Git

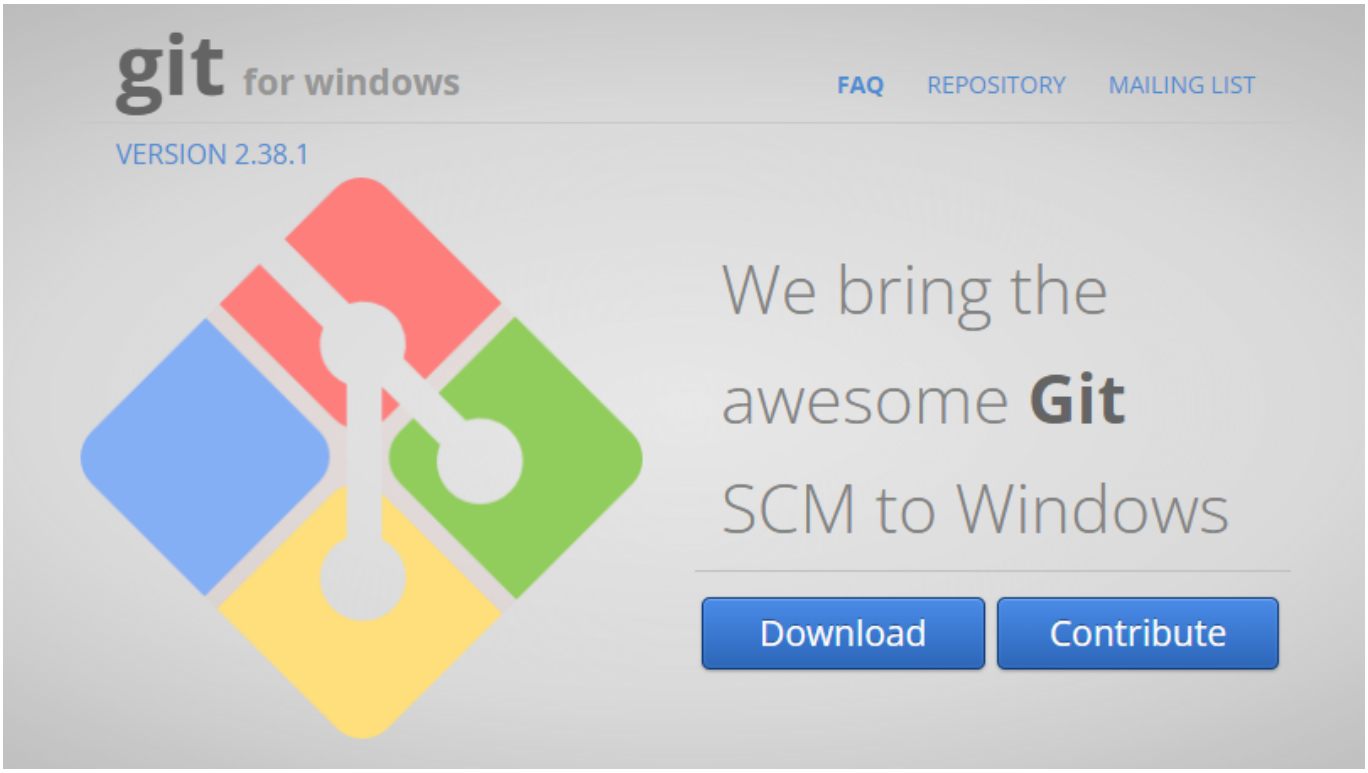
Git es un software de control de versiones diseñado por Linus Torvalds, quien también creó Linux. Git permite a los desarrolladores llevar un seguimiento de las diferentes versiones de un archivo o conjunto de archivos mientras se crea un nuevo software.

Esto es útil porque permite a los desarrolladores volver a una versión anterior de un archivo si hicieron un cambio que resultó en un error. También permite a los desarrolladores colaborar en el mismo proyecto al mismo tiempo.

Para usar git, primero debe instalarlo en su computadora. Luego, debe configurarlo para que sepa quién está haciendo cambios en qué archivos. Esto se hace mediante el uso de una cuenta de git. Una vez que haya configurado git, puede empezar a usarlo para controlar las versiones de sus archivos.

Paso 1: Descargar e instalar Git

Antes de nada, necesitas tener Git instalado en tu sistema Windows.



Paso 2: Configurar Git

Una vez que tengas Git instalado, necesitas configurarlo con tu nombre y dirección de correo electrónico. Esto se hace para que Git pueda asociar tus commits con tu identidad:

```
git config --global user.name " tu nombre "
git config --global user.email " tu@email.com "
```

Paso 3: Crear un repositorio

Para empezar a trabajar con Git, lo primero que necesitas hacer es crear un **repositorio**. Un repositorio de Git es un espacio en el que se almacenan los archivos de un proyecto y todo su historial de cambios. Puedes crear un repositorio de dos maneras: mediante la interfaz gráfica de Git o mediante la línea de comandos.

