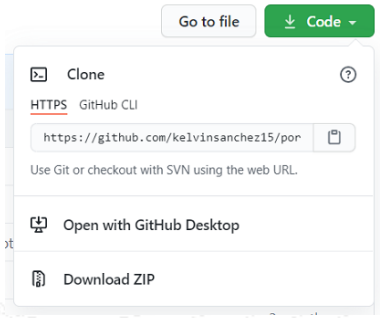
Planear actividades de construcción del software de acuerdo con el diseño establecido.

Producción y compilación del contenido digital Inst. Angélica M. Triana
Solo fines académicos

git clone -recorderis

Tengo un archivo que quiero enviar al repositorio remoto que me compartieron. Para ello realizo lo siguiente:

Usa la opción Clone -> HTTPS, y copia el enlace proporcionado.copy-https-url2



1. Ejecuta git clone https://github.com/nombreDeUsuario/repositorio.git en la terminal. Aquí, nombrelDeUsuario y repositorio serán reemplazados por los valores proporcionados en el enlace copiado.
2. Ejecuta git init en la terminal. Esto inicializará la carpeta/repositorio que tienes en tu computador local.
3. Ejecuta git add . en la terminal. Esto hará un seguimiento de los cambios realizados en la carpeta de tu sistema desde el último commit. Si es la primera vez que haces commit a los contenidos de la carpeta, se añadirán todos.
4. Ejecuta git commit -m "inserta Mensaje aquí". Esto preparará los cambios añadidos/rastreados en la carpeta de tu sistema para empujar a Github. Puedes reemplazar inserta el Mensaje aquí con cualquier mensaje de confirmación relevante de tu elección.
5. Finalmente, ejecuta git push origin main para empujar tus archivos a Github. Ten en cuenta que la última palabra en el comando main, no es una entrada fija cuando se ejecuta git push, puede ser reemplazada por cualquier “nombre_de_rama” relevante.

Diferencia entre git fetch y git pull

git fetch es el comando que le dice a tu git local que recupere la última información de los metadatos del original (aunque no hace ninguna transferencia de archivos. Es más bien como comprobar si hay algún cambio disponible).

git pull por otro lado hace eso Y trae (copia) esos cambios del repositorio remoto.

Recuperado de:
https://www.freecodecamp.org/espanol/news/git-fetch-vs-pull-cual-es-la-diferencia-entre-los-comandos-git-fetch-y-git-pull/

Contenido

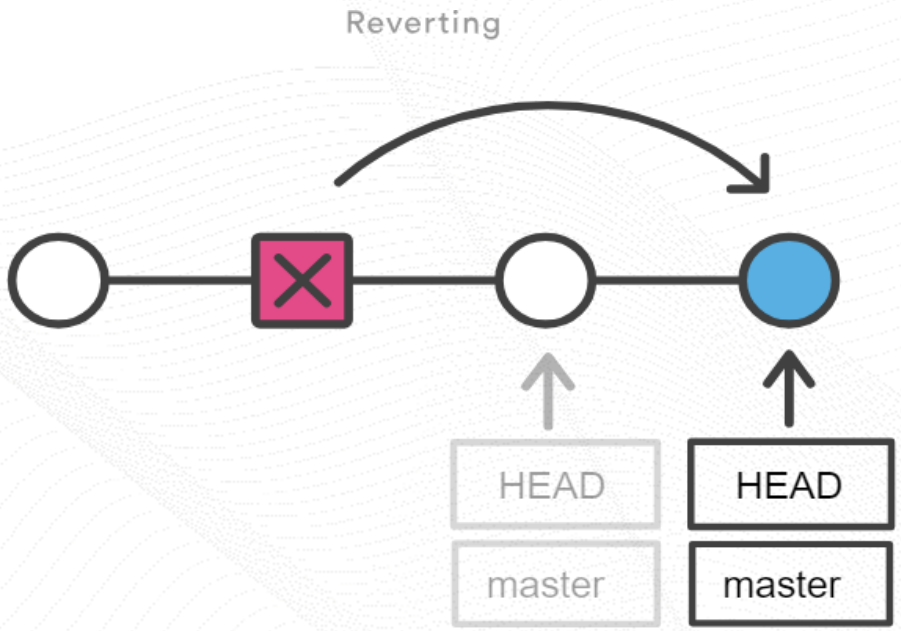
git revert/git reset
git remote
git config y git clean
Generalidades
Concepto fork

