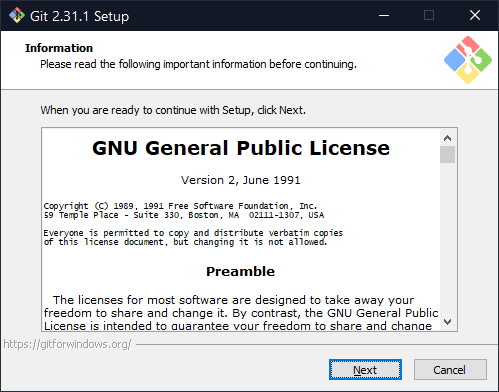
Instalación de Git.

Descargamos el instalador desde su página:

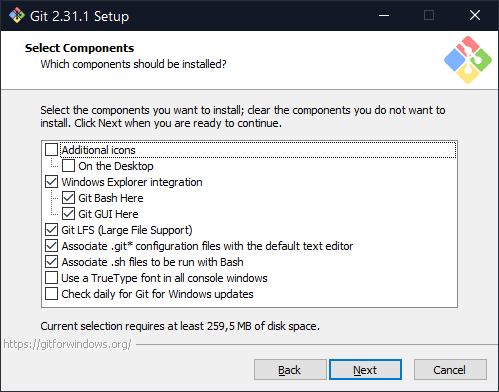
<https://git-scm.com>

Ejecutamos el archivo de instalación de la última versión (abril 2021: Git-2.31.1-64-bit.exe).

Aceptamos la licencia.



Si es necesario, modificamos el directorio de instalación, así como los componentes a instalar.

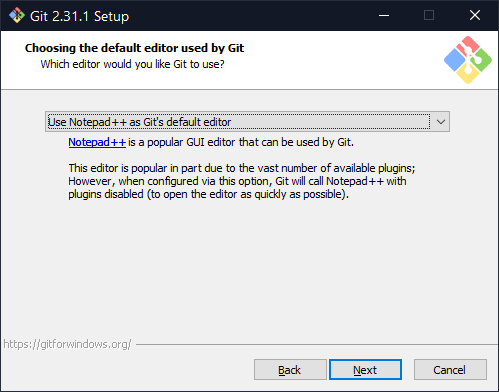


En particular interesa activar la integración con el explorador.

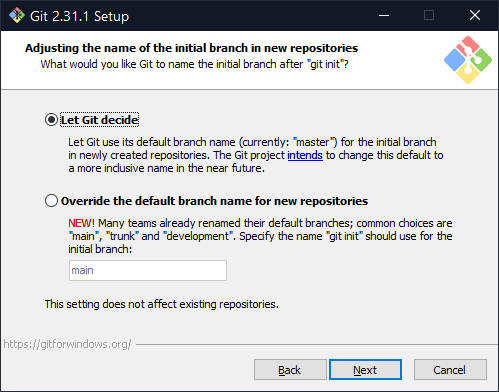
Seleccionamos la carpeta del menú de inicio o indicamos que no se cree.

Seleccionamos el editor por defecto.

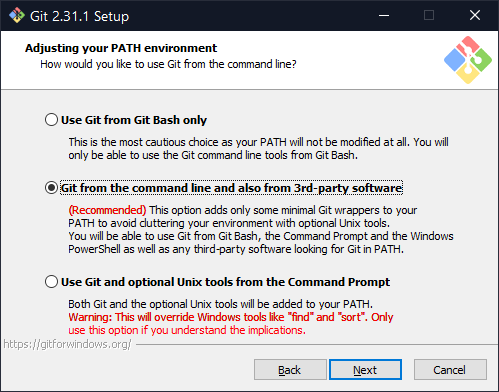
Podemos, por ejemplo, seleccionar el Notepad++ que ya hemos utilizado anteriormente. O bien utilizar otra de las posibilidades, vim es una muy buena opción que además pesa muy poco aunque es poco intuitivo si bien es el editor utilizado en las primeras versiones de git, otra posibilidad también muy recomendable es el editor nano, también de muy poco peso pero mucho más intuitivo, de hecho podemos usar cualquier editor incluso el editor de visual studio code, las opciones son ilimitadas, en general podéis elegir el editor con el que os encontréis más a gusto.



