Comandos básicos de GIT

Cuando hablamos de controlar la versión de sistemas hay muy pocos que hagan el estupendo trabajo que hace **GIT**, tanto en rendimiento como en predominio. GIT fue desarrollado por Linus Torvalds en el 2005 y hoy en día, millones de compañías lo usan como un eficiente administrador de código y control de versiones en sus proyectos.

Este software de código abierto se puede descargar tanto para Linux, Windows, Mac y Solaris, y en este tutorial aprenderás los comandos básicos de GIT para sacarle el mejor provecho.

¿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.email sam@google.com
```

- **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 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

- **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"`
- **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 push**
  Este es uno de los comandos más básicos. Un simple push envía los cambios que se han hecho en la rama principal de los repertorios remotos que están asociados con el directorio que está trabajando. Por ejemplo:

  ```
  git push  origin master
  ```

- **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 remote**
  El comando git se usa para conectar a un repositorio remoto. El siguiente comando muestra los repositorios remotos que están configurados actualmente:

  ```
  git remote -v
  ```

Este comando te permite conectar al usuario con el repositorio local a un servidor remoto:

```
git remote add origin <93.188.160.58>
```

- **git branch**
  Este comando se usa para listar, crear o borrar ramas. Para listar todas las ramas se usa:
- `git branch`
-
- `para borrar la rama:`
-
  ```
  git branch -d <branch-name>
  ```

- **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
  ```

- **git merge**
  Este comando se usa para fusionar una rama con otra rama activa:

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

- **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>
  ```

  El siguiente comando se usa para ver los conflictos que hay entre ramas que están por ser fusionadas para poder fusionarlas sin problemas:

  ```
  git diff <source-branch> <target-branch>
  ```

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

  ```
  git diff
  ```

- **git tag**
  Etiquetar se usa para marcar commits específicos con asas simples. Por ejemplo:

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

- **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
```

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

  ```
  git rm filename.txt
  ```

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

Tomado de: https://www.hostinger.co/tutoriales/comandos-de-git