Control de versiones

Martín Dell’Oro

Control de Versiones

-Gestión de diversos cambios en algún producto

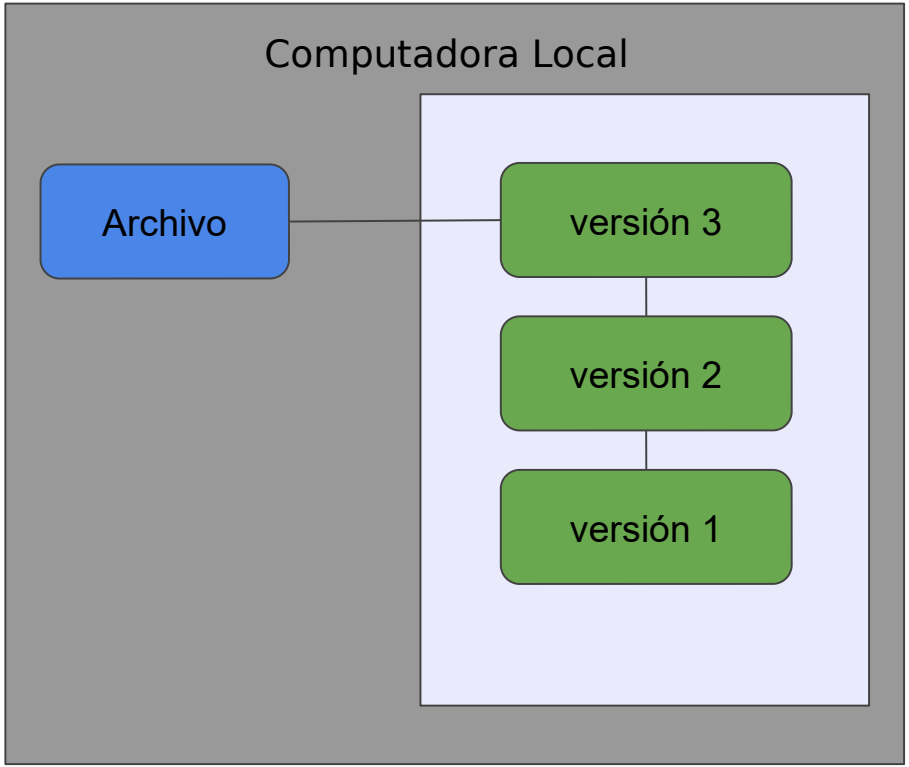
-Versión -> estado del producto en algún momento dado.

