GIT

¿Qué necesitas?

- Tener GIT instalado en tu sistema.

Comandos básicos de GIT

git config

Uno de los comandos más usados en git es git config, que puede ser usado para establecer una configuración específica de usuario, como sería el caso del email, un algoritmo preferido para diff, nombre de usuario y tipo de formato, etc... Por ejemplo, el siguiente comando se usa para establecer un email:

```
git config --global user.name "Marcos Fdez"
git config --global user.email "sam@google.com"
git config --global github.user repitaneo
git config --global github.token f24702a53eb455d9d.....6f5205d20a3
```

git init

Este comando se usa para crear un nuevo repertorio GIT:

```
git init
```

git add

Este comando puede ser usado para agregar archivos al index. Por ejemplo, el siguiente comando agrega un nombre de archivo temp.txt en el directorio local del index:

```
git add temp.txt
git add .
```

git commit

El comando commit es usado para cambiar a la cabecera. Ten en cuenta que cualquier cambio comprometido no afectara al repertorio remoto. Usa el comando:

```
git commit –m “Message to go with the commit here”
```

Una opción para ver los valores que se eliminan y añaden. Permite editar el fichero commit y añadir el mensaje en la primera línea. Este modo accede al editor VI (incluso en Windows)

```
git commit -v
```

Para saltarse el stage se puede usar

```
git commit -a -m “nombre de version”
```

No obstante conviene hacer los commit sin parámetros, abrir vim, y especificar todos los cambios detallando bien el commit.

```
git commit
```

git status

Este comando muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser añadidos o comprometidos.

```
git status
```

git branch

Este comando se usa para listar, crear o borrar ramas. Para listar todas las ramas se usa:

```
git branch <nombre>
```

para borrar la rama:

```
git branch -d <branch-name>
```

git checkout

El comando checkout se puede usar para crear ramas o cambiar entre ellas. Por ejemplo, el siguiente comando crea una nueva y se cambia a ella:

```
command git checkout -b <banch-name>
```

Para cambiar de una rama a otra solo usa:

```
git checkout <branch-name>
```

git merge

Este comando se usa para fusionar una rama con otra rama activa:

```
git merge <branch-name>
```

git log

Ejecutar este comando muestra una lista de commits en una rama junto con todos los detalles. Por ejemplo:

```
commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw
Author: Alex Hunter <alexh@gmail.com>
Date:   Mon Oct 1 12:56:29 2016 -0600
```

Algunas opciones útiles para visualizar mejor son:

```
git log --oneline --graph --all
```

git reset

Para resetear el index y el directorio que está trabajando al último estado comprometido se usa este comando:

```
git reset --hard HEAD
```

git revert

Para resetear un cambio agregando otro commit:

```
git revert HEAD
```

git diff

Este comando se usa para hacer una lista de conflictos. Para poder ver conflictos con el archivo base usa:

```
git diff --base <file-name>
```

Entre commits:

```
git diff <id> <id> | HEAD HEAD~1
```

Para solo ver una lista de todos los conflictos presentes usa:

```
git diff
```

Subida de repositorio a github

```
git remote add origin https://github.com/usuario/repositorio.git
git push -u origin master
```

Comandos (un poco) menos usados de GIT

git clone

Este comando se usa con el propósito de revisar repertorios. Si el repertorio está en un servidor remoto se tiene que usar el siguiente comando:

```
git clone alex@93.188.160.58:/path/to/repository
```

Pero si estás por crear una copia local funcional del repertorio, usa el comando:

```
git clone /path/to/repository
```

Sirve para importar un repositorio para unirse, por ejemplo a un proyecto.

.gitignore

Fichero con listas de ficheros para ser ignorados en los commits. Permite obviar elementos tales como logs, ... Contenido de ejemplo de un fichero.

```
#comentario
$cat .gitignore
*.[o]a
*.~
```

git rm

Este comando se puede usar para remover archivos del index y del directorio que está trabajando:

```
git rm filename.txt
```

git mv

Este comando se puede usar para mover o renombrar ficheros

```
git mv filename.txt filename.pep
```

git fetch

Este comando le permite al usuario buscar todos los objetos de un repositorio remoto que actualmente no reside en el directorio local que está trabajando. Por ejemplo:

```
git fetch origin
```

git rebase

Este comando se usa para la re aplicación de los compromisos en otra rama. Por ejemplo:git rebase master

git pull

Para poder fusionar todos los cambios que se han hecho en el repositorio local trabajando, el comando que se usa es:

```
git pull
```

Comandos poco usados o avanzados de GIT

git tag

Etiquetar se usa para marcar commits específicos con asas simples. Por ejemplo:

```
git tag 1.1.0 <instert-commitID-here>
```

git stash

Este es uno de los comandos menos conocidos, pero ayuda a salvar cambios que no están por ser comprometidos inmediatamente, pero temporalmente:

```
git stash
```

git show

Se usa para mostrar información sobre cualquier objeto git. Por ejemplo:

```
git show
```

git ls-tree

Para ver un objeto de árbol junto con el nombre y modo de cada uno de ellos, y el valor blob´s SHA-1, se usa:

```
git ls-tree HEAD
```

git cat-file

Usando el valor SHA-1, se puede ver el tipo de objeto usando este comando. Por ejemplo:

```
git cat-file -p d670460b4b4aece5915caf5c68d12f560a9fe3e4
```

git grep

Este comando le permite al usuario buscar en los árboles de contenido cualquier frase o palabra. Por ejemplo, para buscar por www.tupaginaweb.com en todos los archivos se usaría:

```
git grep “www.tupaginaweb.com”
```

gitk

Este es la interfaz gráfica para un repositorio local que puede invocar escribiendo y ejecutando:

```
gitk
```

git instaweb

Con este comando un servidor web puede correr interconectado con el repositorio local. Un navegador web también está automáticamente dirigido a el:

```
git instaweb –http=webrick
```

git gc

Para optimizar el repositorio por medio de una recolección de basura, que limpiara archivos innecesarios y los optimizara, usa:git hc

```
git archive
```

Este comando le permite al usuario crear archivos zip o tar que contengan los constituyentes de un solo árbol de repositorio:git archive – -format=tar master

git prune

Con este comando los objetos que no tengan ningún puntero entrante serán eliminados:git prune

git fsck

Para poder hacer un chequeo de integridad del sistema de archivos git, usa este comando. Cualquier objeto corrompido será detectado:git fsck