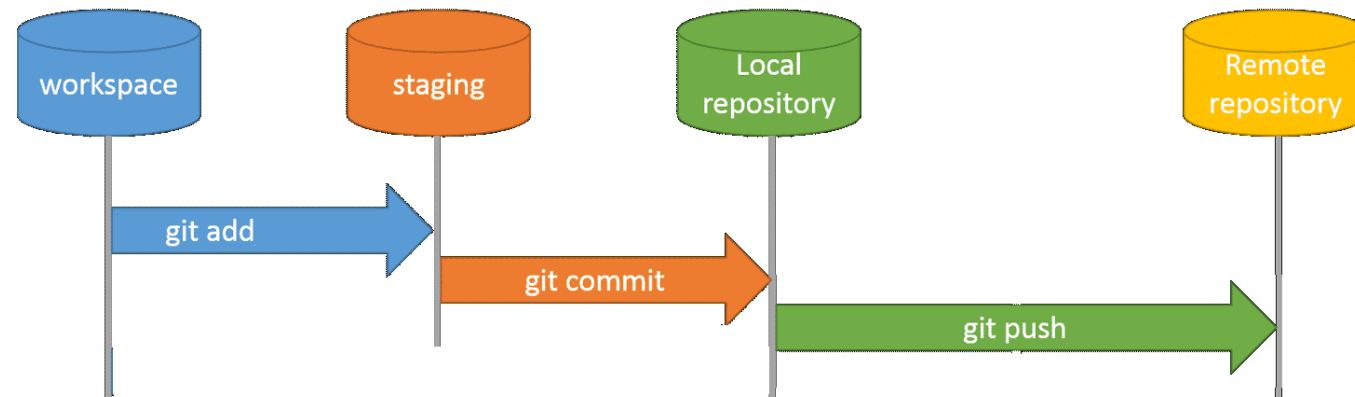
Flujo de trabajo (WorkSpace)



- Árbol de Trabajo (WorkSpace):

Definición: El WorkSpace es simplemente el directorio de tu sistema de archivos en el que estás trabajando. Contiene todos los archivos y carpetas del proyecto, incluidos los cambios que has realizado desde la última confirmación.

Puede tener archivos modificados, archivos nuevos o archivos eliminados en comparación con la última confirmación.



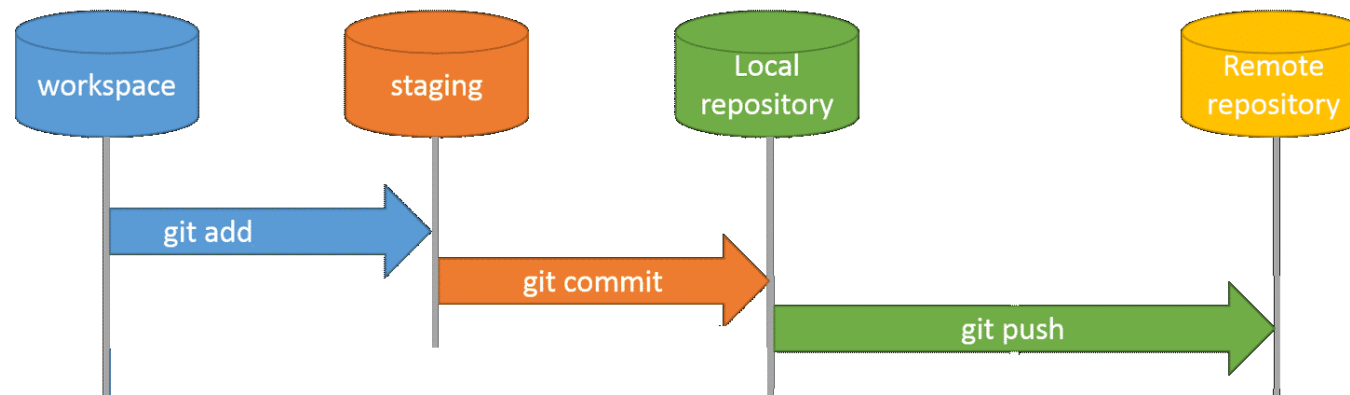
Flujo de trabajo (Staging)



- Staging (Index o Staging Area):

Definición: Conocida como zona de preparación es un área intermedia entre el workspace y el Local Repository. Es donde preparas y seleccionas los cambios que se incluirán en la próxima confirmación.