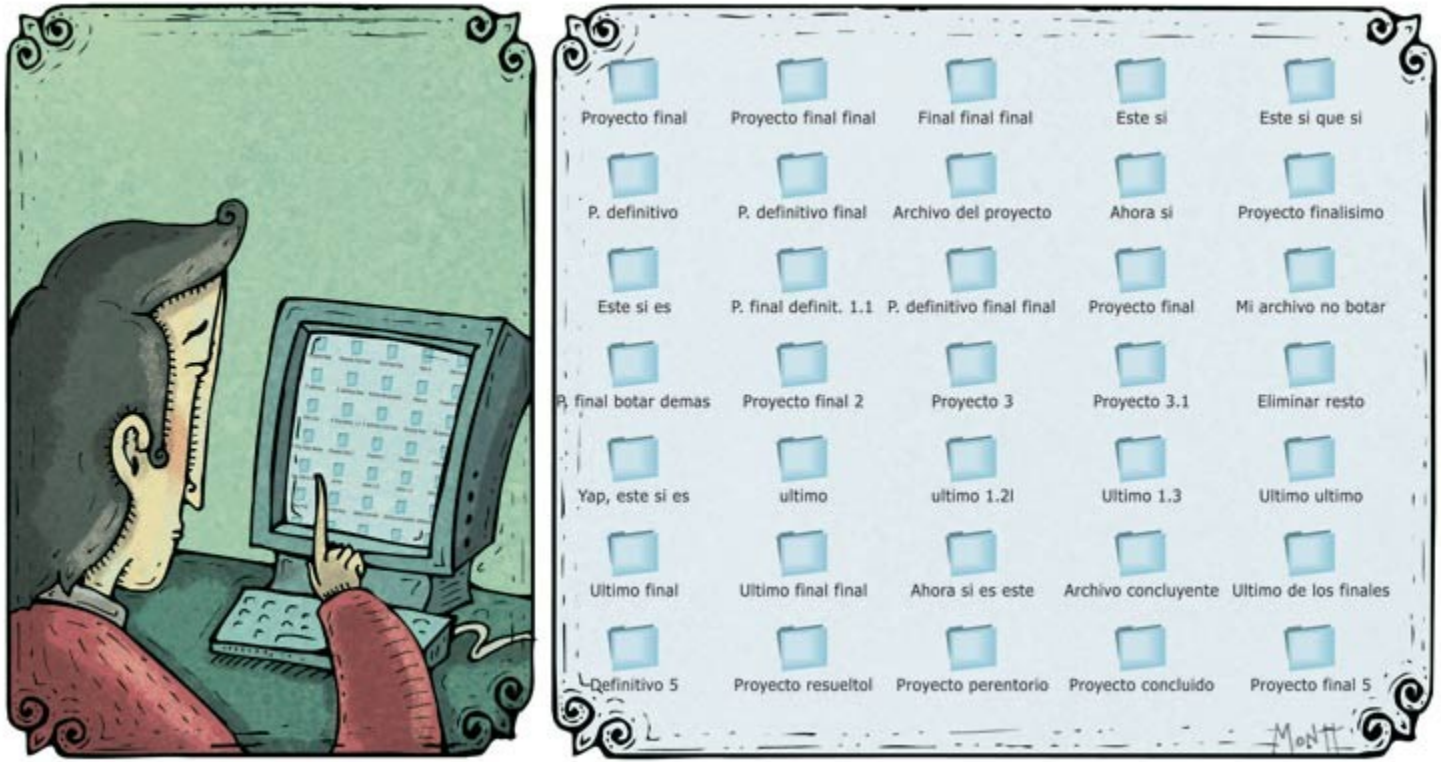
Clasificación

Podemos clasificar los SVC de acuerdo a la arquitectura utilizada

Locales
Centralizados
Distribuidos

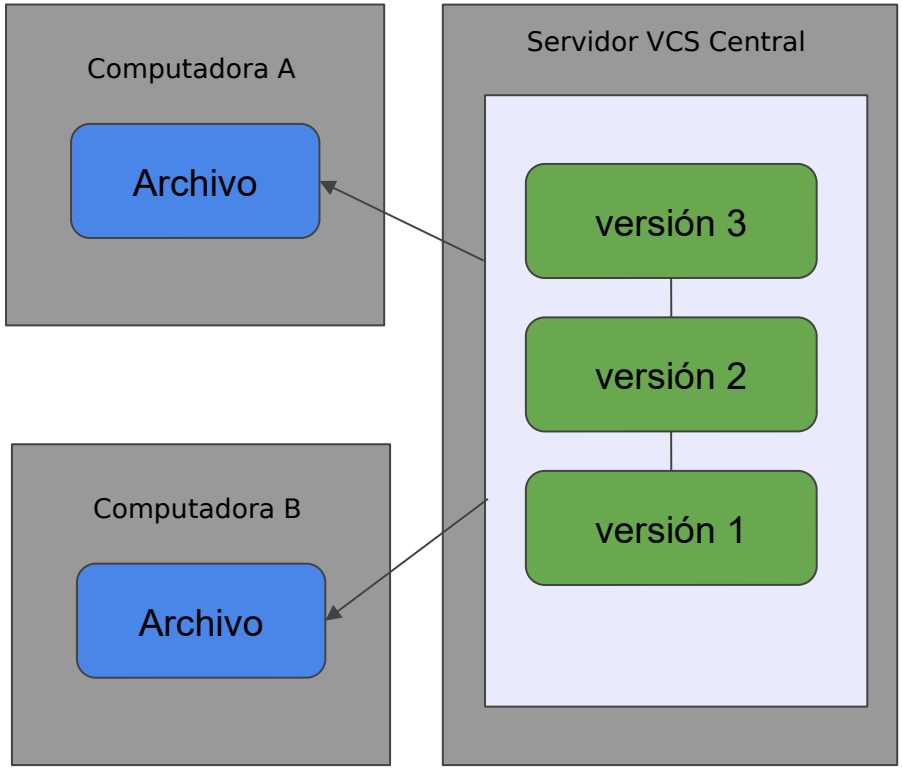
Arquitectura Local




Proyecto final
Proyecto final final
Final final final
Este si
Este si que si
P. definitivo
P. definitivo final
Archivo del proyecto
Ahora si
Proyecto finalisimo
Este si es
P. final definit. 1.1
P. definitivo final final
Proyecto final
Mi archivo no botar
P. final botar demas
Proyecto final 2
Proyecto 3
Proyecto 3.1
Eliminar resto
