GIT

GIT
INTRODUCCIÓN

Sistemas de Control de Versiones

En el proceso de desarrollo de software es un requisito casi indispensable mantener un registro de los cambios que se realizan sobre el código fuente a lo largo del tiempo. Es debido a esto que cobran importancia los sistemas de control de versiones.

EL CONCEPTO DE VERSIÓN

(También llamado revisión o edición) de un proyecto (código fuente) hace referencia al estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Los sistemas de control de versiones distribuidos permiten en cambio continuar el trabajo aún cuando el repositorio de referencia no está disponible. En estos sistemas los clientes no solo descargan la última copia del código, sino que se replica completamente el repositorio con los cambios históricos (versiones). De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo.

VENTAJAS

- HISTORIAL DE CAMBIOS
- CREACIÓN Y FUSIÓN DE RAMAS
- TRAZABILIDAD DE CAMBIOS EN SOFTWARE

GIT

Es un proyecto de código abierto maduro y con un activo mantenimiento desarrollado originalmente por Linus Torvalds.

Este sistema de control de versiones distribuido que funciona bajo cualquier plataforma (Windows, MacOS, Linux, etc) y está integrado en una amplia variedad de entornos de desarrollo (IDEs).

CARACTERÍSTICAS

- RENDIMIENTO
- SEGURIDAD
- FLEXIBILIDAD

GIT
INTRODUCCIÓN

Sistemas de Control de Versiones

En el proceso de desarrollo de software es un requisito casi indispensable mantener un registro de los cambios que se realizan sobre el código fuente a lo largo del tiempo. Es debido a esto que cobran importancia los sistemas de control de versiones.

EL CONCEPTO DE VERSIÓN

(También llamado revisión o edición) de un proyecto (código fuente) hace referencia al estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

Los sistemas de control de versiones distribuidos permiten en cambio continuar el trabajo aún cuando el repositorio de referencia no está disponible. En estos sistemas los clientes no solo descargan la última copia del código, sino que se replica completamente el repositorio con los cambios históricos (versiones). De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios disponibles en los clientes puede ser copiado al servidor con el fin de restaurarlo.

VENTAJAS

- **HISTORIAL DE CAMBIOS**
- **CREACIÓN Y FUSIÓN DE RAMAS**
- **TRAZABILIDAD DE CAMBIOS EN SOFTWARE**

GIT

Es un proyecto de código abierto maduro y con un activo mantenimiento desarrollado originalmente por Linus Torvalds.

Este sistema de control de versiones distribuido que funciona bajo cualquier plataforma (Windows, MacOS, Linux, etc) y está integrado en una amplia variedad de entornos de desarrollo (IDEs).

CARACTERÍSTICAS

- **RENDIMIENTO**
- **SEGURIDAD**
- **FLEXIBILIDAD**

Este sistema presenta una arquitectura distribuida, es decir que, cada desarrollador posee una copia del trabajo en un repositorio local donde puede albergar el historial completo de todos los cambios y, mediante determinados comandos, realiza sincronizaciones al repositorio remoto.

Rendimiento: Los algoritmos implementados en Git aprovechan el profundo conocimiento sobre los atributos comunes de los auténticos árboles de archivos de código fuente, cómo suelen modificarse con el paso del tiempo y cuáles son los patrones de acceso. El formato de objeto de los archivos del repositorio de Git emplea una combinación de codificación delta (que almacena las diferencias de contenido) y compresión, y guarda explícitamente el contenido de los directorios y los objetos de metadatos de las versiones.

Seguridad: la principal prioridad es conservar la integridad del código fuente gestionado. El contenido de los archivos y las verdaderas relaciones entre estos y los directorios, las versiones, las etiquetas y las confirmaciones, están protegidos con un algoritmo de hash criptográficamente seguro llamado "SHA1". De este modo, se salvaguarda el código y el historial de cambios frente a las modificaciones accidentales y maliciosas, y se garantiza que el historial sea totalmente trazable.

Flexibilidad: es flexible en varios aspectos, en la capacidad para varios tipos de flujos de trabajo de desarrollo no lineal, en su eficiencia en proyectos tanto grandes como pequeños y en su compatibilidad con numerosos sistemas y protocolos. Se ha ideado para posibilitar la ramificación y el etiquetado como procesos de primera importancia y las operaciones que afectan a las ramas y las etiquetas (como la fusión o la reversión) también se almacenan en el historial de cambios.

