Lab 02 Git

Laboratorio Vision por Computador

Para qué un repositorio?

El objetivo de un repositorio es el de administrar un conjunto de archivos y sus cambios con el tiempo.

Para casi cualquier cosa

Usualmente para codigo, pero funcionapara cualquier otro documento de texto plano.

Que es un repositorio

La estructura de datos que administra los archivos es el repositorio. El repositorio contiene toda la historia de un proyecto. Entre otros podemos:

- Ver la historia de un archivo
- Saber quién lo creo
- Saber quiénes han hecho cambios
- Ver todos los cambios en el proyecto en cualquier momento
- Comparar archivos individuales en dos momentos diferentes
- Revertir a un estado anterior
- Saber Por qué se realizaron los cambios (idealmente)

Un poco mas alla

Usualmente podemos adicionar un log de cambios arbitrario. Si lo usamos adecuadamente podemos saber.

- ¿Qué hace este archivo?
- ¿Para que se agregó esta funcionalidad?
- ¿Qué hace este fragmento de código?
- ¿En qué momento surgió este problema? ¿Qué pretendía hacer el autor en ese entonces?

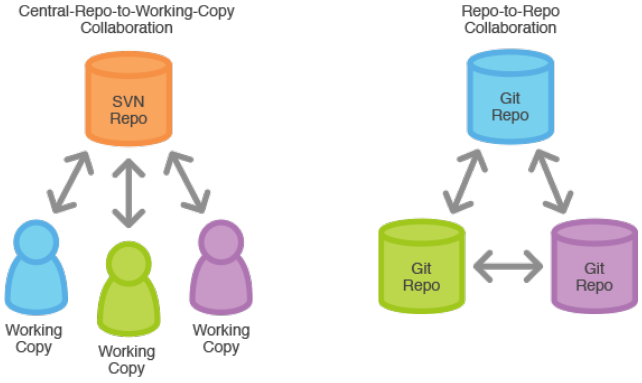
Git

Git apareció como una alternativa para el manejo de versiones en el proyecto del kernel de linux.

Linux Kernel

- Involucra a miles de personas al rededor del mundo
- El costo estimado de desarrollarlo de nuevo estaría por encima de los 1140 millones de dólares.
- Tiene mas de 10 millones de lineas de codigo.

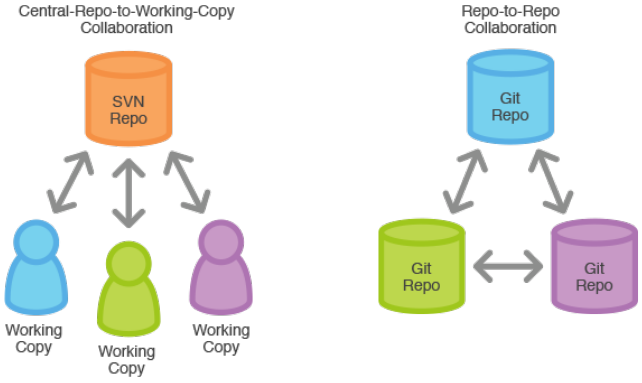
Colaboración



http://techidiocy.com/wp-content/uploads/2013/08/svn-repo.png

Git permite que varias personas puedan trabajar al mismo tiempo en un mismo proyecto.

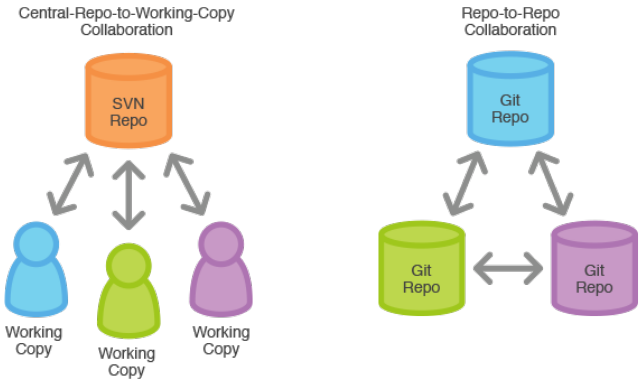
Colaboración



http://techidiocy.com/wp-content/uploads/2013/08/svn-repo.png

Sin embargo la principal forma de colaboración es entre repositorios. (Volveremos sobre esto al final)