Actividades de la sesión

git revert

git revert (parte 1)

El comando git reset te permite **restablecer** tu estado actual a un estado específico. Puedes restablecer el estado de archivos específicos, así como el de todo una rama. Esto es útil si **aún no has subido tu commit a GitHub** o a otro repositorio remoto.



git revert (parte 2)

Resulta primordial entender que **git revert solo deshace una confirmación**; no "revierte" el proyecto a un estado anterior eliminando todas las confirmaciones posteriores. **La idea es revertir pero que no nos borre lo sucedido**.

Para ello, clone un repositorio que contenga un historial de commits:

Ejemplo>
git clone https://github.com/amtriana62/nodia.git
cd nodia //nos ubicamos en la carpeta clonada
git status //se realiza un git status para revisar en que rama nos encontramos
git log –online //para revisar los commits realizados.

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop
(main|MERGING)
$ git clone https://github.com/amtriana62/nodia.git

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop
(main|MERGING)
$ cd nodia

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git log --oneline
```

git revert (parte 3)

En este caso se va a deshacer el último commit. El que esta en HEAD.

Para ello podemos hacer en la terminal un **git revert** HEAD:

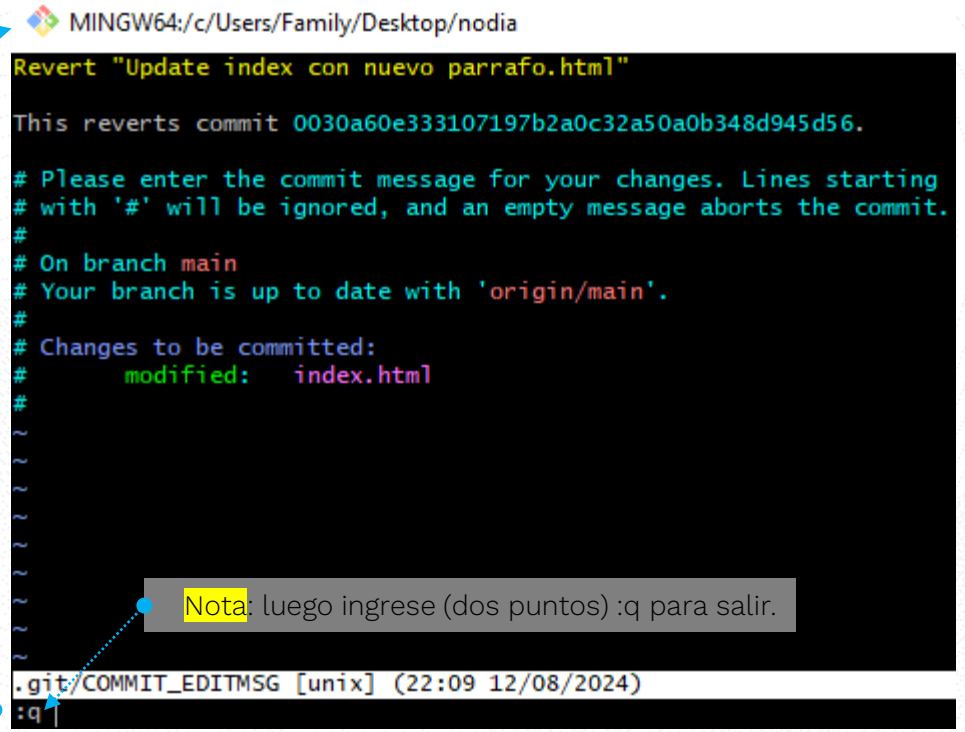
```
$ git log --oneline
0030a60 (HEAD -> main, tag: v2.0.0, tag: v1.0.0,
origin/main, origin/HEAD) Update index con nuevotml
071bf46 Create README.md
32b1406 index no dia

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git revert 0030a60
```

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git revert 0030a60
[main fc5b9fd] Revert "Update index con nuevo
parrafo.html"
 1 file changed, 1 insertion(+), 2 deletions(-)
```

Se cargará un mensaje al respecto al "delete", pero, si revisamos no ha eliminado del todo el commit de los logs.

