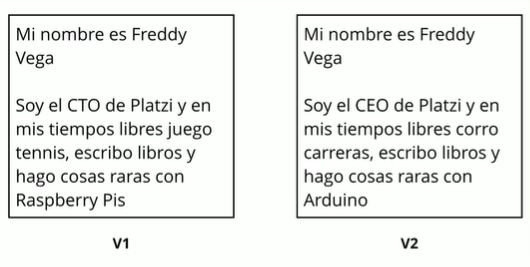
PLATZI: Curso Profesional de Git y GitHub – by Freddy Vega.

Ing. Luis Felipe Narvaez Gomez. E-mail: [ing.felipenarvaez017@gmail.com](mailto:ing.felipenarvaez017@gmail.com)

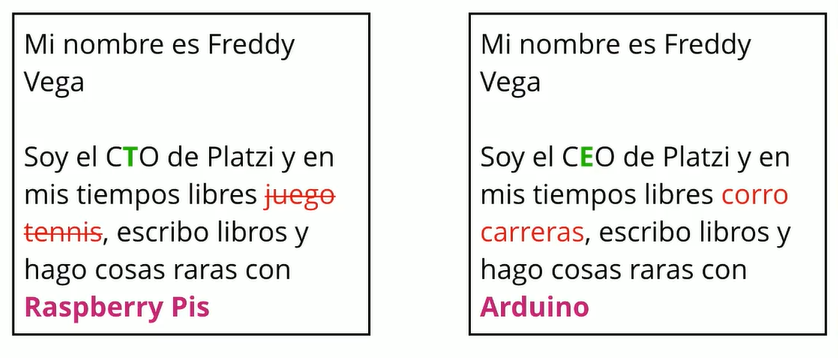
NOTA: Estos son apuntes personales tomados en el momento de la realización de este curso y la adquisición de información y conocimiento similar. No pretende ser un documento académico sin embargo esta expuesto de forma pública mediante el GitHub <https://github.com/RuisoArt> para las personas que quieran hacer uso de la información que a continuación se presenta sin ánimos de nada más que la presentación de apuntes de estudio.

APUNTES INTRODUCCIÓN:

En el momento que nosotros hacemos diferentes versiones de sistemas, documentos o algún contenido digital, lo común es que por cada cambio realizado creemos todo un NUEVO ARCHIVO y lo bauticemos a nuestro acomodo para vislumbrar en un tiempo posterior que versión del contenido es con la que estamos trabajando, algo como “tesis.doc”, luego “tesis-ve2.doc”, después “tesis final.doc”, ya luego la nombramos “tesis final final.doc” ya así hasta tener un “tesis finalfinall esta si ve 2456 unica.doc”.



En todas esas versiones del documento, lo que cambio fueron partes del contenido, pero no todo el contenido en si, no el suficiente para que ameritase una creación total de un nuevo documento por cada uno de los cambios realizados y es este problema al momento de hacer versiones que se plantea solucionar.

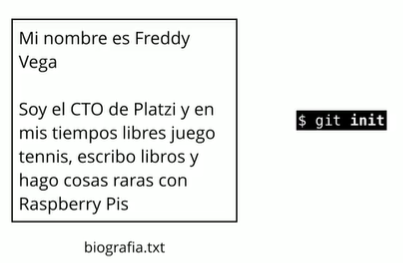


La mejor manera y mas eficiente de controlar estas versiones entre archivos es tener un sistema que solo se centre en reconocer estos mínimos cambios entre las versiones del documento, evitando la creación de todo un documento completo ( solo guardar los cambios ). Este proceso de solo guardar los cambios es algo ideal al momento que no solo trabajamos con documentos de texto si no con proyectos de software donde el código es intervenido por varias personas generando cambios en las estructuras de texto en cada momento.

Para la anterior se creo los sistemas de Control de versiones los cuales se enfocan en reconocer los cambios realizados en determinado archivo, guardarlos, fecharlos, mencionar quien los hizo, volver a una versión anterior, entre otras funciones. El sistema que se manejara en este documento es Git.



El archivo por si mismo se mantiene, no se crea otro, solo se guardan los cambios que se realizaron. Para poder iniciar un control de versiones de un documento cualquiera, la carpeta donde se aloja el mismo debe abrirse desde terminal y escribir el comando “git init” esto hace que se cree un repositorio en este folder donde se guardaran las versiones con sus cambios y demás metadatos correspondientes al historial de procesos de creación de los documentos que contiene.



