Sistemas de versionado

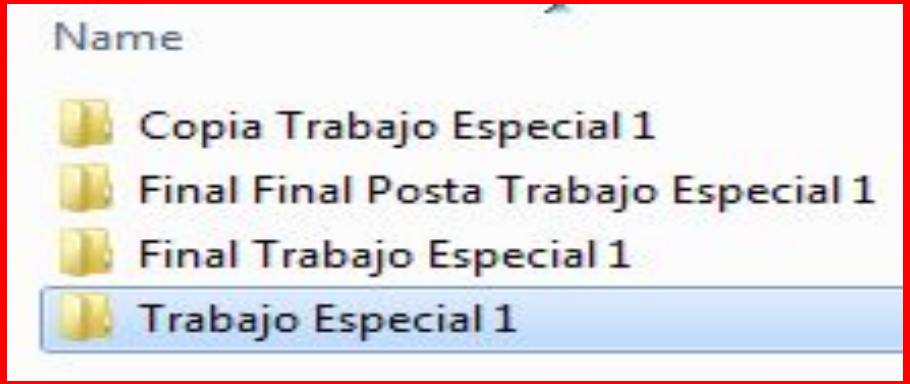
¿Qué es?

Gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo.

¿QUÉ ES?

Un Sistema de Versionado de Código (SVC) es lo que nos permite **compartir el código** fuente de nuestros desarrollos y a la vez **mantener un registro de los cambios** por los que va pasando.

Copiar y Pegar Archivos



NO

Es control de versiones

Software de Control de Versiones

SCV LOCALES:

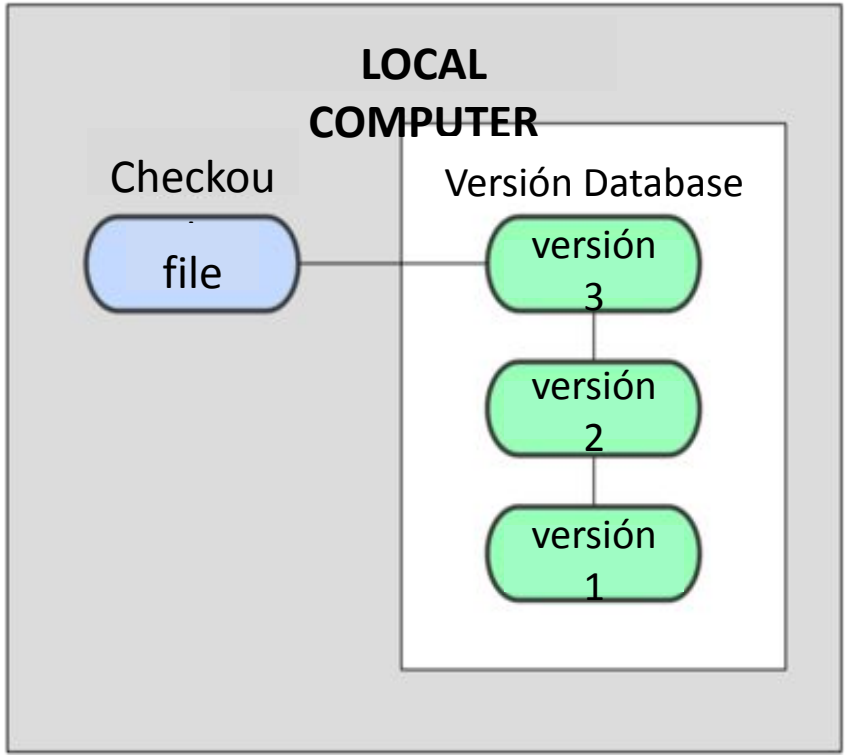
Funciona guardando las diferencias entre archivos de una versión a otra en el disco. Otra manera de hacer esto es copiar los archivos a otro directorio.

VENTAJAS:

- Es simple.

DESVENTAJAS:

- No permite trabajar en conjunto con otros desarrolladores.
- Propenso a errores.



