MANUAL BÁSICO DE GIT

TRABALLANDO CON GIT

DESENVOLVEMENTO DE INTERFACES

CS DESENVOLVEMENTO DE APLICACIÓNS MULTIPLATAFORMA

Autor: Manuel Pacior Pérez

Índice

1 INTRODUCIÓN

Git (pronunciado "guit") é un software de código aberto (GPLv3) para o control de versións, deseñado por Linus Torvalds (Linux kernel). Foi implementado co propósito de forneces mecanismos eficientes e fiables de versións de aplicacións que contan cun gran número de ficheiros de código fonte e sobre os que traballan moitas persoas á vez.

Git converteuse nun sistema de control de versións totalmente funcional. Existen unha gran cantidade de proxectos que usan este sistema, un deles é o grupo de programación do núcleo de Linux.

Entre as características máis relevantes están:

- Forte soporte ao desenvolvemento non lineal: facilita a xestión de ramas de desenvolvemento e a fusión entre diferentes versións.
- Ferramentas específicas para acceder ao historial de desenvolvemento.
- Xestión distribuída. Git ofrece a cada desenvolvedor unha copia local de todo o historial. Os cambios propáganse entre os repositorios locais e pode ser fusionados na rama remota do mesmo xeito que na local.
- Os repositorios pódense publicar: Permite facelo a través de HTTP, FTP, rsync ou a través dun protocolo nativo, ben a través dunha simple conexión TCP / IP ou mediante cifrado SSH.
- Xestión eficiente de grandes proxectos: en boa medida a causa da rapidez coa que xestiona as diferenzas entre ficheiros.

2 INSTALACIÓN

2.1 Debian

Para instalar Git en Debian (ou Ubuntu) tan so debe executar o seguinte comando por consola con permisos de administracor, ou buscar a aplicación "git" no xestor de paquetes e seleccionala para a súa instalación:

```
apt-get install git
```

2.2 Windows

Existen varios xeitos de instalar Git en Windows, nós imos seguir o oficial, para o que tan so debemos seguir os seguinte pasos:

- Ir á ligazón web de descargas de git para Windows
- Baixar o instalador de Windows para a nosa arquitectura (32 ou 64 bits)
- Executar o instalador e completar o proceso

3 APRENDENDO GIT

A continuación indicamos os conceptos básicos para manexarnos con este sistema de control de versións. Parte da información contida neste apartado tivo como referencia a que se atopa no seguinte enlace.

3.1 Crear un novo repositorio

Para facer esta operación temos que crear un novo directorio, debemos ir a el a través dun terminal e executar:

```
git init
```

Isto inicializará nesta carpeta un novo repositorio de Git.

3.2 *Checkout* a un repositorio

Esta operación serve para crear en local, unha copia do repositorio. Para facelo debe executarse:

```
git clone /ruta/ao/repositorio
```

No caso de usar un servidor Git remoto sería:

```
git clone usuario@host:/ruta/ao/repositorio
```

No caso do repositorio de git que tedes creado para no gitlab para traballar sobre el sería:

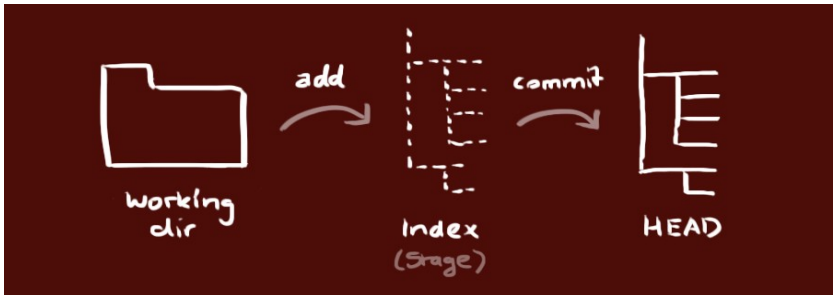
```
git clone https://gitlab.com/mpacior/dint-dam-20-21/codigo_de_usuario
```

Pediranos o noso nome de usuario de Gitlab (será o mesmo que o código da nosa conta de Gmail do IES San Clemente), e a nosa contrasinal (ver instrucións de como obter o nome de usuario e poñer o contrasinal).

3.3 Fluxo de traballo

O repositorio local está composto por tres "árbores" xestionadas polo Git:

1. O primeiro é o teu [Directorio de traballo] que contén os ficheiros.
2. O segundo é o [Index] que actúa como zona intermedia.
3. O terceiro e último, é o [HEAD] que apunta ao último commit feito.



3.4 *Add & commit*

Podemos gravar os cambios (engadilos ao ***[Index]***) usando

```
git add <filename>

git add .
```

Este é o primeiro paso do fluxo de traballo básico. Para cometer estes cambios use:

```
git commit -m "Mensaxe de confirmación"
```

Agora o ficheiro está incluído no ***[HEAD]***, pero aínda non está no seu repositorio remoto.

3.5 Envío de cambios

Os nosos cambios están agora no ***[HEAD]*** da copia local. Para enviar estes cambios ao seu repositorio remoto execute:

```
git push origin master
```

Substitúe o *master* pola rama á que queremos enviar os nosos cambios.

