TEMA 04 – GIT AVANZADO

Tags

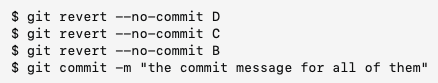
* Introducción
  + Introducción
    - Git permite identificar commits como importantes (tags)
    - Normal-: releases
  + Crear etiquetas ligeras
    - Referencia alfanumérica que se especifica en un commit
    - ‘git tag <nombreEtiqueta>’
    - Si no se especifica nada más, la crea donde señala el HEAD
    - Para commit concreto: ‘git tag <nombreEtiqueta> <hash>
    - Si intentamos crear una etiqueta que ya existe, da error
      * -f, --force: sobreescribe
  + Crear etiquetas anotadas
    - Además, contiene creador, fecha i mensaje
    - ‘git tag -a <nombreEtiqueta> -m “Mensaje”’
    - Si no se pone el “-m”, abrirá el editor de texto por defecto
  + Mostrar etiquetas
    - ‘git tag’, ‘git tag -l’, ‘git tag –list’
    - Muestra en orden alfabético
    - ‘git tag -l “patrón”: permite aplicar patrones de búsqueda
  + Consultar etiquetas
    - Para consultar la información de las etiquetas i del commit al que hacen referencia
    - ‘git show’
    - Etiquetas ligeras: solo mostrará la información del commit
    - Etiquetas anotadas: información del commit, persona que la crea i mensaje
    - ‘git tag <nombreEtiqueta>’: para ver una etiqueta en concreto
  + Borrar etiquetas
    - ‘git tag -d <nombreEtiqueta>’
  + Etiquetas en el remoto
    - Por defecto, el ‘push’ NO publica las etiqueta
    - ‘git push --tags’
    - Eliminar etiqueta del repositorio remoto ‘origin’:
      * ‘git push origin –delete <nombreEtiqueta>’
* Etiquetas en GitHub
  + Permite gestionar etiquetas del repositorio remoto de manera muy simple
  + Se crean mediante Releases (sistema específico de GitHun no vinculado con Git en sí)
  + Se pueden consultar las releases en:
    - https://github.com/<owner>/<repository>/releases
  + Las etiquetas se pueden consultar en:
    - https://github.com/<owner>/<repository>/tags
  + Crear una etiqueta
    - Hace falta hacer un nuevo Release
    - Clicar en “Draft a nre release”
    - Al crearlo, indicar qué etiqueta se quiere vincular
      * Etiqueta existente
      * Escribir el nombre de una nueva etiqueta
  + Visualizar el estado de una etiqueta
    - Desde GitHub se puede hacer ‘checkout’ de rama, commit o etiqueta
    - Ramas o etiquetas:
      * https://github.com/<owner>/<repository>/tree/<branch or tag>
    - Commits:
      * https://github.com/<owner>/<repository>/commit/<commit hash>
  + Actualizar y borrar etiquetas
    - No permite modificar una etiqueta ya creada desde la web
    - Habría que borrarla y volverla a crear
    - Borrar: primero se tienen que eliminar las releases vinculadas desde Releases
    - Una vez eliminadas, se puede eliminar la etiqueta desde Releases > Tags
    - Hay que crear en la etiqueta para poder eliminarla

Stash

* Introducción
  + Almacén que permite guardar temporalmente los cambios que todavía no se quieren “commitear”
  + Útil si:
    - Hemos hecho cambios en el directorio de trabajo y necesitamos cambiar a otra rama para trabajar en otra cosa
    - Para incorporar cambios que entren en conflicto con el que estábamos trabajando
* Mostrar el stash
  + Permite guardar múltiples ‘stash’ [0..N]
  + Stash 0 es el más reciente
  + ‘git stash list’: muestra los stash
* Guardar los cambios
  + ‘git stash’
  + Incluirá los cambios en la primera posición: stash@{0} e incrementarà el índice de los stash existentes en 1
* Mostrar los cambios
  + ‘git stash show’
  + Si, adicional-, queremos mostrar los cambios podemos indicar la opción ‘-p’
  + También se puede indicar el índice que se quiere consultar (por defecto el 0)
  + ‘git stash show [-p] [index]’
* Recuperar los cambios
  + ‘git stash apply’
  + Volverá a aplicar los cambios guardados en el directorio de Trabajo
  + También se puede indicar el índice que se quiere aplicar (por defecto el 0)
    - ‘git stash apply [index]
  + ‘git stash pop [index]: los cambios se aplica y se borran del stash
* Descartar los cambios
  + ‘git stash drop [index]