Para que el repositorio sepa con que documentos trabajara dentro del mismo folder donde se estableció el repo, se le debe añadir con el comando “git add nombredocumento.extencion”.



Ahora cada que se haga un cambio y estos se quieran subir al repositorio con el control de versiones se de utilizar el comando “git commit -m “comentario sobre los cambios realizados””.



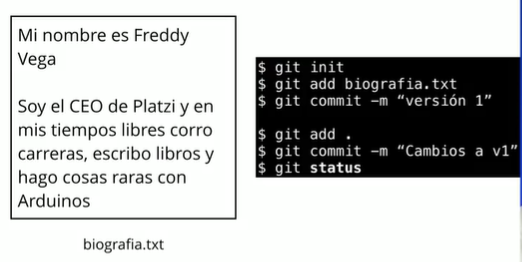
Ahora volvemos a hacer cambios en el archivo, tenemos que volver a agregar el archivo, bien podemos utilizar el comando “git add nombredocumento.extencion” o de manera mas sencilla “git add .” lo cual toma todos los documentos y archivos de la carpeta donde iniciamos el repo en local y los sube nuevamente al repo, guardando solo los cambios en cada uno de ellos.



Nuevamente hacemos el commit y quedan de una vez guardados todos nuestros cambios.

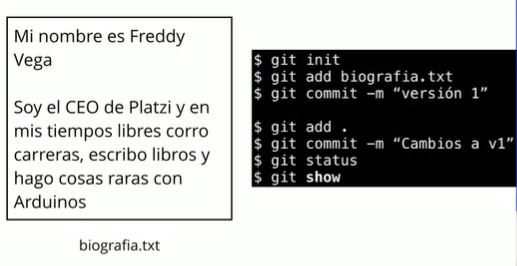


Podemos ver el estado de la base de datos de nuestro respositorio con el comando “git status” aquí nos señalara si hay cambios en los documentos que de local falta subirlos a la base de datos de nuestro repositorio o si por el contrario todo el repose encuentra actualizado con el estado de versiones de la carpeta que tenemos en local en nuestro ordenador.



Podemos ver esto como que, al momento de iniciar un GIT INIT el folder donde lo hicimos se convertirá en un carpeta que será la copia en local de un repositorio en una base de datos en la nube, habrá un archivo de git que guardara los cambios de los archivos que la carpeta contenga, haciendo del controlador de versiones de cada uno y comparara cada cambio con su versión en línea o el repositorio en línea, si el repo esta desactualizado en cambios con respecto a su contraparte en local nos avisara en “status” asegurándonos que siempre el repositorio este actualizado y los archivos este siempre en su versión más reciente.

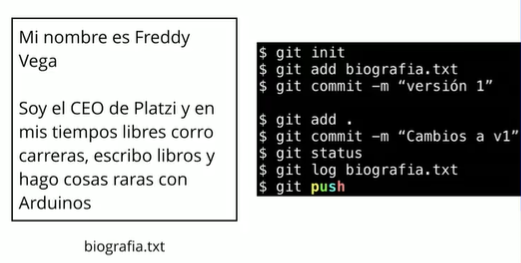
Otro comando muy importante es “git show” este nos mostrara un historial de todos los cambios realizados en nuestro repositorio incluyendo que cambios específicos, cuando se hicieron y quien los hizo.



Para ver los cambios de un archivo en especifico y por ende su historial utilizamos el comando “git log nombrearchivo.extencion”.



Para enviar los archivos a un servidor remoto se utiliza el comando PUSH con “git push” y para traer cambios realizados de ese servidor remoto a la parte local PULL con “git pull”.



APUNTES ¿QUE ES GIT?

Se utiliza el siguiente enlace a Youtube como parte de la dinámica del curso: <https://www.youtube.com/watch?v=DinilgacaWs> También los párrafos que se presentan a continuación de color morado son los propios dados por el curso.

