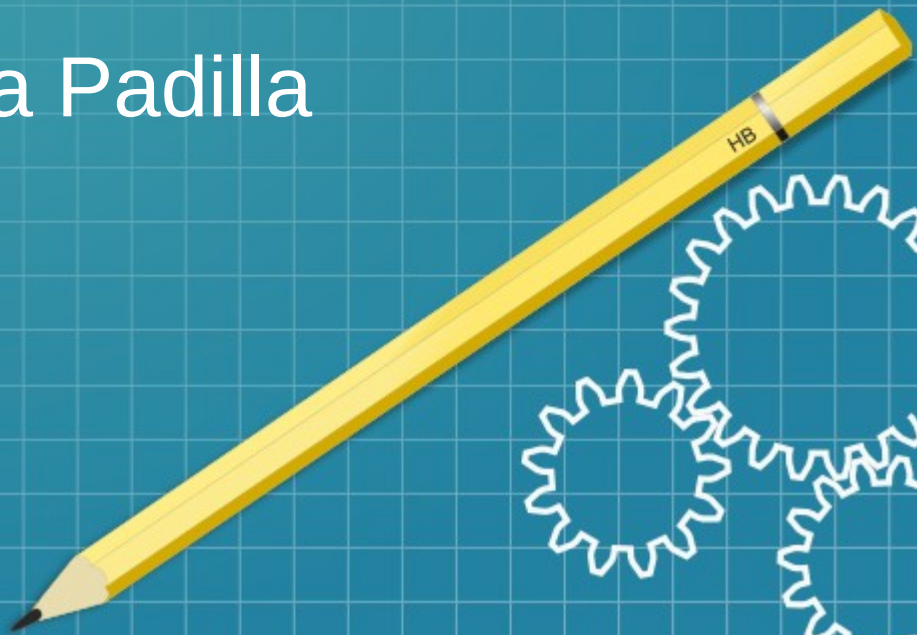
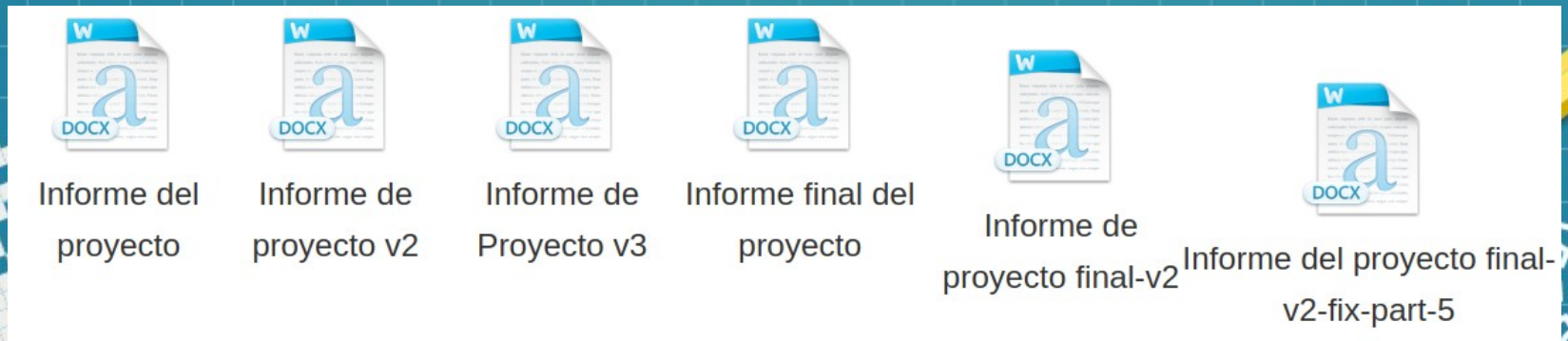
Control de Versiones

Rosa Victoria Villa Padilla



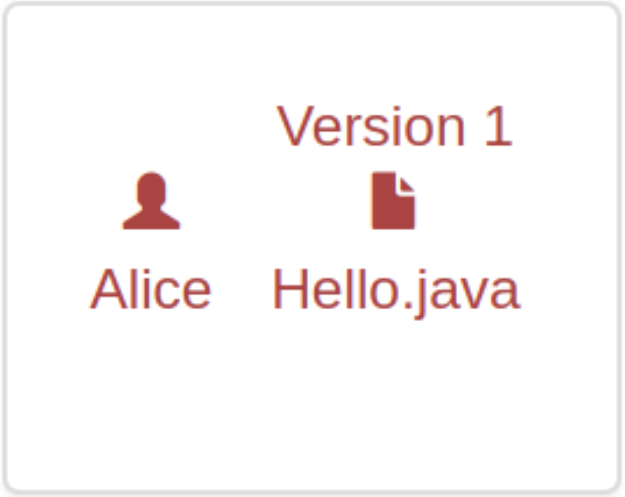
Sistemas de control de versiones que ya has usado.