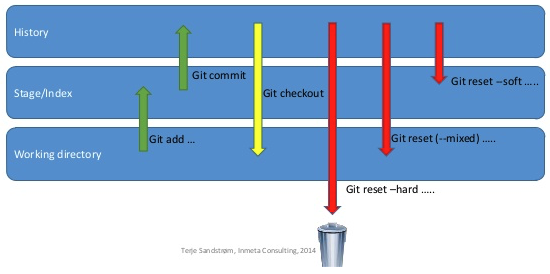
Revert

* Sirve para revertir los cambios realizados en un commit anterior
* Crea un nuevo commit que deshace los cambios hechos, haciendo que el código vuelva al estado en el que se encontraba
* Es una buena forma de revertir los cambios sin alterar la historia del repositorio
* Útil sobre todo cuando se trabaja con repositorios compartidos con otros desarrolladores
* 'git revert <commit>’
* Para revertir múltiples commits:

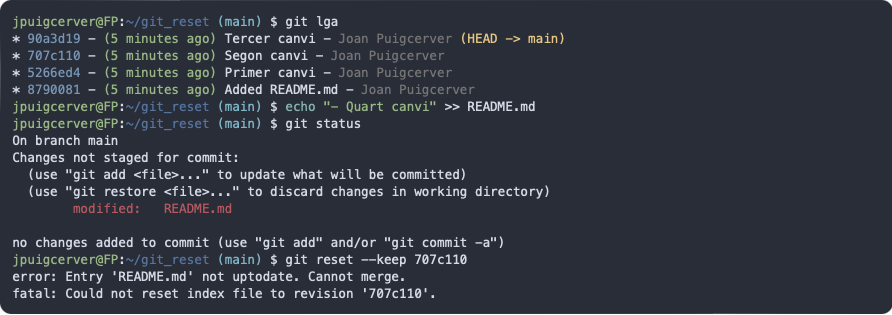


Reset & Amend

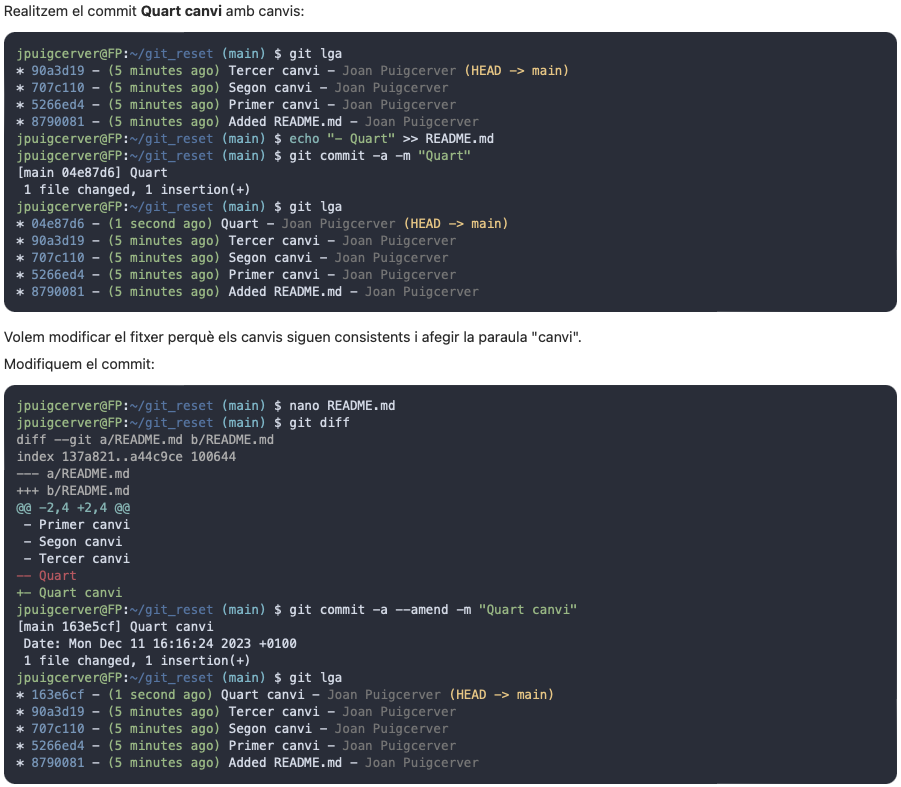
* Introducción
  + Permiten modificar la historia del repositorio local para ajustar o rehacer la historia a nuestras necesidades
  + Algunas de las acciones que se pueden hacer:
    - Deshacer o modificar commits anteriores
    - Reorganizar los commits antes de publicarlos
    - Mover punteros de las ramas
  + Modificar la historia, sobre todo si se ha hecho ‘push’ se tiene que hacer con precaución, ya que puede ocasionar problemas con los colaboradores del repo
* Reset
  + Modifica el estado actual del repositorio a un commit en concreto
  + Mueve la referencia de la rama actual, donde está el HEAD
  + El comportamiento depende de las opciones especificadas:



