Control de versiones

Clase - Aplicaciones Interactivas

¿Cuál es la importancia de un control de Versiones?

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

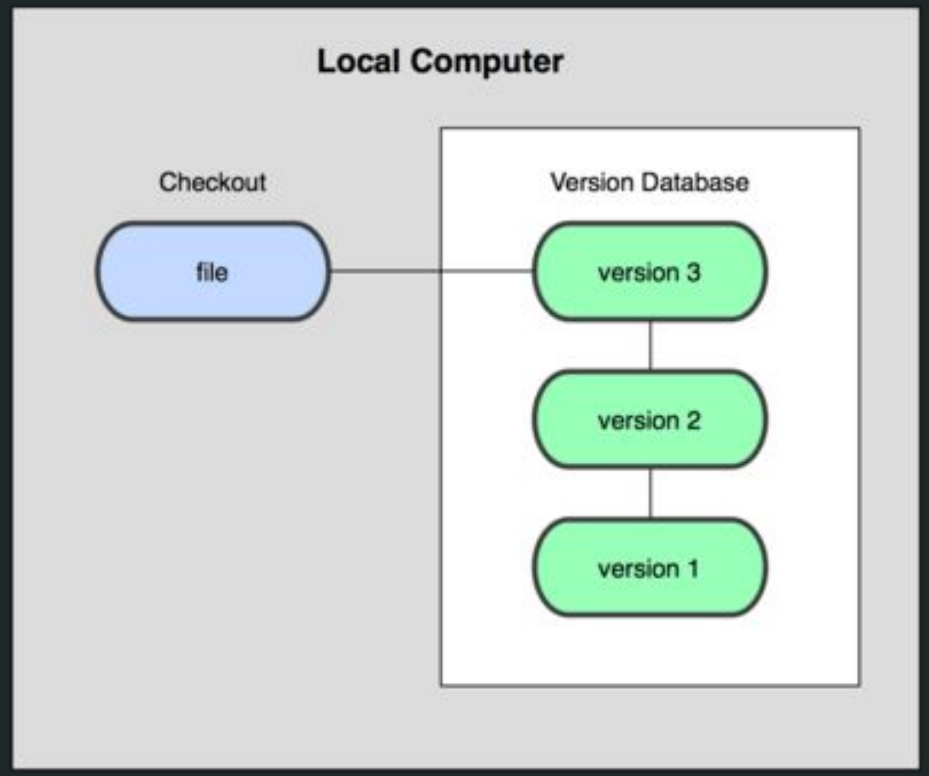
- Sistemas de control de versiones locales
- Sistemas de control de versiones centralizados (CVCS)
- Sistemas de control de versiones distribuidos (DVCS)

Sistemas de control de versiones locales

Un método de control de versiones usado por mucha gente es copiar los archivos a otro directorio. Este enfoque es muy común porque es muy simple, pero también tremendamente propenso a errores.

Es fácil olvidar en qué directorio se encuentra, guardar accidentalmente en el archivo equivocado o sobrescribir archivos que no querías.