**Git es un sistema de control de versiones distribuido, diseñado por Linus Torvalds. Está pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Git está optimizado para guardar todos estos cambios de forma atómica e incremental.**

**Se obtiene su mayor eficiencia con archivos de texto plano, ya que con archivos binarios no puede guardar solo los cambios, sino que debe volver a grabar el archivo completo ante cada modificación, por mínima que sea, lo que hace que incremente demasiado el tamaño del repositorio.**

**“Guardar archivos binarios en el repositorio de git es una mala práctica, únicamente deberían guardarse archivos pequeños (como logos) que no sufran casi modificaciones durante la vida del proyecto. Los binarios deben guardarse en un CDN”.**

**¿Qué es un sistema de control de versiones?**

**El SCV o VCS (por sus siglas en inglés) es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas llevar el historial del ciclo de vida de un proyecto, comparar cambios a lo largo del tiempo, ver quién los realizó o revertir el proyecto entero a un estado anterior.**

**Cualquier tipo de archivo que se encuentre en un ordenador puede ponerse bajo control de versiones.**

**¿Qué es Github?**

**Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se emplea principalmente para la creación de código fuente de programas de computadora.**

**Github puede considerarse como la red social de código para los programadores y en muchos casos es visto como tu curriculum vitae, pues aquí guardas tu portafolio de proyectos de programación.**

