GIT

→ Sistema de control de versiones distribuido

¿Para qué sirve?

① Guardar el historial de cambios a los archivos de un proyecto.

② Permitir a varios contribuir de forma o simultánea sobre el mismo proyecto.

Uso

Aplicación de consola línea de comandos:

git <subcomando>

Comandos

① git init → Inicializa un repo para utilizar Git en un proyecto. (1 vez x proyecto)

② git status → Muestra el estado del proyecto, cambios en los archivos que va hacer integrados en el por commit

③ git add . → Agregar archivos (Todos) al index para el próximo commit

④ git commit -m "mensaje" → copia de la última versión del proyecto

↳ Se identifica con 40 caracteres, pero no se puede referir a ellos con los primeros 7.

⑤ git log → historial del proyecto

Ramas

Permiten desviarse de la línea ppal de desarrollo

↳ Uso ① Hacer cambios que puede que integremos o no a la rama ppal del producto

② Mantener #s estados de desarrollo del producto

Cuando se crea un repo, x defecto, se crea una rama MASTER, es la rama ppal

Crear rama

6. git branch nombre-rama
7. git checkout nombre-rama → pasarse de rama

↳ git checkout -b nombre_rama → 2 pasos anteriores

8. ~~touch nombre.txt → permite crear un archivo vacío~~

Integrar cambios de una rama a la ppal

1. git checkout master → nos pramos en la rama master
2. git merge mi-rama → Integra los cambios de mi-rama a MASTER
3. git branch → Para ver las ramas de mi proyecto
10. git branch -d mi-ram → elimina rama

Repositorios Remoto

Tener en cuenta

1. Un repositorio remoto es independiente del repositorio local. => No se sincronizan automáticamente hay que hacerlo manualmente

Se debe registrar el repositorio

git remote add origin URL del repo remoto
↑ subcomando ↑ Nombre local ↑ ejc. git@github.com:user/repo.git

git push -u origin master → Empuja commits locales al repo remoto
↑ para no repetir nombre del repo remoto y la rama a la que queremos empujar esos commits

git pull → Jalar los cambios del repo remoto.

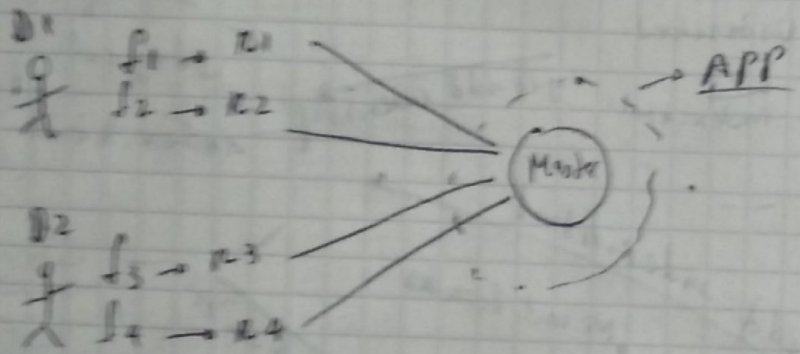
Se debe sincronizar rama x rama => lo que realmente se sincronizan en las ramas son los commits

Clonar Repositorio

→ Para cuando alguien quiere aportar en nuestro repositorio

git clone url del repo nombre de la carpeta que quiero el repo.

→ git remote -v para ver de ya registró repo remoto



GIT carpeta.git

.git

carpetas

① objects → commits, archivos y carpetas

② refs → heads, remotes, tags → ramas locales, etiquetas

rama = Ref a un commit.