- Drive, Dropbox
- Deshacer / rehacer el búfer
- Mantener múltiples copias de archivos con números de versión



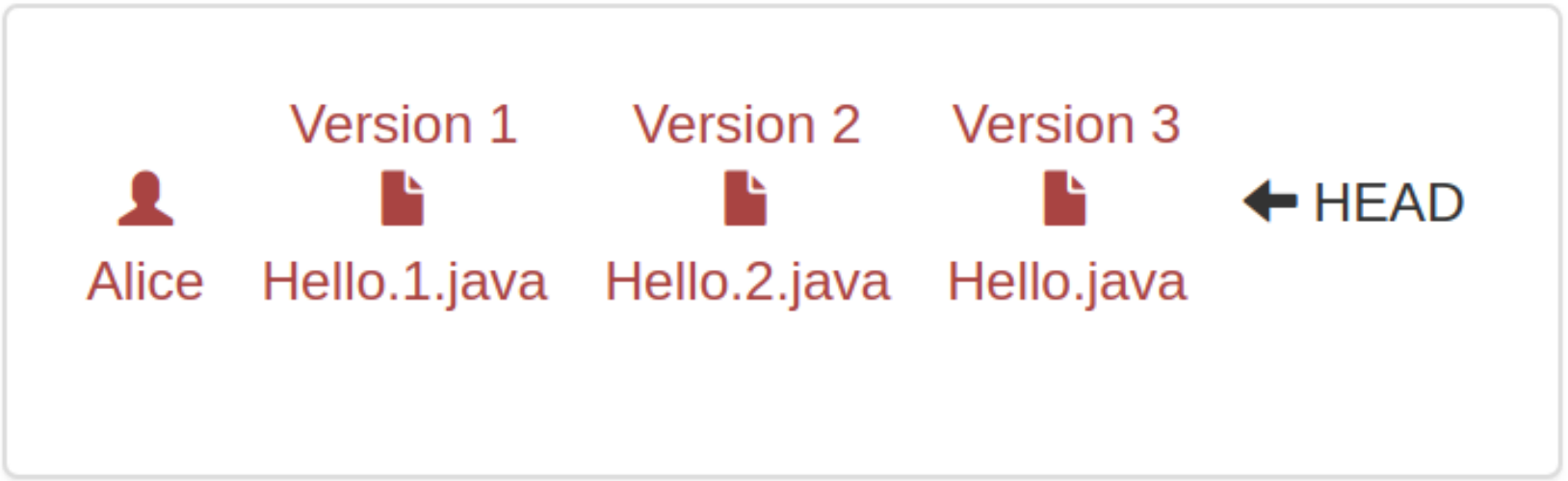
Inventando el control de versiones.

- Alice trabaja en un archivo durante varios días.
- En el último minuto, antes de que necesite entregar su código para que la califiquen, se da cuenta de que ha hecho un cambio que lo rompe todo.



Inventando el control de versiones.

- Una simple disciplina de guardar archivos de copia de seguridad haría el trabajo.
- Alice usa su juicio para decidir cuándo ha alcanzado algún punto que justifique guardar el código.



