What is Git?

Snapshots, Not differences

Los sistemas VCS usuales tienen un control de versiones *delta-based*. Guardan listas de cambios por archivo.

Git guarda snapshots de todos los archivos modificados y guarda apuntadores a cada estado. Cada repositorio de git es una copia completa y funcional del proyecto, no requiere dependencias externas y es todo local.

Git hace checksums en cada commit, lo cual hace difícil o imposible modificar sin que Git se de cuenta o perder información porque todo se verifica y valida.

The Three States

Git tiene tres estados en los que puede estar un archivo en cualquier momento dado: - **Modified**: El archivo tiene modificaciones pero aún no se le ordena a Git incluir los cambios en el siguiente "commit" o confirmación.

- **Staged / Preparado**: El archivo fue modificado y marcado para que Git lo observe y guarde su versión actual en el siguiente commit.
- **Commited / Confirmado**: Los cambios al archivo ya fueron observados y confirmados. Se guarda el estado actual del archivo en la base de datos de Git.

Adicionalmente hay archivos que pueden estar dentro del directorio, en disco, y no existir dentro de la base de datos Git. Los tres estados anteriores solo aplican para archivos *rastreados* o *tracked*. Los archivos *no-rastreados* o *untracked* siguen dentro del alcance de búsqueda de Git, y éste aún los puede ver, pero tiene instrucciones de no rastrear modificiaciones.

Similarmente, hay tres secciones o "espacios" en cada proyecto de Git:

1. Working Directory.
2. Staging Area.
3. Git Repository.

Working Directory es el directorio "fisico", local en tu máquina en donde estás trabajando. Es como cualquier otra carpeta, pero tiene la particularidad de que hay una instancia de Git observándola. El *Working Directory* es el $_{\text{checkout}}$ de una versión particular del proyecto. Los archivos fueron descomprimidos de la base de datos de Git, y puestos en el disco para poder ser modificados como cualquier otro archivo.

Staging Area se puede pensar como el lugar físico a donde se mandan los archivos que están marcados como listos para ser confirmados (commited). En realidad es un archivo, pero la abstracción de lugar es más útil.

Git repository es donde vive la base de datos de Git, y todos los metadatos asociados. Aqui se guardan las diferentes versiones comprimidas de cada commit hecho, asi como apuntadores y metadatos de documentación. El archivo .git es el que contiene toda esta información, y es el que obtiene uno al clonar un repositorio.

Un archivo particular se considera *commited* (confirmado) si fue modificado, añadido al *staging area*, y se hizo un *commit* (confirmación). Los cambios que se le hicieron a ese archivo ahora están grabados en git. Un archivo se considera *staged* (preparado) si se modificó y se añadió al *staging area*. Un archivo se considera *modified* (modificado) si sufrió cambios desde la última versión que conoce Git, pero aún no se añade al *staging area*. Los archivos *modified* que no sean añadidos al *staging area* antes de efectuar un *commit* no serán rastreados en ese *commit*, y sus cambios no se guardarán.

Basic Git Workflow

1. Modifica archivos locales en disco.
2. Se elige qué archivos se desea rastrear (*track*) añadiéndolos al *staging area*. Son estos y solo estos los archivos que serán parte de la siguente confirmación.
3. Se lleva a cabo una *confirmación*, en la que se toman un snapshot del estado actual del *staging area* y se guarda permanentemente en el repositorio Git, junto con datos de identificación y mensajes de confirmación.

Command Line Interface

Para obtener ayuda sobre el comando

```
$ git add -h
usage: git add [<options>] [--] <pathspec>...

    -n, --dry-run         dry run
    -v, --verbose         be verbose

    -i, --interactive     interactive picking
    -p, --patch           select hunks interactively
    -e, --edit            edit current diff and apply
    -f, --force           allow adding otherwise ignored files
```

```
-u, --update          update tracked files
--renormalize         renormalize EOL of tracked files (implies -u)
-N, --intent-to-add   record only the fact that the path will be added later
-A, --all             add changes from all tracked and untracked files
--ignore-removal      ignore paths removed in the working tree (same as --no-all)
--refresh             don't add, only refresh the index
--ignore-errors       just skip files which cannot be added because of errors
--ignore-missing      check if - even missing - files are ignored in dry run
--chmod (+|-)x        override the executable bit of the listed files"
```

Git Basics

Getting a Git Repository

Usualmente uno obtiene un repositorio de Git en dos maneras:

1. Tomas cualquier carpeta local en tu disco, y la conviertes en un repositorio con `git init`, o bien
2. *Clonas* un repositorio de Git existente de algún otro lugar.

Para crear un repositorio nuevo desde una carpeta local, basta con navegar hasta la carpeta deseada a través de la terminal de comandos, y ejecutar el comando `git init` que creará un repositorio nuevo con todo lo necesario para usar Git, y rastrear cambios en los archivos de ese directorio. El comando `init`` crea un "esqueleto de git". En este punto, ningun archivo está siendo rastreado, todos aparecen como _untracked_. Si deseas empezar a rastrear cambios, debes cambiar el estatus de los archivos de _untracked_ a _tracked_ a través del commandogit add. Cabe mencionar que los archivos nuevos, es decir los originales en un directorio antes de corrergit init“‘, o aquellos creados o añadidos después de haber creado el repositorio, siempre aparecerán por primera vez como *untracked*, hasta que se le señale a Git que se deben rastrear.

> Notas: El comando `git add` tiene dos funciones: Cambia el estatus de un archivo de *untracked* a *tracked*, y además añade archivos *modified* al *staging area* para prepararlos para un *commit*.
>
> `add` recibe como argumento nombres de archivos, o patrones *glob*.

Tradicionalmente una vez que se han rastreado los archivos de interés se hace un ~primer commit~