ÁREAS Y ESTADOS

Para trabajar con git es fundamental entender los estados por los que pueden pasar los archivos durante todo el flujo de desarrollo.

COMANDOS BÁSICOS

Es un proyecto de código abierto maduro y con un activo mantenimiento desarrollado originalmente por Linus Torvalds.

Este sistema de control de versiones distribuido que funciona bajo cualquier plataforma (Windows, MacOS, Linux, etc) y está integrado en una amplia variedad de entornos de desarrollo (IDEs).

Ver gráfico

SECCIONES FUNDAMENTALES

- **DIRECTORIO DE TRABAJO**
- **ÁREA DE PREPARACIÓN**
- **DIRECTORIO DE GIT**
- **REPOSITORIO REMOTO**

`git init`

Es el comando para inicializar un directorio como repositorio Git, se ejecuta dentro del directorio del proyecto y, como resultado, crea un subdirectorio .git que contiene todos los archivos para poder realizar el seguimiento de los cambios, etiquetas, etc.

`git add <file>`

Luego de la creación, modificación o eliminación de un archivo, los cambios quedan únicamente en el área de trabajo, por lo tanto es necesario pasarlos al área de preparación mediante el uso del comando git add, para que sea incluido dentro de la siguiente confirmación (commit).

`git status`

Es un comando que permite conocer en qué estado se encuentran los archivos

`git commit`

Con este comando se confirman todos los cambios registrados en el área de preparación, o lo que es lo mismo, se pasan los cambios al repositorio local.

`git push`

Es el comando que se utiliza para enviar todas las confirmaciones registradas en el repositorio local a un repositorio remoto.

`git pull`

Funciona al inverso de git push, trayendo todos los cambios al repositorio local, pero también dejándolos disponibles directamente para su modificación o revisión en el área de trabajo. Es importante mencionar que se utiliza cuando ya se tiene un repositorio local vinculado a uno remoto, al igual que con el comando git push.

`git clone`

En el caso de necesitar “bajar” un repositorio remoto de algún proyecto ya existente se puede ejecutar este comando. Genera un directorio (con el nombre del repositorio o uno especificado explícitamente) que contiene todo lo propio al proyecto, además del subdirectorio .git necesario para poder gestionar los cambios y todo lo pertinente al repositorio Git.

DIRECTORIO DE TRABAJO
(WORKING DIRECTORY)

Es una copia de una versión del proyecto, Son archivos sacados de la base de datos comprimida y se colocan en el disco para poder ser usados o modificados.

DIRECTORIO DE GIT
(LOCAL REPOSITORY)

Es el lugar donde se almacenan los metadatos y la base de datos de objetos del proyecto. Es lo que se copia cuando se clona un repositorio desde otra fuente.

REPOSITORIO REMOTO
(REMOTE REPOSITORY)

Es el repositorio que se encuentra en un servidor remoto y con el que eventualmente se sincronizan los trabajos entre los diferentes integrantes del equipo.

Sistemas de Control de Versiones

En el proceso de desarrollo de software es un requisito casi indispensable mantener un registro de los cambios que se realizan sobre el código fuente a lo largo del tiempo. Es debido a esto que cobran importancia los sistemas de control de versiones.

Estos sistemas son herramientas que permiten realizar un seguimiento de los cambios y también permitir proteger el código de errores humanos accidentales. Además, un sistema de control facilita el trabajo en equipo a la hora de desarrollar software, ya que mientras un integrante trabaja en alguna función específica, otro podría estar trabajando en alguna corrección de errores o bien en otra función, para luego integrar las soluciones y realizar una sincronización del trabajo de cada uno.

El uso de un sistema de control de versiones tiene tres ventajas principales:

1. Gracias al historial de cambios se puede saber el autor, la fecha y notas escritas sobre los cambios realizados. También permite volver a versiones anteriores para ayudar a analizar causas raíces de errores y es crucial cuando hay que solucionar problemas de versiones anteriores.
2. Creación y fusión de ramas. Al tener varios integrantes del equipo trabajando al mismo tiempo, cada uno en una tarea diferente, pueden beneficiarse de tener flujos de trabajo independientes. Posteriormente se pueden fusionar estos flujos de trabajos o ramas a una principal. Los sistemas de control de versiones tienen mecanismos para identificar que los cambios entre ramas no entren en conflicto para asegurar la funcionalidad y la integración.
3. Trazabilidad de los cambios que se hacen en el software. Poder conectar el sistema de control con un software de gestión de proyectos y seguimiento de errores, ayuda con el análisis de la causa raíz de los problemas y con la recopilación de información.