Luego se muestra lo siguiente:



Nota: luego ingrese (dos puntos) :q para salir.

git revert (parte 4)

Se realiza un **git log --oneline** en el repositorio. Observe que en el historial esta el revert y el id_commit que se revirtió:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git log --oneline
fc5b9fd (HEAD -> main) Revert "Update index con nuevo parrafo.html"
0030a60 (tag: v2.0.0, tag: v1.0.0, origin/main, origin/HEAD) Update
index con nuevo parrafo.html
071bf46 Create README.md
32b1406 index no dia
```

Se utiliza **git show HEAD** para revisar donde quedo:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git show HEAD
commit fc5b9fdd4ff468de2938b1dfbdc58d7802f7a903 (HEAD -> main)
Author: amtriana62 <amtriana62@misena.edu.co>
Date:   Mon Aug 12 22:09:27 2024 -0500

    Revert "Update index con nuevo parrafo.html"

    This reverts commit 0030a60e333107197b2a0c32a50a0b348d945d56.
```

Ahora realice un cambio local, guárdelo y envíelo al repositorio remoto.

git revert (parte 5)

Siguiendo el ejemplo>

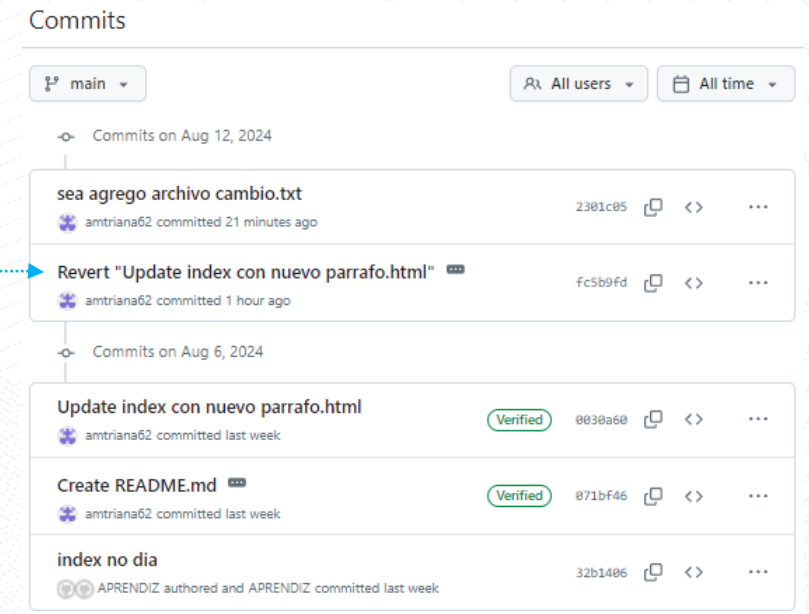
```
$ git status
On branch main
... muestra más info
        ejemploCambio.txt ...

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git add .

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git commit -m "sea agrego archivo cambio.txt"
[main 2301c05] sea agrego archivo cambio.txt
 1 file changed, 1 insertion(+)
 create mode 100644 ejemploCambio.txt

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia
(main)
$ git push origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 631 bytes | 157.00
KiB/s, done.
Total 6 (delta 0), reused 2 (delta 0), pack-reused 0
To https://github.com/amtriana62/nodia.git
   0030a60..2301c05  main -> main
```

