**Git & GitHub**

**git init:** Inicializa un repertorio de Git vacío donde se guardarán los archivos. Se ejecuta estando en la dirección del archivo que queremos subir.

**git add** NombreArchivo**:** Notifica a Git la existencia del archivo especificado. Usando punto (**.**) selecciona todos los archivos de la carpeta.

**git rm - -cached** NombreArchivo**:** Comando contrario a *git add.* Quita el archivo especificado de la memoria ram y del seguimiento de git.

**git commit -m "**versión 1**":** Envía los últimos cambios del archivo a la base de datos del sistema de control de versiones.

Después que se editó algún archivo:

git add . > git commit -m "Cambios a v1" >

**git status:** Revisa el status del Git. Se notificará cuando algún archivo haya tenido alguna edición.

**git show** NombreArchivo**:** Muestra los cambios históricos realizados. Qué líneas, cuándo y quién ha realizado los cambios.

**git log** NombreArchivo**:** Muestra la historia entera de un archivo.

**VIM:** Salir de la interfaz **Shift + Esc + Z + Z**

**git diff:** Compara la información guardada en staging/ram previa a realizar un commit, con nuevas modificaciones realizadas.

**git diff** “código de versión 1” “código de versión 2”: Compara los cambios hechos en dos versiones diferentes del mismo archivo. El código de versión se puede obtener con *git log*.

Al ejecutar **git init** se crea**:**

Staging: Es una especie de memoria ram o área de preparación intermedia donde se colocan los cambios mediante **git add** antes de ser totalmente subidos al repositorio mediante un **commit**.

Un archivo en este punto se considera como tracked y fuera de este untracked.

**/.git**: Es directorio oculto que se crea en el directorio raíz, almacena toda la información necesaria para gestionar el control de versiones y el historial de cambios del repositorio. Nombre de la rama principal por defecto → **master**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ramas comunes del repositorio git:

* Master: Rama principal donde suele estar la versión funcional y en uso.
* Development: Rama experimental donde se trabajan en cambios para ser unidos a la rama master.
* Hotfix: Cambios que se realizan en el instante debido a fallos o problemas en la rama master.

**git checkout “código de versión” NombreArchivo**: Importar una versión de un archivo almacenado en el repositorio. Retorna el archivo a una versión anterior pero el cambio queda pendiente de commit.

**git checkout master NombreArchivo:** Se retorna a la última versión o versión master del archivo especificado. Se utiliza después de ver alguna versión pasada usando “git checkout + código de versión anterior” para retornar a la última versión del head.

**git reset** **“código de versión” --hard**: Retorna el archivo a la versión que tenía según el código enviado. Elimina cambios sin guardar.

**git reset “código de versión” --soft**: Retorna el archivo a la versión que tenía según el código enviado. Los cambios desechos pasan al área de preparación donde se pueden revisar antes de realizar un commit.

**git branch** **NombreRama**: Crea una nueva rama para trabajar en el directorio.

**git checkout** **NombreRama**: Selecciona la rama en la que se está trabajando. Se puede revisar la rama que se está utilizando actualmente en “git status” en la parte de “On branch…”.

**git branch:** Muestra las ramas existentes en el sistema.

**git merge NombreRama -m:** Se utiliza fusionar distintas ramas del git.

**git show--branch** / **git show--branch --all:** Ambos comandos muestran un resumen de las branch.

**git log --all --graph --decorate --oneline:** Muestra un resumen corto de los cambios realizados en el repositorio.

**Generación de Key**

**ssh-keygen -t rsa -b 4096 -C “correo”:** Genera la llave/key pública y privada.

**ssh-add ~/.ssh/id\_rsa:** Agrega al sistema la clave privada generada.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**alias NombreAlias=“CódigoAContenerEnElAlias”:** Asigna el *código* o comando especificado a la palabra que designemos como *NombreAlias*. Luego se podrá utilizar en sustitución de código completo.

Tags

**git tag -a NombreTag -m “Mensaje de registro” “código de versión”:** Crea un tag que sirve de punto de referencia a una versión específica del proyecto.

**git tag:** Muestra la lista de tags existentes.

**git show-ref --tags:** Muestra la lista de tags con detalles de código y descripción.

**git push origin --tags:** Envía a GitHub los tags creados localmente.

**git tag -d NombreTag:** Elimina el tag especificado. (Únicamente de manera local)

**git push origin : refs/tags/NombreTag:** Combinado con el comando para borrar de manera local un tag previamente ejecutado; elimina el tag del repositorio en GitHub.

**gitk:** Abre un programa la historia de los cambios hechos en el proyecto.

**git push origin NombreRama/Branch:** Envía al repositorio en GitHub la rama especificada.

**git clone LinkRepositiorioGitHub:** Importa una versión de un repositorio desde GitHub.

Una vez clonado el repositorio desde GitHub se necesitará que el dueño habilite el usuario que lo importó para poder realizar cambios y posteriormente realizar

**git push origin rama/branch:** Sube al repositorio en línea los cambios realizados de forma local.

**git pull origin rama/branch:** Descarga la última versión el repositorio en línea a nuestros archivos locales.