El concepto de versión (también llamado revisión o edición) de un proyecto (código fuente) hace referencia al estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. Los sistemas de control de versiones utilizan repositorios para almacenar el proyecto actualizado junto a sus cambios históricos. Los sistemas de control de versiones centralizados almacenan todo el código en un único repositorio, es decir que un único servidor contiene todos los archivos versionados. Esto representa un único punto de falla dado que si el servidor no está disponible por un tiempo nadie podrá colaborar o guardar cambios en archivos en los que hayan estado trabajando.

Los sistemas de control de versiones distribuidos permiten en cambio continuar el trabajo aún cuando el repositorio de referencia no está disponible. En estos sistemas los clientes no solo descargan la última copia del código, sino que se replica completamente el repositorio con los cambios históricos (versiones). De esta manera, si un servidor deja de funcionar y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios estarán disponibles para los clientes y puede ser copiado al servidor con el fin de restaurarlo.

Git

Git es un proyecto de código abierto maduro y con un activo mantenimiento desarrollado originalmente por Linus Torvalds. Este sistema de control de versiones distribuido funciona bajo cualquier plataforma (Windows, MacOS, Linux, etc.) y está integrado en una amplia variedad de entornos de desarrollo (IDEs). Este sistema presenta una arquitectura distribuida, es decir que, cada desarrollador posee una copia del trabajo en un repositorio local donde puede albergar el historial completo de todos los cambios y, mediante comandos determinados, realiza sincronizaciones al repositorio remoto.

Git fue diseñado teniendo en cuenta las siguientes características:

- **Rendimiento**: Los algoritmos implementados en Git aprovechan el profundo conocimiento sobre los atributos comunes de los auténticos

árboles de archivos de código fuente.El formato de objeto de los archivos del repositorio de Git emplea una combinación de codificación delta (que almacena las diferencias de contenido) y compresión, guardando explícitamente el contenido de los directorios y los objetos de metadatos de las versiones.

•●**Seguridad**: la principal prioridad es conservar la integridad del código fuente gestionado. El contenido de los archivos y las verdaderas relaciones entre estos y los directorios, las versiones, las etiquetas y las confirmaciones, están protegidos con un algoritmo de hash criptográficamente seguro llamado "SHA1". De este modo, se salvaguarda el código y el historial de cambios frente a las modificaciones accidentales y maliciosas, garantizando que el historial sea totalmente trazable.

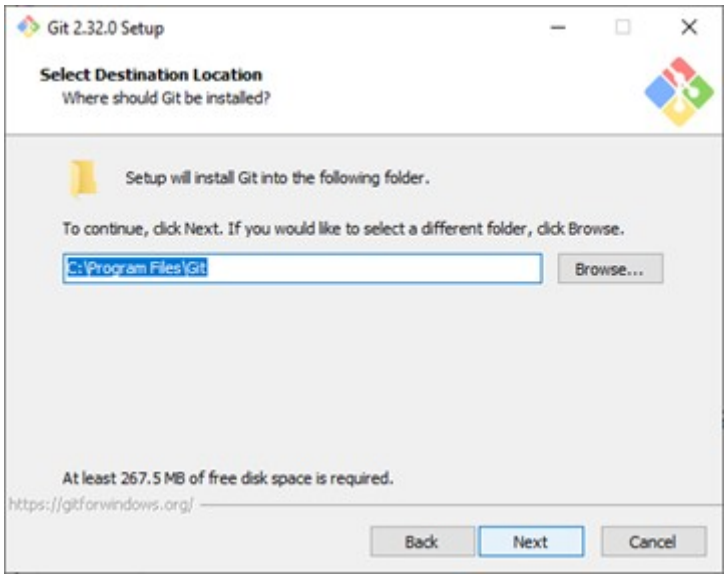
•●**Flexibilidad**: es flexible en varios aspectos, en la capacidad para varios tipos de flujos de trabajo de desarrollo no lineal, en su eficiencia en proyectos tanto grandes como pequeños y en su compatibilidad con numerosos sistemas y protocolos. Se ha ideado para posibilitar la ramificación y el etiquetado como procesos de primera importancia y las operaciones que se refieren a las ramas y las etiquetas (como la fusión o la reversión) también se almacenan en el historial de cambios.