A continuación podemos decidir si git usará ‘master’ para indicar la rama principal o bien utilizará otro nombre para designarla.

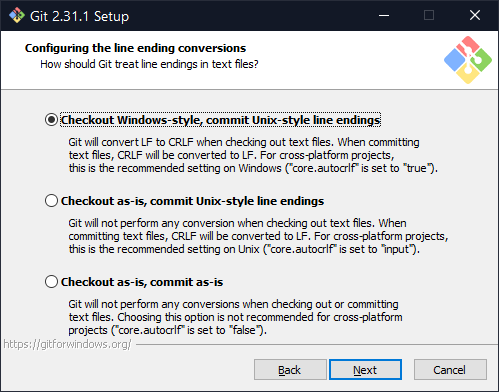


A continuación decidimos cual es el modo de utilización de git, el modo más conservativo es utilizar git únicamente desde su bash, aunque si queremos utilizar otras herramientas o alguno de los GUI disponibles es recomendable configurar el uso desde las consolas del sistema (opción por defecto).



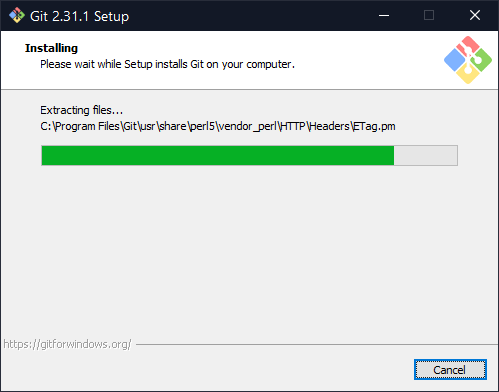
A continuación, indicamos el procedimiento para crear conexiones seguras en las conexiones, mantenemos la opción por defecto (OpenSSL).

La siguiente ventana nos pregunta como vamos a realizar los saltos de línea. Si todos los usuarios utilizan el mismo sistema operativo, se puede utilizar la última opción (Checkout as-is, commit as-is), en caso contrario es recomendable la primera para maximizar la compatibilidad.



Luego nos pregunta por la consola que se deberá utilizar con el git bash, por defecto se utiliza MinTTY.

En las siguientes ventanas dejamos las opciones por defecto y se procede con la instalación.



Con esto finalizamos la instalación de Git.

Comandos de Git.

Desde cualquier carpeta de Windows podemos lanzar con el menú contextual del ratón el Git Bash, se abre una ventana de consola en el directorio correspondiente a dicha carpeta que permite ejecutar los comandos de Git.

Lo primero que debemos realizar es crear un repositorio, con el comando git init hacemos que la carpeta actual pase a albergar un repositorio de git (con esa orden se crea una carpeta oculta .git que incluye los metadatos que utiliza git para realizar el seguimiento de los cambios realizados en la carpeta).

Si vamos a trabajar con un repositorio remoto ya existente, deberemos clonar dicho repositorio:

git clone https//github.com/jablafer/pruebagit

Se creará una nueva carpeta ‘pruebagit’ en la cual se descargarán todos los archivos de la rama principal del proyecto pruebagit, además se creará el correspondiente directorio .git con los metadatos existentes en el repositorio remoto (se realiza un duplicado completo del repositorio del proyecto).

Algunos comandos básicos para uso en git:

CONFIGURACIÓN BÁSICA DE USUARIO PARA REPOSITORIOS LOCALES:

Establece el nombre que desea esté anexado a sus transacciones de commit:

$ git config --global user.name "[name]"

Establece el e-mail que desea esté anexado a sus transacciones de commit:

$ git config --global user.email "[email address]"

Habilita la útil colorización del producto de la línea de comando:

$ git config --global color.ui auto

CREAR REPOSITORIOS U OBTENERLOS DESDE UNA URL:

Crea un nuevo repositorio local con el nombre especificado:

$ git init [project-name]

Descarga un proyecto y toda su historia de versión:

$ git clone [url]

EFECTUAR CAMBIOS Y TRANSACCIONES:

