Manual básico de Git

Actualmente es necesario usar un software de control de versiones para el desarrollo colaborativo, sobre todo cuando se habla de proyectos grandes.

Un software de control de versiones es un grupo de aplicaciones para gestionar los cambios en el código fuente, dando la posibilidad de poder regresar a versiones viejas del mismo código fuente si así es requerido.

Esto se logra con un historial de versiones que va guardando el software en un repositorio, que puede ser una carpeta o conjunto de carpetas donde se almacenan estas versiones o cambios del proyecto y al cual tienen acceso los desarrolladores del mismo.

El control de versiones es una potente herramienta colaborativa.

La necesidad de usar un software de control de versiones es que, el trabajo en equipo es muy complicado, sobre todo si se esta trabajando remotamente; si dos personas están modificando el mismo archivo, solo se preservaran los cambios de la última persona que guarde el archivo.

El software de control de versiones, facilita este trabajo en equipo, permitiendo que se varias personas puedan trabajar un mismo archivo al mismo tiempo evitando que se pierdan los cambios.

Ejemplos de software de control de versiones son: CVS, Tortoise, CVS, Clear Case, SourceSafe, etc

**Virtudes de Git**

Git es un software creado por Linus torvalds, el desarrollador del núcleo del S.O. Linux, tiene una curva de aprendizaje baja, es rápido, es eficiente al gestionar proyectos grandes, es eficiente , open source y permite el uso de ramas (branch).

2.1

Comandos básicos de la consola de comandos de Windows msdos.

En este post recogemos los **principales comandos**utilizados por el antiguo sistema operativo **DOS**, comandos que hoy en día siguen resultando imprescindibles para muchas aplicaciones o para la creación de archivos por lotes para automatizar procesos.

Existen muchos más comandos útiles que recogeremos a lo largo del tiempo detallando su utilidad y aplicación. Por el momento os dejamos estos para terminar de perder el miedo a las tan temidas pantallas negras.

* **COMANDO:**MD
* **ACCIÓN:** Crea directorios
* **SINTAXIS:** MD [unidad:\ruta\] <nombre>

* **COMANDO:**RD
* **ACCIÓN:** Borra directorios (solo si se encuentra vacío)
* **SINTAXIS:** RD [unidad:\ruta\] <nombre>

* **COMANDO:** CD ; CD.. ; CD\
* **ACCIÓN:** Sirve para moverse por los distintos directorios
* **SINTAXIS:** CD <nombre de la carpeta> , CD.. , CD\

* **COMANDO:**TREE
* **ACCIÓN:** Muestra la el árbol de directorios de la unidad
* **SINTAXIS:** TREE [Unidad:\ruta]
* **PRINICPALES MODIFICADORES:**/F Presenta los nombres de los archivos que contiene cada directorio.  /A Utiliza caracteres de texto en lugar de caracteres gráficos.

* **COMANDO:** DELTREE
* **ACCIÓN:** Borra el directorio aunque no esté vacío.
* **SINTAXIS:** DELTREE [unidad:\ruta\] <nombre>
* **PRINICPALES MODIFICADORES:**/Y No pide confirmación antes de borrar.

* **COMANDO:**DIR
* **ACCIÓN:** Muestra el contenido de un directorio
* **SINTAXIS:** DIR [unidad:\directorio]
* **PRINICPALES MODIFICADORES:**/P Pausa la pantalla para poder ver todo. /S Muestra también subdirectorios.

* **COMANDO:**MOVE
* **ACCIÓN:** Mueve ficheros de un directorio a otro
* **SINTAXIS:** MOVE <origen> <destino>
* **PRINICPALES MODIFICADORES:**/Y No pregunta por la confirmación de reemplazo

* **COMANDO:**TYPE
* **ACCIÓN:** Visualiza archivos de texto
* **SINTAXIS:** TYPE [Unidad:\ruta\Archivo]

* **COMANDO:**XCOPY
* **ACCIÓN:**Copia directorios enteros
* **SINTAXIS:** XCOPY <Origen> <destino>
* **PRINICPALES MODIFICADORES:**/P Pide confirmación antes de copiar cada archivo. /S La copia debe ser extensiva a todos los directorios (los vacíos no). /E Copia también los directorios Vacíos.

* **COMANDO:**REN / RENAME
* **ACCIÓN:**Renombra ficheros
* **SINTAXIS:** RENAME <nombre actual> <nombre nuevo>

* **COMANDO:**DEL
* **ACCIÓN:**Elimina ficheros
* **SINTAXIS:** DEL [Unidad:\ruta\Archivo]
* **PRINICPALES MODIFICADORES:**/P Pide confirmación.