Instalación Git

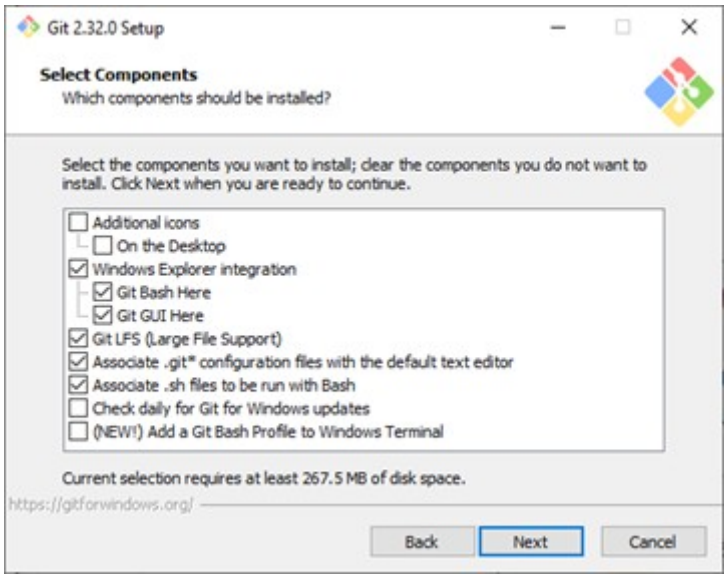
1. Descargar la versión adecuada, según el sistema operativo, desde este enlace

2.Ejecutar el instalador (capturas del instalador versión Git-2.32.0-64-bit), leer y aceptar los términos y condiciones de las Licencias.

