Instalar Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.
Hay distribuciones de Git para sistemas Linux y POSIX en el sitio oficial Git SCM.

Git para toda plataforma: http://git-scm.com

Configurar herramientas

Configura la información del usuario para todos los repositorios locales

- Establecer el nombre que estará anexado a los commits
$ git config --global user.name "Your Name"

- Establecer el email que estará anexado a los commits
$ git config --global user.email "your-email@correo.com"

- Habilitar la coloración en la línea de comandos
$ git config --global color.ui auto

Crear repositorios

Inicia un nuevo repositorio u obtiene uno de una URL existente

- Crear un nuevo repositorio local con el nombre especificado
$ git init "Nombre del proyecto"

- Descargar un proyecto y toda su historia de versión
$ git clone "URL"

www.portallinux.es/librodetrucos

Libro de trucos

Para saber como usar la herramienta Git

Efectuar cambios

Revisa las ediciones y elabora una transacción de commit

-Enumerar todos los archivos nuevos o modificados que se deben confirmar
$ git status

- Mostrar las diferencias de archivos que no se han enviado aún al área de espera
$ git diff

- Tomar una instantánea del archivo para preparar la versión
$ git add "archivo"

- Mostrar las diferencias del archivo entre el área de espera y la última versión del archivo
$ git diff --staged

- Mover el archivo del área de espera, pero preserva su contenido
$ git reset "archivo"

- Registrar las instantáneas del archivo o de los archivos permanentemente en el historial de versión
$ git commit -m "mensaje" -a

Cambios grupales

Nombra una serie de commits y combina esfuerzos ya culminados

- Enumerar todas las ramas del repositorio actual
$ git branch

-Crear una nueva rama
$git branch "nombre-de-la-rama"

www.cshluesocc.org

Creado con Inkscape

-> Esta nota se tiene que mantener <-

Si quieres la plantilla envíame un correo!
Se permite la copia de "Libro de trucos" en cualquier formato, ya sea sin ánimo de lucro o con fines comerciales, siempre y cuando no se modifique su contenido, se respete su autoría y esta nota se mantenga. Gracias por respetar mi trabajo. PortalLinux.es

Contacto
Twitter: @zaguirito
Facebook: Buscar por PortalLinux
Correo: zaguriblog@gmail.com
DeviantART: zagur.deviantart.com

Libro de Trucos v1.3

Sobre "Libro de trucos"

"Libro de trucos" es una creación de Jesús Camacho, autor y editor principal de PortalLinux.

www.portallinux.es/librodetrucos

"Libro de trucos" es similar a los famosos cubos de trucos pero con un formato distinto. Parte de la idea del autor que está cansado de que se le acaben achafando los cubos o no sepa donde guardarlos. Con el fin de solventar este problema, ha decidido pasar los famosos cubos en un formato más agradable para el usuario ya que se puede guardar en el bolsillo sin ningún problema.

El formato libro pequeño es perfecto para guardarlo y tenerlo siempre encima.

Para buscar más información de los comandos usa el comando man NombreDelComando para entrar en el manual del comando.

Cambios grupales

- Cambiar a la rama especificada y actualiza el directorio activo
$ git checkout "nombre-de-la-rama"

- Combina el historial de la rama especificada con la rama actual
$ git merge "rama"

- Borrar la rama especificada
$ git branch -d "nombre-de-la-rama"

Nombre del archivo de refactorización

Reubica y retira los archivos con version

- Borrar el archivo del directorio actine y pone en el área de espera el archivo borrado
$ git rm "archivo"

- Retira el archivo del control de versiones, pero preserva el archivo a nivel local
$ git rm --cached "archivo"

- Cambia el nombre del archivo y lo prepara para el commit
$ git mv "archivo-original" "archivo-renombrado"

Suprimir tracking

Excluye los archivos temporales y las rutas

- Un archivo llamado ".gitignore" suprime la creacion accidental de versiones de archivos y rutas que concuerdan con los patrones espeficicados, ej. del contenido del archivo .gitignore:
*.log
build/
temp-*

www.portallinux.es/librodetrucos

www.cshluesocc.org

Comunidad de Software y Hardware Libre
Twitter: @cshluesocc
Facebook: fb.me/cshluesocc
Email: cshluesocc4@gmail.com

Sincronizar cambios en un repositorio fork

Cuando se trabaja con forks es importante saber como sincronizar nuestro repositorio local con el repositorio remoto original.

1- Agregar el repositorio padre como origen remoto
$ git remote add "alias" "URL del repositorio padre"

2- Obtener los cambios que hay en el repositorio original pero sin mezclarlo en nuestro repositorio
$ git fetch "alias"

3- Mezclar la rama especificada del repositorio padre a nuestra rama local
$ git merge "alias"/"rama"

Ejemplo:
Nos situamos dentro del directorio de nuestro proyecto:
$ cd proyecto

Paso 1:
$ git remote add upstream https://github.com/user/proyecto.git

Paso 2:
$ git fetch upstream

Paso 3:
$ git merge upstream/master

Suprimir tracking

- Enumerar todos los archivos ignorados en este proyecto
$ git ls-files --other --ignored --exclude-standard

Guardar fragmentos

Almacena y restaura cambios incompletos

- Almacena temporalmente todos los archivos tracked modificados
$ git stash

- Restaurar los archivos guardados más recientemente
$ git stash pop

- Enumerar todos los sets de cambios en guardado rápido
$ git stash list

- Eliminar el set de cambios en guardado rápido más reciente
$ git stash drop

Repasar historial

Navega e inspecciona la evolución de los archivos de proyecto

- Enumerar el historial de la versión para la rama actual
$ git log

- Enumera el historial de versión para el archivo, incluido los cambios de nombre
$ git log --follow "archivo"

www.cshluesocc.org

www.portallinux.es/librodetrucos

- Descarga el historial del marcador e incorpora los cambios
$ git pull

- Carga todos los commits de la rama local al repositorio remoto
$ git push "alias" "rama"

- Combina la rama del marcador con la rama local actual
$ git merge "marcador"/"rama"

- Descarga todo el historial del marcador del repositorio
$ git fetch "marcador"

Registrar un marcador de repositorio e intercambiar historial de version

Sincronizar cambios

- Deshecha todo el historial y regresa al commit especificado
$ git reset --hard "commit"

- Deshace todos los commits despues de "commit", preservando los cambios locales
$ git reset "commit"

Borra errores y elabora historial de reemplazo

Rehacer commits

- Produce metadatos y cambios de contenido del commit especificado
$ git show "commit"

- Muestra las diferencias de contenido entre dos ramas
$ git diff "rama-uno" "rama-dos"

Repasar historial