< Curso Profesional de Git y GitHub

TODOS los comandos del curso v2.0

70

FixingMind5 22569

<h1>Comandos de git</h1>
Este tutorial era parte del curso pasado así que decidí simplemente actualizar los comandos que enseña en Freddy en este curso, espero les pueda ser útil. 😉

⚙ **CONFIGURACIONES**

git config --global user.email me@email.com Para definir un correo

git config --global user.name **nombreUsuario** Para definir un nombre de usuario

git config --global color.ui true Para hacer git más dinámico xD

git config --global core.editor "atom --wait" para colocar algún editor, en este caso Atom

git config --list para ver la configuración por defecto de tu git

git config --list --show-origin Ver dónde están las configuraciones guardadas

— Atom > Install Shell Commands para solucionar los errores de arranque en MacOS

🛠 **FLUJO DE TRABAJO**

git init nombreDelRepo Para iniciar un nuevo repositorio

git init Si nos encontramos en el directorio que queremos como repo

rm -rf .git de algún repo para que éste deje de serlo

git add nombreArchivo para agregar un *Untracked file* del working directory al staging area

git add -A para agregar todos los archivos que tenemos

git add . funciona como el anterior

git rm --cached (Sin el espacio) para remover un archivo del staging area

git status para saber qué y dónde se encuentra

git commit -m 'message' Para comprometer lo que modificaste -m para agregar un mensaje

git commit --amend Por si te equivocaste y quieres agregar un cambio, no crea un nuevo commit, se fusiona con el commit anterior, tienes que hacer un add antes

git commit --am para hacer **git add** y **git commit** al mismo tiempo. Sólo funciona con archivos a los que ya les hiciste add antes, a los completamente nuevos no les hará nada.

🏷 **TAGS**

git tag -a 'version' agregará una etiqueta al último SHA

— hacer **git tag -a 'versión' SHA1234567890abcdf* para especificar a que SHA tageas

git tag -l Para ver las tags de los SHA

git tag -d Para borrar una etiqueta

git tag -f -a 'version' -m 'messge' SHA1234567890abcdf para renombrar un tag. OJO: después hay que eliminarlo porque si no se queda así

git push origin --tags mandas los tags al repositorio remoto

git push origin :refs/tag/nombre-del-tag Para remover el tag del repositorio remoto

git push origin :refs/tag/nombre-del-tag Para remover el tag del repositorio remoto

⌨ **LOGS**

git log Rastro de nuestro proyecto

git log --oneline muestra los SHA en una sola línea

git log --graph muestra las ramas

En el curso anterior nuestro compañero bustosfredy dejó este config para tener un log más cómodo:

**git config --global alias.superlog "log --graph --abbrev-commit --decorate --date=relative --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all" **

Sólo queda que ustedes hagan en lugar de **git log**, **git superlog** y encontrarán belleza

COMPARACIONES

git show <archivo> Para ver los cambios del archivo

git diff SHA1 Para comparar el estado final con el commit

git diff SHA2 SHA1 Para comparar entre dos commits (mostrará que se borraron cosas)

git diff SHA1 SHA2 mostrará que se agregaron cosas, sigues comparando commits

git checkout SHA1 <archivo> te lleva a ese momento de la historia. No hagas commit si no quieres que ese cambio se quede para siempre, de lo contrario, hazlo 😉

git checkout master <archivo> vuelves a la última versión

CTRL + Z

git reset --soft SHA1 desde qué commit quiero quitar los cambios, ese commit se va quedar tal cual y se eliminarán los cambios posteriores

git reset --mixed SHA1 hace lo mismo sólo que el commit que pusiste se deja en el Working Directory

git reset --hard borra absolutamente todo. Actúa desde donde está

—Archivos recién agregados en Working Directory no son afectados, están fuera de la jurisdicción de git

git reflog ... Créeme, lo vas a necesitar. 👇

⚠ WARNING.

Si eliminaste un SHA por accidente con **git reset --hard** puedes traerlo de vuelta con **git reflog** aparecerá el historial de git, después haces **git reset --hard SHA** y lo traes de vuelta a la vida

RAMAS

git branch nombreRama para crear nueva rama

git branch -l para listar ramas

git branch -d nombreRama para eliminar cierta rama, "D" en lugar de "d" para forzar la eliminación

git branch -m nombrePrevio nombreNuevo para renombrar ramas

git checkout nombreRama para movernos entre ramas

git checkout SHA para movernos a una instancia de nuestro proyecto y sacar de ahí una nueva rama

git checkout -b nombreRama creas una nueva rama y te mueves a ella

git show-branch --all Muestra las ramas que hay y una breve historia de como se han comportado

gitk Muestra como se han comportado las ramas d una manera muy visual

MEZCLA DE RAMAS

git merge ramaAMezclar Mezcla la rama que escribiste con la rama en la que estás

Si hay un conflicto de mezcla, puede ser que tengas que meter mano al editor

git rebase nombreRama para tener un seguimiento lineal sobrescribiendo la historia de nuestro código

git rebase -i nombreRama Nos abrirá el editor para acceder a otras modalidades

GUARDAR ESTADO

git stash guarda un estado en el que te encontrabas para poderte mover a otra rama sin hacer commit, el cambio tiene que estar en staging area

git stash list muestra las referencias a los diferentes stash que ya tenías
git stash drop stash@{x} para eliminar un stash
git stash apply Vamos a aplicar nuestros cambios
git stash apply aplica el ultimo estado guardado
git stash apply stash@{x} agrega un stash al que tú ya hiciste referencia
git stash pop trae el último cambio que guardaste sin agregarlo a staging
git stash branch nombreDeLaNuevaRama crea una rama y las últimas modificaciones que fueron **"stasheadas"** en otra rama las trae a la nueva sin agregarla a staging