Colaboración



http://techidiocy.com/wp-content/uploads/2013/08/svn-repo.png

Tambien funciona de forma local

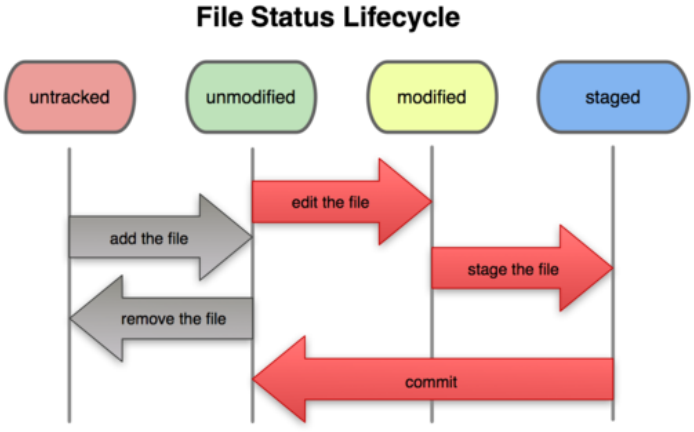
Git NO require de un servidor. El repositorio es simplemente un carpeta especial (.git) dentro del sistema de archivos.

Git - Linea de comandos

La interfaz principal de git es la línea de comandos.
Existen muchas interfaces gráficas que proveen pueden simplificar el trabajo con repositorios locales y remotos.

En http://git-scm.com/downloads/guis encontaran una lista de interfaces para diferentes sistemas operativos.

Git workflow



http://ben-denham.github.io/git-advanced/images/

Ejemplo - Creando un repositorio

En nuestra máquina linux

- crear una nueva carpeta (mkdir)
- crear un nuevo repositorio de git (git init).

Hay algo nuevo en la carpeta? que? (ls -l o ls -la)

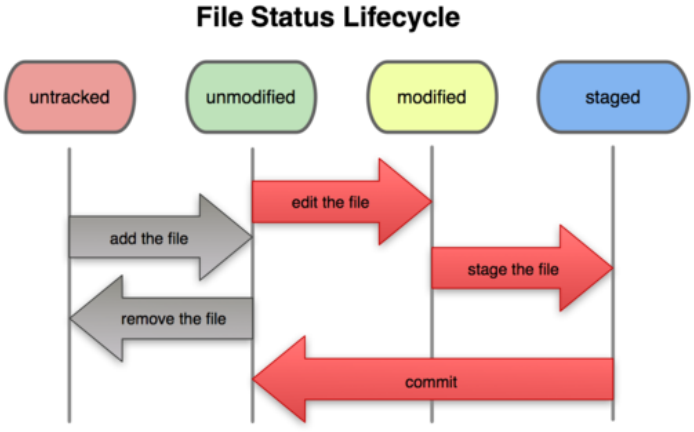
Ejemplo - Añadiendo archivos

- Creen un archivo de texto (touch)
- Editenlo con algún texto de prueba (vim / nano / gedit)
- Usen 'git status', que es esa salida?

git status

"Displays paths that have differences between the index file and the current HEAD commit, paths that have differences between the working tree and the index file, and paths in the working tree that are not tracked by Git"

Git workflow

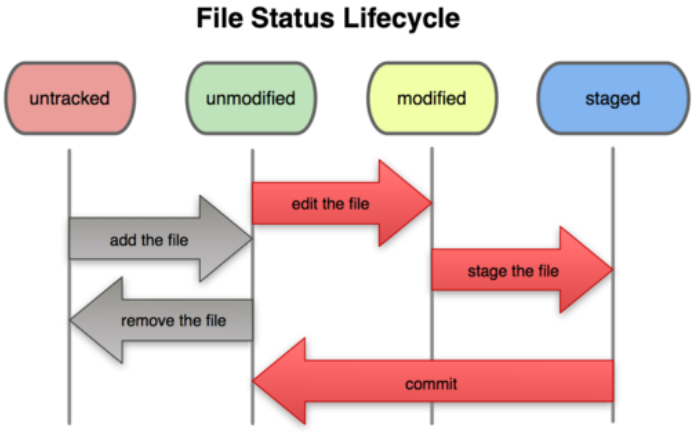


http://ben-denham.github.io/git-advanced/images/

Ejemplo - Agregando al 'Staging area'

- Agregamos el archivo al staging area (git add)
- Cual es nuevo estado del archivo? (git status).
- Enviamos los cambios al repositorio, (git commit)
- Cual es nuevo estado del archivo en el control de versiones? (git status).

```
git commit -m 'El log de el commit'
```



http://ben-denham.github.io/git-advanced/images/

Que pasa si hacemos más cambios al archivo?, como nos vamos a mover dentro del workflow

Git - Versiones

Y donde esta el registro de los cambios?

