Comandos Esenciales en Git Cheat Sheet

by Astray via cheatography.com/173464/cs/36443/

Configurar un Repositorio

git init

Crea un nuevo repositorio en la ruta donde se ejecuta el comando

git clone [url]

clona un repositorio al directorio actual

git clone [url] [nuevo directorio]

clona un repositorio a un nuevo directorio

Collaboración y Sincronización con GitHub

Estos comandos se utilizan para trabajar colaborativamente con el repositorio remoto

Configurar Acceso al Repositorio en GitHub

Accede a tu repositorio en GitHub y haz clic en la pestaña "Settings" (Configuración)

En la página de configuración, haz clic en la pestaña "Collaborators" (Colaboradores)

En el campo "Search by username" (Buscar por nombre de usuario), escribe el nombre de usuario de la persona que quieres añadir como colaborador

Selecciona el nombre de usuario de la lista de resultados y haz clic en "Add collaborator" (Añadir colaborador)

La persona que has añadido recibirá una solicitud de colaboración en su correo electrónico de GitHub. Cuando la acepte, podrá empezar a colaborar en el repositorio

Para añadir colaboradores a un repositorio en GitHub, primero debes tener permisos de propietario del repositorio o ser un usuario con permisos de escritura.

Tipos de permisos de acceso para colaboradores

Acceso de lectura: Los colaboradores con este tipo de acceso pueden ver y descargar el código del repositorio, pero no pueden realizar cambios ni enviar solicitudes de pull.

Tipos de permisos de acceso para colaboradores (cont)

Acceso de escritura: Los colaboradores con este tipo de acceso pueden realizar cambios y enviar solicitudes de pull, pero no pueden eliminar el repositorio ni cambiar la configuración del repositorio.

Acceso de administrador: Los colaboradores con este tipo de acceso tienen acceso completo al repositorio y pueden realizar cambios, enviar solicitudes de pull, eliminar el repositorio y cambiar la configuración del repositorio.

Puedes cambiar el tipo de acceso de un colaborador en cualquier momento en la página de configuración del repositorio, en la sección "Collaborators" (Colaboradores).

Git push

git push [remote repo] [branch name]

Pública los commits del repositorio local al remoto. Se puede omitir el repositorio si se desea seguir usando el mismo repositorio
Ejemplo: git push origin master

git push [remote repo] --all

Pública los commits de todas las ramas al repositorio remoto.

git push [remote repo] --tags

Pública tags locales al repositorio remoto

- Las Tags no se sincronizan automaticamente con el repositorio remoto con otros comandos push.
https://www.atlassian.com/git/tutorials/syncing#git-push

Git pull

git pull [remote repo]

Trae los cambios desde repositorio remoto. Fast forward merge. Se puede omitir el repo si ya se ha configurado.

git pull [remote repo]/[branch name]

Trae los cambios de una rama y los combina al repositorio local.

- Git pull command = git fetch + git merge
- Más información y ejemplos:
https://www.atlassian.com/git/tutorials/syncing#git-pull

Git fetch

git fetch [remote repo name]

Muestra los cambios de todas las ramas del repositorio remoto

git fetch [remote repo name] [branch]

Muestra los commits en una rama del repositorio.

git fetch --dry-run

Ver los cambios de un repositorio remoto antes de hacer pull

- Se utiliza para ver en que han trabajado los demás colaboradores
- El contenido *fetched* se representa como una rama remota y no afecta al repositorio local.
https://www.atlassian.com/git/tutorials/syncing#git-fetch

Git remote

git remote

Verifica si se tiene un repositorio remoto. Si se hace en un repositorio clonado, retorna el repositorio original

git remote -v

Muestra el path completo del repositorio remoto

git remote add origin [github url]

Añade un repositorio remoto

git remote [url] [branch name]

Apunta rama remota a la url correcta

git remote rm [remote repo name]

Remover conección al repositorio especificado

Ramas y Repositorios

git switch [remote branch]

Trae los cambios de una rama remota al repositorio local y crea la rama localmente.

git push --set-upstream [repository] [new branch name]

Este comando debe hacerse desde la rama en cuestión. Git crea en el repositorio remoto la nueva rama y la asocia a la rama actual.
Ejemplo: git push –set-upstream origin new-branch

Viajar en el tiempo con Git

Estos comandos se utilizan para deshacer cambios, revertir estados y recuperar el código a cualquier punto en el historial de Git.

Git restore

git restore [file name]

Recupera el estado del archivo en cuestión segun el HEAD

git restore [commit id] [file name]

Recupera el estado del archivo en cuestión segun un commit concreto

git restore --staged [file name]

Saca un archivo del área de staged a unstaged

Este es un nuevo comando que ayuda a deshacer operaciones. Ya que es nuevo la mayoría de los tutoriales no lo mencionan, sus funciones pueden ser reemplazadas con git checkout.

Git checkout

git checkout [commit id]

Devuelve el proyecto a un commit concreto en estado de detached HEAD

git checkout HEAD~[número]

Devuelve el proyecto moviendose [número] posiciones hacia átras desde HEAD. HEAD~1 se refiere al commit anterior a HEAD. HEAD~2 se refiere a 2 commits antes.

