Mantenimiento de repositorios con Git en Debian o Ubuntu

Pablo Vivar Colina

Julio 2018

1. Introducción

El software libre es genial, sobretodo sus sistemas operativos, los sistemas GNU/Linux vienen con muchas variaciones, pero las más utilizadas en el mercado son las basadas en Ubuntu y Debian.

2. Debian y Ubuntu

2.1. Debian

Debian es un sistema operativo (S.O.) libre, para su computadora. El sistema operativo es el conjunto de programas básicos y utilidades que hacen que funcione su computadora.

Debian ofrece más que un S.O. puro; viene con 51000 paquetes, programas precompilados distribuidos en un formato que hace más fácil la instalación en su computadora.

2.2. Ubuntu

Ubuntu viene con todo lo que necesitas para dirigir tu organización, escuela, hogar o empresa. Todas las aplicaciones esenciales, como una suite de oficina, navegadores, correo electrónico y aplicaciones multimedia vienen preinstaladas y miles de juegos y aplicaciones más están disponibles en el Centro de Software de Ubuntu.

Ubuntu siempre ha sido libre de descargar, usar y compartir. Creemos en el poder del software de código abierto; Ubuntu no podría existir sin su comunidad mundial de desarrolladores voluntarios.

Con un firewall incorporado y un software de protección antivirus, Ubuntu es uno de los sistemas operativos más seguros del mercado. Y las versiones de soporte a largo plazo le ofrecen cinco años de parches y actualizaciones de seguridad.

La informática es para todos, independientemente de su nacionalidad, sexo o discapacidad. Ubuntu está totalmente traducido a más de 50 idiomas e incluye tecnologías de asistencia esenciales. [1]Traducción realizada con el traductor DeepL [2]

2.3. Superusuario

El superusuario en sistemas libres es una "herramienta"que nos permite realizar ciertas acciones con más permisos en la computadora que los que podriamos realizar con los permisos por defecto

1. sudo <comando>*Ubuntu*
2. su (acceso terminal root) *Debian*

2.4. Navegación Directorios

En Debian y Ubuntu con los entornos de escritorios por defecto tenemos acceso a navegación de directorios en entornos gráficos, por ejemplo en Ubunto con gnome tenemos el navegador *nautilus* y en Debian con XFCE tenemos el navegador *thunar*, éste último podemos accesar a él a través de invocarlo con su nombre desde la terminal.

Para navegar entre carpetas en la terminal podemos utilizar los siguientes comandos.

1. ls (muestra el contenido en la carpeta en la cual nos encontramos)
2. cd <carpeta>(accesa a carpeta)
3. cd .. (sale de carpeta)

Es importante mencionar que con la tecla <TAB>podemos autocompletar nombres de directorios, programas, accesos, etc. y que con el comando cd podemos accesar a un directorio (lejano) usando diagonales </>por ejemplo:

$$cdusuario/Documentos/Repositorio \tag{1}$$

Podemos notar que en 1 se utiliza una dirección directa, con esa única línea de comando ya nos podemos localizar en el directorio deseado.

3. Repositorios en Git

3.1. Introducción

Git es un sistema de control de versiones distribuido, gratuito y de código abierto, diseñado para gestionar todo tipo de proyectos, desde pequeños hasta muy grandes, con rapidez y eficacia.

Git es fácil de aprender y tiene una huella diminuta con un rendimiento increíblemente rápido. Supera a las herramientas SCM como Subversion, CVS, Perforce y ClearCase con características como sucursales locales baratas, áreas de preparación convenientes y múltiples flujos de trabajo. [3] Traducción realizada con el traductor DeepL. [2]

3.2. Trabajando con Git

3.2.1. Directorio de Git

El directorio.git es donde Git almacena los metadatos y la base de datos de objetos para el repositorio.

3.2.2. Directorio de trabajo

Una copia de una versión del proyecto git, tomada de la base de datos comprimida en el directorio.git

3.2.3. Área de puesta a disposición/Índice

Archivo que almacena información sobre qué se va a confirmar en el repositorio git

3.2.4. Configurar herramientas

git config –global user.name "[nombre]"
Establece el nombre que desea adjuntar a sus transacciones de confirmación.

git config –global user.email "[dirección de correo electrónico]"
Establece el correo electrónico que desea adjuntar a sus transacciones de confirmación.

3.2.5. Realizar modificaciones

git status
Enumera todos los archivos nuevos o modificados que se van a confirmar.

git diff
Muestra las diferencias de archivos que aún no se han preparado.

git add [archivo] Instantáneas del archivo en preparación para el versionado (la bandera $--all$ añade todos los cambios nuevos).

git diff $--etapas$ Muestra las diferencias de archivo entre la puesta a disposición y la última versión del archivo.

git reset[archivo]
Desescribe el archivo, pero conserva su contenido.

git commit -m "[mensaje descriptivo]"
Registra las instantáneas del archivo permanentemente en el repositorio.

3.2.6. Crear Repositorios

git init[nombre del proyecto]
Crea un nuevo repositorio local con el nombre especificado.

git clone[url]
Descarga un proyecto y toda su historia de versiones.

3.2.7. Sincronizar cambios

git push
Sube todos los compromisos de las sucursales locales al repositorio.

git pull
Descarga el historial de marcadores e incorpora cambios.

[3]Traducción realizada con el traductor DeepL [2].

4. GitLab

Para el mantenimiento de repositorios se estará usando la plataforma GitLab, ya que brinda muchas herremientas adicionales, como levantar issues, comunicación entre usuarios, etc.

5. Conclusión

En la creación y mantenimiento de repositorios de los comandos anteriormente mencionado se hará uso más frecuente de los siguientes y uno adicional para la creación de repositorios

1. git pull
2. git status
3. git add [archivo] (con bandera –all)
4. git commit -m "[mensaje descriptivo]"
5. git push

Referencias

[1] Canonical. *Ubuntu Desktop.*

[2] DeepL. *Traduccion.*

[3] Git. *Git –everything-is-local.*