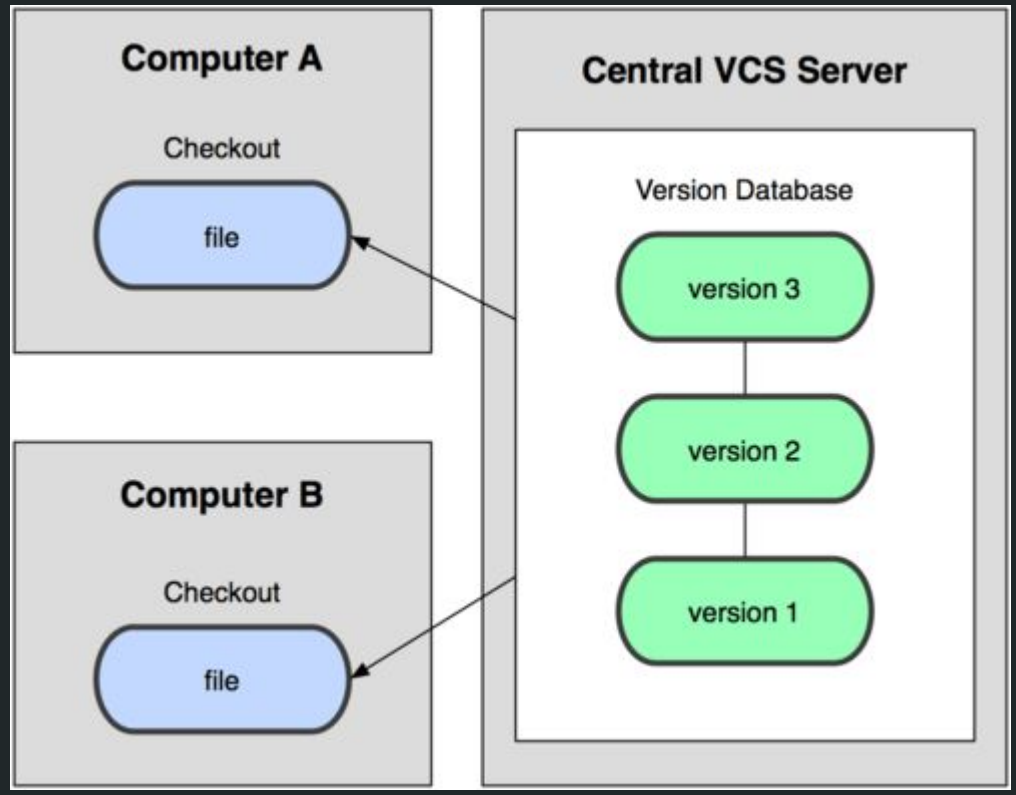
El gran problema que se encuentra es cuando se necesita colaborar con desarrolladores en otros sistemas.



Sistemas de control de versiones centralizados (CVCS)

Tienen un único servidor que contiene todos los archivos versionados, y varios clientes que descargan los archivos desde ese lugar central. Los administradores tienen control de qué puede hacer cada uno; y es mucho más fácil administrar.

Una desventaja, es el punto único de fallo que representa el servidor centralizado. Si ese servidor se cae durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios versionados de aquello en que están trabajando. Cuando tienes toda la historia del proyecto en un único lugar, te arriesgas a perderlo todo.