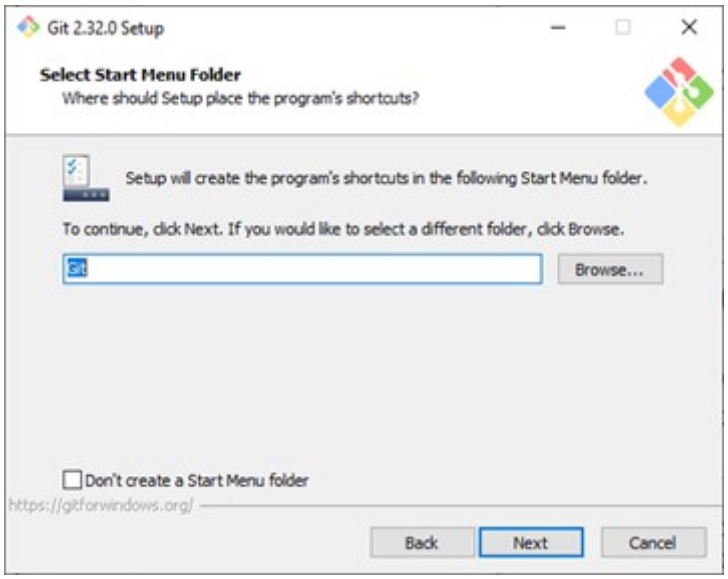
3.Elegir el destino de la instalación



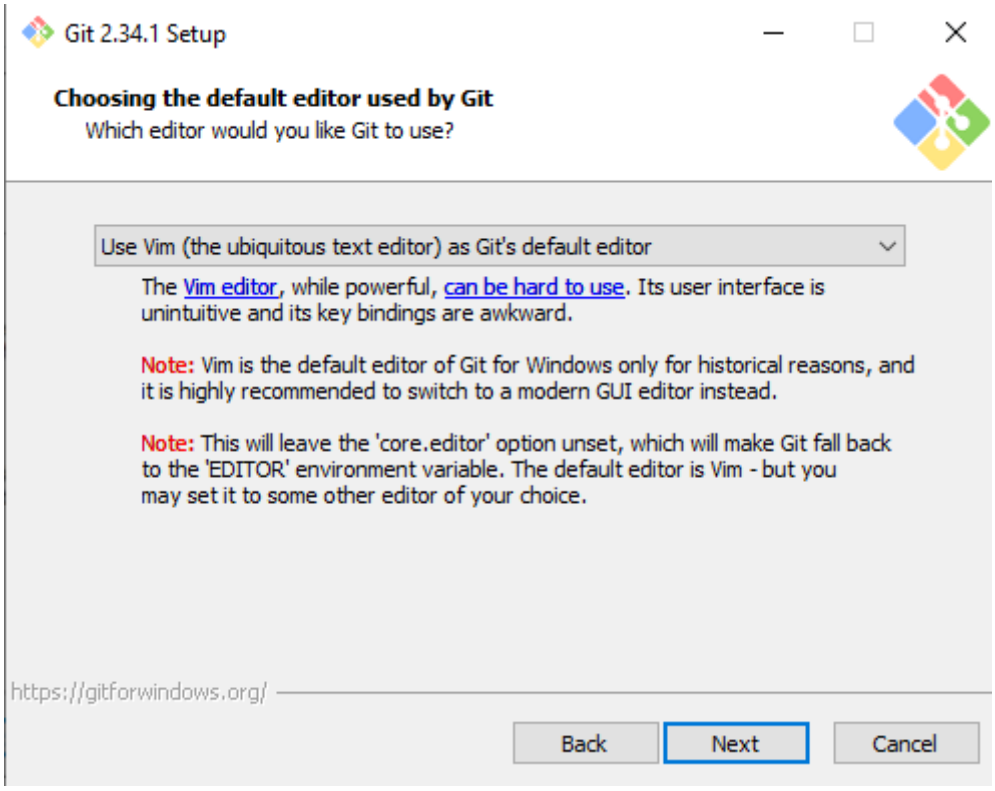
4.Seleccionar los componentes que se van a instalar.



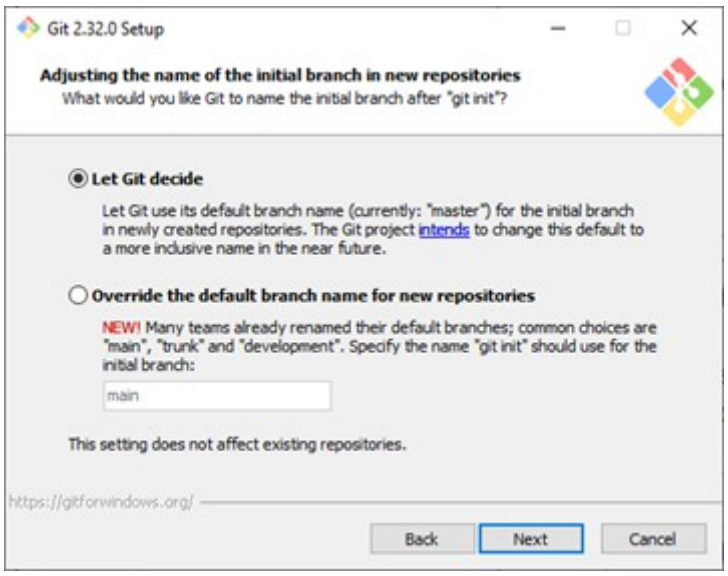
5. Asignar un nombre a la entrada que aparecerá en el menú de inicio de Windows.



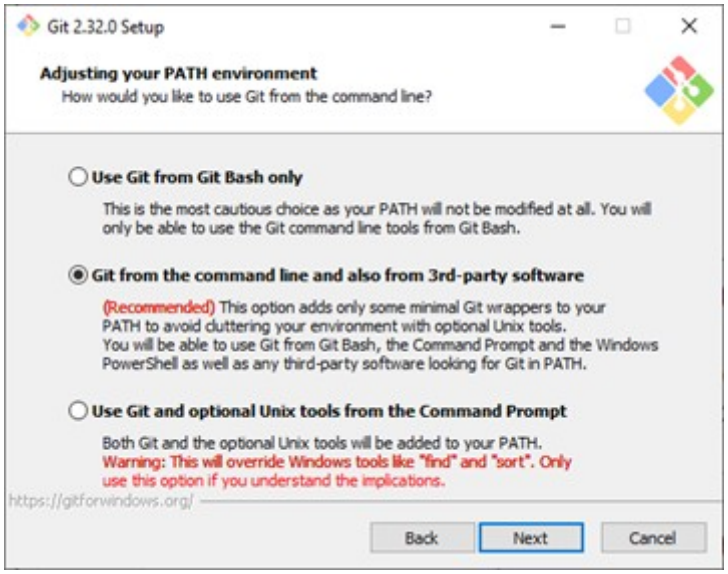
6.Seleccionar el editor que se utilizará para los comentarios y configuraciones de Git.



