**Comandos en Git**

Escala de tiempo

Descripción generada automáticamente

Git add git commit

<https://git-scm.com/book/es/v2>

<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

generar llave pública y privada

publica en el repositorio remoto y privada en la pc

Crear un nuevo repositorio y le tengo que colocar mi nombre de usuario de Git Hub y me creara un readme como presentación para cuando vean mi perfil.

**Git init**

Creará un nuevo repositorio local GIT en el directorio actual.

Git init (nombre del Proyecto) puedes crear un repositorio dentro de un nuevo directorio especificando el nombre del proyecto.

**Git clone**

Se usa para copiar un repositorio. Si el repositorio está en un servidor remoto, usa:

Git clone (enlace del repositorio)

**Git status**

Muestra las modificaciones realizadas en un archivo.

**Git add**

Nos permite pasar los archivos del primer estado al segundo estado (Staging Área).

git add nombredelarchivo

Agrega un solo archivo.

git add.

Agrega todos los archivos que están dentro del repositorio.

**Git commit**

Creará una instantánea de los cambios y la guardará en el directorio git.

git commit –m “El mensaje que acompaña al commit va aquí”

Si se tiene un error ortográfico en el mensaje y quieres cambiarlo se cambia de esta forma:

**git commit –amend -m “<nuevo mensaje>”**

**Si se quiere añadir un nuevo archivo al repositorio, pero no se quiere crear un nuevo commit, lo subes con git add y usas este comando:**

**git commit –amend –no-edit**

**Git push**

Se usa para enviar confirmaciones locales a la rama maestra del repositorio remoto.

git push origin <master>

Reemplaza <master> con la rama en la que quieres enviar los cambios cuando no quieras enviarlos a la rama maestra.

**Git pull**

fusiona todos los cambios que se han hecho en el repositorio remoto con el directorio de trabajo local.

**Git reset**

Sirve para reiniciar el git add y poder hacer un commit en especifico

“git reset nombredelarchivo”

Otra manera en la que podemos resolver nuestro problema cuando sean más de un archivo, podemos hacer “git reset .”

Con este comando podemos deshacer cosas/cambios sobre los commits que hemos hecho anteriormente. Este comando tiene 3 variaciones: soft, mixed, hard

Soft: Lo que hace es regresar los archivos al área de staging.

Mixed: Con el podemos regresar los archivos al directorio del trabajo, nuestros cambios no se tocan, y nos aparecerán como modificaciones.

Hard: Debemos tener cuidado, con hard lo que hace es eliminar todos los cambios del commit que especificamos.

Por defecto el que está configurado es “mixed”.

git reset --(soft, mixed o hard) (hash del commit) sirve para borrar un commit

git reset HEAD~2(el numero representa la cantidad de commits a regresar)

**Git rm**

Lo que podemos hacer es eliminar nuestros archivos de nuestro repositorio y no solo eso, se encarga de agregar ese cambio (Eliminar un archivo) al staging area.

git rm (nombredelarchivo)

-r: esto significa a “recursivo” eliminando todos los archivos que están dentro de alguna carpeta.

–cached: Esta opción es para quitar etapas y eliminar rutas solo del índice. Los archivos de árbol de trabajo, ya sean modificados o no, se dejarán solos.

-f: Forced. se utiliza para eliminar un archivo si el archivo no está actualizado con su última confirmación de salida. Es para evitar que elimine un archivo en el que ha realizado cambios, pero aún no los ha registrado.

-q:Quiet. git rm normalmente genera una línea por cada archivo eliminado. Esta opción suprime esa salida

**Git log**

Podemos ver todos los commits que hemos hecho y a su vez la información sobre ellos, el usuario, email, la fecha e incluso el mensaje que agregamos en él.

git log --oneline Te muestra los commit en una línea junto con su hash (clave del commit) más corto

git log --all: Te muestra todos los commit

git log -n: Te mostrará solo en número de los commits que indiques

git log --graph: Te mostrará una gráfica sobre cada commit y fusiones que hagas

git log –stat:  Imprime tras cada confirmación una lista de archivos modificados, indicando cuántos han sido modificados y cuántas líneas han sido añadidas y eliminadas para cada uno de ellos, y un resumen de toda esta información.

git log --grep= PalabraClaveDelCommit: sirve para buscar algún commit con una palabra

**Git checkout**

Ahora para poder volver en el tiempo solo tenemos que hacer lo siguiente: “git checkout hashdelcommit”

Para volver a nuestro último commit basta con hacer “git checkout master” o la otra forma es con “git reflog” y buscar nuestro último commit por su mensaje y utilizando su hash.

**Git remote**

git remote add origin (url del repositorio remoto)

sirve para subir el repositorio local a GitHub

**Git tag**

Es un punto específico en nuestro proyecto que está ligado a un hash (commit) de nuestro proyecto, en el cual nosotros podríamos especificar la versión de nuestro proyecto.

No cambia y se puede descargar esa versión del proyecto.

Sirve por si se termina un proyecto y se guarde esa versión y crear otra sin temor a perderla.

Git tag

Sirve para ver que versión es.

git tag <versión>

Solo se coloca el numero de la versión que será.

git tag -a v.1.0 -m “mensaje”

Se coloca la versión que es junto con un mensaje

git push –follow-tags

para subir al repositorio en GitHub nuestro tag.

**Git stash**

El comando git stash almacena temporalmente (o guarda en un stash) los cambios que hayas efectuado en el código en el que estás trabajando para que puedas trabajar en otra cosa y, más tarde, regresar y volver a aplicar los cambios más tarde.

Principalmente se usa cuando trabajas en otra rama y todavía no vas a realizar un commit y no se pierda el proceso realizado.

git stash

git stash list

git stash apply stash@{x}

**Git rebase**

Sirve para fusionar ramas pero a diferencia de merge, pero ordena un commit detrás del otro, no es muy recomendable usar por que puede afectar commits pasados.

Git rebase <nombre de la rama>

Git rebase en modo estándar tomará automáticamente las confirmaciones de tu rama de trabajo actual y las aplicará al encabezado de la rama anterior.

**Git clean**

Sirve para borrar los archivos que no se hayan hecho git add.

Se puede combinar con el git reset, los bajas del git add y ya los puedes eliminar con git clean.

git clean -n

Te muestra los archivos que podría eliminar pero no los borra.

git clean -f

Borra los archivos

**Git amend**

Sirve principalmente para dos cosas, si te equivocaste de nombre en el commit y deseas modificarlo es con:

**git commit –amend -m “<nuevo mensaje>”**

**O si quieres añadir un nuevo archivo pero no quieres hacer otro commit, primero lo añades con git add y después usas:**

git commit –amend –no-edit

Con eso se añade al mismo commit.