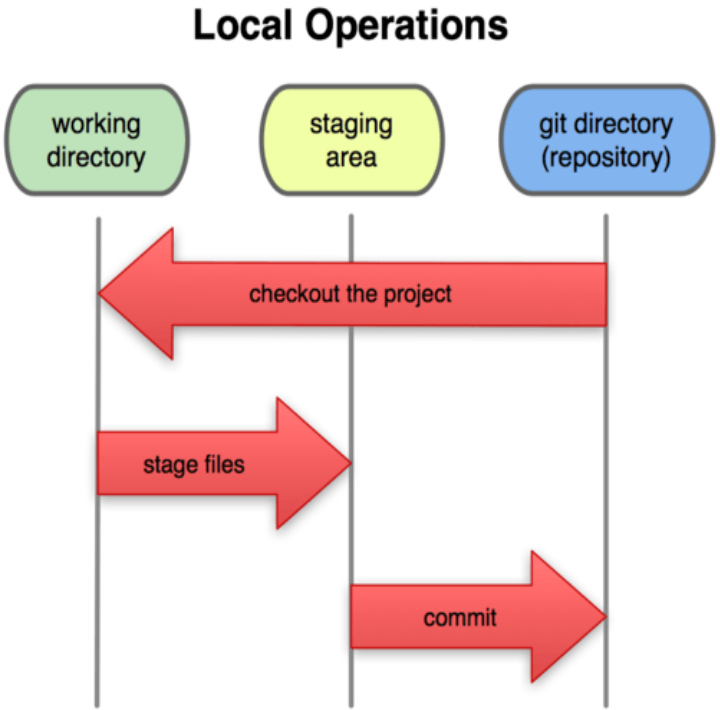

git + GitHub

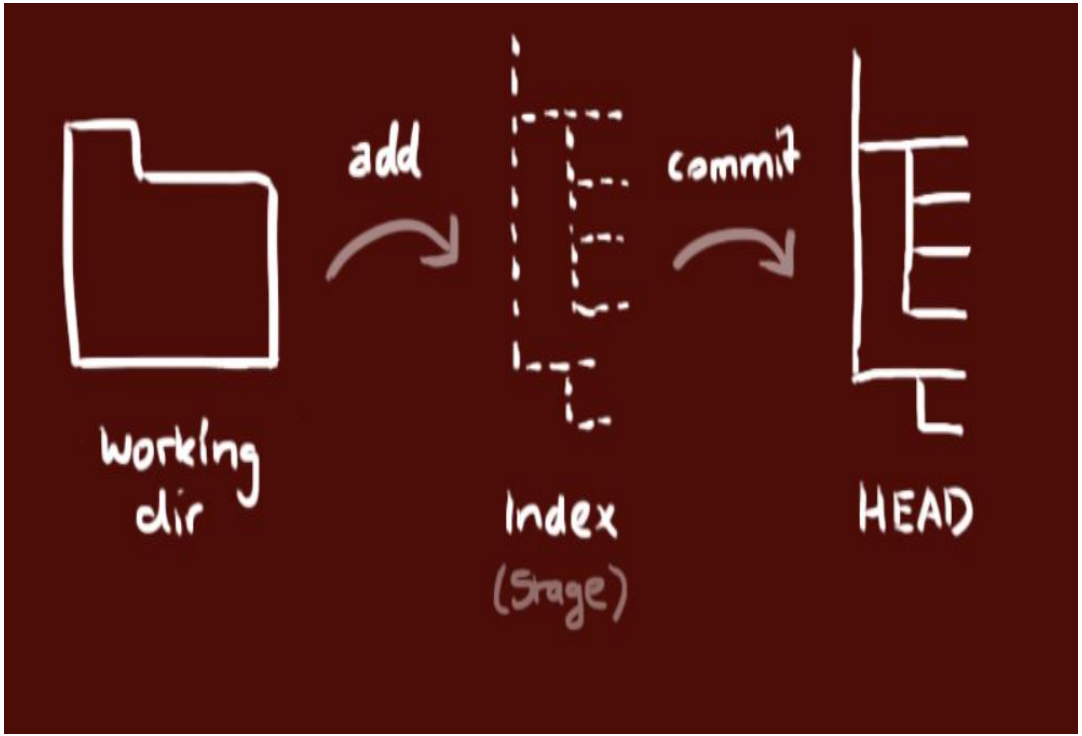
Git: Sistema de Control de Versiones

"Un sistema Que Registra Los cambios realizados sobre un archivo o Conjunto de Archivos a lo largo del Tiempo"

- **Creado por Linus Tolvards.**
- **Casi cualquier Operación es Local**
- **Es de tipo Distribuido.**
- **Los Torrrent manejan el mismo método**
- **Tiene Integridad (SHA1), no se puede corromper.**

3 ESTADOS:


Local Operations
working directory
staging area
git directory (repository)
checkout the project
stage files
commit


add
commit
working dir
Index
(Stage)
HEAD

GitHub: La plataforma

“Es una plataforma en la que se almacena los cambios de un proyecto, y se puede colaborar y ayuda a distribuir”

- **Funciona como una Red Social.**
- **En vez de compartir fotos, compartes Codigo.**
- **Te ayuda a Organizar Tu proyecto.**
- **Es una Website (github.io).**
- **Tiene Integridad (SHA1), no se puede corromper.**

INSTALACION Y CONFIGURACION

una vez instalado:

#git –-versión **//muestra la versión**

Configuración Usuario y Contraseña:

