Tutorial básico de GIT – Parte 1

Introducción



En este segundo post vamos a preparar el entorno para poder trabajar y realizar algunas operaciones básica. También introduciremos algunos conceptos nuevos para trabajar con repositorios remotos, los cuales profundizaremos en el próximo post

Vamos a tratar:

- Operaciones básicas
  - Instalando Git
  - Configuraciones por defecto
  - Crear un nuevo repositorio
  - Hacer commit con un fichero
- Trabajar con un repositorio remoto
  - Hacer push a un repositorio remoto
  - Hacer clone de un repositorio remoto
  - Hacer pull de un repositorio remoto

Operaciones básica

Instalando Git

Bien antes de empezar a trabajar con Git deberemos preparar el entorno de trabajo. Esto es realmente fácil simplemente lo instalamos en GNU\Linux haciendo uso del sistema de paquetes

Para sistema basados en rpm como Fedora:

```
$ yum install git
```

Para sistemas basados en deb como Debian:

```
$ sudo apt-get install git
```

Acabada la instalación y a modo de prueba podemos ejecutar el siguiente comando que nos devolverá la versión instalada:

```
git --version
git version 2.1.0
```

Configuraciones por defecto

Ahora, vamos a configurar el nombre de usuario por defecto y dirección de correo electrónico para que Git identifique la persona que commitea los cambios. Esta configuración sólo hay que hacer una vez.

La configuración de la consola Git se guarda en el archivo .gitconfig en el directorio home del usuario. Se puede editar manualmente el archivo, pero en este tutorial vamos a utilizar el comando «config».

```
$ git config --global user.name "<Username>"
$ git config --global user.email "<Email address>"
```

Configuramos la salida de color de Git

```
$ git config --global color.ui auto
```

También puede configurar los alias para los comandos de Git. Por ejemplo, se puede abreviar «checkout» para «co» y usarlo

para ejecutar el comando.

```
$ git config --global alias.co checkout
```

Crear un nuevo repositorio

Vamos a empezar por la creación de un nuevo repositorio local. Nuestro objetivo es crear un directorio de pruebas y ponerlo bajo control de versión con Git. Utilizaremos este directorio lo largo del tutorial.

Crearemos un directorio tutorial en cualquier lugar de nuestro equipo. Después accederemos al directorio y utilizaremos el comando «init» para convertir ese directorio en un repositorio Git local.

```
$ git init
```

Mediante los siguientes comandos crearemos el nuevo directorio tutorial en un repositorio Git.

```
$ mkdir ~/tutorial
$ cd ~/tutorial
$ git init
Initialized empty Git repository in /home/yourname/tutorial/.git/
```

Hacer commit de un fichero

En el directorio del tutorial que hemos creado anteriormente, vamos a añadir un nuevo archivo y lo registraremos en el repositorio.

Crea el archivo «sample.txt» en ese directorio con un texto cualquiera, por ejemplo:

```
echo "After three days without programming, life becomes meaningless." > sample.txt
```

Podemos usar el comando «status» para confirmar el estado de nuestro «working tree» e «index» en Git.

```
$ git status
```

Obteniendo el siguiente resultado

```
On branch master

Initial commit

Untracked files:
  (use "git add ..." to include in what will be committed)

        sample.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Como podemos deducir de respuesta de estado, «sample.txt» actualmente no se está bajo seguimiento. Tendremos que añadir «sample.txt» en el «index» primeramente antes de poder registrar y trabajar con sus cambios.

Para hacerlo simplemente utilizaremos el comando «add», seguido por el que queremos añadir al «index». Si queremos añadir varios archivos al «index», podemos hacerlo separándolos con espacios.

```
$ git add <file1> <file2>
```

Si queremos añadir todos los ficheros de un directorio al «index» simplemente utilizaremos «.»

```
$ git add .
```

Ahora, vamos a comprobar que «sample.txt» se ha añadido con éxito al «index».

```
$ git add sample.txt
$ git status
On branch master
```

```
Initial commit

Changes to be committed:
  (use "git rm --cached ..." to unstage)

        new file:   sample.txt
```

Ahora que «sample.txt» se ha añadido al «index», podemos hacer un commit del archivo. Utilizaremos el comando «commit» tal y como se muestra a continuación.

```
$ git commit -m ""
```

Despues de ejecutar el commit comprobaremos el estado.