Podemos usar 'git log' y encontrar el registro de los cambios que hemos hecho, junto con los logs que hemos ingresado.

Git - Versiones

Y si no me gusta el cambio?

Podemos volver a versiones (revisiones) anteriores de nuestro archivo (Repositorio), con 'git checkout [revision]'.
[revisión] no es otra cosa que el hash que aparece cuando usamos git status.

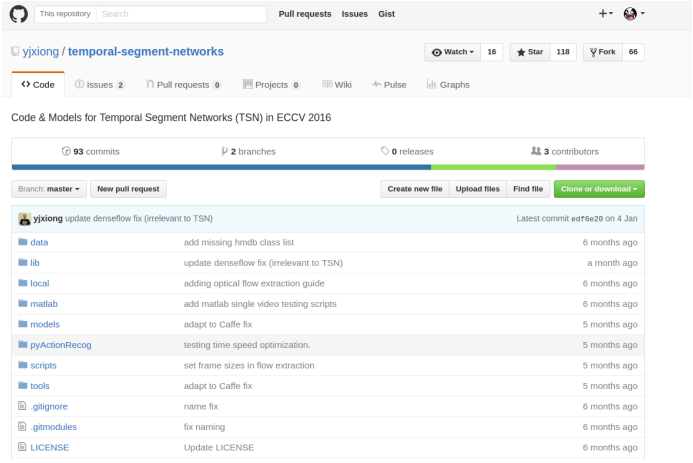
Estas son las operación más básicas de git. Sin embargo existen opciones mas avanzadas que son útiles en proyectos más grandes.

- Branches
- Tags
- Merge/rebase
- Reset
- Revert
- Bisect

Probablemente no las vamos a necesitar en el curso. Así que no las vamos explorar en clase

GitHub

Github es un servicio de para alojar y administrar repositorios de git en la nube.

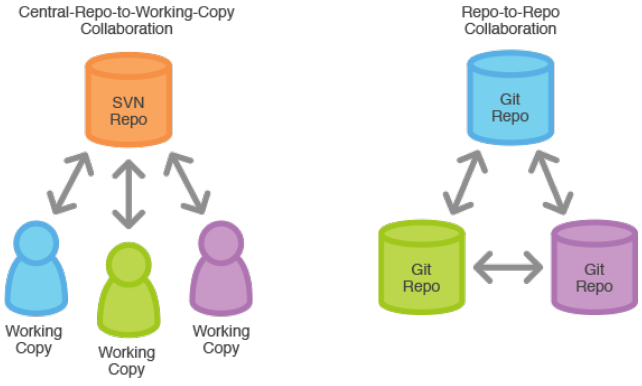


Tiene varias herramientas que fomentan la colaboración en proyectos.

Demo Git

- Ingresemos a github,
- Creemos una cuenta de usuario, si vamos crear repositorios open source (no requiere datos personales, ningún tipo de pago ni suscripción)
- Ingresemos al repositorio de git del curso (https://github.com/fuankarion/Vision17)
- Seleccionen clonar (el botón verde arriba a la derecha), utilicen la url como argumento del comado 'git clone'

Que acabamos de hacer?, de que sirve?



http://techidiocy.com/wp-content/uploads/2013/08/svn-repo.png

Dónde están los repositorios?, quien es el dueño de cada repositorio?, que pasa si quiero hacer cambios?

En realidad no vamos a trabajar con 'Pull Request' (no tiene mucho sentido en el laboratorio).

Lo que realmente necesitamos son **'Forks'**, copias del repositorio que se almacenan en la nube.

Esquema de trabajo de laboratorio

- Creen un fork de el laboratorio.
- Envienme el link del fork, asegurense que cualquier persona tiene acceso a su repositorio (no lo marquen como privado)
- Antes de cada deadline suban el reporte a su propio repositorio (primero al local, luego al remoto). Asegurense que el reporte sea sencillo de encontrar
- Voy copiar cada repositorio a la hora de entrega.