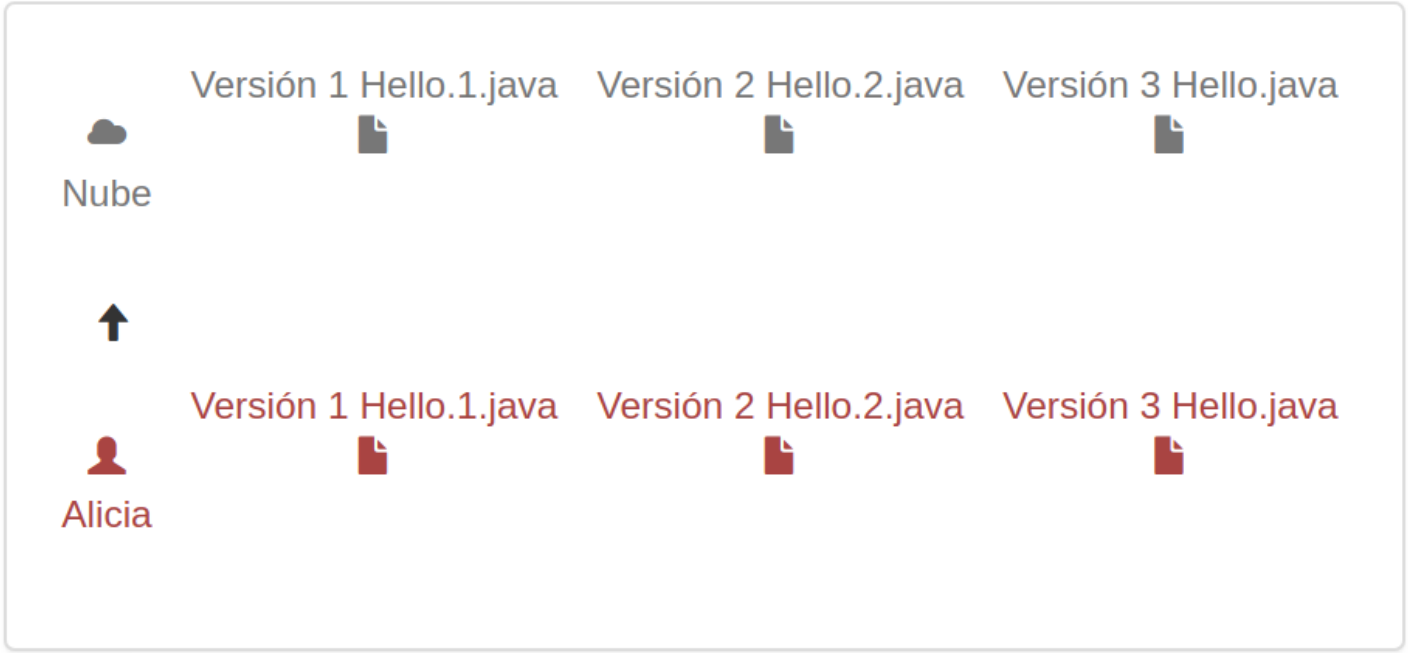
Inventando el control de versiones.

- Ahora, cuando Alice se da cuenta de que la versión 3 tiene fallas fatales, puede copiar la versión 2 nuevamente en la ubicación de su código actual. Se evitó el desastre! Pero, ¿y si la versión 3 incluía algunos cambios que eran buenos y otros que eran malos?

Inventando el control de versiones.

- Alice puede comparar los archivos manualmente para encontrar los cambios y clasificarlos en cambios buenos y malos. Entonces ella puede copiar los cambios buenos en la versión 2.
- En UNIX tenemos herramientas como diff

- También quiere estar preparada en caso de que su computadora portátil sea atropellada o simplemente un día explote, por lo que guarda una copia de seguridad de su trabajo en la nube

