**Comandos de GIT básicos**

* Git init creará un nuevo repositorio local GIT. El siguiente comando de Git creará un repositorio en el directorio actual:

Git init

* Como alternativa, puedes crear un repositorio dentro de un nuevo directorio especificando el nombre del proyecto:

Git init [nombre del proyecto]

* Git clone se usa para copiar un repositorio. Si el repositorio está en un servidor remoto, usa:

Git clone nombredeusuario@host:/path/to/repository

* A la inversa, ejecuta el siguiente comando básico para copiar un repositorio local:

Git clone /path/to/repository

* Git add se usa para agregar archivos al área de preparación. Por ejemplo, el siguiente comando de Git básico indexará el archivo temp.txt:

Git add <temp.txt>

* Git commit creará una instantánea de los cambios y la guardará en el directorio git.

Git commit –m “El mensaje que acompaña al commit va aquí”

* Git config puede ser usado para establecer una configuración específica de usuario, como el email, nombre de usuario y tipo de formato, etc. Por ejemplo, el siguiente comando se usa para establecer un email:

Git config –global user.email [tuemail@ejemplo.com](mailto:tuemail@ejemplo.com)

* La opción -global le dice a GIT que vas a usar ese correo electrónico para todos los repositorios locales. Si quieres utilizar diferentes correos electrónicos para diferentes repositorios, usa el siguiente comando:

Git config –local user.email [tuemail@ejemplo.com](mailto:tuemail@ejemplo.com)

* Git status muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser preparados o confirmados.

Git status

* Git push se usa para enviar confirmaciones locales a la rama maestra del repositorio remoto. Aquí está la estructura básica del código:

Git push origin <master>

* Git checkout crea ramas y te ayuda a navegar entre ellas. Por ejemplo, el siguiente comando crea una nueva y automáticamente se cambia a ella:

Command git checkout -b <branch-name>

* Para cambiar de una rama a otra, sólo usa:

Git checkout <branch-name>

* Git remote te permite ver todos los repositorios remotos. El siguiente comando listará todas las conexiones junto con sus URLs:

Git remote -v

* Para conectar el repositorio local a un servidor remoto, usa este comando:

Git remote add origin <host-or-remoteURL>

* Por otro lado, el siguiente comando borrará una conexión a un repositorio remoto especificado:

Git remote <nombre-del-repositorio>

* Git branch se usa para listar, crear o borrar ramas. Por ejemplo, si quieres listar todas las ramas presentes en el repositorio, el comando debería verse así:

Git branch

* Si quieres borrar una rama, usa:

Git branch -d <branch-name>

* Git pull fusiona todos los cambios que se han hecho en el repositorio remoto con el directorio de trabajo local.

Git pull

* Git merge se usa para fusionar una rama con otra rama activa:

Git merge <branch-name>

* Git diff se usa para hacer una lista de conflictos. Para poder ver conflictos con respecto al archivo base, usa:

Git diff –base <file-name>

* El siguiente comando se usa para ver los conflictos que hay entre ramas antes de fusionarlas:

Git diff <source-branch> <target-branch>

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

Git diff

* Git tag marca commits específicos. Los desarrolladores lo usan para marcar puntos de lanzamiento como v1.0 y v2.0.

Git tag 1.1.0 <instert-commitID-here>

* Git log se usa para ver el historial del repositorio listando ciertos detalles de la confirmación. Al ejecutar el comando se obtiene una salida como ésta:

Commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw

Author: Alex Hunter [alexh@gmail.com](mailto:alexh@gmail.com)

Date: Mon Oct 1 12:56:29 2016 -0600

* Git reset sirve para resetear el index y el directorio de trabajo al último estado de confirmación.

Git reset - -hard HEAD

* Git rm se puede usar para remover archivos del index y del directorio de trabajo.

Git rm filename.txt

* Git stash guardará momentáneamente los cambios que no están listos para ser confirmados. De esta manera, pudes volver al proyecto más tarde.

Git stash

* Git show se usa para mostrar información sobre cualquier objeto git.

Git show

* Git fetch le permite al usuario buscar todos los objetos de un repositorio remoto que actualmente no se encuentran en el directorio de trabajo local.

Git fetch origin

* Git ls-tree te permite ver un objeto de árbol junto con el nombre y modo de cada ítem, y el valor blob de SHA-1. Si quieres ver el HEAD, usa:

Git ls-tree HEAD

* Git cat-file se usa para ver la información de tipo y tamaño de un objeto del repositorio. Usa la opción -p junto con el valor SHA-1 del objeto para ver la información de un objeto específico, por ejemplo:

Git cat-file –p d670460b4b4aece5915caf5c68d12f560a9fe3e4

* Git grep le permite al usuario buscar frases y palabras específicas en los árboles de confirmación, el directorio de trabajo y en el área de preparación. Para buscar por [www.hostinger.com](http://www.hostinger.com) en todos los archivos, usa:

Git grep [www.hostinger.com](http://www.hostinger.com)

* Gitk muestra la interfaz gráfica para un repositorio local. Simplemente ejecuta:

Gitk

* Git instaweb te permite explorar tu repositorio local en la interfaz GitWeb. Por ejemplo:

Git instaweb –http=webrick

* Git gc limpiará archivos innecesarios y optimizará el repositorio local.

Git gc

* Git archive le permite al usuario crear archivos zip o tar que contengan los constituyentes de un solo árbol de repositorio. Por ejemplo:

Git archive - -format=tar master

* Git prune elimina los objetos que no tengan ningún apuntador entrante.

Git prune

* Git fsck realiza una comprobación de integridad del sistema de archivos git e identifica cualquier objeto corrupto

Git fsck

* Git rebase se usa para aplicar ciertos cambios de una rama en otra. Por ejemplo:

Git rebase master