Estado: Los archivos en la zona de preparación son aquellos que han sido marcados y están listos para ser enviados al Local Repository o Head.



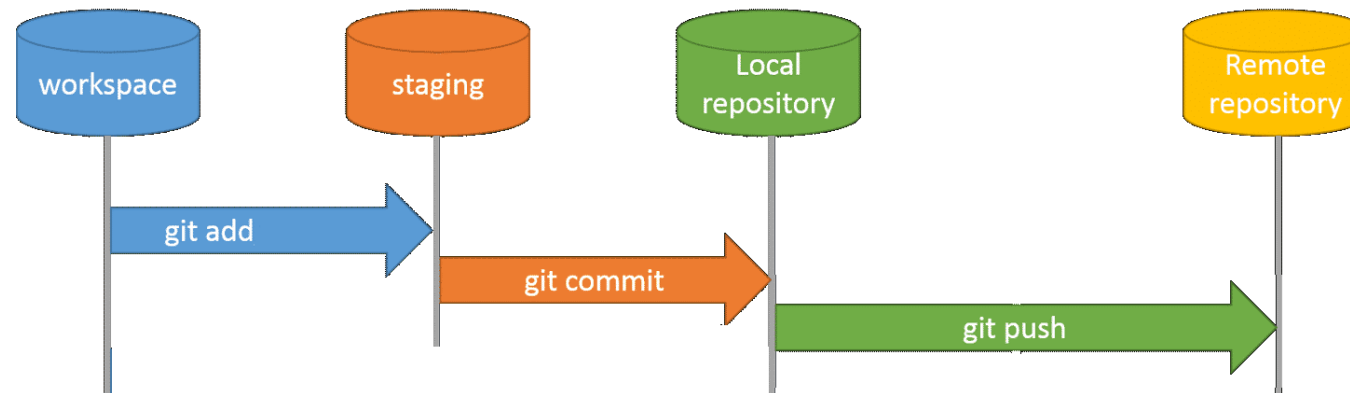
Flujo de trabajo (Local Repository)



- Local Repository (Commit History o HEAD):

Definición: El árbol de confirmación es una secuencia de instantáneas (commits) que representan los estados del proyecto en diferentes puntos en el tiempo. Cada confirmación contiene una referencia al árbol de confirmación anterior, creando una cadena histórica de cambios.