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git log --oneline
2301c05 (HEAD -> main, origin/main, origin/HEAD) sea
agrego archivo cambio.txt
fc5b9fd Revert "Update index con nuevo parrafo.html"
0030a60 (tag: v2.0.0, tag: v1.0.0) Update index con
nuevo parrafo.html
071bf46 Create README.md
32b1406 index no dia
```



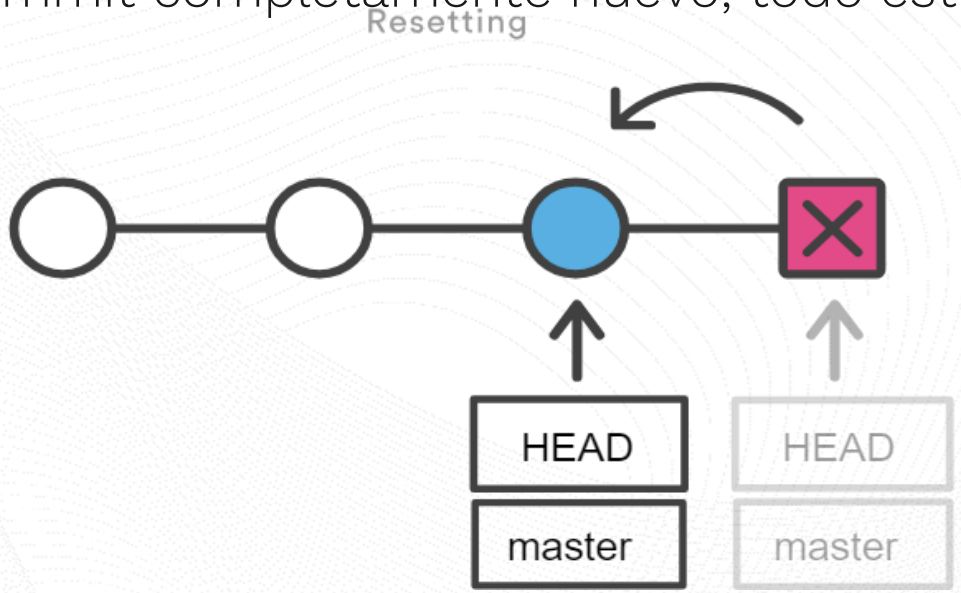
La idea es revertir pero que no nos borre lo sucedido.

git reset

git reset (parte 1)

Si ya has subido tu commit a un repositorio remoto, se recomienda que no uses git reset, ya que reescribe el historial de commits. Esto puede hacer que **trabajar en un repositorio con otros desarrolladores y mantener un historial** consistente de commits sea muy difícil.

En su lugar **es mejor usar git revert**, que deshace los cambios realizados por un commit anterior creando un commit completamente nuevo, todo esto sin alterar el historial de commits.



git reset (parte 2)

git reset ya se ha utilizado para “devolvernos en el tiempo” pero aquí lo recordaremos:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git log --oneline
2301c05 (HEAD -> main, origin/main, origin/HEAD) sea agrego archivo cambio.txt
fc5b9fd Revert "Update index con nuevo parrafo.html"
0030a60 (tag: v2.0.0, tag: v1.0.0) Update index con nuevo parrafo.html
071bf46 Create README.md
32b1406 index no día

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git reset --hard 0030a60
HEAD is now at 0030a60 Update index con nuevo parrafo.html
```

Quiero volver a este punto.

Se realiza el git reset --hard

El historial después del punto del reset se elimina:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git log --oneline
0030a60 (HEAD -> main, tag: v2.0.0, tag: v1.0.0) Update index con nuevo parrafo.html
071bf46 Create README.md
32b1406 index no dia
```

- Git reset es más forzoso, elimina puntos anteriores hasta el señalado.
- Usar reset es estar convencido que se requiere hacer esta acción porque lo borra también del log.

Actividad de la sesión

De forma individual,

git reset tiene varios modos de trabajo, algunos de ellos son **git reset --hard** y **git reset --soft**, por favor investigue este último y realice un ejemplo al respecto.

Tenga en cuenta que:

--hard: restablece el índice y el árbol de trabajo. Cualquier cambio en los archivos rastreados en el árbol de trabajo desde el commit son descartados.

--soft: no restablece el fichero índice o el árbol de trabajo, pero restablece HEAD para commit. Cambia todos los archivos a "Cambios a ser commited".

Posteriormente, se seleccionará un aprendiz para que socialice y explique el ejemplo.


GIT RESET --HARD
TIME TRAVEL!
imgflip.com
GIT RESET
FUTUBANDERA.CL
¡SIGNIFICA PELIGRO!
imgflip.com

git remote

git remote

El comando git remote te permite crear, ver y eliminar conexiones con otros repositorios. Las conexiones remotas se asemejan más a marcadores que a enlaces directos con otros repositorios.

Este comando ya se ha utilizado cuando se vinculo el repositorio local con el remoto con las siguientes líneas de código:

```
git remote add origin url
git branch –M main
git push –u origin main
```

Otros comandos relacionados son:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git remote
origin
```

git remote
Enumera las conexiones remotas que tienes con otros repositorios.

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git remote -v
origin  https://github.com/amtriana62/nodia.git (fetch)
origin  https://github.com/amtriana62/nodia.git (push)
```

git remote –v
Lo mismo que el comando anterior, pero incluye la URL de cada conexión.

Eliminaciones-generalidades

Eliminaciones (parte 1)

Para eliminar una url remota, usamos el comando **git remote rm** seguido del nombre del remoto para eliminar un remoto.

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git remote
origin

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git remote rm origin

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git push //verificamos
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using
    git remote add <name> <url>
```

Para eliminar el seguimiento de un solo archivo del repositorio local es **git rm** *nombreArchivo*:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git status
On branch main
nothing to commit, working tree clean

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git rm README.md
rm 'README.md'
```

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ git status //verificamos
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    README.md
```

Eliminaciones (parte 2)

¿Cómo eliminar un repositorio local en Git?. Para eliminar el seguimiento del repositorio local, es decir, esa carpeta ya no será la del repositorio use el comando: **rm –rf *.git***:

Revisamos que exista la carpeta del repositorio con el comando **ls -al**:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ ls -al
total 17
drwxr-xr-x 1 Family 197121  0 Aug 13 00:22 ./
drwxr-xr-x 1 Family 197121  0 Aug 12 21:51 ../
drwxr-xr-x 1 Family 197121  0 Aug 13 00:29 .git/
-rw-r--r-- 1 Family 197121 39 Aug 12 23:41 index.html

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main)
$ rm -rf .git
```

Para ver los archivos podemos usar el comando ls (no es comando git pero nos sirve).

Verificamos que ya no existe, es decir, ya no se le hace seguimiento:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (main|MERGING)
$ ls -al
total 13
drwxr-xr-x 1 Family 197121  0 Aug 13 00:35 ./
drwxr-xr-x 1 Family 197121  0 Aug 12 21:51 ../
-rw-r--r-- 1 Family 197121 39 Aug 12 23:41 index.html
```

Eliminaciones (parte 3)

Recuerde que para eliminar una rama local es **git branch –d *nombreRama*** para mas información diríjase al siguiente enlace: https://www.freecodecamp.org/espanol/news/como-borrar-una-branch-de-git-en-ambos-repositorios-local-y-remoto/

Para borrar los archivos en un repositorio de GitHub, consulte el siguiente enlace: https://docs.github.com/es/repositories/working-with-files/managing-files/deleting-files-in-a-repository

Cambiar mensaje de un commit

Cambiar el mensaje de un commit

Se utiliza git commit *archivo.extension* --amend –m “*nuevo mensaje*”

Ejemplo:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git commit -m "primercoment"
[master (root-commit) 57f03a8] primercoment
 1 file changed, 2 insertions(+)
 create mode 100644 index.html

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git log --oneline
57f03a8 (HEAD -> master) primercoment

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git commit index.html --amend -m "primer archivo index"
[master f166feb] primer archivo index
 Date: Tue Aug 13 00:43:50 2024 -0500
 1 file changed, 2 insertions(+)
 create mode 100644 index.html

