Ejercicios Git solucionados

# Estás trabajando en un proyecto y has realizado un commit (C2) con un mensaje que contiene un pequeño error tipográfico en el mensaje, y además te faltó incluir un archivo antes de hacer el commit.

git init

touch archivo1

touch archivo2

git add -A

git commit -m "C1. Commit inicial"

echo "Hola mundo desde archivo1" >> archivo1

echo "Hola mundo desde archivo2" >> archivo2

git add archivo1

git commit -m "C2. Hinicializa archivo1"

* Corrige el mensaje del commit C2 para arreglar el error tipográfico y asegúrate de que *archivo2* también sea incluido en el commit C2.
* Verifica que los cambios se han aplicado sin crear un nuevo commit en el historial.

Solución:

git add archivo2

git commit --amend -m "C2. Inicializa archivo1 y archivo2"

# Estás trabajando en un proyecto que genera archivos temporales y archivos de configuración locales que no deberían ser incluidos en el control de versiones. Configurar el archivo .gitignore para que Git ignore estos archivos:

* Archivos con extensión “.log”.
* Archivos temporales que comienzan con tmp\_”.
* Cualquier archivo dentro de la carpeta “config/” excepto un archivo llamado “config/settings.conf”.

git init

touch archivo1.log

touch archivo2.log

touch tmp\_archivo.txt

mkdir config

touch config/settings.conf

touch config/local.conf

Solución:

El archivo .gitignore debería ser:

# Ignorar todos los archivos .log

\*.log

# Ignorar todos los archivos que comiencen con 'tmp\_'

tmp\_\*

# Ignorar todos los archivos dentro de la carpeta 'config' excepto 'config/settings.conf'

config/\*

!config/settings.conf

# Estás trabajando en un proyecto con varios commits y necesitas revisar el historial utilizando diferentes criterios de búsqueda. Para esto, vas a usar un comando que te permita navegar por el historial de commits de manera eficiente y personalizada.

* Muestra solo los últimos 3 commits de tu proyecto en formato simplificado con una línea por commit.
* Filtra los commits que se hicieron después del 1 de septiembre de 2024.
* Busca commits realizados por “Juan Pérez”.
* Muestra los commits que contienen la palabra clave “bug” en el mensaje del commit.

Solución:

git log -n 3 --oneline

git log --since="2024-09-01"

git log --author="Juan Pérez"

git log --grep="bug"

# Has estado trabajando en un proyecto y accidentalmente eliminaste un archivo que necesitabas. Afortunadamente, el archivo estaba presente en el penúltimo commit, y ahora necesitas recuperarlo. ¿Cómo recuperas *archivo1* en esta situación?

git init

touch archivo.txt

echo "Contenido del archivo" > archivo.txt

git add -A

git commit -m "C1. Añade archivo.txt con contenido"

git rm archivo.txt

git commit -m "C2. Elimina archivo.txt "

Solución:

Primero debemos identificar el último commit donde estaba presente *archivo1* con:

$ git log --oneline

ae644ec (**HEAD** -> **main**) C2. Elimina archivo1

969fa27 C1. Añade archivo1 y archivo2

Y usamos el hash del commit C1 para recuperar el archivo con git checkout:

$ git checkout 969fa27 -- archivo1

# Has estado trabajando en un proyecto y has realizado varios commits. Sin embargo, te das cuenta de que los últimos dos commits contienen cambios que no deberías haber realizado. Decides que quieres deshacer esos cambios por completo y volver a un estado anterior del proyecto, eliminando todos los commits y cambios posteriores de forma permanente.

git init

echo "Contenido inicial" > archivo.txt

git add archivo.txt

git commit -m "C1. Añade archivo.txt con contenido inicial"

echo "Segunda línea de contenido" >> archivo.txt

git add archivo.txt

git commit -m "C2. Añade segunda línea a archivo.txt"

echo "Tercera línea de contenido" >> archivo.txt

git add archivo.txt

git commit -m "C3. Añade tercera línea a archivo.txt"

echo "Cuarta línea de contenido" >> archivo.txt

git add archivo.txt

git commit -m "C4. Añade cuarta línea a archivo.txt"

Solución:

Primero debemos identificar el hash del commit al que queremos ir:

$ git log --oneline

f8b143c (**HEAD** -> **main**) C4. Añade cuarta línea a archivo.txt

2c2cd0b C3. Añade tercera línea a archivo.txt

dd7a148 C2. Añade segunda línea a archivo.txt

61415ef C1. Añade archivo.txt con contenido inicial

Ahora usamos el hash del commit C2 para recuperar el archivo con git checkout:

$ git reset --hard dd7a148

# Estás trabajando en un proyecto y decides crear una nueva rama para implementar una nueva funcionalidad. Después de realizar un commit en esa rama, intentas fusionarla con la rama principal (*main* o *master*), pero surge un conflicto debido a modificaciones en el mismo archivo en ambas ramas. Resuelve el conflicto y completa la fusión correctamente.

* Realiza un commit inicial en la rama principal (*main* o *master*).

git init

echo "Inicialización de archivo1" > archivo1

git add -A

git commit -m "C1. Commit inicial en main"

* Crea una nueva rama llamada *nueva-rama*, realiza un cambio en el archivo y haz un commit en esa nueva rama.

echo "Línea añadida en la nueva rama" >> archivo1

* Cambia de nuevo a la rama principal (*main* o *master*), realiza un cambio en el mismo archivo y haz un commit.

echo "Línea añadida en la rama principal" >> archivo1

* Intenta fusionar la nueva rama con *main* y resuelve el conflicto que ocurra.
* Completa la fusión y verifica que los cambios se hayan aplicado correctamente, de manera que *archivo1* quede así:

Inicialización de archivo1

Línea añadida en la nueva rama

Línea añadida en la rama principal

* Elimina la rama que has creado.
* Comprueba que el log queda de la siguiente manera:

$ git log --oneline --graph --all

\* e732a62 (**HEAD** -> **main**) Merge branch 'nueva-rama'

|\

| \* 73173a0 C2. Commit en nueva-rama

\* | 6ab9e22 C3. Commit en main

|/

\* 991b9df C1. Commit inicial en main

Solución:

git init

echo "Inicialización de archivo1" > archivo1

git add -A

git commit -m "C1. Commit inicial en main"

git switch -c nueva-rama

echo "Línea añadida en la nueva rama" >> archivo1

git add -A

git commit -m "C2. Commit en nueva-rama"

git switch main

echo "Línea añadida en la rama principal" >> archivo1

git add -A

git commit -m "C3. Commit en main"

git merge nueva-rama

Resolvemos el conflicto y hacemos commit.

Finalmente eliminamos la rama creada:

$ git branch -d nueva-rama

Deleted branch nueva-rama (was 73173a0).