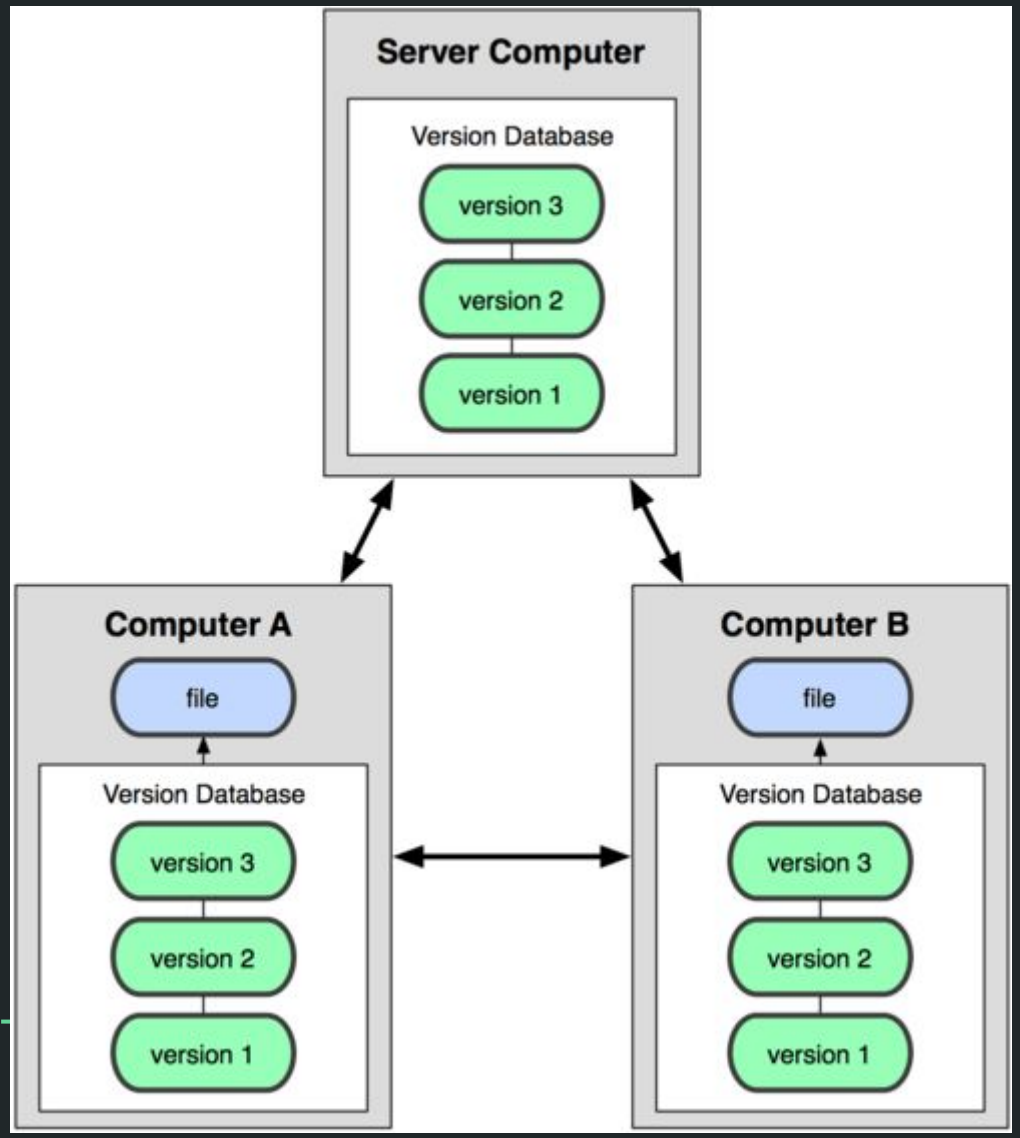


Sistemas de control de versiones distribuidos (DVCS)

En un DVCS , los clientes no sólo descargan la última instancia de los archivos si no que replican completamente el repositorio.

Por ejemplo, si un servidor se cae, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo.

Cada vez que se descarga una instancia, en realidad se hace una copia de seguridad completa de todos los datos.



Git

¿Qué es Git?

Es un producto para control de versiones de archivos.

Nos permite manejar nuestros archivos de texto o binarios y llevar un control de los cambios que en ellos realizamos.

Funcionamiento

GIT modela sus datos como un conjunto de instancias de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en GIT, básicamente hace una foto del de todos tus archivos en ese momento, y guarda una referencia a esa instancia

Integridad

Todo en GIT es verificado mediante una suma de comprobación antes de ser almacenado, y es identificado a partir de ese momento mediante dicha suma. Esto significa que es imposible cambiar los contenidos de cualquier archivo o directorio sin que GIT lo sepa.

Estados

El directorio de GIT es donde GIT almacena los metadatos y la base de datos de
objetos para tu proyecto. Es la parte más importante de GIT, y es lo que se copia cuando clonas un repositorio desde otro ordenador.

GitIgnore

GIT tiene un archivo interno en el cual se puede especificar que tipos de archivos van a ser ignorados. Es opcional, puede estar o no dentro del repositorio. Si existe, su contenido va a indicar todos los archivos, patrones de archivos y carpetas que GIT literalmente tiene que ignorar.

Ejemplo:

```
# a comment - this is ignored
# no .a files
*.a
# but do track lib.a, even though you're ignoring .a files above
!lib.a
# only ignore the root TODO file, not subdir/TODO
/TODO
# ignore all files in the build/ directory
build/
# ignore doc/notes.txt, but not doc/server/arch.txt
doc/*.txt
# ignore all .txt files in the doc/ directory
doc/**/*.txt
```

Comandos especiales

- git init → Crea un repositorio Git.
- git clone [url] → Crea un repositorio Git y hace un fetch del codigo.
- git fetch [nombre de branch] → Se obtiene la ultima version del codigo.
- git status → Muestra el estado de los archivos de mi repositorio local contra el repositorio remoto.
- git add . → Prepara todos archivos que fueron modificados para ser “commiteados”.
- git add [nombre de archivo] → Prepara solo el archivo indicado para ser “commiteado”.
- git commit –m “[comentario descriptivo de los cambios hechos]” → Incorporo a mi repositorio local los archivos que agregados recientemente.

