1º DAW. Entornos Desarrollo

Tema 3: Control de Versiones,
Optimización y Documentación

Práctica 3.1.1
Control de Versiones con git

Luis Garcia <lgarcia@ausiasmarch.net>

Introducción

El objetivo de la práctica es empezara familiarizarse con la gestión de repositorios con git desde línea de comandos. Para la mayor parte de la práctica, sólo necesitamos tener instalado git y un navegador con conexión a Internet para acceder a github, por lo que puede realizarse directamente desde los ordenadores del aula de informática, sin necesidad de ninguna otra máquina virtual.

En el último apartado, utilizaremos una segunda maquina para mostrar las posibilidades de colaboración y la resolución de conflictos. El segundo equipo será una MV con Ubuntu Desktop que hemos preparado como entorno de pruebas en la práctica anterior. Para instalar git, usaremos "apt-get install" contra los repositorios oficiales de la distribución.

1 Acceso local a un repositorio

1.0 Antes de empezar a hacer nada, inicializa las variables user.name y user.email usando valores reales apropiados en los comandos:

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

1.1 Crea un directorio ~/pruebas-git, y dentro de este un subdirectorio llamado "repo01"
Entra al directorio repo01 y crea los ficheros test01, test02 y test03 (pueden estar vacíos)
Crea un subdirectorio datafiles y (dentro de éste) el fichero datafiles/data.txt
Introduce algo de texto en test01

1.2 Inicializa un repositorio git en el directorio.
Agrega todos los archivos y haz un commit de los cambios con el comentario "Commit inicial".
Comprueba con git status que todos los ficheros y cambios se han guardado y muestra el historial de revisiones (solo una de momento!!!) con git log. Adjunta una captura.

1.3 Modifica los ficheros test01 y test02 y utiliza git diff para ver los cambios.
Haz un nuevo commit sin usar la opción "-a". Comprueba si se guardan los cambios.
Realiza finalmente el commit con la opción "-a" y el comentario "Nuevos cambios". Comprueba de nuevo que se guardan los cambios

1.4 Crea un nuevo archivo "nonsense.txt" y agregalo al control de versiones. Haz un commit con el comentario "Incorporado nuevo archivo"

1.5 Elimina el archivo del repositorio y consolida los cambios con el comentario "Eliminado el nuevo archivo". Comprueba que se guardan los cambios correctamente.

1.6 Edita el fichero datafiles/data.txt e introduce algo de texto. Guarda los cambios usando "data.txt actualizado" como comentario. Borra ahora el archivo "de manera accidental" en la copia de trabajo usando el comando "rm" (NO con "git rm"). ¿Que indica el comando git status ahora? Recupera el fichero usando:

```
git  checkout  datafiles/data.txt
```

Comprueba que no se han perdido los cambios.

1.7 Elimina el contenido de datafiles/data.txt pero sin borrar el archivo, puedes usar un editor o el comando:

```
:> datafiles/data.txt
```

Haz un commit usando “contenido de data.txt eliminado”. Comprueba que el contenido de data.txt se ha perdido.

1.8 Usa ahora git log para ver el número de revisión del commit justo anterior. Utilizando este número, recupera el contendido borrado con el comando:

git checkout NUM_REVISION datafiles/data.txt

Comprueba que se ha recuperado y guárdalo en un commit “Restaurado data.txt”

1.9 Elimina de nuevo el contenido de datafiles/data.txt, y además borra test01 con “git rm”. Haz persistentes los cambios en un commit con el comentario “Principio del fin”. Borra después test02 y haz otro commit “multiples ficheros perdidos”.

Utiliza git log para determinar el número de revisión ANTERIOR a “Principio del fin”. Vamos a usarlo para volver “temporalmente” a dicha revisión con el comando:

```
git checkout NUM_REVISION
```

Puedes comprobar que los ficheros se han recuperado, pero el repositorio está ahora en una situación “provisional” (Puedes comprobar con “git branch -a” que no hay ninguna rama activa)

1.10 Podemos crear una rama alternativa para guardar este estado. La llamaremos “estable” y la creamos con:

```
git checkout -b estable
```

Comprueba que se ha creado con “git branch -a” y que no hay cambios pendientes con git status

1.11 Vuelve a la rama principal con:

```
git checkout master
```

Puedes comprobar que siguen desaparecidos los ficheros que eliminamos en 1.9. Ahora, restauraremos el estado anterior pero de forma permanente en la propia rama “master”, para ello volveremos a necesitar el numero de revisión anterior a Principio del fin” y lo utilizaremos en el comando:

```
git reset --hard NUM_REVISION
```

Puedes comprobar que se han restaurado los ficheros y que además han desaparecido permanentemente las revisiones posteriores a la que hemos recuperado.