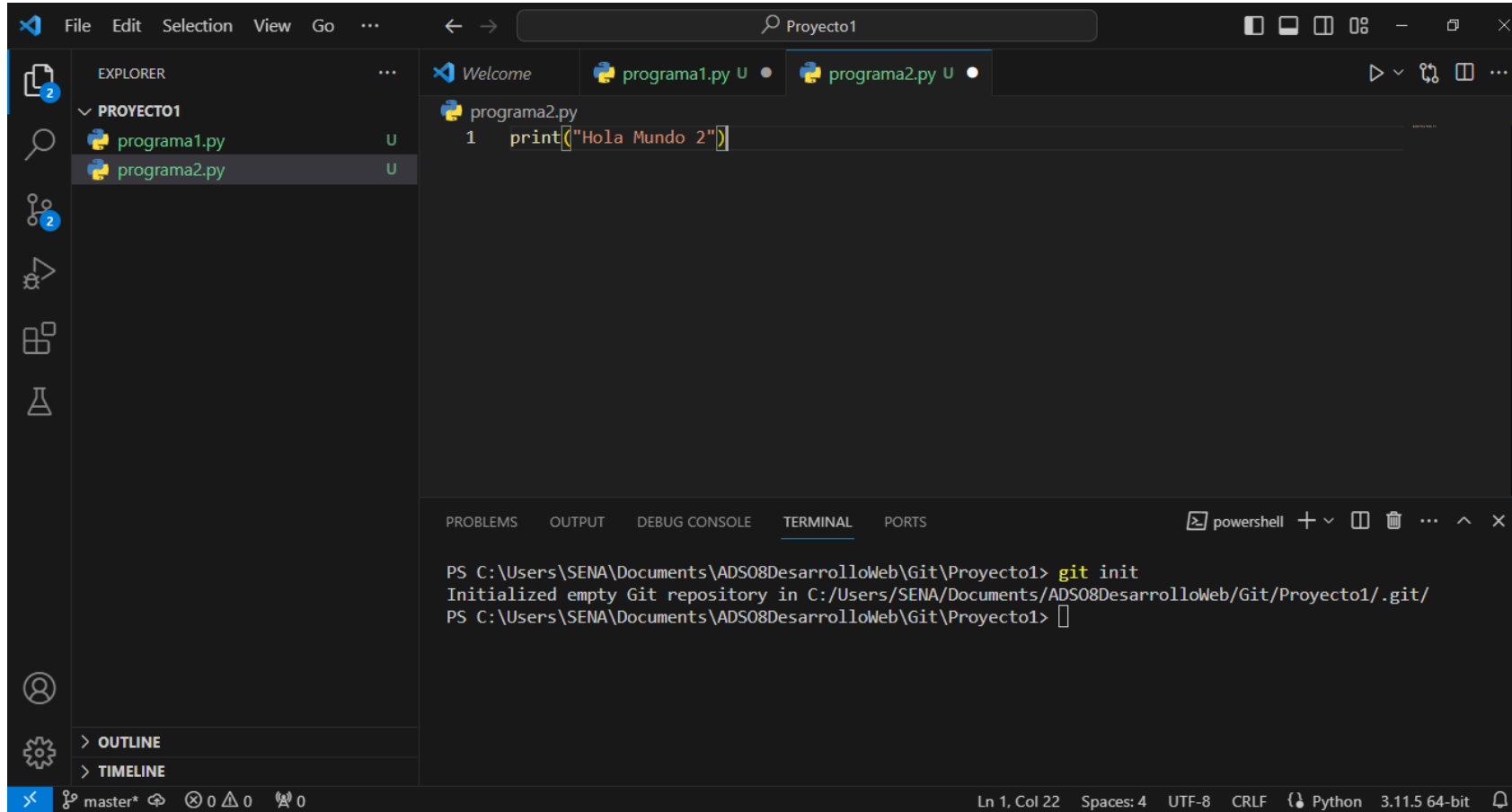
Estado: Cada commit es inmutable y representa un estado específico del proyecto en un momento dado. El último commit de la rama actual se encuentra en la referencia especial llamada HEAD.



Flujo de trabajo



- Procedamos a crear dos archivos con código fuente, por ejemplo en Python:



En este caso VS Code está integrado con git y mediante el caracter "U" (Untracked files que significa archivos sin seguimiento.) indica que el archivo está sin seguimiento