Comandos especiales

- git pull → Obtiene todos los cambios del repositorio remoto y los “mergea” con mi repositorio local.
- git push → Envio todos mis commits de mi branch local al branch remoto en el que se encuentra asociado.
- git checkout –b [nombre de branch] → Se crea un branch [nombre de branch] local a partir del branch situado.
- git merge [nombre de branch] → Une los cambios del [nombre de branch] al branch local.
- git branch → Lista todos los branches existentes en nuestro repositorio local.
- git branch –d [nombre de branch] → Borra el branch [nombre de branch] del repositorio local.
- git log → Muestra un historial con todos los commits realizados.

¿Qué es Github?

GitHub es una plataforma donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código.

Almacenar tu código en un "repositorio" en GitHub te permite:

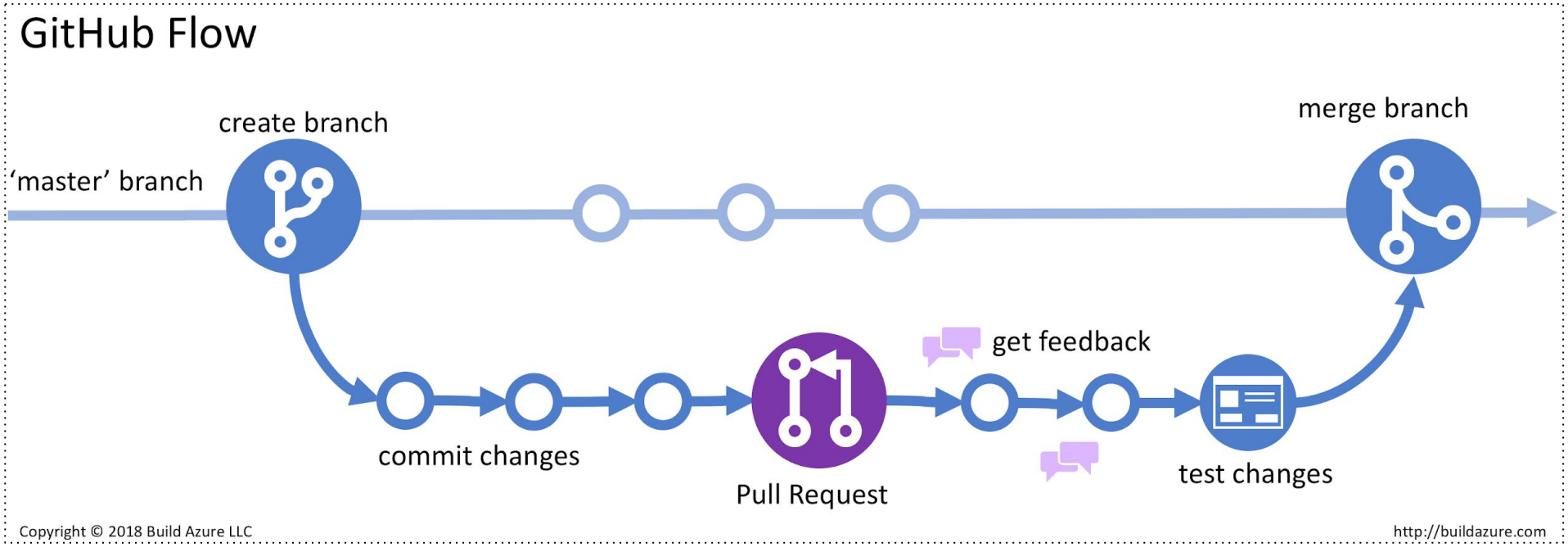
- Presentar o compartir el trabajo.
- Seguir y administrar los cambios en el código a lo largo del tiempo.
- Dejar que otros usuarios revisen el código y realicen sugerencias para mejorarlo.
- Colaborar en un proyecto compartido, sin preocuparse de que los cambios afectarán al trabajo de los colaboradores antes de que esté listo para integrarlos.

El trabajo colaborativo, una de las características fundamentales de GitHub, es posible gracias al software de código abierto **Git**, en el que se basa GitHub. Dentro del mismo, contaremos con varias acciones disponibles, entre ellas **Pull Request**

Git Branching Flows

Definir una estrategia de Branch es crucial para tener una forma de trabajar con github y en equipo. Existen varias definiciones de estrategias de branches (Gitflow, gitlab flow, github flow, Ship show Ask). Para fines practicos de esta clase, hoy veremos **Github flow**

.



Vamos a la practica