CEREZAS

git cherry-pick SHA sobre una rama para traer un commit de algún lugar y dejarlo en la rama actual, donde se supone debería ir

CLONANDO

git clone dirHtml para clonar tu repositorio
ssh-keygen -t rsa -b 4096 -C "email@email.com" Para crear una llave publica
pbcopy < ~/.ssh/id_rsa.pub para copiar al clipboard
git remote add origin direcciónDeTuRepo para crear un repositorio remoto de nombre Origin por convención
git remote -v para revisar si tenemos el repositorio remoto
git remote remove origin para remover repositorio remoto origin, por convención
git fetch origin traer archivos desde el directorio remoto, origin por convención
git pull origin master para traer del repositorio remoto lo que hay en esas ramas
git push origin master a dónde vamos a enviar nuestros archivos
git push origin master --tags enviar los tags de nuestros proyectos
git push Origin algunaRama para enviar otra rama a Internet, a nuestro repositorio remoto

LIMPIAR

git clean --dry-run Para visualizar qué archivos se van a borrar. Archivos, no carpetas
git clean -f Para borrar los archivos que visualizaste: Ignorará lo que esté dentro del archivo .gitignore
git clean -df Borra archivos y carpetas

BUSCAR PALABRAS

git grep <palabra> busca la palabra en todos tus archivos dependiendo de la rama
git grep -n <palabra> Te dice dónde está y la línea
git grepp -c <palabra> Te dice cuántas veces usaste esa palabra por archivo
Si en algún momento estás haciendo cosas raras y la palabra no la... ¿acepta?, ponle comillas. Ejemplo: **<p>**, no va a funcionar **"<p>"** sí

Otros comandos útiles

git config --global alias.stats "shortlog -sn --all --no-merges" después sólo haz **git stats** . Te mostrará cuántos commits han hecho cada miembro del equipo, de una manera más abreviada y sin contar los merges
git blame Para ver quién tuvo la culpa de qué.
git blame rutaArchivo -L11,22 Te dirá quién modificó qué del archivo que pusiste. La -L es para indicar de qué línea a qué línea quieres ver el comando.
git branch -r ves ramas remotas.

git branch -a ves todas las ramas.
La ip de GitHub es 192.168.252.153 - 192.168.252.154

15 hace 3 años Apple

Escribe tu comentario +2

Ordenar por: Top

4 **sam-rojas** 2097 Puntos 3 años
que genial este aporte!

0 **FixingMind5** 3 años
Gracias 😁

3 **mael358** 5751 Puntos 2 años
Muy buen aporte! Gracias! 😁

0 **FixingMind5** 2 años
A ti! 😃

3 **AndresDel** 3778 Puntos 3 años
Excelente. muchas gracias,

0 **FixingMind5** 3 años
Gracias 😃

3 **Abrilhdezv** 2967 Puntos 2 años
Excelente aporte. Gracias.

0 **FixingMind5** 2 años
A ti 😃

3 **JuanGaybre** 14201 Puntos 2 años
Genial hermano!!
Gracias.

0 **FixingMind5**
Gracias! 🙏

2 **joservilleda** 3086 Puntos — un año

Muchas gracias

2 **josephmonsalve** 3620 Puntos — 2 años

Que buena esa prro!

0 **FixingMind5** — 2 años

Gracias!! 😁

2 **sebitas-dlq** 11456 Puntos — un año

Muchas gracias, por tomarte el tiempo de hacer esto. Hiciste una biblia para todos nosotros ;')

0 **FixingMind5** — un año

Jajaja, gracias a ti por tomarte el tiempo de leerla.

2 **Jmorillg** 773 Puntos — un año

Gracias por tu esfuerzo y permitir que estudiantes como yo que andamos un poco mal de tiempo podamos disponer de unos apuntes así de rápido!

0 **FixingMind5** — un año

Gracias a ti!

2 **rafa-romero** 3973 Puntos — un año

😃 buen aporte...

2 **blink55f** 3660 Puntos — un año

ufff buenazo, gracias!

2 **Ivan12** 11569 Puntos — un año

Muchas gracias

0 **julian-loaiza-lopez** 1075 Puntos — un año

Muy bueno el aporte

0 **epadilla** 59601 Puntos

Especial para tenerlo en cuenta, gracias.

0

 charly2045 5667 Puntos

un año

Excelente aporte, gracias

Entradas relacionadas

1

Git Fetch y merge vs git Pull, ¿cual hacer?

¿Qué es Git Fetch? Git Fetch crea una nueva rama en nuestro repositorio, ahí tenemos una copia exacta de todo lo que sucede en el repositori

 maucoder1

2

¿Qué es Git, Github y Por qué usarlos?

 iris-val

2

Buenas prácticas

Ramas Cuando un desarrollador quiere corregir un error o implementar una característica, crea una nueva rama por fuera de la rama principa

 rieramoss