* **COMANDO:**UNDELETE
* **ACCIÓN:**Recupera archivos borrados con DEL
* **SINTAXIS:** UNDELETE <fichero>
* **PRINICPALES MODIFICADORES:**/LIST Presenta una lista de los archivos recuperables. /ALL Recupera todos los archivos.

* **COMANDO:**FC
* **ACCIÓN:**Compara que 2 ficheros sean iguales
* **SINTAXIS:** FC fichero1 fichero2

* **COMANDO:**EDIT
* **ACCIÓN:**Editor de texto
* **SINTAXIS:** EDIT <fichero>

* **COMANDO:**FORMAT
* **ACCIÓN:**Formatea unidades
* **SINTAXIS:** FORMAT <Unidad:>
* **PRINICPALES MODIFICADORES:**/Q  Da un formato rápido. /S  Copia al disco los archivos de sistema.

* **COMANDO:**SYS
* **ACCIÓN:**Copia el sistema a otra unidad
* **SINTAXIS:** SYS <Unidad:>

* **COMANDO:**CHKDSK
* **ACCIÓN:**Chequea el disco y nos muestra información sobre él.
* **SINTAXIS:** CHKDSK [Unidad:][fichero]

* **COMANDO:**DISKCOPY
* **ACCIÓN:**Copia el contenido total de un disco, sirve para hacer copias de seguridad.
* **SINTAXIS:** DISKCOPY <Unidad origen> <Unidad destino>

* **COMANDO:**CLS
* **ACCIÓN:**Limpia la pantalla
* **SINTAXIS:** CLS

* **COMANDO:**VER
* **ACCIÓN:**Nos muestra la versión de MSDOS que utilizamos
* **SINTAXIS:** VER

* **COMANDO:**DATE
* **ACCIÓN:**Muestra y cambia la fecha
* **SINTAXIS:** DATE

* **COMANDO:**TIME
* **ACCIÓN:**Muestra y cambia la hora
* **SINTAXIS:** TIME

* **COMANDO:**FIND
* **ACCIÓN:**Busca una cadena específica de texto
* **SINTAXIS:** FIND [modificadores] “Cadena” [Unidad:\ruta\archivo]  (“Cadena”= Texto a buscar)
* **PRINICPALES MODIFICADORES:**/V Presenta todas las líneas que no contengan la cadena especificada. /I   Indica que no se haga distinción entre mayúsculas y minúsculas.

* **COMANDO:**LABEL
* **ACCIÓN:**Cambia la etiqueta del disco
* **SINTAXIS:** LABEL

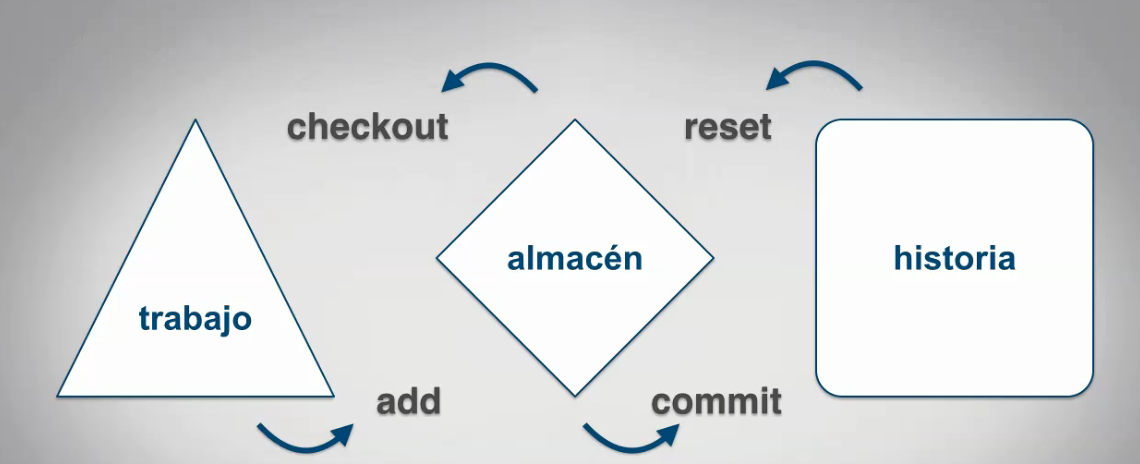
* **COMANDO:**VOL
* **ACCIÓN:**Muestra el número de serie del disco o unidad
* **SINTAXIS:** VOL

* **COMANDO:**ATTRIB
* **ACCIÓN:**Asocia a los archivos distintos atributos
* **SINTAXIS:** ATTRIB [Modificadores] <Fichero>

Git se puede descargar desde la url <https://git-scm.com>, la ventaja de Git es que existen versiones para varios sistemas operativos.