```
$ git commit -m "first commit"
[master (root-commit) 0742011] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 sample.txt

$ git status
On branch master
nothing to commit, working directory clean
```

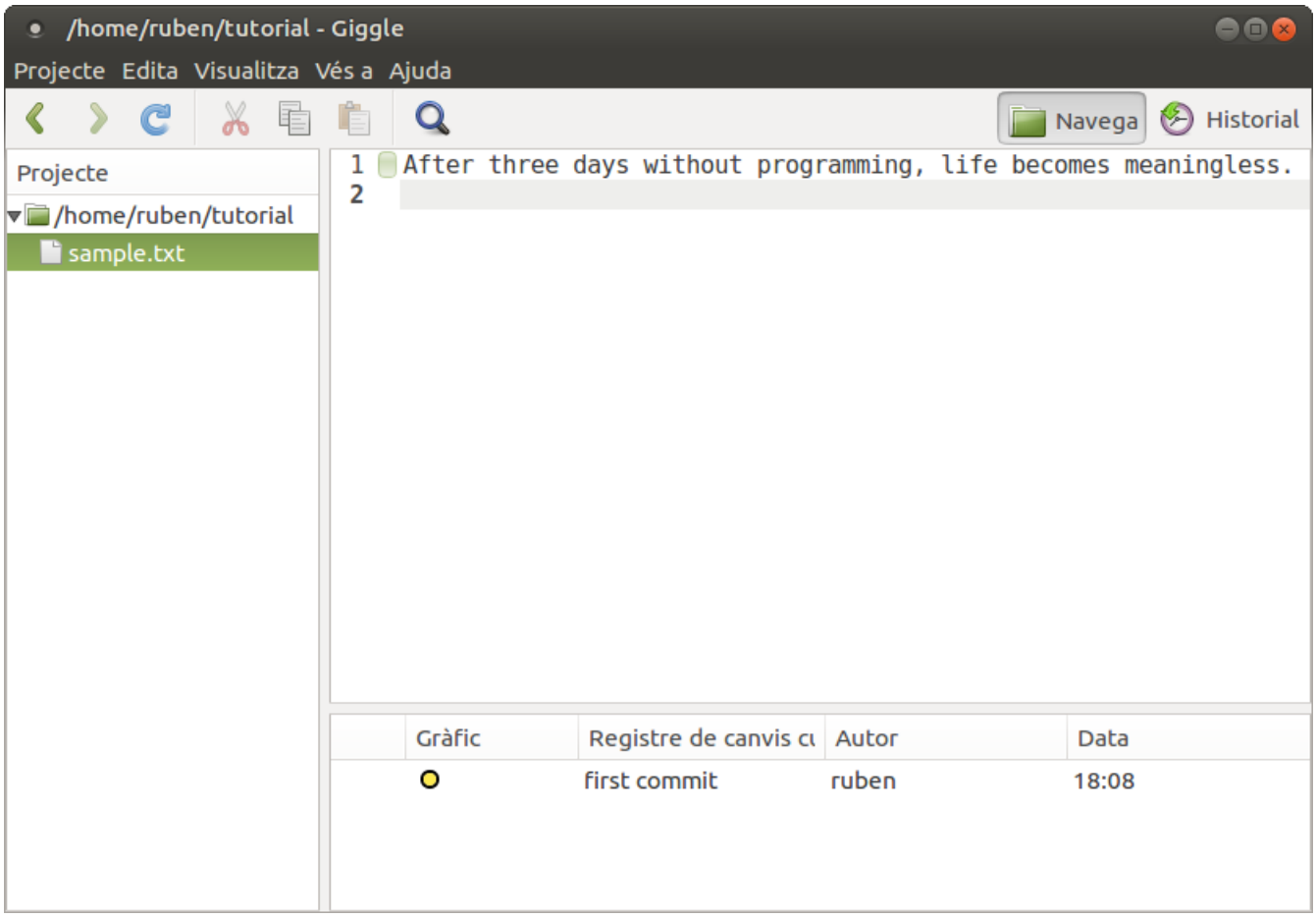
El resultado del comando «status» nos indica que no hay nuevos cambios a ser comiteados.

Podemos ver lo que se acaba de agregar mediante commit en el registro histórico del repositorio con el comando «log».

```
$ git log
commit 0742011aaf70e0e3a611fb22500c73d633f755c1
Author: someone <someone@gmail.com>
Date:   Fri Jan 30 18:08:42 2015 +0100

    first commit
```

Como referencia diremos que hay varios entorno gráficos que permiten realizar estas tareas. Por ejemplo tenemos Giggle



que podemos instalar fácilmente mediante los siguientes comando:

```
sudo apt-get install giggle giggle-terminal-view-plugin giggle-personal-details-plugin
```

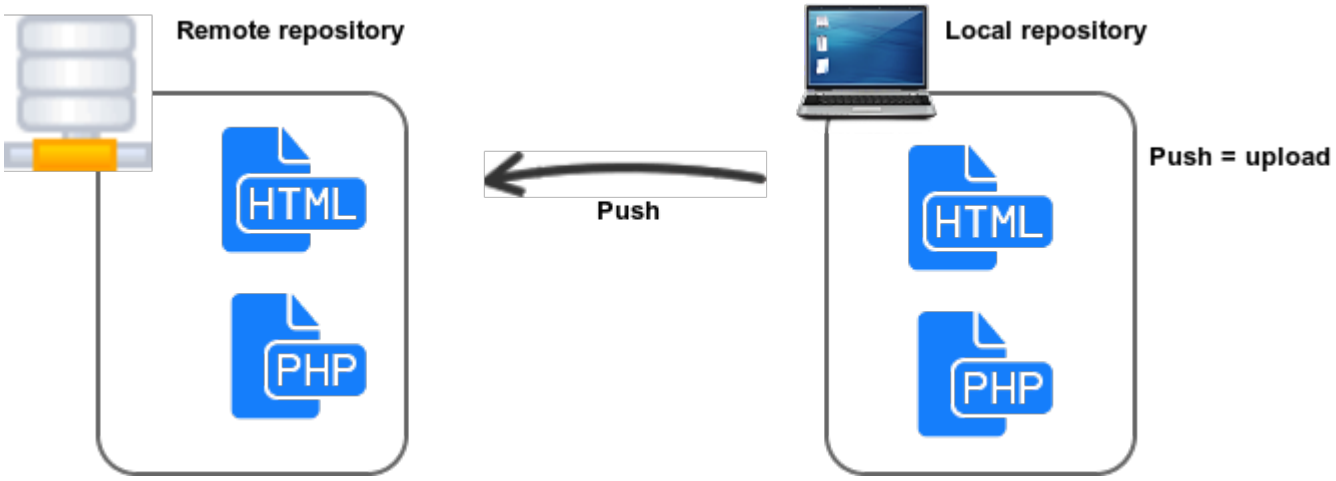
Trabajar con un repositorio remoto

Hacer push a un repositorio remoto

Hasta ahora, sólo hemos trabajado con repositorios locales. A partir de ahora trabajaremos con repositorios remotos que nos permitirán que compartir nuestros cambios con otros miembros del equipo.

Para empezar a compartir nuestros cambios, tenemos que subirlos a un repositorio remoto con el comando «push». Esto

hará que el repositorio remoto se actualice y sincronice con nuestro repositorio local.



Hacer clone de un repositorio remoto

Si ya existe un repositorio remoto, se puede recuperar una copia y guardarla en tu máquina local para y empezar a trabajar.

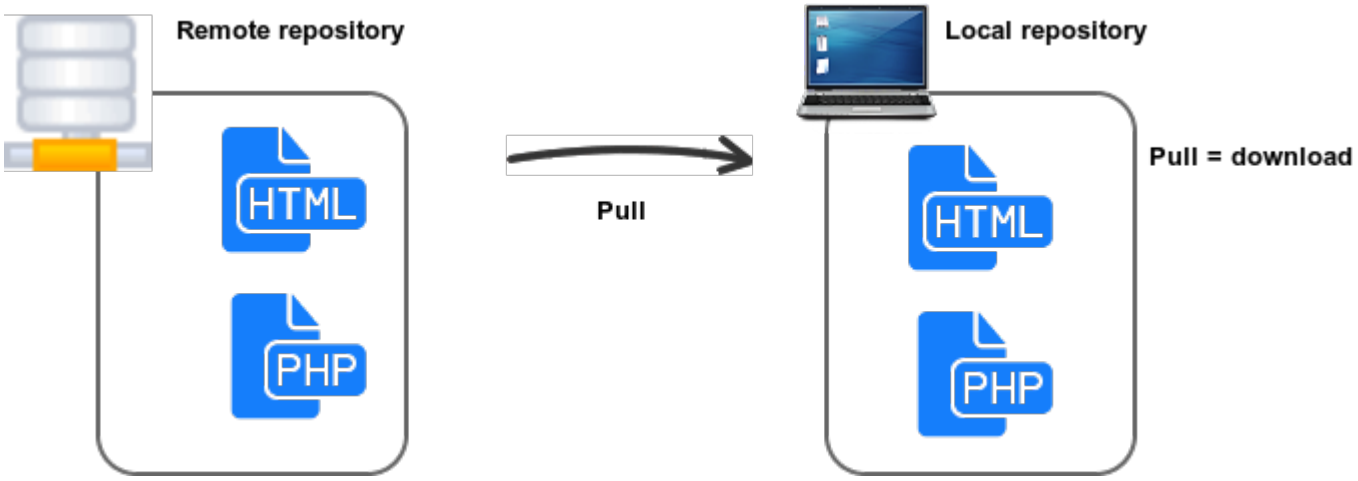
Se utiliza el comando «clon» para copiar un repositorio remoto. Por defecto, el comando «clon» sería configurar automáticamente una rama master en local a partir de la rama master remota des de la cual se ha realizado el clonado.

Un repositorio clonado tendrá el mismo registro histórico que el repositorio remoto. Se puede consultar y revertir cualquier de los commits en el repositorio local.

Hacer pull de un repositorio remoto

Cuando se realiza un «push» de los cambios realizados al repositorio remoto compartido, el repositorio local se queda desincronizado respecto a la última versión remota. Mediante Git, es bastante fácil de sincronizar el repositorio local con el repositorio remoto que ha sido actualizado.

Mediante una operación «pull» se pueden recuperar los cambios que se encuentran en el repositorio remotol. Cuando se ejecuta un «pull», la última revisión se descarga desde el repositorio remoto y se importa al repositorio local.



Observaciones

En este segundo post hemos visto como crear un repositorio trabajar con el, hacer un commit… Todo ello en local. Hemos introducido los conceptos necesarios para trabajar con un repositorio remoto, en el próximo post mostraremos paso a paso como hacerlo.

Ruben.