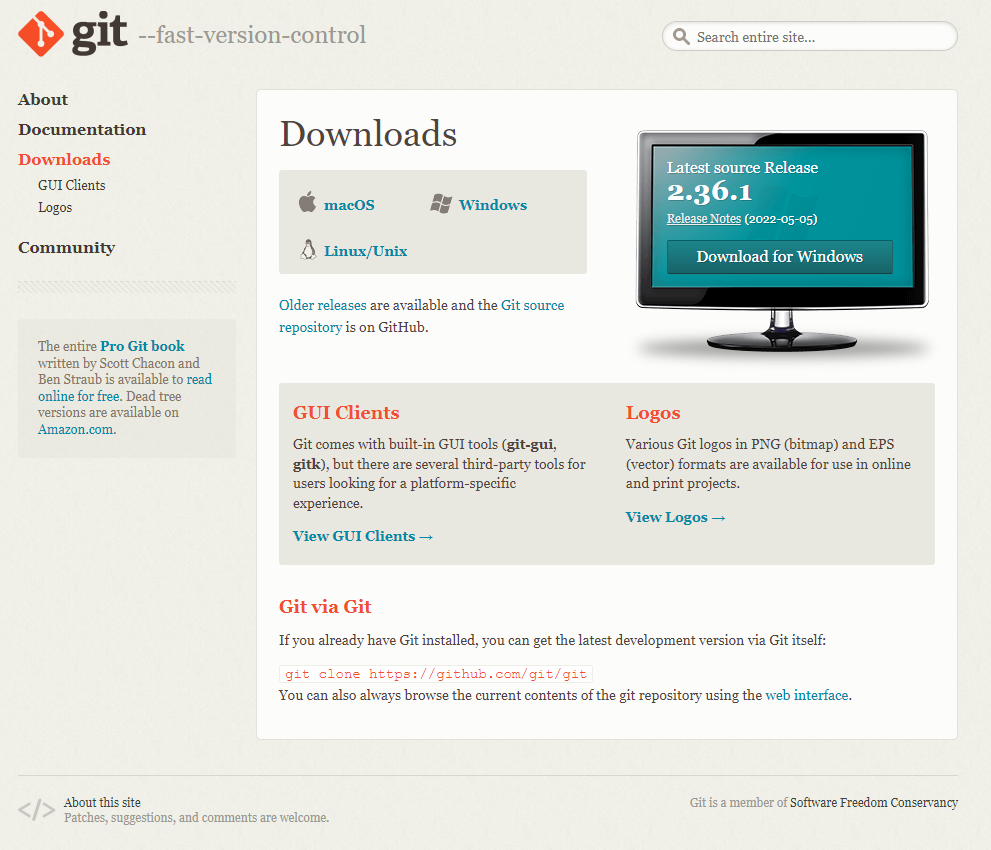
**Aporte creado por: Franco Coloccini**

APUNTES INSTALANDO GIT

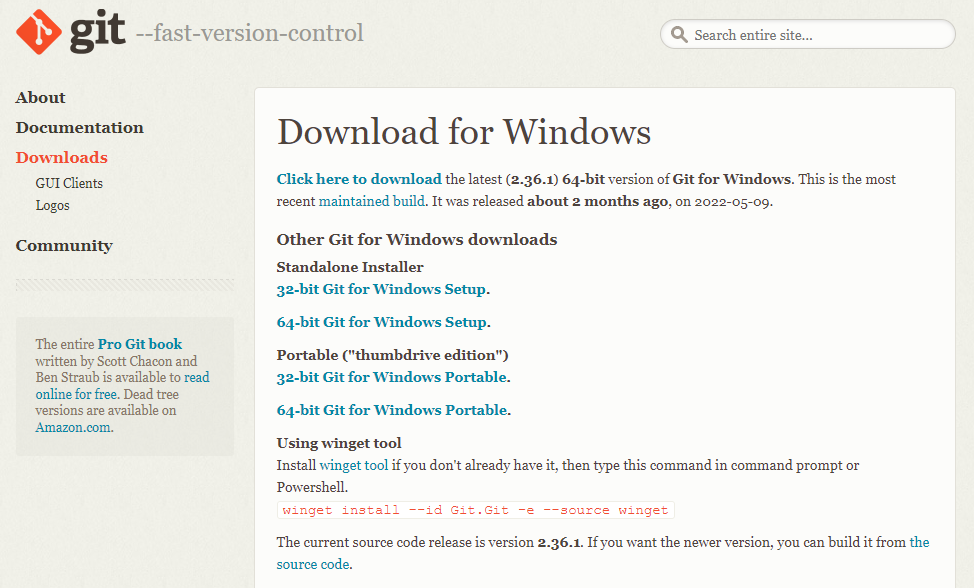
En mi caso lo instalare solo para el sistema operativo de Windows, para mi computadora al momento de realizar el curso es Windows 10 Home ve 21H2 compilación del SO 19044.1766.

Para poder empezar a trabajar con GIT el sistema de control de versiones de nuestros archivos debemos ir al siguiente enlace donde ubicaremos como instalar esta herramienta.