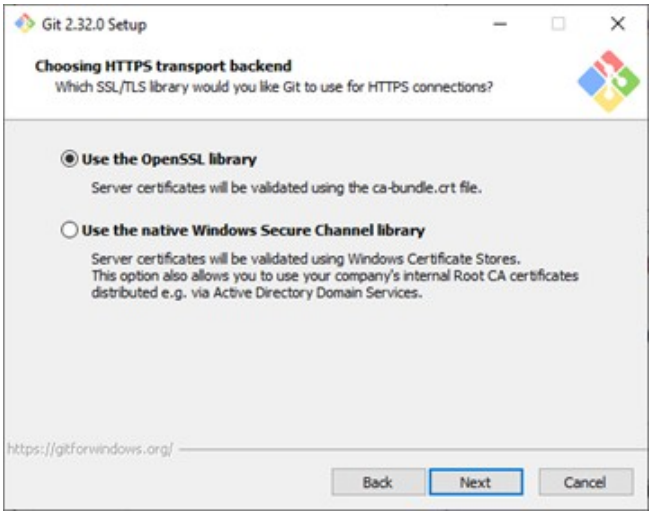
7. Elegir el nombre que se le asignará a la rama principal durante la creación de un nuevo repositorio.



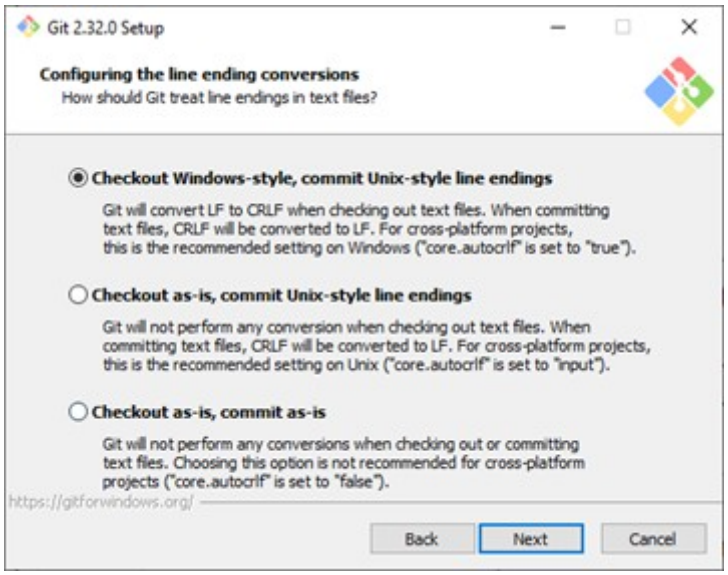
8. Configurar desde dónde podrán ser accesibles los comandos de git.



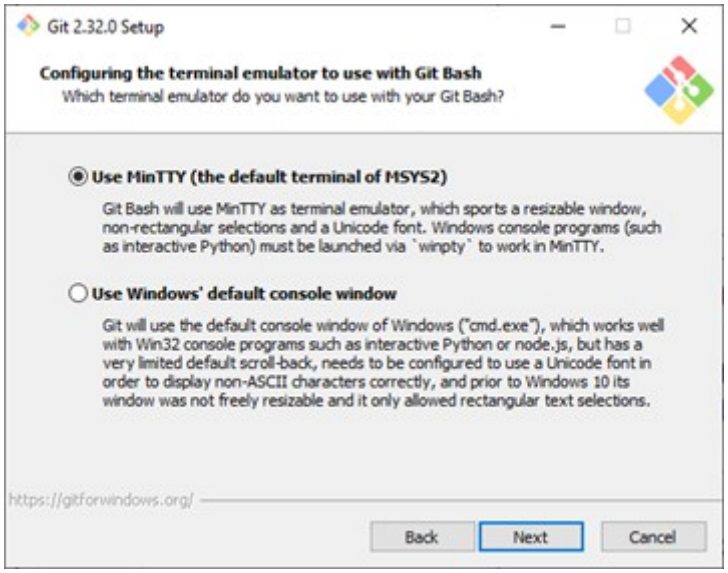
9. Seleccionar qué librería de seguridad se utilizará para las comunicaciones.