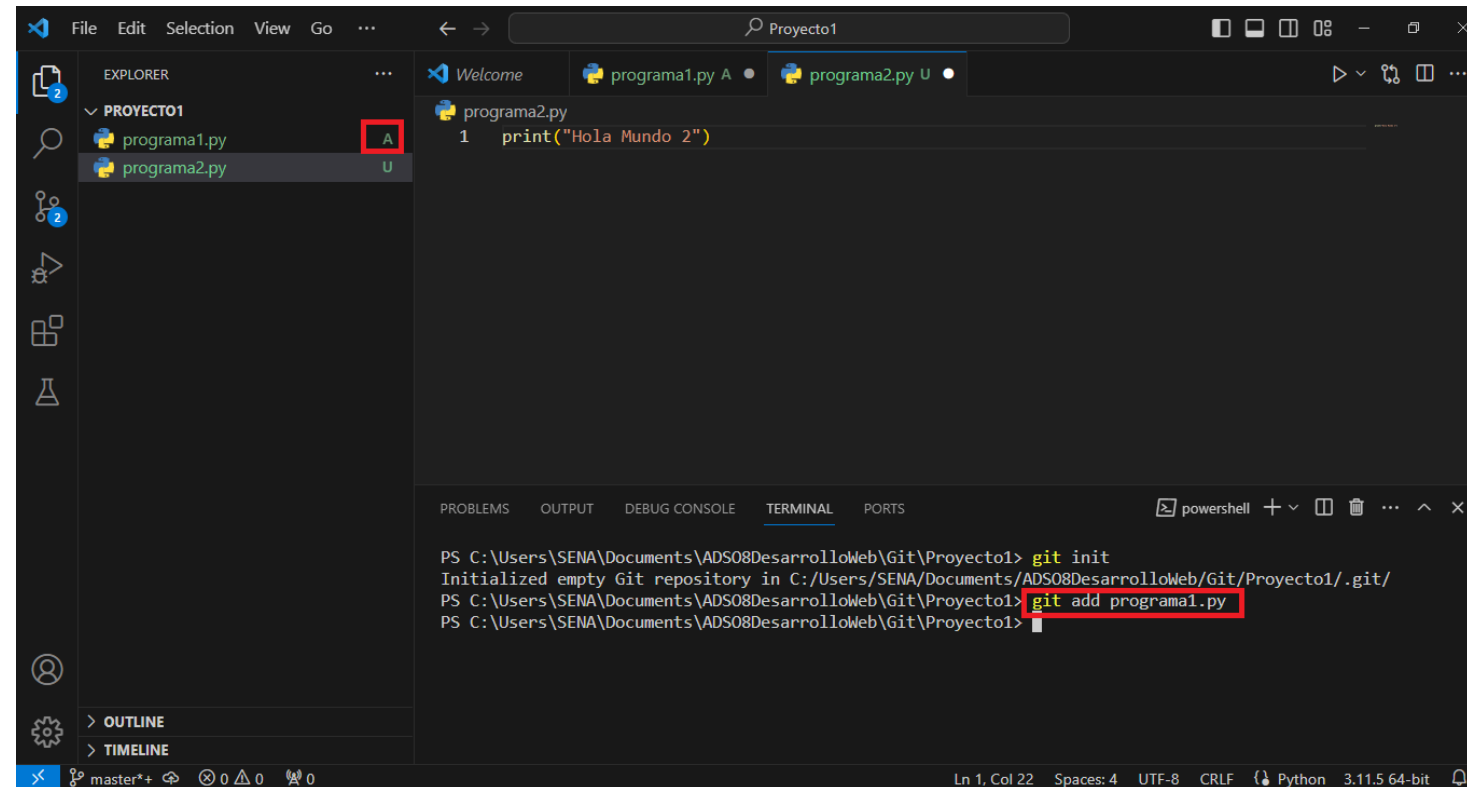
Flujo de trabajo



- Para comenzar a rastrear un archivo nuevo debemos utilizar el comando add:

git add programa1.py

Luego de esto el archivo pasa al área de preparación. Podemos ver que VS Code cambia el caracter 'U' por 'A'



Flujo de trabajo



Para confirmar el o los archivos preparados (en nuestro caso solo tenemos preparado el archivo programa1.py) debemos utilizar el comando commit.

git commit -m "primer commit para añadir programa1.py al proyecto".

pasamos el parámetro -m (que indica que añadimos un mensaje) y el mensaje propiamente dicho entre comillas. Es importante este mensaje con una descripción de los cambios que estamos haciendo en nuestro proyecto (generalmente cada vez que añadimos funcionalidades a nuestra aplicación efectuamos un commit y con un mensaje indicamos dichas funcionalidades)

A screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py'. The main editor area shows the content of 'programa2.py', which contains a single line of Python code: `print("Hola Mundo 2")`. The Terminal panel at the bottom displays the output of a series of Git commands executed in a PowerShell window. The commands are: `git init`, `git add programa1.py`, and `git commit -m "primer commit para añadir programa1.py al proyecto"`. The output shows that a new Git repository was initialized, the file was added to the staging area, and a new commit was created with the specified message. The commit hash is `2980a05`.