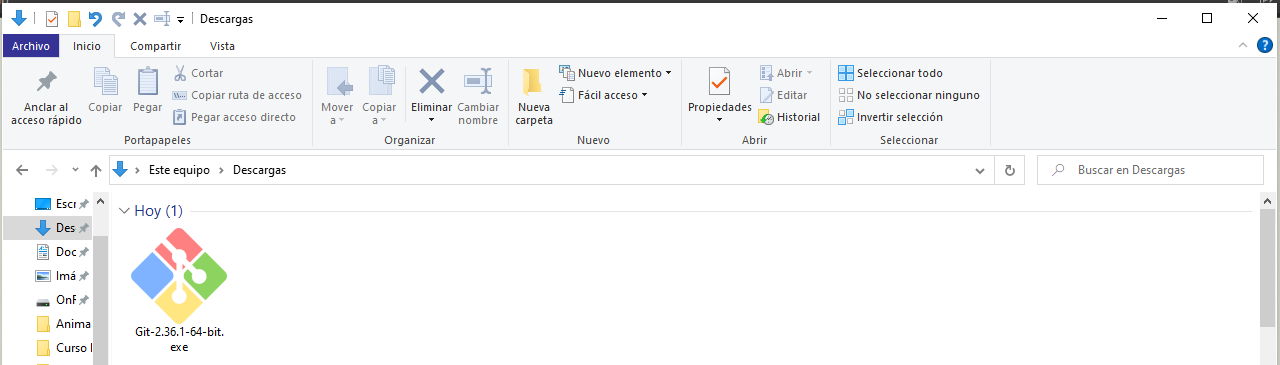
<https://git-scm.com/downloads>

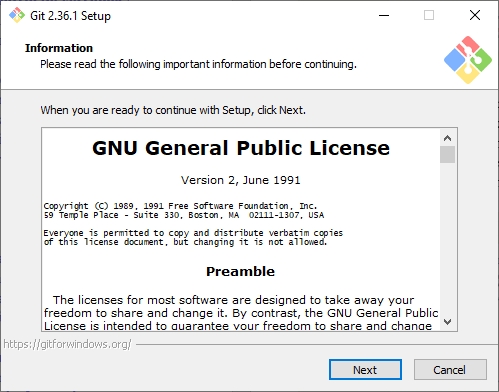


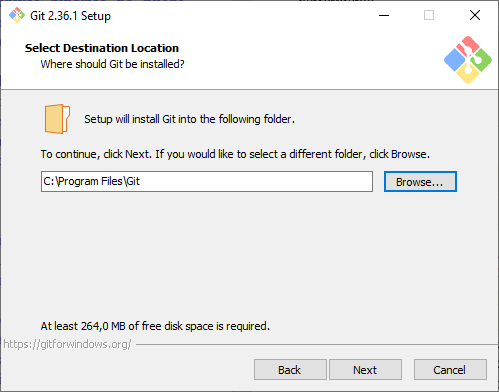
En Windows no hay mayor problema con la instalación de este programa es el clásico “siguiente, siguiente, siguiente y Finallizar”. En mi caso selecciono la versión que mas se acomode a mis necesidades y arquitectura de procesador de mi computadora, x64-bit Standalone Installer.

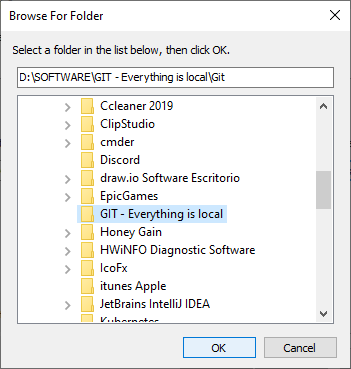


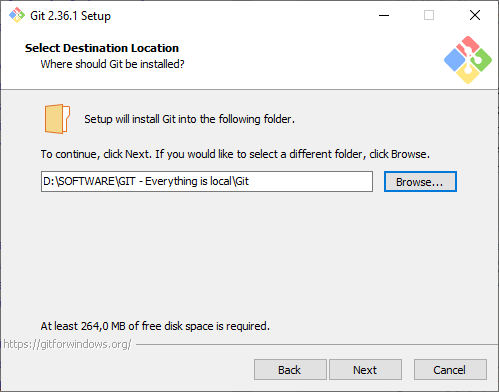
Una vez descargado lo buscamos en el equipo y lo ejecutamos como administradores para evitar inconvenientes.