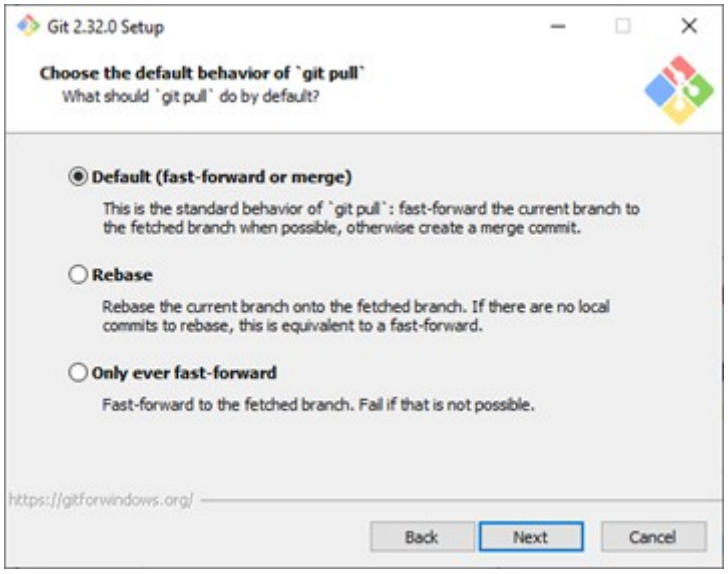
10. Configurar los finales de línea en los archivos de texto.



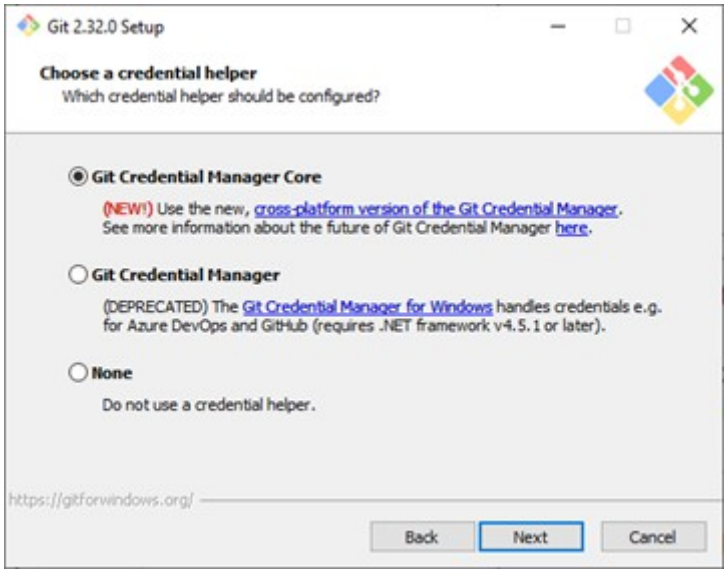
11. Configurar qué emulador de terminal se elegirá para usar con Git Bash.



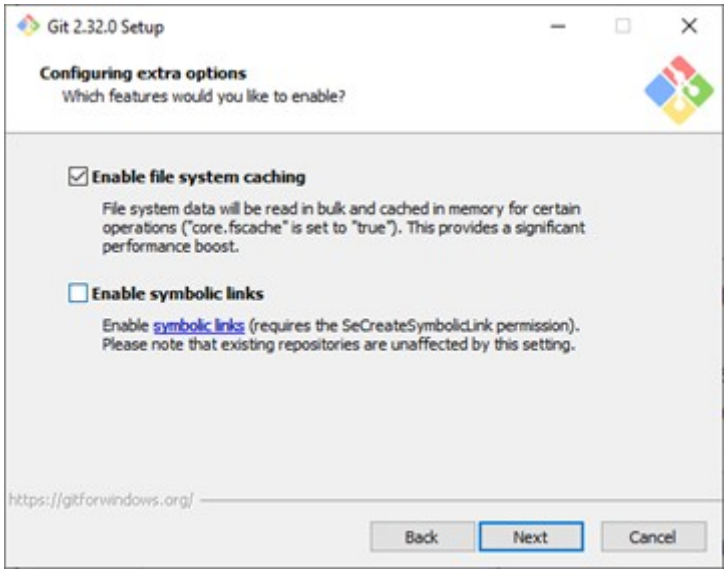
12. Elegir cuál será el comportamiento por defecto al ejecutar el comando “git pull”.



13.Configurar el gestor de credenciales que se utilizará para mantener información de usuarios asociados.



14. Configurar si se desea activar el sistema de caché y/o enlaces simbólicos.



15.Esperar a que finalice la instalación