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git init
Initialized empty Git repository in C:/Users/SENA/Documents/ADS08DesarrolloWeb/Git/Proyecto1/.git/
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git add programa1.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git commit -m "primer commit para añadir programa1.py al proyecto"
[master (root-commit) 2980a05] primer commit para añadir programa1.py al proyecto
1 file changed, 1 insertion(+)
 create mode 100644 programa1.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Flujo de trabajo



Como podemos ver solo queda el programa2.py sin seguimiento.

Procedamos a hacer el seguimiento del archivo programa2.py, igual que hicimos con el archivo1.py:

git add programa2.py

git commit -m "se añade el archivo programa2.py al proyecto"

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git init
Initialized empty Git repository in C:/Users/SENA/Documents/ADS08DesarrolloWeb/Git/Proyecto1/.git/
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git add programa1.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git commit -m "primer commit para añadir programa1.py al proyecto"
[master (root-commit) 2980a05] primer commit para añadir programa1.py al proyecto
1 file changed, 1 insertion(+)
create mode 100644 programa1.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git add programa2.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git commit -m "se añade el archivo programa2.py al proyecto"
[master ab646a0] se añade el archivo programa2.py al proyecto
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 programa2.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Flujo de trabajo



Vamos a introducir una línea más en cada uno de los archivos de Python:

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py'. Both files have a green 'M' icon next to them, indicating they are modified. The Editor panel shows the content of 'programa2.py', which contains two lines of Python code: '1 print("Hola Mundo 2")' and '2 print("Fin")'. The line numbers 1 and 2 are highlighted in yellow. The Terminal panel at the bottom shows the output of Git commands: 'git init', 'git add programa1.py', 'git commit -m "primer commit para añadir programa1.py al proyecto"', 'git add programa2.py', and 'git commit -m "se añade el archivo programa2.py al proyecto"'. The status bar at the bottom indicates the current file is 'programa2.py' at line 2, column 13, with a UTF-8 encoding and CRLF line endings.

Flujo de trabajo



Los movemos ahora a los dos archivos al área de preparación (de una sola vez movemos todos los archivos con extensión py):

git add *.py también se podría utilizar **git add .**

Con esto ya tenemos a los archivos en el área de preparación:

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py', both marked with a green 'M' for modified. The main editor area shows the content of 'programa2.py', which contains two lines of Python code: 'print("Hola Mundo 2")' and 'print("Fin")'. The bottom panel shows the integrated terminal with a PowerShell prompt. The commands 'git add *.py' and 'git status' have been entered and are highlighted with a red rectangle. The output of 'git status' shows that both files are staged for commit. The status bar at the bottom indicates the current branch is 'master+', there are 0 commits, 0 files staged, and 0 files tracked. The file encoding is UTF-8, line endings are CRLF, and the Python interpreter is 3.11.5 64-bit.

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git add *.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   programa1.py
        modified:   programa2.py

PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Flujo de trabajo



Finalmente los confirmamos para que quede la nueva versión del proyecto mediante el comando commit:

`git commit -m "se cargan las nuevas versiones de los archivos programa1.py y programa2.py"`

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py'. The editor window shows the content of 'programa2.py', which contains two lines of Python code: `print("Hola Mundo 2")` and `print("Fin")`. The Terminal panel at the bottom shows the execution of the `git commit -m "se cargan las nuevas versiones de los archivos programa1.py y programa2.py"` command. The output of the command is displayed, showing the commit message and the files changed. The command and its output are highlighted with red boxes.