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git log --oneline
f166feb (HEAD -> master) primer archivo index
```

Erramos el comentario.

Revisamos el log para ver el error.

Nuevo mensaje.

Nombre del archivo con su extensión.

Nuevamente un log para ver la corrección.

git config

git config

Al instalar Git se debe establecer el nombre de usuario y dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información, y es introducida de manera inmutable en los commits que se envían:

(solo se hace una única vez)

$ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

Para ver esta configuración ingresar git config --global –l

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git config --global -l
gui.recentrepo=C:/Users/Family/Documents/Arduino/hardware/espressif/esp32
user.email=amtriana62@misena.edu.co
user.name=amtriana62
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
```

Cuando es doble guion -- la palabra es completa cuando es un guion – es un comando reducido.

git clean

git clean

El comando clean actúa en archivos sin seguimiento, este tipo de archivos son aquellos que se encuentran en el directorio de trabajo, pero que aún no se han añadido al índice de seguimiento de repositorio con el comando add. Por ejemplo:

```
Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sin_seguimiento.txt

nothing added to commit but untracked files present (use "git add" to track)

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git clean --dry-run
Would remove sin_seguimiento.txt

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git clean -f
Removing sin_seguimiento.txt

Family@LAPTOP-QCE12K6U MINGW64 ~/Desktop/nodia (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Revisa que archivos no tienen seguimiento.

Eliminar los archivos listados de no seguimiento

Actividad de la sesión

De forma individual*

Consulte los siguientes enlaces :

- Vadillo, J. (2020). Curso GIT - Parte 3: Fork de un repositorio [Youtube] Recuperado de: Clic aquí
- Alvarez, M. (2021). Qué es un fork y para qué sirve, cuáles son los pasos para crear un fork de un repositorio Git en GitHub. Recuperado de: Clic aquí
- GitHub Docs. Fork a repo. Recuperado de: Clic aquí

Y responda a las preguntas:

- ¿Qué es un fork?, ¿Cuál es su propósito?
- ¿Dónde se crea un fork?
- ¿Cuál es la diferencia entre git clone y git fork?

Posteriormente, participe en la socialización de estos conceptos con la ficha.

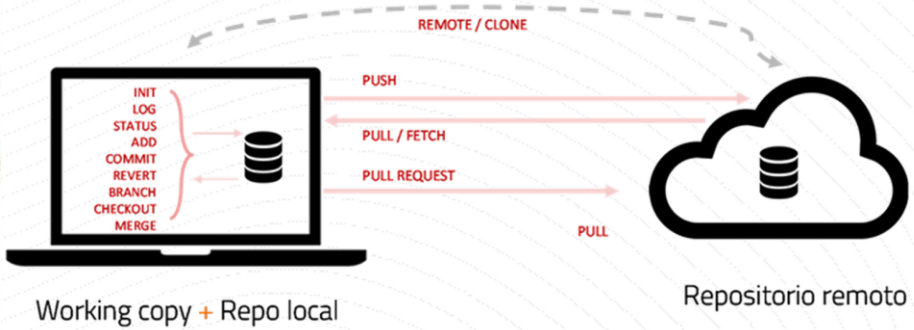
Finalmente

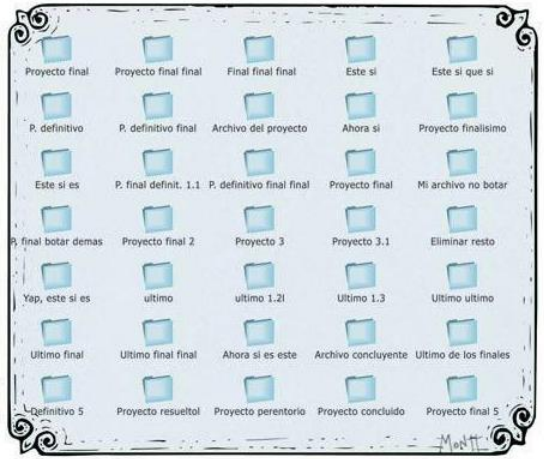

COMANDOS BÁSICOS DE git
GIT es un sistema de control de versiones que nos ayuda a llevar el historial completo de modificaciones de un proyecto.
Todo desarrollador sin importar el lenguaje debe dominar Git.
Prof. Beto Quiroga
GIT INIT
Inicia un nuevo repositorio.
GIT CLONE
Clona un repositorio existente.
GIT ADD
Añade un archivo a la zona de montaje. git add * añade uno o más archivos a la zona de montaje.
GIT CONFIG
Establece el nombre del autor, el correo y demás parámetros que Git utiliza por defecto.