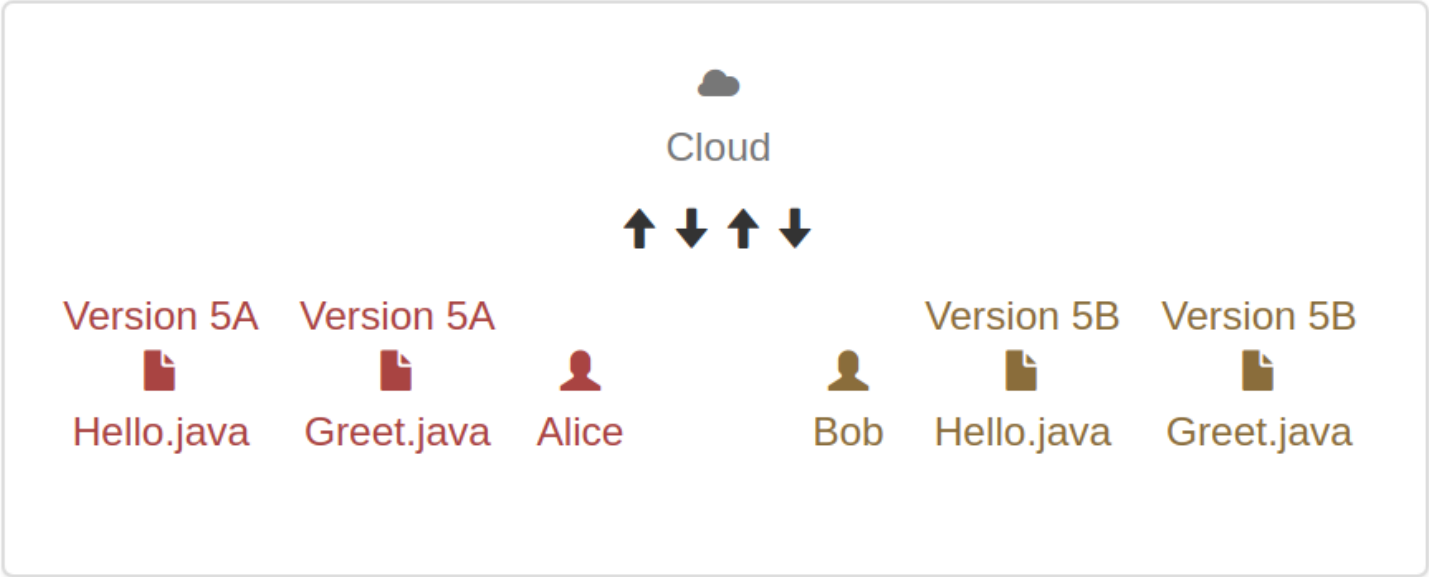


- ¿Puede haber problemas de versiones?