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git commit -m "se cargan las nuevas versiones de los archivos programa1.py y programa2.py"
[master 2a5889e] se cargan las nuevas versiones de los archivos programa1.py y programa2.py
2 files changed, 4 insertions(+), 1 deletion(-)
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Flujo de trabajo (Saltar área de preparación)



En un proyecto muy sencillo si queremos podemos saltar el área de preparación agregando un parámetro al comando commit.

Continuando con el proyecto2 del concepto anterior procedamos a modificar los dos archivos programa1.py y programa2.py, borrando la última línea de cada archivo (los archivos deben quedar con el caracter 'M' de modificado):

A screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files, 'programa1.py' and 'programa2.py', both marked with a red 'M' icon indicating they are modified. The main editor area shows the 'programa1.py' file with two lines of code: '1 print("Hola Mundo 1")' and '2'. The Terminal panel at the bottom shows a PowerShell session where the command 'git commit -m "se cargan las nuevas versiones de los archivos programa1.py y programa2.py"' has been executed. The output of the command is displayed, showing the commit hash '[master 2a5889e]' and the changes: '2 files changed, 4 insertions(+), 1 deletion(-)'. The status bar at the bottom indicates the current file is 'programa1.py' on the 'master' branch, with 0 errors and 0 warnings.

Flujo de trabajo (Saltar área de preparación)



Hemos borrado el print de cada uno de los archivos. Según el flujo que vimos en el concepto anterior primero los debíamos agregar a los archivos al área de preparación y finalmente mediante el comando commit actualizar los cambios en el repositorio. Podemos saltar el área de preparación agregando un parámetro (-a) al comando commit:

git commit -a -m "se borraron la última línea de los archivos programa1.py y programa2.py"

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py'. The Editor panel shows 'programa1.py' with the following code:

```
1 print("Hola Mundo 1")
2
```

The Terminal panel at the bottom shows the execution of the command `git commit -a -m "se borraron la última línea de los archivos programa1.py y programa2.py"`. The output of the command is displayed in red text:

```
[master 1455cab] se borraron la última línea de los archivos programa1.py y programa2.py
2 files changed, 2 deletions(-)
```

The status bar at the bottom indicates the current branch is 'master' and shows 0 changes.

Eliminar archivos y cambiar sus nombres



Cuando hacemos el seguimiento de archivos de un proyecto con git, tenemos que administrar e informar a git los archivos que quitaremos del proyecto o renombraremos. Mediante el comando 'rm' de git procedemos a eliminar un archivo. Por ejemplo si queremos eliminar el archivo programa1.py para que git lo deje de seguir:

git rm programa1.py

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'PROYECTO1' with two files: 'programa1.py' and 'programa2.py'. The main editor area shows the content of 'programa1.py', which contains three lines of Python code:

```
1 print("Hola Mundo 1")
2 print("mañana")
3 print(" ")
```

. The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt. The command `git rm programa1.py` has been entered and executed, resulting in the output `rm 'programa1.py'`. The status bar at the bottom indicates the current file is 'programa1.py' at line 3, column 11, using UTF-8 encoding and CRLF line endings.

El archivo se elimina del área de trabajo y queda en el área de preparación hasta que se ejecute el próximo commit git.

Eliminar archivos y cambiar sus nombres



De forma similar para cambiar el nombre de un archivo que estamos siguiendo con git debemos utilizar un comando:

git mv programa2.py programafinal.py

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project named 'PROYECTO1' with a file 'programafinal.py' highlighted. The main editor area shows a Python file 'programa1.py' with the following code:

```
1 print("Hola Mundo 1")
2 print("mañana")
3 print(" ")
4
```

The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt. The following commands have been executed:

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git rm programa1.py
rm 'programa1.py'
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git mv programa2.py programafinal.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

The status bar at the bottom indicates the current file is 'Ln 3, Col 11', using 'Spaces: 4', 'UTF-8' encoding, 'CRLF' line endings, and is a 'Python 3.11.5 64-bit' file.