* + Soft
    - ‘git reset --soft <ref>’
    - Modifica el estado del repo a la referencia especificada
    - Conserva los cambios realizados en el “working directory” y el “staging área”
  + Mixed
    - ‘git reset [--mixed] <ref>’
    - Comporta- por defecto
    - Modifica el estado del repo a la referencia especificada
    - Conserva los cambios realizados en el “Working Directory” solamente
  + Hard
    - ‘git reset --hard <ref>’
    - Modifica el estado del repo a la referencia especificada
    - Los cambios no se conservan
  + Keep
    - ‘git reset --keep <ref>’
    - Muy similar al comporta- por defecto
    - No permite realizar el ‘reset’ si comporta sobreescribir los cambios actuales del “working directory” que no han sido confirmados (commit)

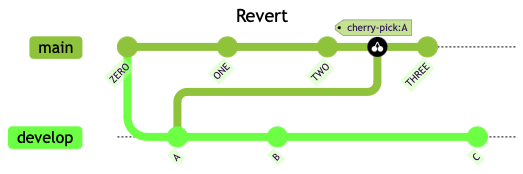


* Amend
  + ‘git commit --amend’
  + Permite realizar cambios en el último commit realizado
  + Permite modificar el mensaje del último commid, añadir nuevos ficheros o añadir nuevos cambios a los ficheros del repositorio, incluido los que han sido modificados en este último commit
  + Si no se especifica ningún mensaje, se abrirá el editor
  + Borra el commit y crea uno nuevo realmente (el hash es diferente)