La versión de Git para Windows, contiene una consola “tipo Linux” para realizar el trabajo con Git en ella.

Diagrama de Git



Este es un diagrama básico de como trabaja Git; el área de trabajo o working directory, es donde editamos nuestro proyecto.

El almacén también es llamado staging area y es donde se preparan los archivos con el comando git add, antes de ser confirmado con el comando git commit y que se guarde la versión en el repositorio.

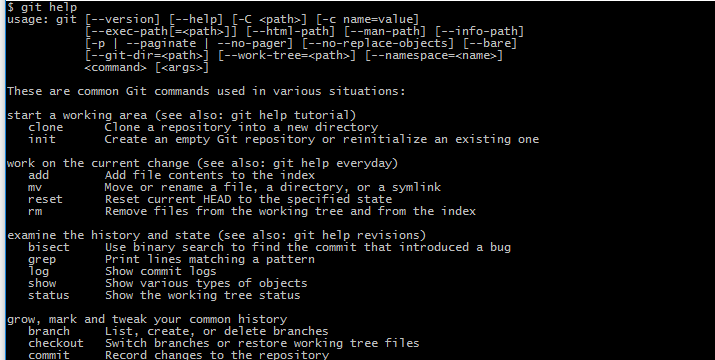
Una vez que se le da commit a los archivos se almacenan sus versiones en el repositorio, ahí podemos regresar a versiones anterior en el staging area.

Al área de trabajo, almacén y a la parte donde se almacena el historial ,se les conoce como los tres estados de trabajo.

Comando de Git.

Git help *comando de Git*

Te muestra ejemplos y una explicación del comando de Git, es el comando de ayuda.



Git config

te permite obtener y establecer variables de configuración, que controlan el aspecto y funcionamiento de Git.

Git config –l

Para ver la configuración actual.

Los siguientes comandos son importantes al instalar git, ya que son datos que aparecen al hacer commit.

$ git config --global user.name "John Doe"

Para Establecer nombre de usuario al usar git

$ git config --global user.email johndoe@example.com

Para establecer tu correo en la configuración de git

También podemos configurar el editor que se usa por default para Git, el alias, etc.

Todos estos parámetros de configuración se guardan en el archivo de gitconfig.

Glosario de Git

Git init: Inicializa un repositorio nuevo (creamos un repositorio en la carpeta actual para comenzar a trabajar).

Se crea la carpeta oculta .git, con todos los archivos de configuración requeridos para comenzar a trabajar.

Git clone dirección: Nos traemos una copia de un repositorio existente a nuestra carpeta local para trabajar en ella.

Git add nombre del archivo [opcional .]: Se preparan archivos para subir al repositorio, se suben al staging area hasta que el commit los sube al repositorio.

Con el comando opcional [.], se suben todos los archivos disponibles para ser subidos de tu carpeta de working area.

Git commit

Es el comando para “confirmar” los cambios añadidos y preparados con el comando add en el staging area, al ejecutar el comando se guarda la versión en el repositorio y se pide que se guarde un mensaje que describa los cambios guardados en el archivo confirmado.

Commit es un comando que tiene muchos subcomandos que se pueden combinar.

*git commit* [-a | --interactive | --patch] [-s] [-v] [-u<mode>] [--amend]

[--dry-run] [(-c | -C | --fixup | --squash) <commit>]

[-F <file> | -m <msg>] [--reset-author] [--allow-empty]

[--allow-empty-message] [--no-verify] [-e] [--author=<author>]

[--date=<date>] [--cleanup=<mode>] [--[no-]status]

[-i | -o] [-S[<keyid>]] [--] [<file>…​]

Los más usados son:

Git commit –m

Se guarda en el repositorio el archivo y se le añade un mensaje.

Git commit –am

Se guardan todos los archivos preparados y se añade un comentario al commit.

*git rm* [-f | --force] [-n] [-r] [--cached] [--ignore-unmatch] [--quiet] [--] <file>…​

Se borra un archivo o carpeta del working directory y del stage.

Con –cached se borra del repositorio pero se deja en el working area.

git mv [-v] [-f] [-n] [-k] <source> <destination>

Mover o renombrar un archivo.

Git status: Informa el estado del árbol de trabajo.

Git diff: Muestra los cambios que están en la working area y que no están en el stage.