Se non clonáramos un repositorio existente e queremos conectar o noso repositorio local a un repositorio remoto, entón executamos:

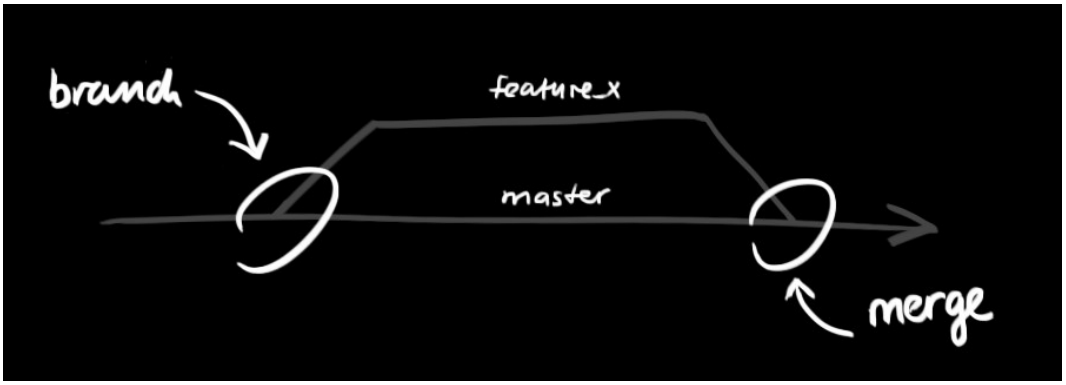
```
git remote add origin <server>
```

Substituímos *<server>* pola dirección ao noso repositorio Git, deste xeito poderíamos gardar os nosos cambios no repositorio remoto seleccionado.

3.6 Ramas

As ramas úsanse para desenvolver funcionalidades illadas entre si. A rama *master* será a rama "predeterminada" cando creamos un repositorio.

Podemos crear novas ramas durante o desenvolvemento, e *fusionalas* á rama principal cando rematemos.



Neste exemplo creamos unha nova rama chamada "feature_x" e cambiamos a ela usando:

```
git checkout -b feature_x
```

Se quixéramos volver á rama principal so temos que executar:

```
git checkout master
```

Para eliminar a rama teríamos que escribir:

```
git branch -d feature_x
```

Debemos ter en conta que unha nova rama non vai estar dispoñible para outras persoas do equipo de desenvolvemento mentres non se suba ao repositorio remoto común. Para facer isto teríamos que executar:

```
git push origin <branch>
```

3.7 Actualizar e fusionar

Para actualizar o noso repositorio local ao último *commit*, executamos no noso directorio de traballo:

```
git pull
```

deste xeito baixaremos e fusionaremos os cambios remotos.

Para fusionar outra rama coa nosa rama activa (por exemplo *master*) usamos

```
git merge <branch>
```

Nestes dous casos Git tentará fusionar automaticamente todos os cambios. Por desgraza, isto non sempre será posible, e poden producirse conflitos. Nese caso, somos nós os responsables de combinar eses conflitos, para elo editaremos manualmente os ficheiros amosados Git. Despois de modificalos, cómpre marcalos como fusionados con:

```
git add <filename>
```

Antes de fusionar os cambios, podemos revisalos usando:

```
git diff <source_branch> <target_branch>
```

3.8 Etiquetas

Recoméndase crear etiquetas para cada nova versión lanzada dun software. Este concepto non é novo, xa que estaba dispoñible en Subversion. Podes crear unha nova etiqueta chamada 1.0.0 executando.

```
git tag 1.0.0 3a7e1f63cd
```

3a7e1f63cd refírese aos 10 caracteres do *commit* ID aos que quere facer referencia coa súa etiqueta. Podemos obter o ID *commit* con

```
git log
```

3.9 Substituír os cambios locais

No caso de que fagamos algo mal, sempre podemos anular os cambios locais usando o comando:

```
git checkout - <nome do ficheiro>
```

Este comando substitúe os cambios no noso directorio de traballo polo contido do *[HEAD]* máis recente. Os cambios que xa se engadiron ao *[Index]*, así como os novos ficheiros, manteranse sen cambios.

Por outra banda, se qeremos desfacer todos os cambios locais e commits, pode traer a última versión do servidor e apuntar á súa copia local principal deste xeito

```
git fetch origin

git reset --hard origin/master
```

3.10 Datos útiles

3.10.1 Interface gráfica predeterminada

A interface gráfica de usuario por defecto é *Gitk* aínda que existen varias alternativas que se poden consultar nesta ligazón.

3.10.2 Cores especiais para a consola

Esta opción actívase co seguinte comando:

```
git config color.ui true
```

3.10.3 Amosar só unha liña por confirmación

Para amosar só unha liña para cada confirmación na traza debemos executar:

```
git config format.pretty oneline
```

3.10.4 Engade ficheiros de forma interactiva

Para engadir os ficheiros de xeito interactivo, tan so temos que executar:

```
git add -i
```