Nombre rama y el último commit que hace ref esa rama

③ HEAD → Ref a la rama, tag o commit del workspace cada vez que hacemos git checkout, este archivo se va actualizando

④ index → Archivos listos para hacer commit

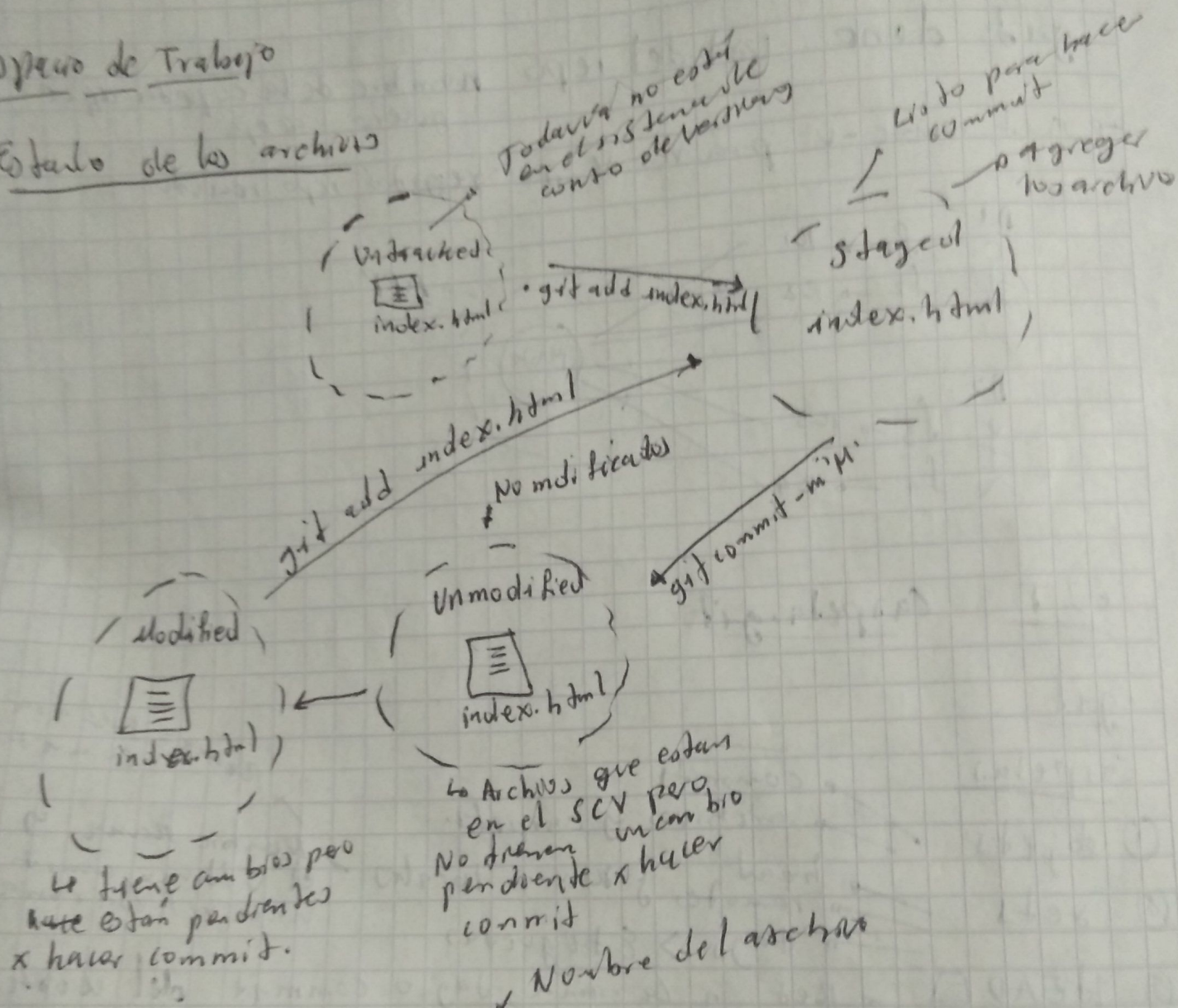
⑤ Config → Archivo de conf del repo.