Múltiples desarrolladores

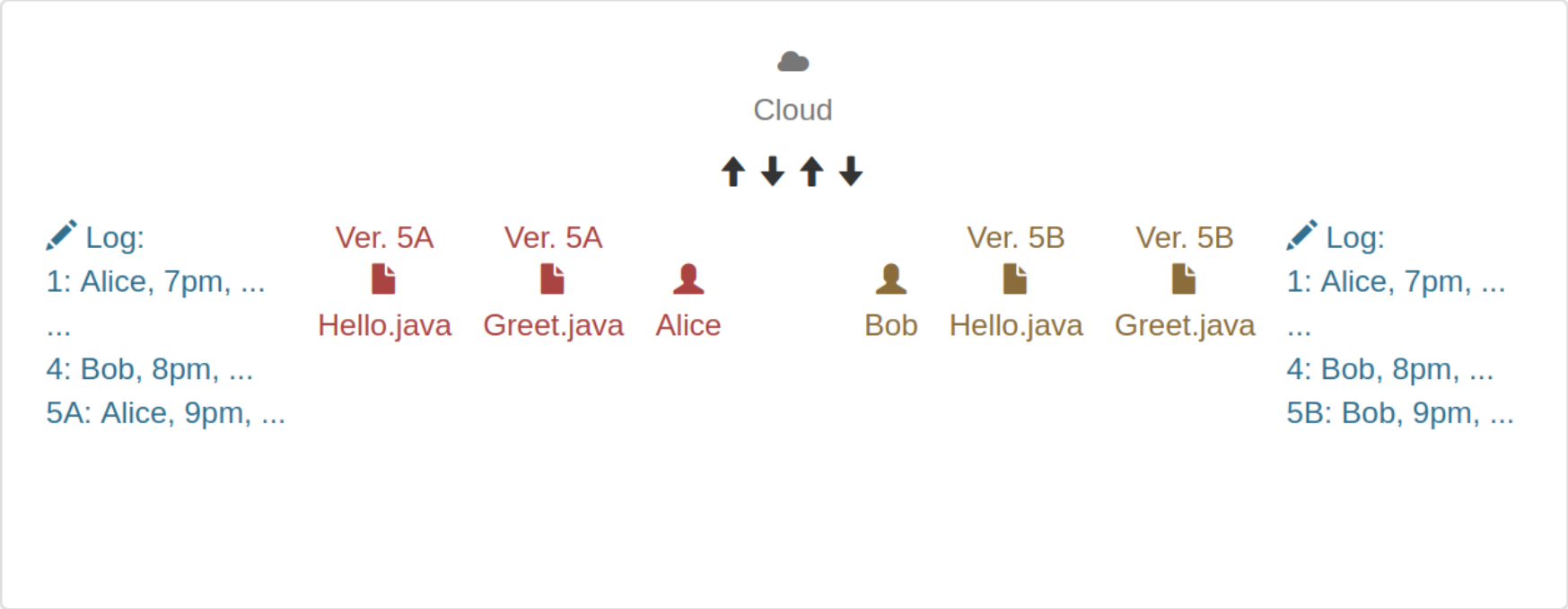
Múltiples desarrolladores

- Ya no comparten un cerebro, lo que hace que sea aún más importante seguir una disciplina estricta al subir y bajar del servidor de la nube compartida. Los dos programadores deben coordinar un esquema para llegar a los números de versión



Múltiples desarrolladores

- Agreguemos un registro que registre a cada versión que lo escribió, cuándo se finalizó y cuáles fueron los cambios, en forma de un breve mensaje de autoría humana.



Múltiples ramas

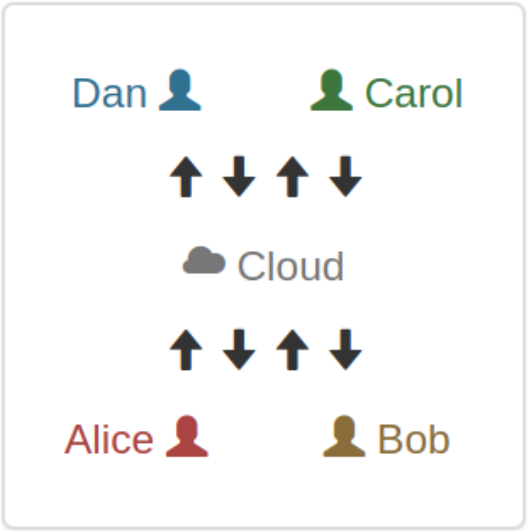
- A veces tiene sentido que un subconjunto de desarrolladores se vaya y trabaje en una rama, un universo de código paralelo para experimentar con una nueva característica. Los otros desarrolladores no quieren utilizar la nueva característica hasta que esté lista, incluso si se crean varias versiones coordinadas mientras tanto.

La impactante conclusión.

- Por supuesto, resulta que no hemos inventado nada aquí: **Git** hace todas estas cosas por usted, al igual que muchos otros sistemas de control de versiones como Bitbucket.

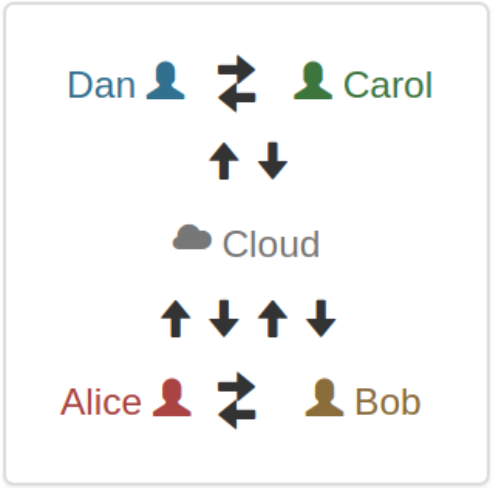
Distribuido vs. centralizado

- En un sistema **centralizado**, todos deben compartir su trabajo hacia y desde el repositorio principal. Los cambios se guardan de forma segura en el control de versiones si están en el repositorio principal , porque ese es el único repositorio.
- CVS y Subversion



Distribuido vs. centralizado

- En un sistema **distribuido**, todos los repositorios son iguales, y los usuarios pueden asignarles diferentes roles. Diferentes usuarios pueden compartir su trabajo desde y hacia diferentes repositorios, y el equipo debe decidir qué significa que un cambio esté en el control de versiones.
- Git y Mercurial



Configuración

- Descarga git para OSX
- Descarga git para Windows
- Descarga git para Linux

Crea un repositorio nuevo

Crea un directorio nuevo, ábrelo y ejecuta

git init

para crear un nuevo repositorio de git.

