GIT

¿Qué es Git?

Git es un sistema de control de versiones distribuido (scvd) escrito en C. Un sistema de control de versiones permite la creación de una historia para una colección de archivos e incluye la funcionalidad para revertir la colección de archivos a otro estado. Otro estado puede significar a otra colección diferente de archivos o contenido diferente de los archivos.

Por ejemplo, cambiar la colección de archivos a un estado de 2 días atrás, o puedes cambiar entre los estados para características experimentales y problemas de producción.

Esta colección de archivos generalmente es llamada "código fuente". En un sistema de control de versiones distribuido todos tienen una copia completa del código fuente (incluyendo la historia completa del código fuente) y puede realizar operaciones referidas al control de versiones mediante esa copia local. El uso de un scvd no requiere un repositorio central.

Si haces cambios al código fuente, haces esos cambios relevantes para el control de versiones (los agregas al staging index) y después los agregas al repositorio (commit).

Git mantiene todas las versiones. Por lo tanto puedes revertir a cualquier punto en la historia de tu código fuente usando Git.

Git realiza los commits a tu repositorio local y es posible sincronizar ese repositorio con otros (tal vez remotos) repositorios. Git te permite clonar los repositorios, por ejemplo, crear una copia exacta de un repositorio incluyendo la historia completa del código fuente. Los dueños de los repositorios pueden sincronizar los cambios vía push (transfiere los cambios al repositorio remoto) o pull (obtiene los cambios desde un repositorio remoto).

Git soporta el concepto de branching, es decir, puedes tener varias versiones diferentes de tu código fuente. Si quieres desarrollar una nueva característica, puedes abrir un branch en tu código fuente y hacer los cambios en este branch sin afectar el principal desarrollo de tu código.

Git puede ser usado desde la línea de comandos, también existen herramientas gráficas.

CREA UN REPOSITORIO NUEVO

Crea un directorio nuevo, ábrelo y ejecuta

```
git init
```

Para crear un nuevo repositorio de git.

HACER CHECKOUT A UN REPOSITORIO

Crea una copia local del repositorio ejecutando

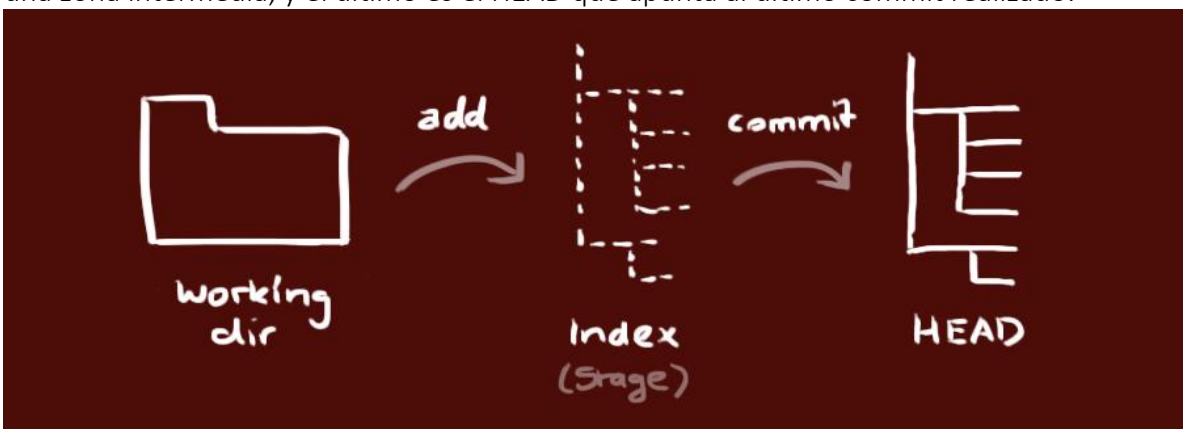
```
git clone /path/to/repository
```

Si utilizas un servidor remoto, ejecuta

```
git clone username@host:/path/to/repository
```

FLUJO DE TRABAJO

Tu repositorio local está compuesto por tres "árboles" administrados por git. El primero es tu Directorio de trabajo que contiene los archivos, el segundo es el Índex que actúa como una zona intermedia, y el último es el HEAD que apunta al último commit realizado.



ADD & COMMIT

Puedes registrar cambios (añadirlos al Index) usando

```
git add <filename>
git add .
```

Este es el primer paso en el flujo de trabajo básico. Para hacer commit a estos cambios usa

```
git commit -m "Commit message"
```

Ahora el archivo esta incluido en el HEAD, pero aún no en tu repositorio remoto.

ENVÍO DE CAMBIOS

Tus cambios están ahora en el HEAD de tu copia local. Para enviar estos cambios a tu repositorio remoto ejecuta

```
git push origin master
```

Reemplaza master por la rama a la que quieres enviar tus cambios.