Este comando es como un comodín, o la navaja suiza de git.
Se suele decir que es overloaded lo que llevó a la adición de los comandos git switch y git restore.
Se puede utilizar para crear ramas, cambiar entre ramas, recuperar archivos y viajar en el historial, pero se recomienda utilizarlo solo para este último uso.

Comandos Exploratorios de Git

git status

Lista los archivos que han sido staged, unstaged, commited, y tracked

git show

Muestra los cambios realizados en el último commit

git show [commit id]

Muestra los cambios realizados en un commit concreto

Comandos Exploratorios de Git (cont)

git shortlog

lista alfabetica de los nombres y mensajes de los commits realizados por cada persona del repositorio

git shortlog -s -n

Muestra el número de commits hecho por cada persona

git log

Muestra el historial completo de todos los commits realizados

git log -n [numero]

Muestra solo la cantidad especificada de commits del historial

git branch

Lista las ramas en el repositorio

git branch -a

Lista las ramas en el repositorio, incluyendo ramas remotas

git tag

Muestra las tags existentes en el repositorio

Guardar Cambios

Este conjunto de comandos se utilizan para añadir cambios al historial de Git

Git add

git add [file name]

Añadir archivos al área de staging

git add .

Añadir todos los archivos al área de staging. Esto incluye aquellos que se encuentran como *untracked*

git add '*[file type]'

Añade todos los archivos de un tipo al área de staging. *Ejemplo: git add .txt*

git add [directory]

Añade todos los cambios de un directorio al área de staging

Más información de este comando:
https://www.atlassian.com/git/tutorials/saving-changes#git-add

Git commit

git commit

Abré un procesador de texto para añadir un mensaje de commit en la línea superior. Añade los cambios del área de staging a un commit.

git commit -m ["commit message"]

Crea un commit con los cambios del área de staging y el mensaje especificado.

git commit -a -m ["commit message"]

Crea un commit con todos los cambios de los archivos en estado *tracked*. No agrega al commit *untracked* files.

Guía para escribir mensajes de commit:
http://udacity.github.io/git-styleguide/

Git diff

git diff

Muestra los cambios en el directorio local (not staged)

git diff --staged

Muestra los cambios respecto al área de staging

git diff HEAD

Muestra los cambios entre archivos staged y unstaged

Git stash

git stash

Añade todos los cambios actuales en staging en un área especial denominada stash, donde pueden ser recuperados más tarde.

git stash pop

Obtiene los cambios del stash y los lleva a la rama actual

El área de stashing es muy útil para guardar cambios temporalmente para cambiar de rama o traer cambios remotos sin realizar commits innecesarios.
Más sobre stash:
https://git-scm.com/docs/git-stash

Comandos Esenciales en Git Cheat Sheet

by Astray via cheatography.com/173464/cs/36443/

Git Branches

git branch

Lista las ramás de un repositorio local

git branch -a

Lista todas las ramás de un repositorio, incluyendo las del repositorio remoto

git branch -r

Lista solo las ramas remotas de un repositorio

git branch [new branch name]

Crea una nueva rama con el nombre indicado

git branch [new branch name] [commit id]

Crea una nueva rama desde el commit especificado

git branch -d [branch name]

Elimina una rama. No se puede eliminar una rama si esta en ella o no se ha hecho merge de los cambios.

git branch -D [branch name]

Elimina una rama por la fuerza (no requiere merge), pero no se puede hacer si se encuentra en la rama.

git switch [branch name]

Sirve para cambiar a la branch en cuestión

git switch -c [branch name]

Crea una nueva rama y cambia a esta el puntero HEAD.

git branch -m [new name]

Renombrar una rama.

git merge [branch name]

Trae el contenido de la rama especificada a la rama actual. No elimina la rama que estamos combinando, esta sigue existiendo. Si ya no se necesita debe ser eliminada manualmente.

Deshacer Commits

git reset [commit id]

Deshace el commit en cuestión y los cambios se dejan en el directorio de trabajo como unstaged

gir reset --hard [commit id]

Deshace el commit y revierte los archivos al estado anterior. Los cambios contenidos en el commit se pierden para siempre. Usar con precaución

git revert [commit id]

Deshace los cambios del commit en cuestión. A diferencia de reset no elimina el commit, sino que crea un nuevo commit revertiendo los cambios.

Si desean deshacer cambios que otros colaboradores ya tienen en sus máquinas, se recomienda usar revert.
Si aún no han hecho push de los cambios, usar reset y nadie nunca lo sabrá.

Opciones en estado de Detached HEAD

Descartar cambios

Para descartar los cambios hacer **git switch** a una rama existente

Crear una nueva branch con los cambios

Para crear una nueva branch desde el estado actual, solo hacer **git switch -c [new branch]** esta rama se realizará desde el estado actual e incluira el trabajo realizado en el directorio local.

Hacer commits, cambios, etc...

Usted puede hacer cambios y commit en este estado y revisar los archivos, pero estos cambios **no serán persistentes** a menos que usted cree una nueva branch. Si cambia a una branch existente, se perderan para siempre.