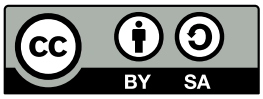
Guión de prácticas: Conceptos básicos

Antonio García Domínguez

4 de agosto de 2008



Índice

1. Clonar un repositorio existente

1. Antes de nada, necesitaremos un repositorio Git. Para ello, *clonaremos* un repositorio existente. Esto difiere del *checkout* de SVN o CVS en que conseguimos no una simple copia de trabajo que aún depende del repositorio central, sino un repositorio completamente independiente.

   Normalmente, los repositorios se clonan usando el protocolo Git (con URL que comienzan por `git://`) en caso de usar acceso anónimo o con acceso autenticado y encriptado mediante SSH (con direcciones de la forma `usuario@host:ruta-repositorio`). Se puede usar HTTP si tenemos problemas con cortafuegos y demás, pero es menos eficiente y requiere ciertos ajustes adicionales.

   Vamos a clonar el repositorio de Git:

   ```
   git clone git://git.kernel.org/pub/scm/git/git.git
   ```

Pregunta 1. *¿A qué se refiere Git por "objetos"?*

Pregunta 2. *¿Cuántos directorios* `.git` *hay en el repositorio creado?*

Pregunta 3. *¿Sabría decir para qué sirven los siguientes conjuntos de rutas del* `.git`*?*

- *config*
- *description*
- *HEAD*
- *hooks/**
- *refs/heads/**
- *refs/remotes/**
- *refs/tags/**

Pregunta 4. *¿Sabría decir cómo consigue Git una transferencia más eficiente de la información del repositorio, a partir de la salida obtenida? Recuerde los conceptos de objetos "sueltos" y objetos "empaquetados".*

2. Ahora vamos a hacer un clonado superficial, que sólo tomará lo necesario para los últimos *commits* que haya en el repositorio:

   ```
   git clone --depth 1 \
       git://git.kernel.org/pub/scm/git/git.git \
       git-superficial
   ```

Fíjese en cómo cambia el número de objetos a descargar.

Pregunta 5. *¿Qué problemas cree que puede presentar un clonado superficial?*

3. Ahora, por último, veremos cómo clonar un repositorio Subversion. Para ello, ejecute la siguiente orden:

```
git svn clone http://osl.uca.es/svn/ForjaDeSoftware
```

Pregunta 6. *¿Qué es lo que está haciendo realmente git-svn aquí?*

2. Navegar por el historial

1. Pruebe a ejecutar las siguiente órdenes dentro del repositorio `git`:

```
git log
git log -p
git log --stat
git log --name-status
git log --pretty=fuller
git log --pretty=oneline
git log -Shello
```

Pregunta 7. *¿Qué ha obtenido con la opción* `-p`*?*

Pregunta 8. *¿Qué significan los "+" y "-" de* `--stat`*?*

Pregunta 9. *¿Ha visto el formato de* `--name-status` *anteriormente?*

Pregunta 10. *¿Por qué cree que hay entradas separadas para Author y Commit con el formato* `fuller`*?*

Pregunta 11. *¿Para qué podría usar la salida de* `oneline`*? Pista: piense en tuberías.*

Pregunta 12. *Consulte la ayuda de* `git-log` *y explique para qué sirve la opción* `-S`*.*

Pregunta 13. *¿Cómo obtendría el historial de las últimas 10 revisiones?*

2. Ahora pruebe a ejecutar esto:

```
git cat-file -p HEAD
git cat-file -p master
```

Pregunta 14. *¿Podría responder ahora a cómo, si no hay números secuenciales de revisión, se mantiene la secuencia de commits?*

Pregunta 15. *Mire en la ayuda de rev-parse y averigüe dónde exactamente se guardan las referencias antes usadas.*

3. Veamos cómo consultar los *commits* por fecha:

```
git log master@{yesterday}
git log --since="2 days ago" --until="yesterday"
```

Pregunta 16. *¿Por qué no funciona la primera orden? Mire en la ayuda de* `rev-parse` *("SPECIFYING REVISIONS") para una pista.*

Pregunta 17. *¿Qué es lo que hace la segunda orden?*

4. Ahora probaremos algunas otras herramientas que nos da Git para navegar por el historial. En primer lugar, probaremos `gitk`, el visor Tcl/Tk incluido con Git.

```
gitk
gitk --all
```

Pregunta 18. *¿Qué diferencia nota entre la primera orden y la segunda?*

5. También podemos conseguir algo al estilo de `svn log`, con números secuenciales de revisiones mediante `-r`. Vaya al repositorio SVN `ForjaDeSoftware` y pruebe estas órdenes:

```
git svn log -r 180:190
git svn log --oneline --show-commit
git cat-file -p a74128c
```

Pregunta 19. *¿Cómo conoce Git los números de revisión originales?*

3. Examinar los objetos a bajo nivel

1. Pruebe a ejecutar lo siguiente:

```
git cat-file -p v1.5.6.4
git cat-file -p v1.5.6.4^{commit}
git cat-file -p v1.5.6.4^{tree}
git cat-file -p v1.5.6.4^{tree}:COPYING
```

Pregunta 20. *Cambie* `-p` *por* `-t` *en las anteriores órdenes. ¿Adivina qué es lo que hace la sintaxis* `^{tipo}`*?*

Pregunta 21. *¿Ha notado algo particular al final de la salida de la primera orden?*

2. Existen muchos sinónimos para lo que acaba de hacer, con ciertas diferencias en su salida. Por ejemplo, pruebe estas órdenes:

```
git show v1.5.6.4
git show v1.5.6.4^{commit}
git show v1.5.6.4^{tree}
git show v1.5.6.4:COPYING
git ls-tree v1.5.6.4
git ls-tree --name-only v1.5.6.4
```

Pregunta 22. *¿Cómo es que no hace falta indicar* `^{tree}` *en las tres últimas órdenes, si lo que se espera es un árbol?*