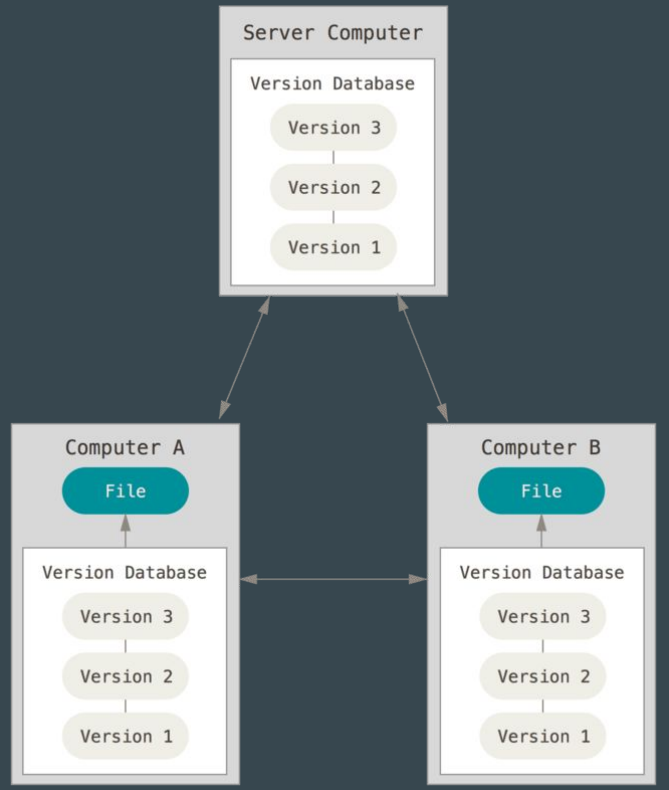
GIT

...

Resumen en base a https://git-scm.com/book/es/v2

¿Qué es un control de versiones, y por qué debería importarte?

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



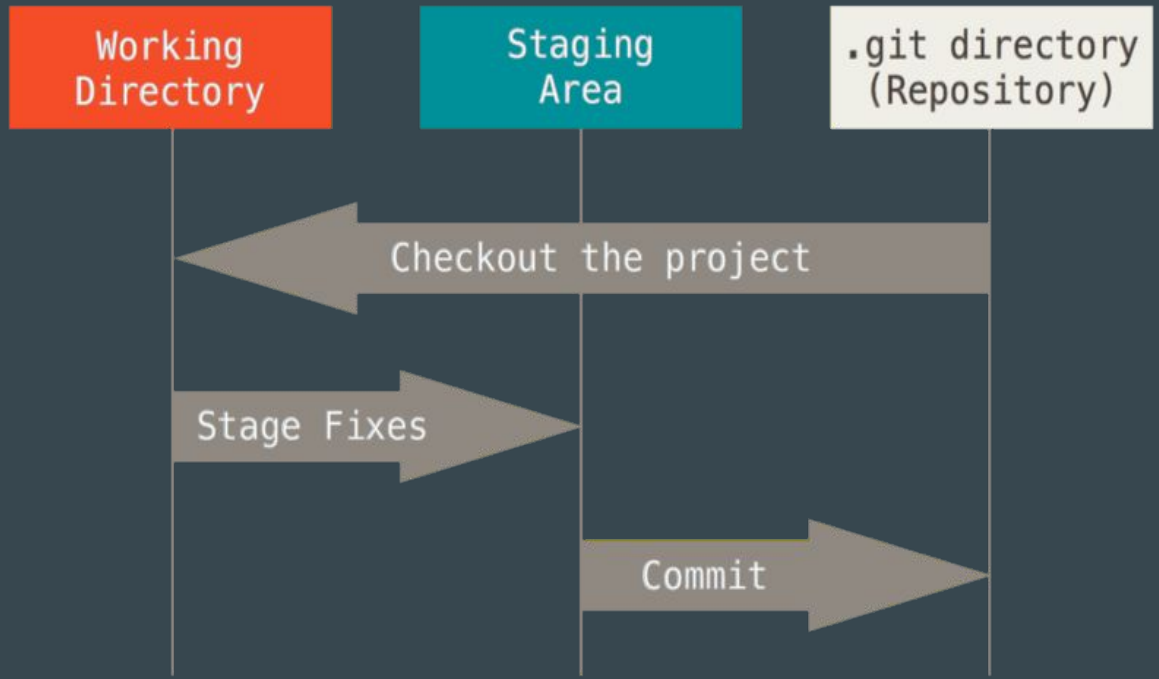
Los tres estados en git (Confirmado, modificado y preparado)

El flujo de trabajo básico en Git es algo así:

Modificas una serie de archivos en tu directorio de trabajo.

Preparas los archivos, añadiéndolos a tu área de preparación.

Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio de Git.



Configurando Git por primera vez

Lo primero que deberás hacer cuando instales Git es establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información

$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
$ git config –global user.password

Para configurar el password, podemos usar el “personal access token classic” para que tenga relación con la cuenta en github, está en Settings, Developer settings, Personal Accses Token, de la página de github.

Para comprobar la configuración:

$ git config –list

Primeros pasos, creando un repo local

Si estás empezando a seguir un proyecto existente en Git, debes ir al directorio del proyecto y usar el siguiente comando:

$ git init

Luego se puede crear un archivo de prueba, y comenzar a controlar las versiones.