#git config –-global user.email correo@gmail.com

#git config –-global user.name nombre-usuario

#git config –-global color.vi true ***//activar colores***

#git config –-list ***//muestra las configuraciones globales***

CREANDO REPOSITORIOS

#git –-init nombre-proyecto //carpeta MAIN del proyecto
#git status //muestra el estado actual
#git add nombre-archivo //agrega determinado archivo
#git rm –cached nombre-archivo //elimina archivo del STAGE AREA
#git add –A nombre-archivo //agrega todos los archivos

#git rm –f nombre-archivo //elimina archivo totalmente

CONFIRMANDO CAMBIOS

#git -m “mensaje” //agregando el commit

#git commit --amend //concatena al commit anterior

#git log //muestra todos los commits hasta el momento

ETIQUETANDO CONFIRMACIONES

#git tag -a 0.5 –a “estable” //se etiqueta como la versión “0.5”

#git tag -l //muestra las versiones del proyecto

#git tag -d 1.1 //borro la etiqueta 1.0

#git tag 1.0 SHA1 //crea una etiqueta 1.0 “a un commit pasado”

#git tag –f -a 0.1 –m “msj” SHA1 //commit y etiqueta a la vez

REVISANDO LA HISTORIA DEL PROYECTO

#git log	//es un comando personalizable.
#git log --oneline	//muestra los commits, resumida.
#git log –oneline --graph	//muestra el grafico de las ramas.
#git log –n	//muestra la cantidad “n” de commits que quisiéramos ver.

CAMBIOS ENTRE VERSIONES

#git diff SHA1	//compara entre ese commit, y el ultimo commit (HEAD) que tengamos.
#git diff SHA1 SHA1	//se compara entre 2 commits.

DESHACIENDO CAMBIOS

#git reset --soft SHA1	**//quita el cambio, pero lo mantiene el STAGE, esperando el nuevo commit.**
#git reset --mixed SHA1	**//quita el cambio, y lo deja en el WORKING DIRECTORY.**
#git reset HEAD [file]	**//quita el ultimo cambio guardado (STAGE), vuelve al WORKING DIRECTORY.**
#git reset –-hard SHA1	**//borra todo, lo que se encuentra en modo commit y STAGE.**

CAMBIANDO EDITOR POR DEFECTO

#git config –global core.editor “editor-nuevo --wait”

- **Posteriormente reiniciar editor.**

#git log --oneline **//muestra los commits, resumida.**
#git log –oneline --graph **//muestra el grafico de las ramas.**
#git log –n **//muestra la cantidad “n” de commits que quisiéramos ver.**

CAMBIOS ENTRE VERSIONES

#git diff SHA1 **//compara entre ese commit, y el ultimo commit (HEAD) que tengamos.**

#git diff SHA1 SHA1 **//se compara entre 2 commits.**

RAMAS(versiones alternas al proyecto)

- **Para que ramas?** Para agregar nuevos features.

#git branch nueva-rama //creando una nueva rama.
#git branch -l //lista todas las ramas.
#git branch –d nombre-rama //elimina la rama.
#git branch –m actual nuevo //RENOMBRA la rama.

MOVIENDONOS ENTRE RAMAS Y VERSIONES

#git checkout nombre-rama //se mueve a la rama nombrada.
#git checkout SHA1 //se mueve a la rama, con ese identificador.
#git checkout -b nueva-rama //crea la rama y también ingresa.

MEZCLANDO RAMAS Y RESOLVIENDO CONFLICTOS

Regla: Ubicarse en la RAMA donde queremos los cambios PRIMERO:

#git merge nombre-rama //mezcla cambios de dicha rama.

#git branch -l //lista todas las ramas.

#git branch –d nombre-rama //elimina la rama.

#git branch –m actual nuevo //RENOMBRA la rama.

GUARDAR CAMBIOS TEMPORALMENTE

- *Guarda un cambio, sin que sea un COMMIT, seria el paso previo.*
- *Previamente, el cambio debe estar en modo STAGE.*

#git stash //guarda el cambio temporalmente.

#git stash list //enlista todos los cambios temporales.

#git stash drop stash@{1} //elimina el stash indicado.

#git stash apply stash@{3} //inserta el stash indicado.

GITHUB

- ***FORK:*** *clona/copia un proyecto, en una determinada cuenta, con el mismo nombre.*

LLAVE SSH

- *Por seguridad, cada vez que suba un cambio, GIT te pedirá USSER Y PASS.*

Generando la llave ssh:

#ssh-keygen –t rsa –b 4096 –C "correo@Gmail.com" (correo de github)

GIT REMOTE

#git remote add nombre-repo https://direccion **//agregando remoto.**

#git remote -v **//muestra todos los repos remotos.**

#git remote remove nombre-repo **//elimina el dicho repositorio.**

PULL/FETCH

#git fetch nombre-remoto rama-remoto //trae los cambios.

#git merge oficial/master –allow-unrelated-histories //crea rama transitoria.

- *Para conbinar los anteriores 2 pasos, en 1, y mas RAPIDO, se usa:*

#git pull nombre-remoto nombre-rama-local //trae los cambios y crea.

PUSH

- *Enviando cambios al repositorio.*

#git push nombre-remoto nombre-rama-local //sube los cambios sin tags.

#git push nombre-remoto nombre-rama-local --tags //sube los cambios

ARCHIVOS NO DESEADOS

- *Ignorando archivos no deseados.*
- *Dentro del proyecto, creamos el archivo* ***.gitignore***