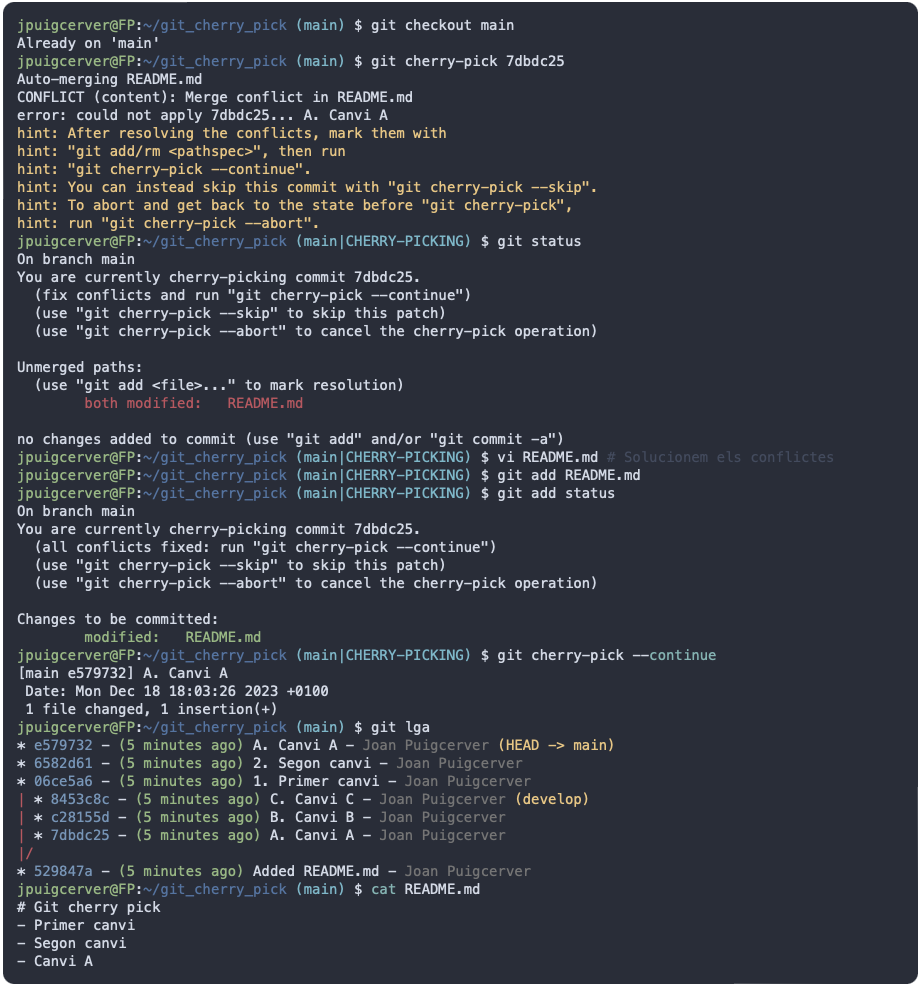


Cherry Pick

* ‘git cherry-pick <hash>’
* Introducción
  + Copia un commit de la historia del repositorio a nuestro HEAD
  + Puede servir para incluir modificaciones realizadas en un commit de una rama a otra

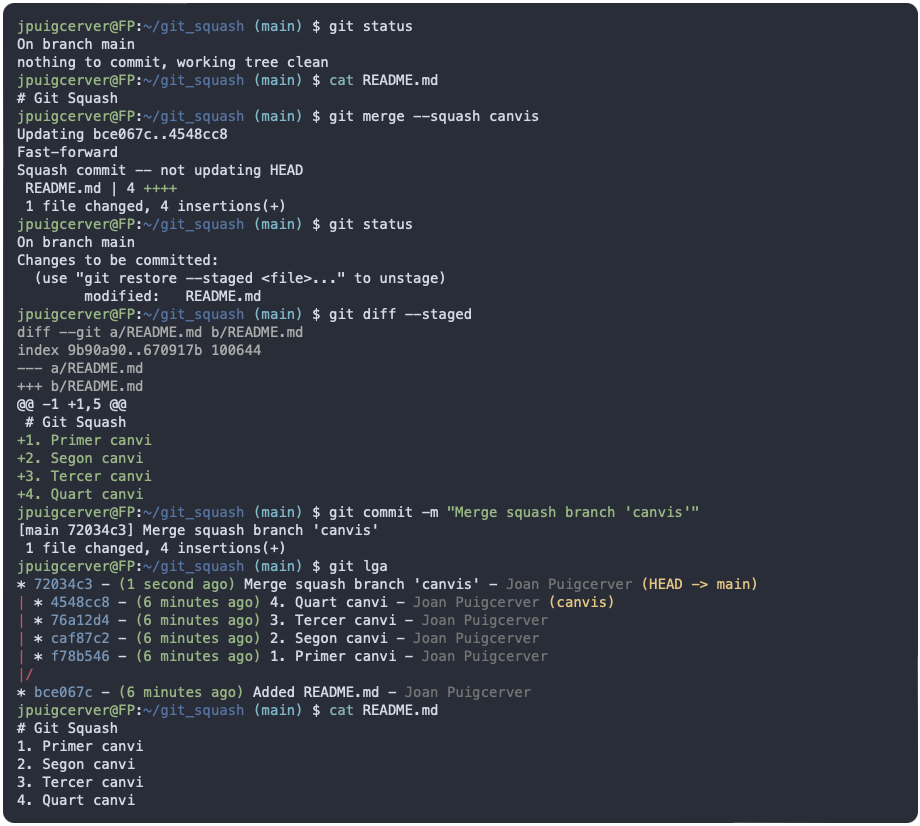


* Ejemplo



Squash

* Introducción
  + ‘git merge –squash <branch\_name>’
  + Permite combinar todos los cambios realizados en una rama (en múltiples commits) en un solo commit
  + Ventaja: simplificar la historia del repositorio
  + Desventaja: se pierde la información de qué se ha modificado en cada commit
  + Útil para fusionar una rama a la cual pensamos que se han realizado demasiados commits con cambios no significativos en todos ellos
* Ejemplo
  + Se han hecho 4 cambios en la rama canvis y queremos incoporporarlos a main
  + Se hace desde main, que es adonde queremos hacer ese commit



TEMA 05 – PROVA DE PROGRAMARI

Test unitaris amb JUnit

* <https://joapuiib.github.io/itb/DAM-ED/UD5/materials/02_junit.html>

F