Copia local

Crea una copia local del repositorio ejecutando

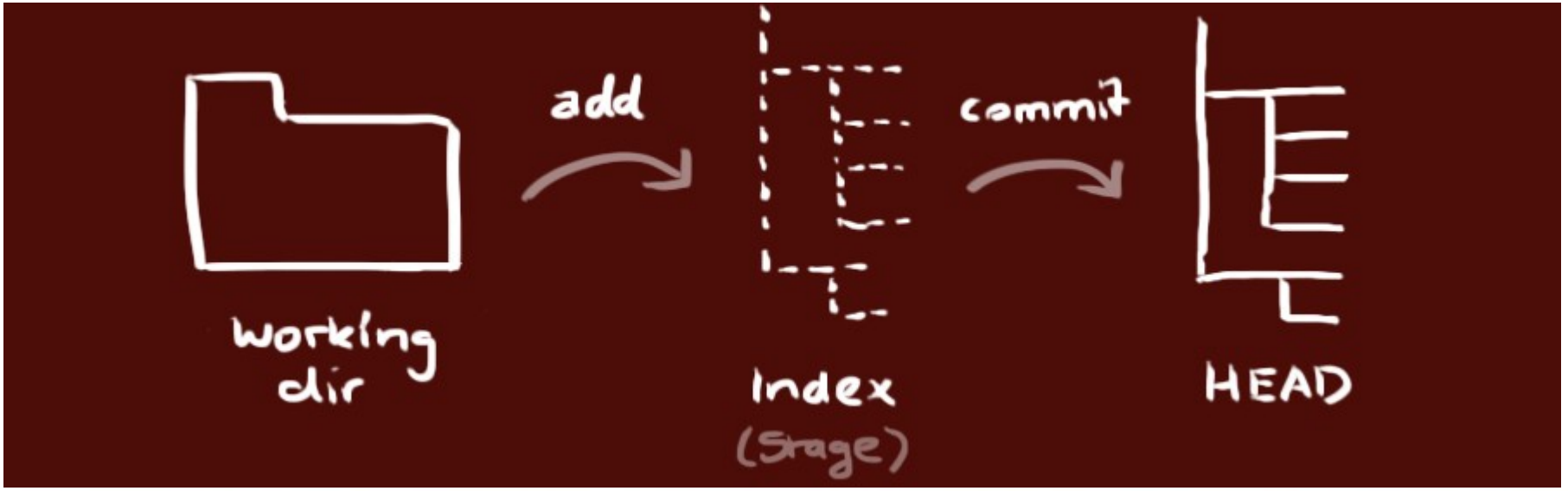
git clone /path/to/repository

Si utilizas un servidor remoto, ejecuta

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

Flujo de trabajo

Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu **Directorio de trabajo** que contiene los archivos, el segundo es el **Index** que actuá 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> solo ese archivo

git add . nuevos archivos o modificados

git add -A todos los cambios

git add * Modificaciones y eliminaciones

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

add & commit

git commit -m "Commit message" agrega un comentario a la versión que estas creando

git commit -am "Commit message" carga los cambios y agrega un comentario a la versión que estas creando

Ahora el archivo esta incluído 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.

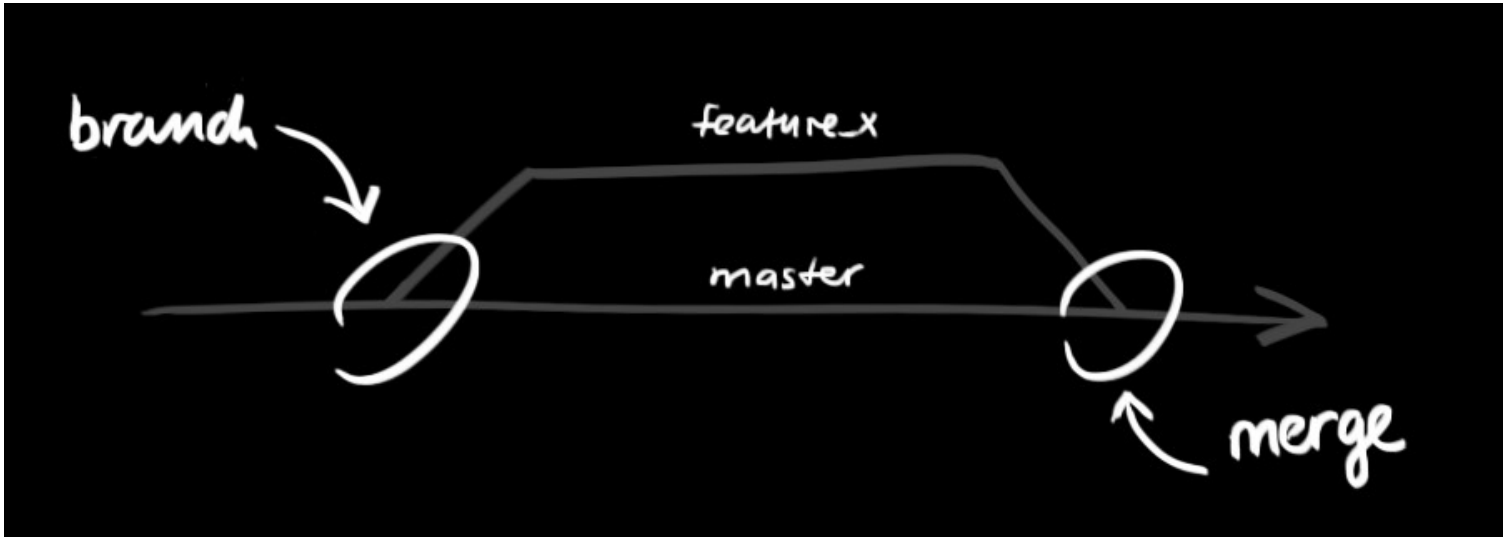
Si no has clonado un repositorio ya existente y quieres conectar tu repositorio local a un repositorio remoto, usa