Eliminar archivos y cambiar sus nombres



Tener en cuenta que tanto el comando `rm` y `mv` requieren un commit para que queden confirmados en el repositorio el borrado y la modificación del nombre de archivo, con `git status` podemos ver las dos últimas acciones realizadas.

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with a file 'programafinal.py' marked with a red 'R' (renamed). The Editor panel shows two open files: 'programa1.py' and 'programa2.py', both marked with a red 'R'. The 'programa1.py' file contains the following code:

```
1 print("Hola Mundo 1")
2 print("mañana")
3 print(" ")
4
```

The Terminal panel at the bottom shows the output of the `git status` command:

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    programa1.py
        renamed:    programa2.py -> programafinal.py

PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

The status bar at the bottom indicates the current branch is 'master+', there are 0 deletions and 0 additions, and the editor is using Python 3.11.5 64-bit.

Eliminar archivos y cambiar sus nombres



Procedemos a aplicar el commit

`git commit -m "se borro el archivo programa1.py y se renombre el archivo programa2.py por programafinal.py"`

A screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows a project named "PROYECTO1" with a file named "programafinal.py". The main editor area shows the content of "programa1.py", which contains three print statements: "Hola Mundo 1", "mañana", and an empty string. The bottom panel shows the TERMINAL with a PowerShell session. The command executed is `git commit -m "se borro el archivo programa1.py y se renombre el archivo programa2.py por programafinal.py"`. The output shows the commit was successful, with 2 files changed and 3 deletions. The status bar at the bottom indicates the current branch is "master" and the file encoding is UTF-8.

```
File Edit Selection View Go ... Proyecto1
EXPLORER
  PROYECTO1
    programafinal.py
programa1.py
1 print("Hola Mundo 1")
2 print("mañana")
3 print(" ")
4
TERMINAL
powershell
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git commit -m "se borro el archivo programa1.py y se renombre el archivo programa2.py por programafinal.py"
[master 536662e] se borro el archivo programa1.py y se renombre el archivo programa2.py por programafinal.py
2 files changed, 3 deletions(-)
delete mode 100644 programa1.py
rename programa2.py => programafinal.py (100%)
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```


Consultar commits realizados



Luego de hacer varias confirmaciones (commit), veremos como conocer cada una de ellas. El comando para hacer esto es log. Una vez mostrado los primeros commit se debe teclear Enter para ver los demás, presiona **q** para salir del commit,

Se nos muestra una lista de las confirmaciones (commit) hechas sobre el repositorio en orden cronológico inverso, es decir el último commit es el que primero se ve.

Es importante notar el porqué del mensaje con una descripción clara del objetivo de cada commit, en el momento que lo hicimos.

De cada confirmación se muestra un valor único por cada commit, el nombre y dirección de correo del autor, la fecha y el mensaje de confirmación.

```
On branch master
Changes to be committed:
  PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git log
  commit 536662e7e3e1be4f5259f3fa910b105ea685ecd9 (HEAD -> master)
  Author: Omar Gutierrez <ogutierrez@sena.edu.co>
  Date: Wed Jan 24 08:07:11 2024 -0500

    se borro el archivo programa1.py y se renombre el archivo programa2.py por programafinal.py

  commit 23422eccc0e8b08ab65be5f4aa5956ad034edb6e
  Author: Omar Gutierrez <ogutierrez@sena.edu.co>
  Date: Tue Jan 23 09:36:21 2024 -0500