SCV CENTRALIZADOS:

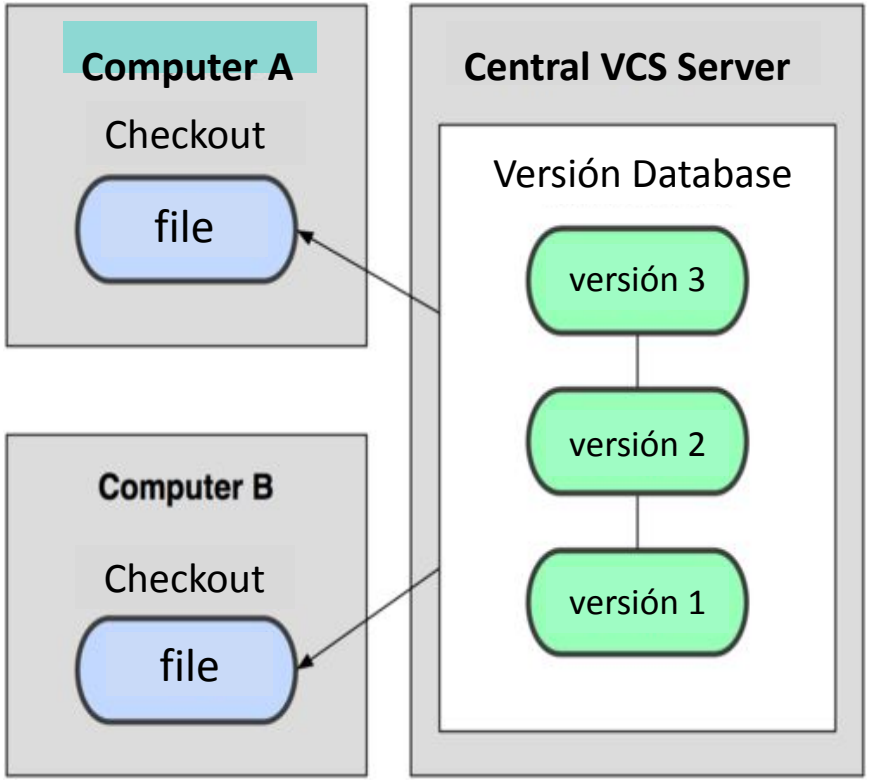
Tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central.

VENTAJAS:

- Es más fácil de administrar.
- Sirve para trabajos en conjunto con otros desarrolladores.

DESVENTAJAS:

- Si el servidor se cae o la base de datos se corrompe pierdes todo.



SCV DISTRIBUIDOS:

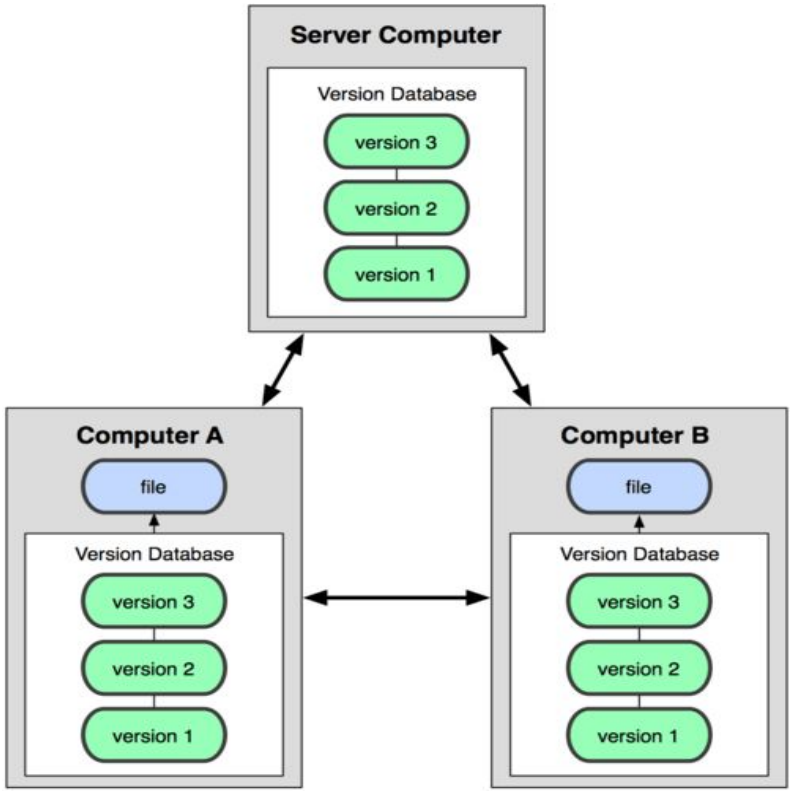
Los clientes no sólo descargan la última instantánea de los archivos: replican completamente el repositorio con cada descarga.

