

elweb

This is the personal blog of Giovanni Collazo.

Git: Tutorial 1 (en Español)

En el post anterior hice una pequeña introducción a Git. Este post es el primero de varios *tutoriales* que me he comprometido a hacer relacionados a Git. Los ejemplos que voy a dar están centrados en el *workflow* que uso comúnmente en mi ambiente de desarrollo, Mac OS X pero funcionan igualmente en Windows y Linux.

Lo primero que hay que entender es que Git es un sistema distribuido. No hay un repositorio central. Todas las copias que existen de un proyecto son un *branch* del proyecto. Cada usuario que tiene una copia puede hacer cambios *commit* y cualquier otra operación sin afectar a ningún otro usuario. Mas poderoso aun es que como no es centralizado varios usuarios pueden hacer *merge* entre ellos sin afectar a mas nadie. Esto es central para el *workflow* usando Git.

Instalar Git

Para instalar git usé la version binaria que bajé desde el site de git (<http://git-scm.com/download>). La instalación fue muy simple, en unos minutos ya estaba en el terminal probando los *features*. Una vez instalado debes probar que esté funcionando. Para esto llamamos a Git desde el terminal a ver si responde.

```
> git
```

Este comando debe producir una lista de los comandos mas comunes que usa git. Ahora veamos que versión tenemos.

```
> git version
```

Esto debe producir el numero de versión que instalamos. En mi caso versión 1.6.5.2

Para buscar ayuda sobre git siempre puedes usar el comando de ayuda.

```
> git help log
```

```
0
```

```
> man git-log
```

Configurar las variables globales de git

Una vez te asegures que git esta funcionando es el momento de asignas las variables globales. Estas son unas variables que se le asignan a todos los repositorios que creas. Es importante mencionar que estas variables se pueden cambiar a proyectos por individual. Para configurar el nombre y correo electrónico hacemos lo siguiente.

```
> git config --global user.name "Tu Nombre Aquí"
```

```
> git config --global user.email tunombre@example.com
```

Crear un Repositorio



Para este ejemplo usaré un proyecto nuevo que cree en Ruby on Rails pero básicamente puede ser cualquier *folder* con algunos archivos dentro.

Voy a crear un proyecto de RoR pero puedes crear cualquier *folder* con algunos archivos de texto dentro.

> ***rails SampleProject***

> ***cd SampleProject***

En este punto me gusta crear un file que se llama `.gitignore` que es una lista de los files que git debe ignorar. Aquí solo voy a decirle a git que ignore un solo tipo de archivo los que se llamen `.DS_Store` que Mac OS pone en todos los directorios. Este file (`.gitignore`) lo puedes crear en cualquier aplicación pero en este ejemplo lo voy a hacer usando el *command line*.

> ***echo '.DS_Store' > .gitignore***

Una vez creado el `.gitignore` es momento de convertir este *folder* en un repositorio de git.

> ***git init***

Ahora hay que añadir los archivos a git.

> ***git add .***

Una vez añadidos los archivos hay que *commit* estos cambios e incluir un mensaje que nos deje saber de que se trata.

> ***git commit -m 'Import Inicial de archivos'***

Listo ya tienes el proyecto en repositorio de git.

Hacer cambios

Ahora la parte mas importante como hacer cambios a los archivos y hacer nuevos “commits”. Para ver el *status* del repositorio puedes hacer:

> ***git status***

En este punto el resultado debería ser: “*nothing to commit (working directory clean)*”. Para ver un listado de los *commit* que has hecho puedes:

> ***git log***

Esto debería producir una lista con los *commits* que se han hecho (en este caso solo uno) con los comentarios que añadimos. Además debes ver un *hash* de cada *commit*. Este *hash* es el numero de versión de ese *commit*. El hash se produce basado en la estructura del repositorio entero al momento del *commit*. Mas adelante hablaremos del *commit hash*.

Ahora voy a editar un file del proyecto en mi editor de texto favorito Textmate.

> *mate README*

Esto abre el file en Textmate. Borré todo el texto y solo añadí una linea que lee: "Sample Project 1". Luego *save* y cierro Textmate por el momento. Ahora que este archivo cambio vamos a ver como git refleja los cambios.

> *git status*

Esto debe indicar que el file README ha sido cambiado. Ahora añadiremos los cambios al repositorio y hacer el *commit*.

> *git add README*

Ahora podemos hacer le *commit*.

> *git commit -m 'Cambiamos el readme file'*

Listo, ya los cambios están en el repositorio. Ahora podemos ver en *log* que nuestro nuevo *commit*.



> *git log*

Esto debe producir una lista donde aparecen nuestros dos *commits*. Ahora vamos a editar nuevamente el file README para ver una forma alternativa de hacer el *add* el *commit* en un solo paso.

> *mate README*

Cambie el texto y *save*. Ahora verifico que git tomó los cambios.

> ***git status***

Esto debe decirte que el file README ha sido modificado. Ahora hacemos el *add* y *commit* con un solo comando.

> ***git commit -a -m 'Commit y add en un solo comando'***

Hasta aquí este *tutorial*. En el próximo discutiremos, *diffs* y *tags*. Si tienes alguna pregunta o sabes alguna forma de hacer esto mejor, escribe un comentario e iniciemos la conversación.

SHARE THIS:



📅 November 21, 2009 👤 Giovanni Collazo

Proudly powered by WordPress