Para “seguir” o “rastrear” un archivo

$ git add prueba.txt

Para dejar de rastrear un archivo

$ git rm --cached prueba.txt

Primeros pasos, realizando un commit

Luego de realizar el seguimiento de uno o un grupo de archivos, se puede crear un commit, para confirmar el seguimiento:

$ git commit -m “comienzo del seguimiento de archivos”

Para ver el estado:

$ git status

Para anular el commit (a partir del segundo commit)

$ git reset --soft HEAD~1

Añadir Repositorios Remotos

Para añadir un repo remoto:

$ git remote add origin https://github.com/manueladriancaceres-ort/ejemplo.git

Realizar el primer push (luego directamente git push, luego del commit):

$ git push -u origin main

Para traer la info del repo remoto:

$ git pull

Para anular un push, se usa:

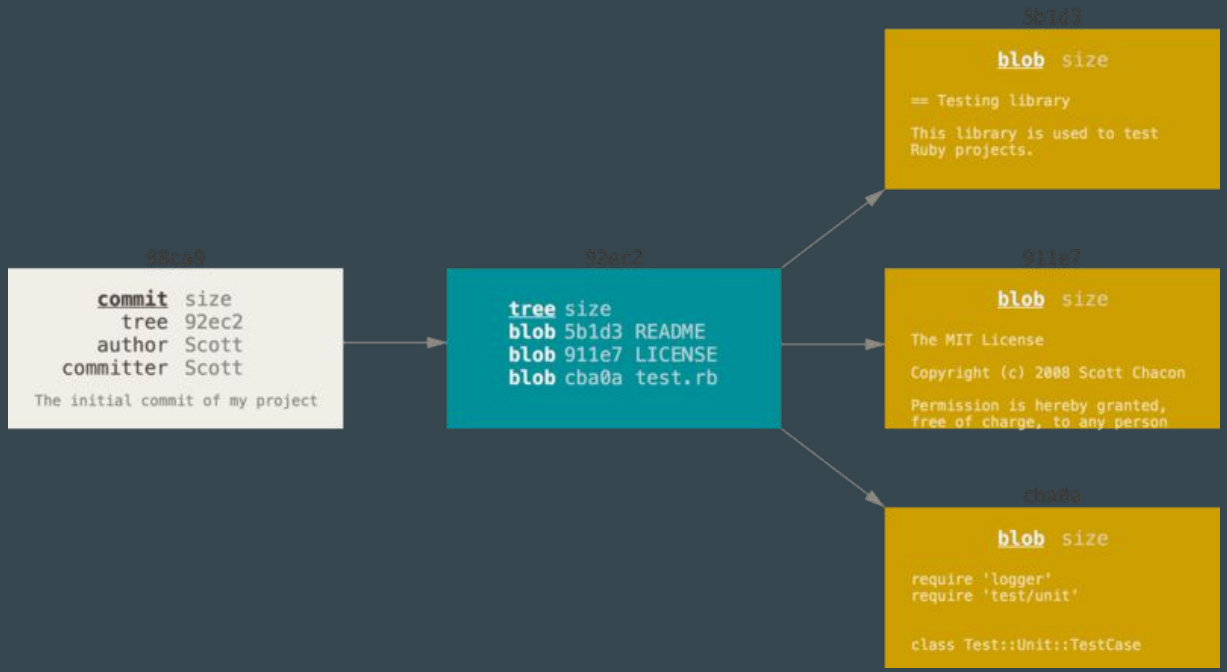
$ gi reset

Ramificaciones en Git - ¿Qué es una rama?

Cualquier sistema de control de versiones moderno tiene algún mecanismo para soportar el uso de ramas. Cuando hablamos de ramificaciones, significa que tú has tomado la rama principal de desarrollo (master) y a partir de ahí has continuado trabajando sin seguir la rama principal de desarrollo.

¿Qué es una rama?

Git no almacena los datos de forma incremental (guardando solo diferencias), sino que los almacena como una serie de instantáneas (copias puntuales de los archivos completos, tal y como se encuentran en ese momento).



Crear una nueva rama

¿Qué sucede cuando creas una nueva rama? Bueno..., simplemente se crea un nuevo apuntador para que lo puedas mover libremente.

$ git branch testing

Para cambiar de rama:

$ git checkout testing

Procedimientos Básicos de Fusión

Supongamos que tu trabajo con el problema #53 ya está completo y listo para fusionarlo (merge) con la rama master. Para ello, vas a fusionar la rama iss53. Simplemente, activa (checkout) la rama donde deseas fusionar y lanza el comando git merge:

$ git checkout master

Switched to branch 'master'

$ git merge iss53

Merge made by the 'recursive' strategy.

index.html | 1 +

1 file changed, 1 insertion(+)

Principales Conflictos que Pueden Surgir en las Fusiones

En algunas ocasiones, los procesos de fusión no suelen ser fluidos. Si hay modificaciones dispares en una misma porción de un mismo archivo en las dos ramas distintas que pretendes fusionar, Git no será capaz de fusionarlas directamente, por lo que verás un conflicto como este:

$ git merge iss53

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html

Automatic merge failed; fix conflicts and then commit the result.