VENTAJAS:

- Existen copias para restaurar el servidor en caso de ser necesario.

DESVENTAJAS:

- Posibilidad de error al modificar un mismo archivo desde lugares diferentes.



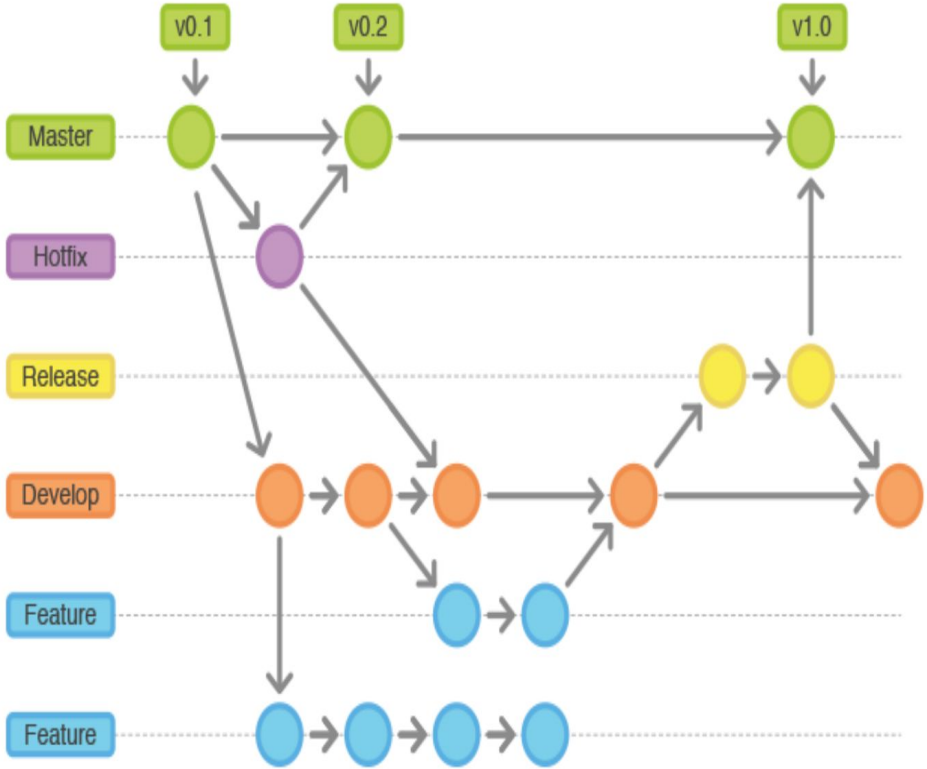
RAMAS/BRANCH:

- **Ramas de largo recorrido:**

  Se tienen varias ramas siempre abiertas, que indican diversos grados de estabilidad del contenido.

- **Ramas puntuales:**

  Se crean de forma puntual para realizar una funcionalidad muy concreta.



TERMINOLOGÍA:

Repositorio: Se almacenan los datos actualizados e históricos de cambios.

Módulo: Conjunto de directorios y/o archivos dentro del repositorio.

Revisión: Es una versión determinada de la información que se gestiona.

Branch: Ramas de modificación de versiones.

Checkout: Un despliegue crea una copia de trabajo local desde el repositorio.

Commit: Sucede cuando una copia de los cambios hechos a una copia local es escrita o integrada sobre el repositorio.

Mas terminología: “conflict”, ”resolve”, ”diff”, ”export”, ”import”, “merge”, “sync”, “workspace”.

Balance

+ Se puede seguir trabajando offline. Incluso si se cae el servidor.
+ Cada repositorio tiene toda la información histórica (Backups replicados).
+ Repositorios más limpios (Dictador Benevolente).
+ Server de Git consume menos recursos.
+ Permite hacer pruebas locales versionadas y subir solo lo relevante.
+ Branching más sencillo.
- Curva de aprendizaje mayor.