2 Aceso remoto a un repositorio

Vamos a probar a acceder a un repositorio remotamente. Este segundo apartado seguiremos realizándolo **desde nuestro ordenador del aula,** (excepto en los momentos en los que se indique otra cosa).

2.1 **Desde la MV crea un directorio ~/repo02** (directamente en el home, **NO dentro de pruebas-git**). Para que el repositorio pueda ser accedido remotamente sin problemas, debe crearse en un formato especial que no puede utilizarse como copia de trabajo. Para ello, entraremos en ~/repo02 e inicializaremos este segundo repositorio con:

```
git init --bare
```

2.2 **Desde el ordenador de clase**, crea un directorio en tu home llamado también ~/pruebas-git. Entra en el directorio, y descarga una copia del repositorio de la MV. El comando que usaremos será:

```
git clone ausias@192.168.56.30 :~/repo02
```

Con esa URI, estamos indicando a git que use ssh para acceder como usuario "ausias" al equipo 192.16.56.30 y que busque un repositorio en su home llamado repo02.

A partir de este momento, tenemos un nuevo repositorio "local" en nuestro ordenador. En este repositorio podemos hacer todo tipo de cambios, commits, etc … como en los ejercicios anteriores, pero el repositorio tiene la particularidad de "recordar" su origen remoto, lo que nos va a permitir además, "subir" los cambios al repositorio original y poderlos compartir asi entre varios equipos.

2.2 Entra en repo02 y crea un fichero denominado test-remoto. Haz el commit de este cambio usando "nuevo fichero test-remoto" como comentario.
Con el commit hemos consolidado la modificación, pero solo en nuestra copia local. Para subir los cambios usaremos "push". La sintaxis a utilizar para actualizar el "origen remoto" del repositorio sería:

```
git push --all
```

Con esto se copiarán los cambios al repositorio remoto (--all fuerza la creación de las nuevas ramas en el repositorio remoto, en nuestro caso es muy importante ya que el repositorio estaba inicialmente vacío). La actualización se realiza con el mismo protocolo con el que lo hemos descargado.

Modifica el fichero test-remoto y haz un commit "modificado test-remoto". Comprueba con git status, e indica que información te proporciona sobre el estado del repositorio local y del remoto.
Por último, sube los cambios de nuevo con "push".

2.3 Vuelve a la máquina virtual. Vamos a crear una copia de trabajo vinculada también a este repositorio. Para ello, entra al directorio ~/pruebas-git y clona el repo02, como está en un directorio local, no necesitamos ssh, basta con ejecutar:

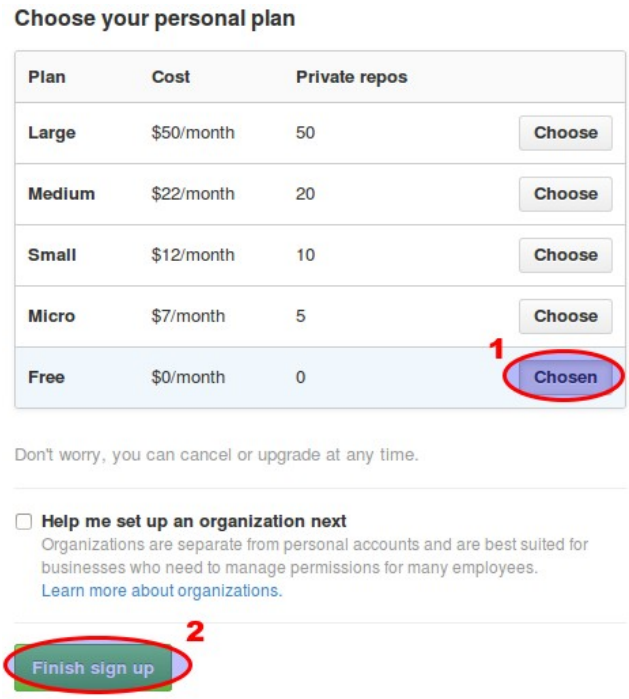
```
git clone ../repo02/
```

Modifica de nuevo test-remoto y haz un commit con el comentario “Cambios desde un ordenador remoto”

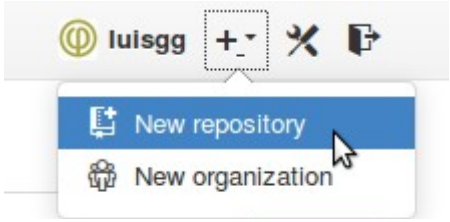
Vuelve al ordenador de clase e intenta actualizar tu copia de repo02 con “push”. ¿Se actualizan estos últimos cambios? ¿Que nos ha faltado en la copia de trabajo de la MV? Indica que es lo que nos hemos dejado por hacer, hazlo y comprueba como se actualiza la copia de la máquina real.