GIT LOG
Se utiliza para listar el historial de versiones de la rama actual.
GIT STATUS
Enumera todos los archivos que deben ser confirmados.
GIT RESET
Descompone el archivo, pero conserva el contenido del mismo.
GIT DIFF
Muestra las diferencias de archivo que aún no se ponen en escena.
¿Qué comandos faltó? Haremos la 2da parte con tus sugerencias
ed.team/cursos/git
EDteam
COMANDOS BÁSICOS DE git (PARTE 2)
Si quieres trabajar en el mundo del desarrollo de software, es obligatorio que sepas GIT.
GIT HELP
Muestra información para saber el uso de cada comando de git.
GIT PULL
Descarga y actualiza los cambios realizados desde un repositorio remoto a tu repositorio local.
GIT CLEAN
Elimina archivos no deseados de un respositorio.
GIT PUSH
Sube los archivos a un repositorio remoto.
GIT BRANCH
Muestra la lista de ramas que existen en un repositorio.
GIT CHECKOUT
Sirve para moverse de una rama a otra y regresar en el tiempo.
GIT MERGE
Fusiona dos o más ramas.
GIT REMOTE
Crea, visualiza y elimina conexiones a otros repositorios.
Si eres programador es obligatorio aprender GIT. Domínalo en:
ed.team/cursos/git
EDteam

La decisión final...

git

Ctrl + Z
Ctrl + Z
Ctrl + Z
Ctrl + Z
Ctrl + Z
Ctrl + Z
Ctrl + Z





Referencias

Arias. E (2022). Git Fetch y git pull 🚀 Cómo funcionan | Diferencias | GitHub | Repositorio remoto | Git. Recuperado de: https://www.youtube.com/watch?v=A7cSX3ZBCws

Cantoral, C. (2017). ¿Que es una guía de código? Recuperado de: https://codigofacilito.com/articulos/guia_codigo

Bose, S. (2021). Coding Standards and Best Practices To Follow. Recuperado de https://www.browserstack.com/guide/coding-standards-best-practices#:~:text=Practices%20To%20Follow-,What%20are%20Coding%20Standards%3F,sophisticated%20and%20highly%20functional%20code.

Bluuweb. (2019) GIT / GITHUB [Tutorial en Español - Parte 1] ♥ Inicio Rápido para Principiantes ♥. Recuperado de: https://youtu.be/hWglK8nWh60

Bluuweb. (2019) GIT / GITHUB ♥ Ramas o Branch, Uniones o Merge ♥ [Tutorial en Español - Parte 3]. Recuperado de: https://youtu.be/tFr0Vg1q9Eg

EDteam (2017). ¿Cómo se deciden las versiones del software?. Recuperado de https://ed.team/blog/como-se-deciden-las-versiones-del-software

Gonzalo, N. (2021). 10 Comandos de Git Que Todo Desarrollador Debería Saber: Clic aquí https://www.freecodecamp.org/espanol/news/10-comandos-de-git-que-todo-desarrollador-deberia-saber/

Kumar, S. (2019). Software coding and testing. Recuperado de https://www.slideshare.net/SandeepKumarNayak1/software-coding-and-testing

Referencias

La rosa, A. (2021). Versionado de Software desde el punto de vista del usuario. Recuperado de https://pandorafms.com/blog/es/versionado-de-software/

Santos, P. (2010) La historia de control de versiones. Recuperado de: http://codicesoftware-es.blogspot.com/2010/11/la-historia-del-control-de-versiones.html

TodoCode. (2021) GIT CLONE + GIT PULL | CONFIGURACIÓN FÁCIL 2021 😁 | Introducción a GIT y GITHUB #5 Recuperado de: https://youtu.be/IWnW0svZ9JQ

G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co