GIT

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Vocabulario

- **Repositorio**: Donde se guardan los archivos y datos asociados a cada commit.
- **Commit**: Cambio de una versión a otra.
- **Verbos GIT**: son comandos GIT específicos para manipular el repositorio.
- **Branch**: un puntero a una versión, se mueve el puntero al hacer commits

Uso de GIT

- Podes usar una GUI, pero eso no te salva de necesitar entender lo que está pasando
  - A veces las GUI te intentan ocultar lo que pasa y termina jugando en contra
  - Son muy utiles para ver los diff
- En servidores remotos (ssh) no te queda otra que la consola
- La herramienta de consola es muy verbose
  - Lee lo que dice, suele ser muy descriptivo
  - EJ: git status

Primeros pasos www.github.com

- A cada commit se le guarda el autor con un username y mail

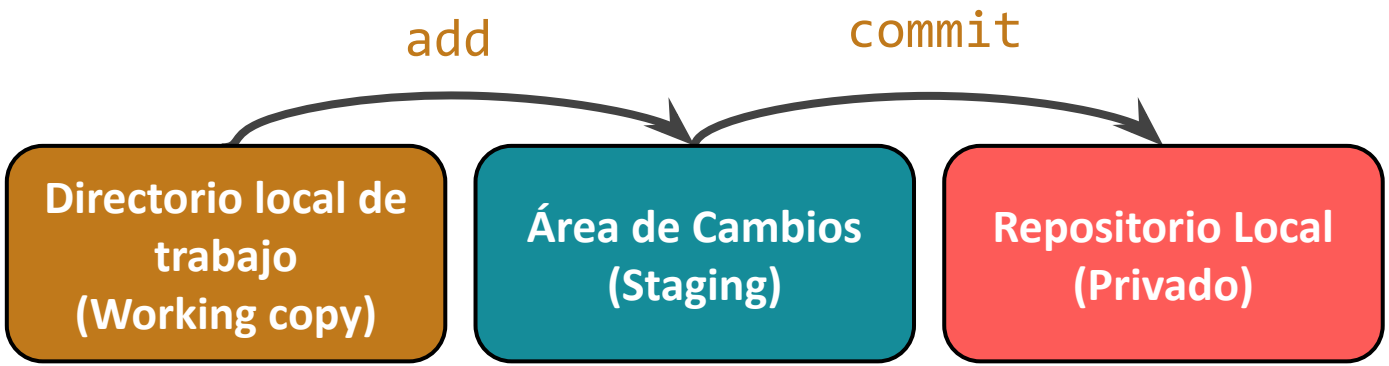
  git config --global user.name "Gisele Keimel"

  git config --global user.email "..."

- --global hace que se guarde como configuración del usuario
  - Sino es solo para cada repositorio
  - Podes commitear en la misma PC con mail diferente (laboral/personal) en cada repositorio

Metafora

- Cada versión es una caja
- Lo que estamos editando es una caja abierta (llamada stage)
- Y tenemos nuestra versión de los archivos (working copy)
- A esa caja le agregamos los archivos modificados
  - git add archivos
- Al cerrar la caja congelamos la versión
  - git commit -m “Mensaje de la versión”
  - Se abre otra caja nueva para la próxima versión (nuevo stage)
- Una vez lista la versión podemos pasarle esa caja al servidor
  - git push servidor branch
  - Si se rompe nuestra PC tenemos una copia
  - Podemos interactuar ahi con otra gente, copiarnos sus cajas y mezclarlas



TU CARPETA

Lo que estás editando

LO QUE YA PREPARASTE PARA COMMITEAR

Lo que ya metiste en la caja

TU HISTORIAL

Todas las cajas que ya cerraste

Este gato ya hizo

git add gatoblanco.txt

En la imagen se ve como hace

git commit -m “Callate”