Yap, este si es
ultimo
ultimo 1.2l
Ultimo 1.3
Ultimo ultimo
Ultimo final
Ultimo final final
Ahora si es este
Archivo concluyente
Ultimo de los finales
Definitivo 5
Proyecto resueltol
Proyecto perentorio
Proyecto concluido
Proyecto final 5

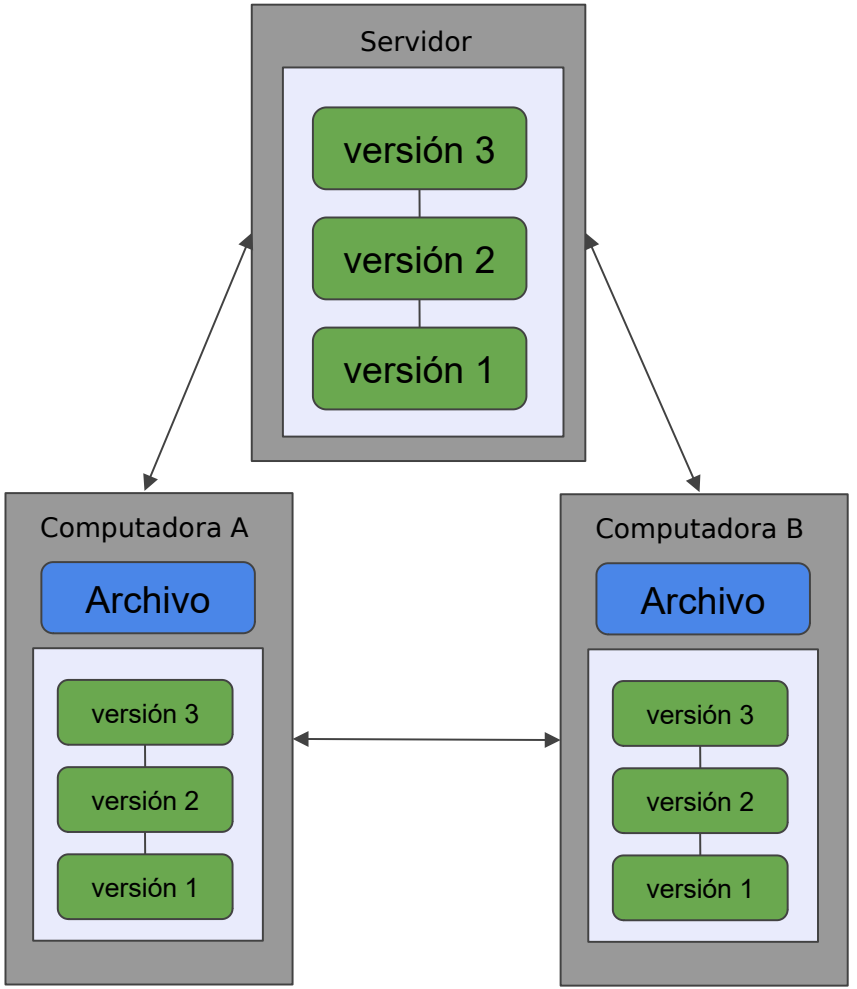
Arquitectura Centralizada

CVS - SVN



Arquitectura Distribuida

GIT - Mercurial



Ventajas

- arquitectura distribuida -

No es necesario estar conectado para guardar cambios.

Posibilidad de continuar trabajando si el repositorio remoto no está accesible.