1. Abrir una terminal y navegar hasta la carpeta "git".
2. A continuación, debe escribir el comando "git init". Este comando creará un repositorio git en su carpeta.

Añadir cambios

Ahora, cada vez que haga un cambio en uno de sus archivos, debe agregar el archivo al repositorio git. Esto se hace con el comando "git add".

Confirmar cambios (commit)

Luego, debe confirmar los cambios con el comando `git commit`. Este comando le permitirá escribir un mensaje que describa los cambios que hizo en el archivo

Historial de cambios

Para ver un historial de los cambios que ha hecho en un archivo, puede usar el comando "git log". Este comando le mostrará todos los commits que ha hecho en el archivo. También puede ver qué líneas de código han cambiado en cada commit.

Paso 4: Clonar un repositorio

Si ya existe un repositorio de Git que deseas utilizar, puedes clonarlo en tu sistema local. Clonar un repositorio crea una copia local del repositorio remoto, lo que te permite tener acceso a todos los archivos del repositorio, así como su historial de cambios.

1. Abra la terminal.
2. Vaya a la carpeta en la que desea almacenar el repositorio clonado.
3. Escriba el siguiente comando y presione Enter:

```
git clone https://github.com/nombre-de-usuario/nombre-repositorio.git
```

4. El repositorio se ha clonado en su computadora.

Paso 5: Crear y confirmar commits

Cada vez que haces cambios en los archivos de tu proyecto, necesitas confirmarlos (hacer un "commit") para que queden registrados en el historial de cambios de Git. Esto se hace mediante el comando "git commit".

Paso 6: Pushear y pullar cambios

Si trabajas en un repositorio remoto, necesitarás pushear tus cambios al repositorio para que otros usuarios puedan acceder a ellos. Pushear es el equivalente de hacer un commit en el repositorio remoto.

Ejemplo:

Vamos a enviar (sincronizar) los cambios realizados en el repositorio **local** al repositorio **remoto**.

```
E:\Docencia\apuntes>git push origin master
Enumerating objects: 36, done.
Counting objects: 100% (36/36), done.
Delta compression using up to 8 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (30/30), 1.97 MiB | 3.53 MiB/s, done.
Total 30 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/danimrprofe/apuntes
   9118fa98..a8c5f455  master -> master
```

El comando se compone de las siguientes partes:

- **git push**: este es el comando para enviar cambios al repositorio remoto.
- **origin**: Este es el nombre del repositorio remoto.
- **master**: este es el nombre de la rama que se enviará al repositorio remoto.

En este ejemplo, los cambios realizados en el repositorio local (E:\Docencia\apuntes) se envían al repositorio remoto (https://github.com/danimrprofe/apuntes) en la rama maestra.

El comando muestra el estado de la inserción, incluidos cuántos objetos se enumeraron, comprimieron y escribieron, y el número total de objetos. El comando también informa la cantidad de objetos locales que se usaron para completar la compresión delta. Finalmente, el comando informa qué rama se empujó y los detalles del empuje.

También puedes pullear cambios del repositorio remoto, lo que es equivalente a hacer un commit en tu repositorio local.

Paso 7: Ramas

Una rama de Git es un conjunto de commits que se encuentran en una línea separada de desarrollo. Las ramas se utilizan para desarrollar funcionalidades independientes, para hacer correcciones de errores o para experimentar con nuevas ideas.

Paso 8: Fusionar ramas

Una vez que hayas terminado de desarrollar una característica o de hacer una corrección de error en una rama, necesitarás fusionar esos cambios de vuelta a la rama principal. Esto se hace mediante el comando "git merge".

Paso 9: Resolver conflictos

A veces, cuando tratas de fusionar ramas, Git no puede hacerlo automáticamente. Esto se debe a que las ramas han divergido demasiado y Git no puede determinar qué cambios deben conservarse. En estos casos, tendrás que resolver los conflictos manualmente.

Paso 10: Etiquetas

Las etiquetas son marcadores que se pueden aplicar a commits específicos. Las etiquetas se utilizan para marcar versiones específicas de un proyecto. Por ejemplo, si estás desarrollando un software, podrías aplicar una etiqueta a cada versión que se libera.

Paso 11: Deshacer cambios

En Git, hay varias formas de deshacer cambios. La forma más sencilla de deshacer cambios es deshacer los últimos cambios confirmados mediante el comando "git reset". También puedes deshacer cambios que no hayan sido confirmados mediante el comando "git checkout".

Paso 12: Revertir commits

A veces, puede que quieras revertir un commit anterior, de forma que los cambios que se hicieron en ese commit se eliminen. Esto se hace mediante el comando "git revert".