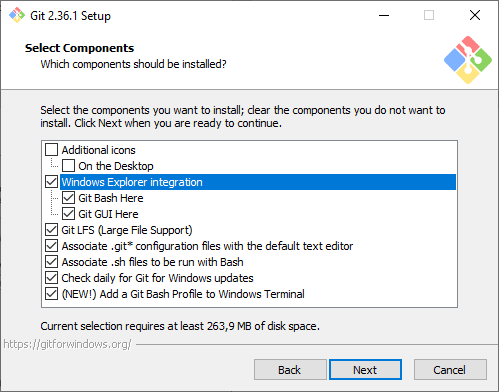


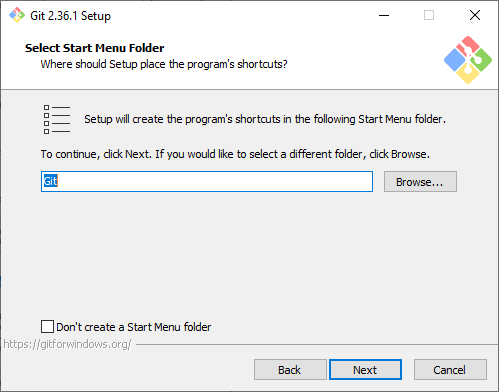


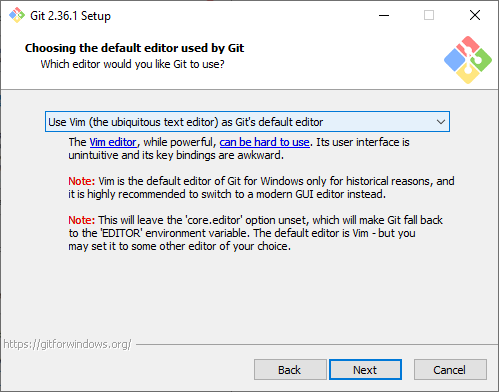


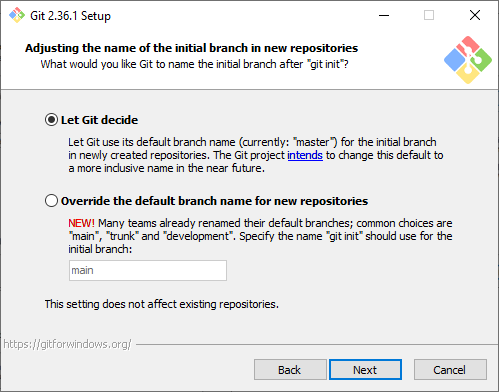




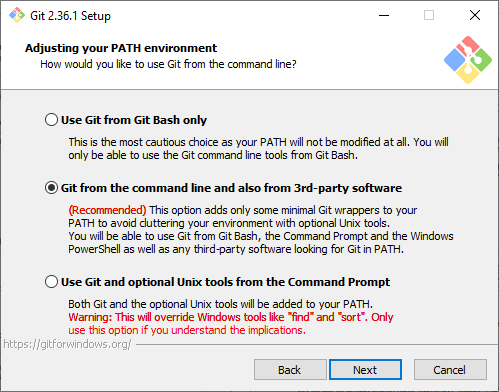


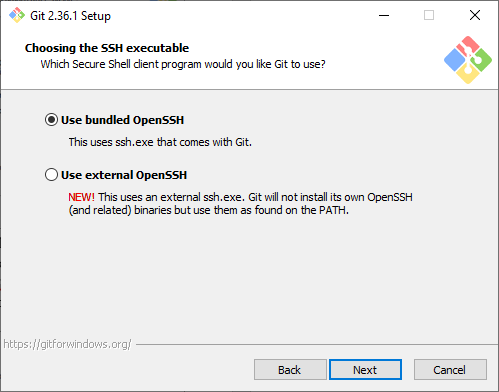




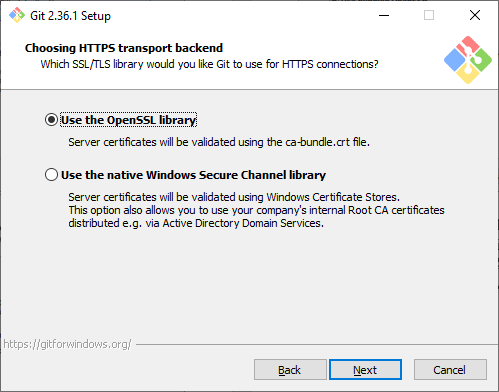


En la siguiente parte selecciono la segunda opción debido a que yo no quiero solo utilizar GIT des de la terminal que intala BASH si no desde cualquier terminal de comandos nativa de Windows como ya la clásica CMD.





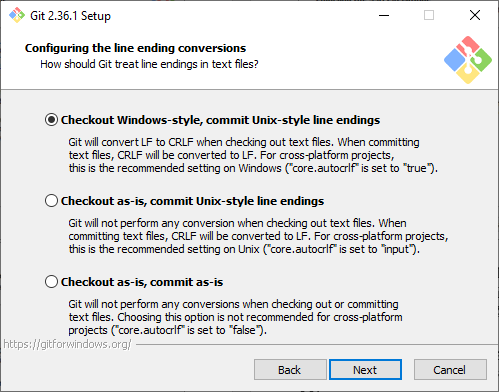
La gran mayoría del mundo de programación utiliza el protocolo de verificación por llaves cripticas de Open SSL, esto dado desde los servidores de internet y SO locales que trabajan con LINUX, sin embargo también existe el protocolo propio de Microsoft (la segunda opción) la cual no es que sea mejor o peor que SSL pero por mi concepto prefiero utilizar e protocolo de Open SSL.



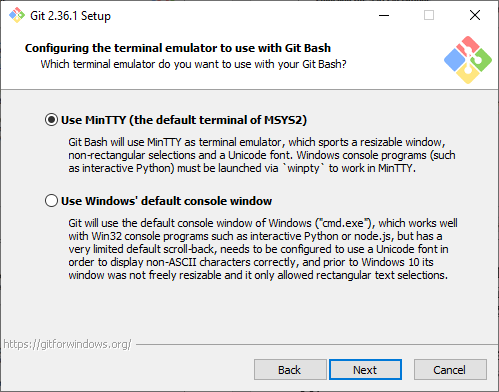
Ahora debemos entender que al momento de ejecutar un comando nosotros damos ENTER pero este interpretado de manera distinta en Windows, en Linux o en MAC dentro de la terminal, mas cuando los mismos ENTER se envían con todo y ficheros entre SO como en un ambiente real de trabajo de programación de Software.