Se necesitan menos recursos para el repositorio remoto.

Más flexibles al permitir gestionar cada repositorio personal como se quiera.

GIT

instalación

LINUX

Fedora: `$ yum install git-core`
Debian: `$ apt-get install git`

WINDOWS

Desde la página de GITHUB:
http://msysgit.github.com

MacOS

`$ brew install git`

GIT

configuración

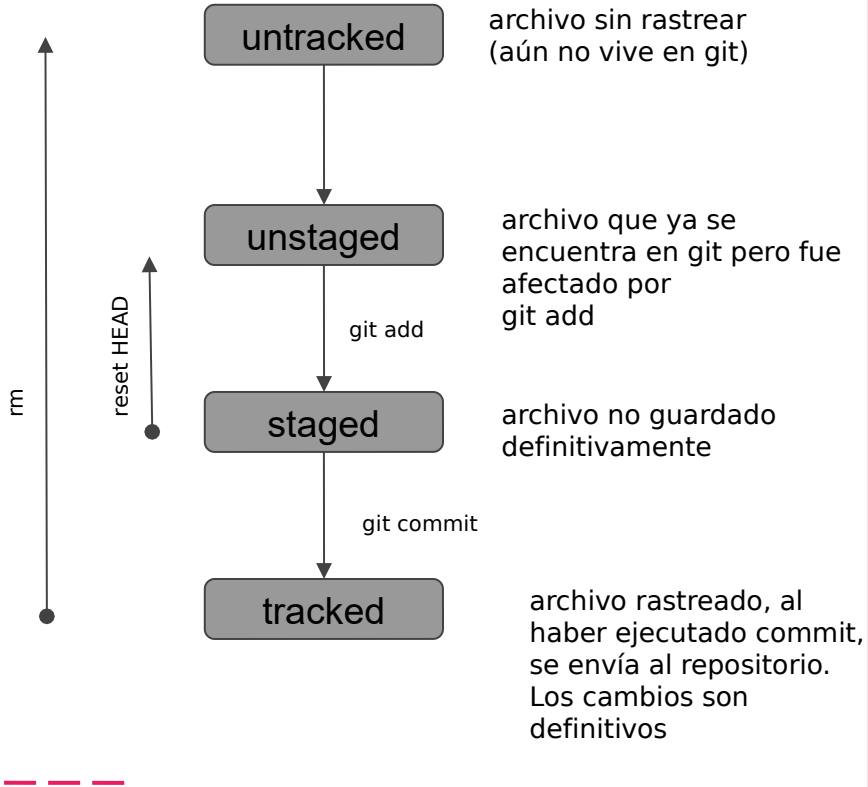
```
$ git config --global user.name "Lucas Baldezzari"
```

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

```
$ git config --global push.default simple
```

Estados

-git status: nos muestra en qué estado se encuentran nuestros archivos



Comandos básicos

El primer paso es inicializar el repositorio

git init

Creará un área conocida como “staging” y un repositorio local

Comandos básicos

git status: ver el estado de nuestros archivos
git add: añade el archivo a staging
git commit: guarda los archivos en el repositorio local
git reset HEAD: quita archivos de staged o los devuelve a su estado anterior
git rm: elimina el archivo de tu directorio de trabajo y no elimina su historial del sistema de versiones

Comandos básicos

git show: muestra los cambios que han existido sobre un archivo

git diff: muestra la diferencia entre una versión y otra.

git log: obtenemos el id de los commits

git checkout -b nombre_rama: para crear una nueva rama y ubicarse sobre la misma

git checkout nombre_rama: es para movernos entre ramas.

Comandos básicos

git push: Sube los cambios hechos en tu ambiente de trabajo a una rama remota

Branches

ramas

Todos los commits se aplican sobre una rama
Por defecto comenzamos en la rama MASTER

Merge

git merge permite tomar las líneas independientes de desarrollo creadas por **git** branch e integrarlas en una sola rama. Esto **significa** que **git merge** se suele utilizar junto con **git** checkout para seleccionar la rama actual y **git** branch -d para eliminar la rama de destino obsoleta.


In case of fire
1. git commit
2. git push
3. leave building