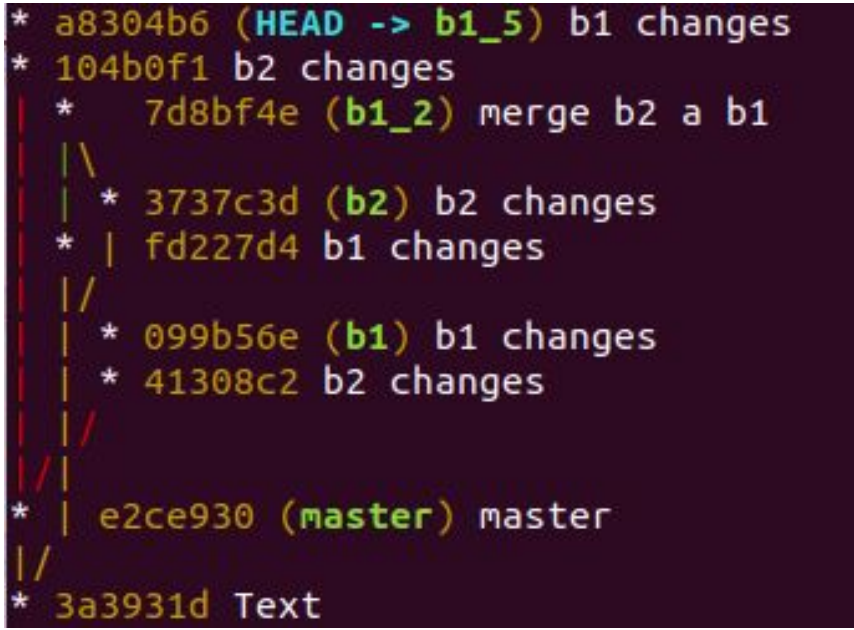
Git Log

Hice muchas versiones, como las veo?

git log

En una interfaz gráfica?

git log --graph --oneline

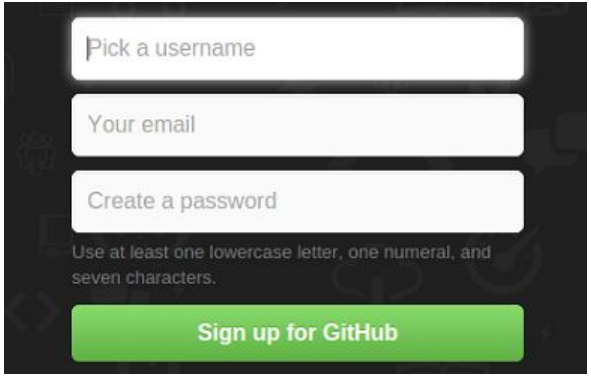


Github

- Plataforma de desarrollo colaborativo, que utiliza Git.
- Ofrece GIT y más cosas juntas
- Gratuito
- Tiene facetas de red social. Se suele usar como CV de proyectos propios.
- Existen otras alternativas (GitLab, BitBucket, etc).

Crear una cuenta en Github

- Entrar en github.com
- Registrarse



Clonarse el repositorio

- Ejecutar en la consola:

```
git clone URL-REPOSITORIO
```

Ejercicio #1 - En clase

- Modificar el readme en la web
- Actualizar nuestro repositorio local

Recapitulamos

`> Modifico primero el readme en la Web, y commiteo los cambios.`

`> git fetch`

Chequeo si hay cambios para traer

`> git pull`

Traigo los cambios

`> git log`

Chequeo los commit realizados

Ejercicio #2 - En clase

- Agregar un archivo a la carpeta del repositorio
- Subirlo al repositorio.

Recapitulamos

```
> git add “Readme.md”
```

Agrego el archivo a GIT. El estado es Untracked.

```
> git commit –m “Primera Version del Readme”
```

Agrego el archivo a mi repositorio local

```
> git pull
```

Traigo los cambios (si hay en el repo origin)

```
> git push
```

Subo los cambios al repo origin

Trabajar solo + Backup

- Por ahora solo trabajamos de a uno en el repositorio.
- Un SCV nos permite trabajar solos y tener un tracking de todo lo que fuimos haciendo.
- Cuando hacemos **push** estamos haciendo un backup en otro repositorio (en nuestro caso GitHub).

Trabajar en equipo

- Podemos trabajar en equipo.
- Resolver los conflictos de código
- Que todos estemos sincronizados con lo último que hay en el repositorio.

.gitignore

- Es un archivo del repositorio (se versiona también).
- Dice que archivos no se van a versionar.
- Permite excluir carpetas de archivos compilados, fotos pesadas, etc.
- Hay muchos ejemplos para cada lenguaje en la web para cada lenguaje (https://github.com/github/gitignore).