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. Puedes crea nuevas ramas durante el desarrollo y fusiónalas a la rama principal cuando termines.



Ramas

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

git checkout -b feature_x

vuelve a la rama principal

git checkout master

y borra la rama

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>

Ver cambios

git status

Muestra una lista de todos los cambios que hay en el repositorio, la rama actual y la version commit

git log

Muestra el historial de versiones de la rama

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.

Pero

"Cada quien haga su parte y mañana lo juntamos todo"

Actualiza & fusiona

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>

Actualiza & fusiona

git diff <branch>

muestra la diferencia entre la version actual y el ultimo commit

git diff <commit> <commit>

muestra la diferencia entre dos commit específicos

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.

Reemplaza cambios locales

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

y otra manera sería con

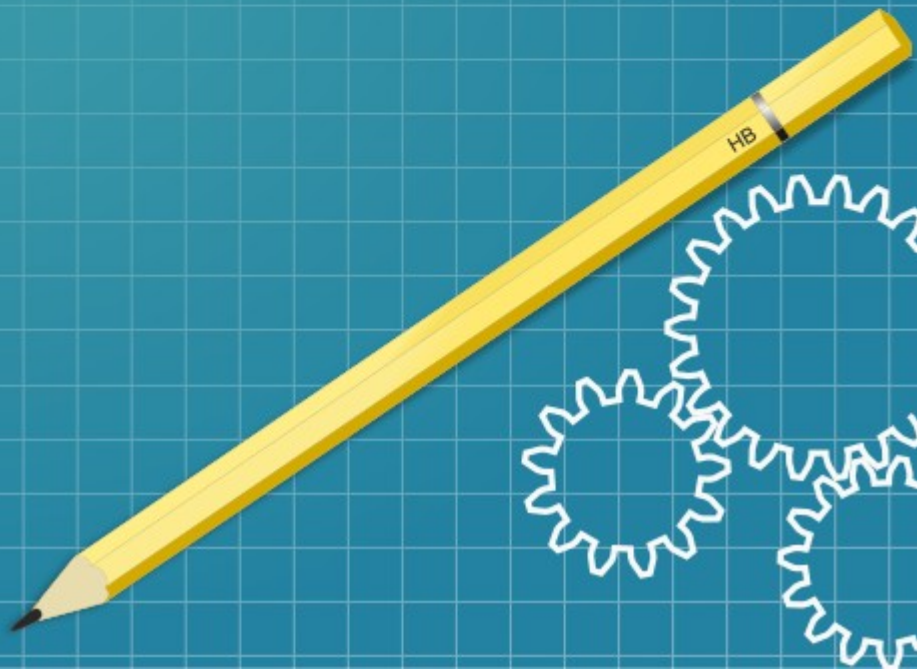
git reset --hard origin/master

Elimina todos los commit sin preservar sus cambios

git reset <commit>

Elimina todos los commit después del commit señalado

Siempre recuerden lo siguiente al trabajar con Git



In case of fire