3 Repositorio en github

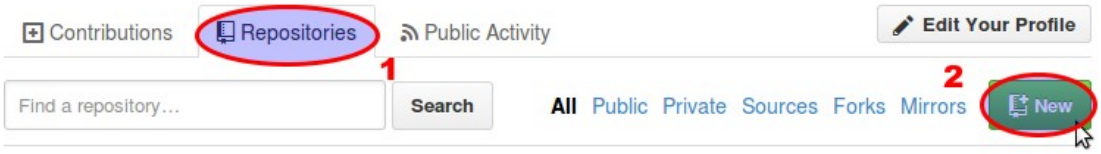
3.1 Accede a la página de github y crea una nueva cuenta para tu uso personal. Solo necesitarás una dirección de correo válida, elegir un nombre de usuario y una contraseña. **Es recomendable usar el mismo correo que usaste al configurar git** en tu equipo. Al elegir el plan, **verifica que SÓLO está seleccionada la opción gratuita** que únicamente permite repositorios públicos!!!



Una vez creada la cuenta, inicia sesión y crea un nuevo repositorio en tu cuenta. Puedes hacerlo o bien desplegando el menú que aparece al hacer clic sobre el “+” junto al nombre de usuario:

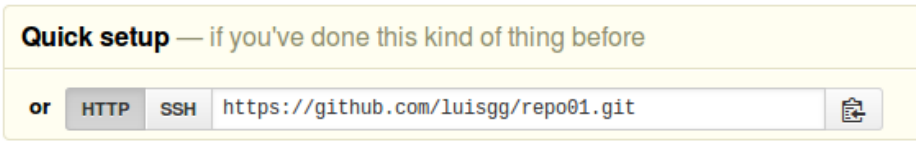


O bien activando la pestaña “Repositories” y pulsando en el botón “New”



Utiliza “repo01” como nombre del repositorio. La descripción es opcional pero puedes poner algo como “Prácticas de clase” o lo que prefieras.

Una vez creado el repo, se mostrará la URL mediante la que es accesible, que tendrá el siguiente aspecto (cambiando el nombre de usuario, claro):



Adjunta una captura como prueba de que lo has creado con éxito.

En la propia página que aparece tras crear el repositorio, se nos indica que tenemos dos posibilidades para usarlo, en función de si ya teníamos creado un repositorio local anteriormente o si vamos a empezar desde cero con el repositorio de github. En los ejercicios siguientes veremos ambos modos.

3.2 **Añadir el repositorio como origen remoto de un repositorio existente**

Entramos al directorio repo01 y ejecutamos (OJO, usar la url correcta!!!, no la del ejemplo!!!):

```
git remote add origin https://github.com/luisgg/repo01.git
```

Si ahora ejecutamos:

```
git push -u origin master
```

se nos pedirá el usuario y la contraseña de github. Si después volvemos a abrir el navegador en nuestra cuenta de github, comprobaremos que el repositorio se ha actualizado en internet y que contiene todo el historial de cambios.

Para los siguientes cambios, ya no será necesario usar “origin master” , bastará con ejecutar:

```
git push
```

O, mejor:

```
git push --all
```

3.3 **Utilizar el repositorio para empezar a trabajar desde cualquier ordenador.**

La ventaja de tener nuestro trabajo en un repositorio accesible por internet es la facilidad para descargarlo en cualquier equipo.
Crea ahora un nuevo directorio ~/copia-github, entra dentro de este directorio y descarga el repositorio (sustituyendo la URL por la tuya correcta) con:

```
git clone https://github.com/luisgg/repo01.git
```

Se habrña creado el subdirectorio repo01. Entra en el subdirectorio, modifica el fichero test02, haz un commit “modificado test02” y sube los cambios con

```
git  push –all
```

Vuelve al directorio de trabajo inicial ~/pruebas-git/repo01 y actualiza la copia con

```
git pull --all
```

Comprueba que obtienes los últimos cambios en test02.