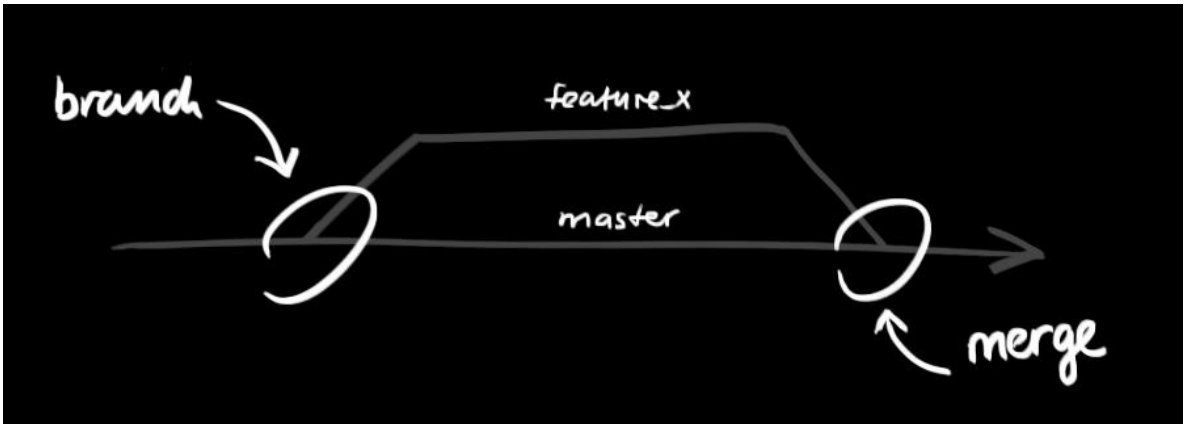
Si no has clonado un repositorio ya existente y quieres conectar tu repositorio local a un repositorio remoto, usa

```
git remote add origin <server>
```

Ahora podrás subir tus cambios al repositorio remoto seleccionado.

RAMAS

Las ramas son utilizadas para desarrollar funcionalidades aisladas unas de otras. La rama master es la rama "por defecto" cuando creas un repositorio. Crea nuevas ramas durante el desarrollo y fusiónalas a la rama principal cuando termines.



Crea una nueva rama llamada "feature_x" y cámbiate a ella usando

```
git checkout -b feature_x
```

Vuelve a la rama principal y borra la rama

```
git checkout master
git branch -d feature_x
```

Una rama nueva no estará disponible para los demás a menos que subas (push) la rama a tu repositorio remoto

```
git push origin <branch>
```

ACTUALIZA & FUSIONA

Para actualizar tu repositorio local al commit más nuevo, ejecuta

```
git pull
```

En tu directorio de trabajo para bajar y fusionar los cambios remotos. Para fusionar otra rama a tu rama activa (por ejemplo master), utiliza

```
git merge <branch>
```

En ambos casos git intentará fusionar automáticamente los cambios. Desafortunadamente, no siempre será posible y se podrán producir conflictos. Tú eres responsable de fusionar esos conflictos manualmente al editar los archivos mostrados por git. Después de modificarlos, necesitas marcarlos como fusionados con

```
git add <filename>
```

Antes de fusionar los cambios, puedes revisarlos usando

```
git diff <source_branch> <target_branch>
```

ETIQUETAS

Se recomienda crear etiquetas para cada nueva versión publicada de un software. Este concepto no es nuevo, ya que estaba disponible en SVN. Puedes crear una nueva etiqueta llamada 1.0.0 ejecutando

```
git tag 1.0.0 1b2e1d63ff
```

1b2e1d63ff se refiere a los 10 caracteres del commit id al cual quieres referirte con tu etiqueta. Puedes obtener el commit id con

```
git log
```

También puedes usar menos caracteres que el commit id, pero debe ser un valor único.

REEMPLAZA CAMBIOS LOCALES

En caso de que hagas algo mal (lo que seguramente nunca suceda ;) puedes reemplazar cambios locales usando el comando

```
git checkout -- <filename>
```

Este comando reemplaza los cambios en tu directorio de trabajo con el último contenido de HEAD. Los cambios que ya han sido agregados al Index, así como también los nuevos archivos, se mantendrán sin cambio.

Por otro lado, si quieres deshacer todos los cambios locales y commits, puedes traer la última versión del servidor y apuntar a tu copia local principal de esta forma

```
git fetch origin
git reset --hard origin/master
```

DATOS

Interfaz gráfica por defecto

```
gitk
```

Colores especiales para la consola

```
git config color.ui true
```

Mostrar sólo una línea por cada commit en la traza

```
git config format.pretty oneline
```

Agregar archivos de forma interactiva

```
git add -i
```

http://blog.santiagobasulto.com.ar/programacion/2011/11/27/tutorial-de-git-en-espanol.html

http://rogerdudler.github.io/git-guide/index.es.html