Hooks Permite crear scripts que se ejecutan en determinados momentos

HEAD → Es un apuntador a $ commits

Espacio de Trabajo

Estado de los archivos



Nombre del archivo

→ git reset HEAD a.html → Esto Sino quiero hacer commit a ese archivo devolverlo del git.add.

Igual si lo hemos modificado

→ Para devolver al anterior commit, y el archivo y e en el control de versiones:

git checkout -- a.html

Pull Requests

Mecanismo para solicitar, revisar y discutir los cambios que se van a integrar a la rama ppal de un proyecto.
Lo puede ser de un fork al proyecto papal o dentro de un mismo proyecto.

Hacerlo

① git checkout -b #Issue o Nombre rama
1-add-title

~~edit~~
git branch -d (Nombre rama) elimina rama.

git diff ~~HEAD~~ Ver #s entre ramas

Wikis

Documentar un proyecto de forma colaborativa
Son páginas

Remote tracking Branches

nombre del repo remoto
git fetch origin ⇒ para actualizar el Remote Tracking branch con el Local branch, cuando alguien crea una rama ~~local~~ remota y no se actualiza la ~~remote~~ Local.

GIT PULL
→ git merge origin/master (Actualizar repo local)

hibernate

.gitignore => Mantiene una lista de archivos que NO deben ser incluidos en el sistema de control de versiones.

↳ Este archivo se debe incluir en el sistema de control de versiones

||

Crear archivo en windows CMD

echo "" > nombre.txt
contenido la extención

BitHub

Github shortcuts

Forks

Copia de un repositorio → hacer cambios al proyecto

↳ para hacer una línea paralela desarrollo

Creo [=] .txt Master

↳ commit

Issues

Son una forma de hacer seguimiento a tareas, mejoras y errores en el proyecto

Rebase

Se utiliza para actualizar una rama con respecto a la ppal u otra (Similar a un merge). También permite reescribir la historia de commits (mover, eliminar, cambiar mensajes, etc)

git rebase -i HEAD~3 → Apaday de ayer mañana

Reflog

Mantiene un registro de todos los cambios sobre las ramas y el HEAD

git reflog

git log --oneline master

Reset → Nos permite devolver al estado en el que está nuestro repo remoto.

git rebase --abort ⇒ Aborta rebase

El head apunta a una rama.

Flujos de Trabajo En equipo

Manager: Cuando se tiene muchos contribuidores externos

Centralizado

→ Se usa en la empresa

Hacer cambio

1) Crea una rama
2) Hacer uno o más commits
3) publica la rama a GitHub
4) Abrir un pull Request (PR)
5) Pedir revisión
6) Integrar cambios (Merge)

Reescribiendo la historia

Modificar la historia de commits de 1 rama

1) Amend

Permite cambiar el último commit x 1 nuevo

git commit --amend

git show → muestra el archivo

Reset

Permite deshacer cambios del área de staging y del espacio de trabajo. También permite retroceder en la historia de commits

⇒ git reset <commit> ⇒ Soft reset Quedan archivos en el espacio de trabajo

⇒ git reset --hard <commit> ⇒ Reset fuerte, No queda espacio de trabajo con cambios

git Reset HEAD^ → padre anterior → El padre del apuntado al que apunta un commit

git reset HEAD-3 → padre, del padre, del padre del head

Tracking Branches

Una rama de seguimiento es una rama local que tiene una relación directa con una rama local

~~git push -u~~ → crea una rama de seguimiento remoto para esta rama local (rama local que no está remotas) y la enlace

git push -u origin nombre-rama ← rama local

↳ si se olvida poner -u

↳ git branch --set-upstream-to=origin/master master

git branch -a → para ver todas las ramas

Stashing

Es una herramienta de Git que permite almacenar temporalmente los cambios al workspace sin necesidad de hacer commit

Proceso

git add .

git stash → limpia workspace, guarda el contenido temporal stash

git stash list → listar los stash, se listan los stash

git stash pop → devuelve a los ~~workspace~~ archivos al espacio de trabajo