    probando add .otra vez
```

Consultar commits realizados



El comando log proporciona gran cantidad de opciones. Veamos algunas de las más usadas:

git log -n : Muestra la cantidad de commits indicada en n ej: git log -3 muestra los últimos tres,

git log -p -n : El parámetro -p, que muestra las diferencias introducidas en cada confirmación y -n muestra la cantidad de commits a revisar

git log --pretty=oneline: oneline imprime cada confirmación en una única línea, lo que puede resultar útil si estás analizando gran cantidad de confirmaciones

git log --since="2022-12-01" --until="2022-12-16": consultar commits por rango de fechas.

git log --author="Omar Gutierrez": consultar todos los commit de un determinado autor

git log -n --stat: ver algunos datos estadísticos

Volver al último commit



Cuando hacemos cambios en un archivo en nuestro directorio de trabajo, podemos volver y retrotraer a un estado anterior que tengamos confirmado en nuestro repositorio.

Veamos los pasos para deshacer un cambio en un archivo, procedamos a crear un archivo en el proyecto1 llamado programa3.py e ingresemos un mensaje "Hola Mundo", luego lo añadimos a git y luego hagamos un commit directo sin pasar por el área de preparación:

git commit -a -m "se agrega el archivo programa3.py"

A screenshot of the Visual Studio Code interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa3.py' and 'programafinal.py'. The editor window shows 'programa3.py' with the content:

```
1 print ("Hola Mundo")
```

. The Terminal panel at the bottom shows the execution of the command `git commit -a -m "se agrega el archivo programa3.py"` in a PowerShell session. The output of the command is:

```
[master d23de39] se agrega el archivo programa3.py
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 programa3.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Volver al último commit



Por un momento procedemos a modificar el archivo programa3.py con el siguiente contenido (es decir modificamos el mensaje, escribiéndolo en inglés):

A screenshot of the Visual Studio Code editor interface. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa3.py' (marked with a green 'M' for modified) and 'programafinal.py'. The main editor area shows the content of 'programa3.py', which is a single line of Python code: `print ("Hello world")`. The bottom panel shows the 'TERMINAL' tab with a PowerShell prompt at the directory `C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>`. The status bar at the bottom indicates the current file is 'programa3.py' at line 1, column 19, with 4 spaces, using UTF-8 encoding and CRLF line endings, and is running Python 3.11.5 64-bit.

```
File Edit Selection View Go ... Proyecto1
EXPLORER
PROYECTO1
  programa3.py M
  programafinal.py
programa3.py
1 print ("Hello world")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
master* 0 0 0
Ln 1, Col 19 Spaces: 4 UTF-8 CRLF Python 3.11.5 64-bit
```

Volver al último commit



Seguidamente nos arrepentimos del cambio y queremos volver al último estado que habíamos confirmado (commit) en nuestro repositorio. Para volver atrás debemos ejecutar el siguiente comando:

git checkout -- programa3.py

The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer panel on the left shows a project named 'PROYECTO1' with two files: 'programa3.py' and 'programafinal.py'. The main editor area has two tabs: 'programafinal.py' and 'programa3.py'. The 'programa3.py' tab is active, showing a single line of code: `1 print ("Hola Mundo")`. Below the editor, the TERMINAL panel is open, showing a PowerShell session. The command `git checkout -- programa3.py` has been entered and executed. The status bar at the bottom indicates the current branch is 'master' and the file encoding is UTF-8.

```
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1> git checkout -- programa3.py
PS C:\Users\SENA\Documents\ADS08DesarrolloWeb\Git\Proyecto1>
```

Este comando reemplaza los cambios en tu directorio de trabajo con el último contenido de HEAD. Los cambios que ya han sido agregados al Index, así como también los nuevos archivos, se mantendrán sin cambio.

Volver a un commit en específico



Para volver a un commit específico en Git, puedes usar el comando `git checkout` seguido del hash del commit al que deseas regresar:

1. Encuentra el Hash del Commit:

Utiliza el comando `git log` para ver el historial de commits y encontrar el hash del commit al que deseas volver.

`git log`

Copia el hash del commit que te interesa.

2. Utiliza `git checkout`

Puedes usar el siguiente comando para volver a un commit específico.

`git checkout <hash_del_commit>`



GRACIAS

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682



www.sena.edu.co