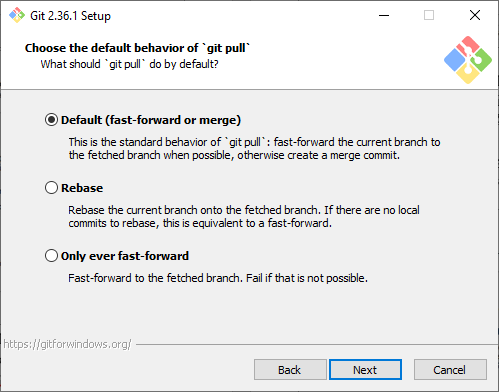
Por tal motivo, el programa nos pregunta que queremos hacer con esta diferencia de interpretación de ENTER entre sistemas operativos. Lo primero es utilizarlo normal en Windows y cuando se envié se traduzca a un sistema basado en UNIX, lo segundo es utilizarlo y que lo envié como caiga, o por último que si se traduce o no sea nuestra elección.

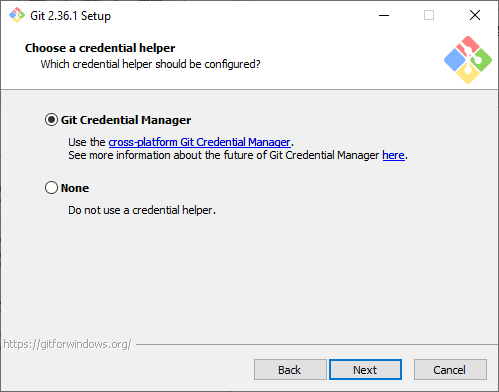
En mi caso prefiero que lo traduzca el solo para evitar tantas complicaciones y errores con los archivos con los que se puede trabajar.

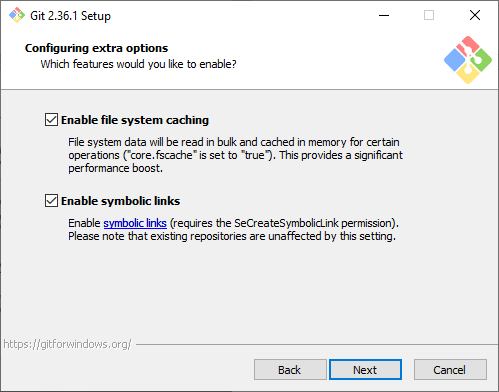


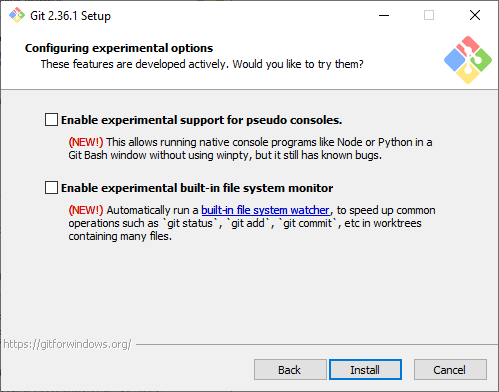
Ahora seleccionamos si queremos utilizar el lenguaje de CMD nativo de Windows (segunda opción) o utilizar uno traducido para que utilicemos comandos de Linux (primera opción), yo elijo esto último.