Enumera todos los archivos nuevos o modificados que se deben confirmar

$ git status

Toma una instantánea del archivo para preparar la versión

$ git add [file]

Mueve el archivo del área de espera, pero preserva su contenido

$ git reset [file]

Muestra las diferencias de archivos que no se han enviado aún al área de espera

$ git diff

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

$ git diff --staged

Registra las instantáneas del archivo permanentemente enel historial de versión

$ git commit -m "[descriptive message]"

CAMBIOS GRUPALES Y FUSIONES DE RAMAS:

Enumera todas las ramas en el repositorio actual:

$ git branch

Crea una nueva rama:

$ git branch [branch-name]

Cambia a la rama especificada y actualiza el directorio activo:

$ git checkout [branch-name]

Combina el historial de la rama especificada con la rama actual:

$ git merge [branch]

Borra la rama especificada:

$ git branch -d [branch-name]

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN:

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

$ git rm [file]

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

$ git rm --cached [file]

Cambia el nombre del archivo y lo prepara para commit

$ git mv [file-original] [file-renamed]

SUPRIMIR TRACKING:

Un archivo de texto llamado .gitignore suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

\*.log

build/

temp-\*

Enumera todos los archivos ignorados en este proyecto

$ git ls-files --other --ignored --exclude-standard

GUARDAR FRAGMENTOS (ALMACENAR Y RESTAURAR CAMBIOS INCOMPLETOS):

Almacena temporalmente todos los archivos tracked modificados

$ git stash

Restaura los archivos guardados más recientemente

$ git stash pop

Enumera todos los sets de cambios en guardado rápido

$ git stash list

Elimina el set de cambios en guardado rápido más reciente

$ git stash drop

REPASAR HISTORIAL:

Enumera el historial de la versión para la rama actual

$ git log

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

$ git log --follow [file]

Muestra las diferencias de contenido entre dos ramas

$ git diff [first-branch]...[second-branch]

Produce metadatos y cambios de contenido del commit especificado

$ git show [commit]

REHACER COMMITS:

Deshace todos los commits después de [commit], preservando los cambios localmente

$ git reset [commit]

Desecha todo el historial y regresa al commit especificado

$ git reset --hard [commit]

SINCRONIZAR CAMBIOS:

Descarga todo el historial del marcador del repositorio

$ git fetch [bookmark]

Combina la rama del marcador con la rama local actual

$ git merge [bookmark]/[branch]

Carga todos los commits de la rama local al GitHub

$ git push [alias] [branch]

Descarga el historial del marcador e incorpora cambios

$ git pull

Tenéis un PDF a modo de hoja de referencia rápida con el resumen de comandos anterior en: <https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf>

Para una referencia más completa (de lo que hemos visto en clase) de git os remito al libro progit que podéis descargar o acceder en línea a través de la dirección <https://git-scm.com/book/es/v2>

Algunos GUI para git.

Sourcetree (<https://www.sourcetreeapp.com/>)

Se trata de un cliente gratuito para sistemas Windows y Mac. Es sencilla pero también potente, lo que la hace una herramienta útil tanto para principiantes como para usuarios con experiencia.

Tortoise Git (<https://tortoisegit.org/download/>)

Se trata de un cliente de software libre y código abierto. Tortoise es un referente en el uso de control de versiones, si bien no es tan extensa como otras herramientas ni tan intuitiva, no deja de ser un referente.

GitKraken (<https://www.gitkraken.com/download>)

Es un cliente con un interface muy intuitiva y agradable, es gratuita para usos no comerciales y enmarca algunas de las tendencias actuales en el uso de herramientas de programación.

SmartGit (<http://www.syntevo.com/smartgit/download>)

Se trata de otra herramienta comercial, también con la posibilidad de uso gratuito en proyectos no comerciales, en este caso el enfoque es en maximizar la facilidad de uso, añadiendo cierta inteligencia a dicho interface.

GitHub Desktop (<https://desktop.github.com/>)

Si el repositorio remoto es GitHub puede que GitHub Desktop sea la herramienta más útil.

Se trata de un cliente gratuito y de código abierto enfocado al uso de git con GitHub.