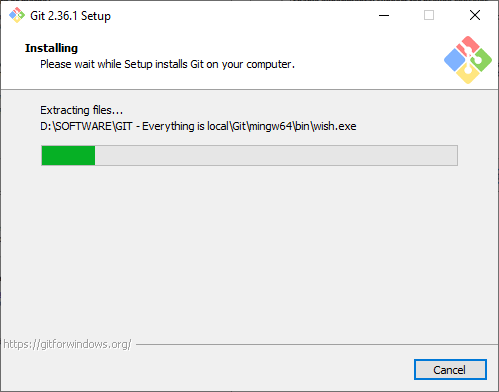


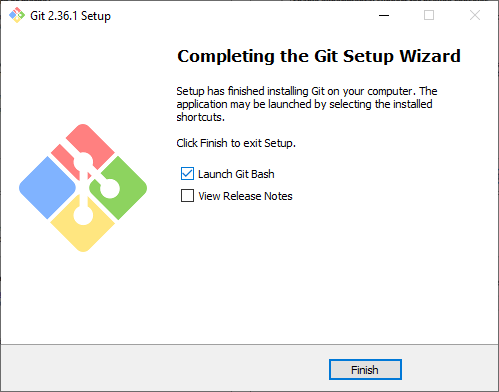




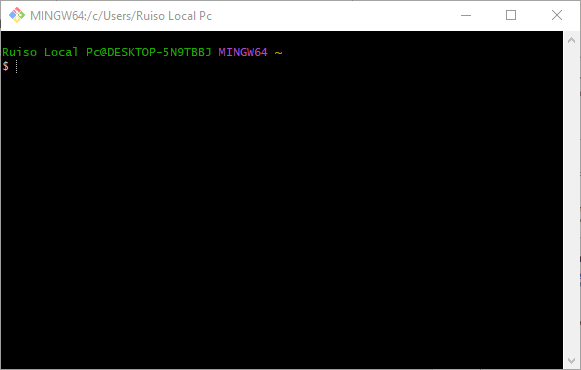








Y listo ya tenemos GIT en nuestro sistema con su consola propia PARECIDA de Linux que maneja lenguaje nativo de los SO basados en UNIX.



Si queremos abrir esta consola podemos localizarla como GIT BASH dentro del equipo.



El siguiente texto en color morado hace parte del texto propio dado por el curso.

**Windows y Linux tienen comandos diferentes, graban el enter de formas diferentes y tienen muchas otras diferencias. Cuando instales Git Bash en Windows debes elegir si prefieres trabajar con la forma de Windows o la forma de UNIX (Linux y Mac).**

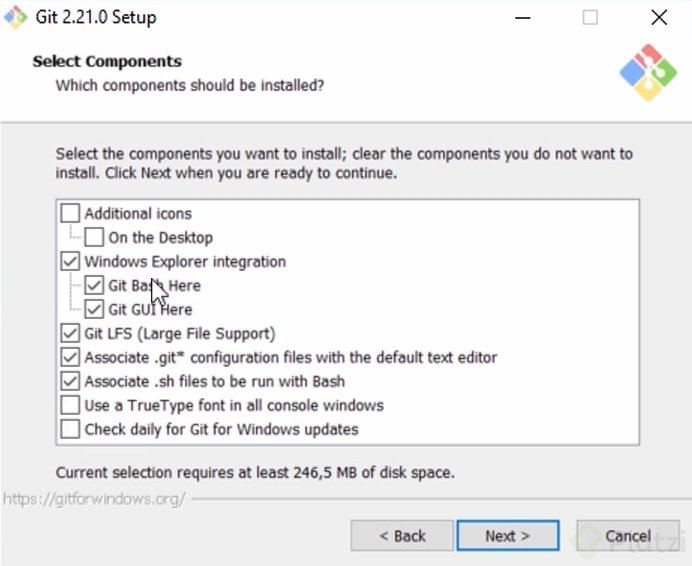
**Ten en cuenta que, normalmente, los entornos de desarrollo profesionales tienen personas que usan sistemas operativos diferentes. Esto significa que, si todos podemos utilizar los mismos comandos, el trabajo resultará más fácil para todos en el equipo.**

**Los comandos de UNIX son los más comunes entre los equipos de desarrollo. Así que, a menos que trabajes con tecnologías nativas de Microsoft (por ejemplo, .NET), la recomendación es que elijas la opción de la terminal tipo UNIX para obtener una mejor compatibilidad con todo tu equipo.**

**Cómo instalar Git en Windows**

**Para instalar este sistema de control de versiones en Windows, simplemente vas al repositorio de descarga. Luego de descargarlo, lo ejecutas como cualquier otra aplicación de Windows.**

**Al momento de iniciar el instalador, asegúrate de haber marcado la opción de instalar Git Bash en el computador (que es lo que nos permitirá correr comandos de Linux en la consola sin problemas para trabajar con Git).**

****

**Aporte creado por: Diego Camacho**