

Amilcar Alexander Medina Méndez

Homework-1

Que es GIT

Git es un sistema de control de revisiones de código abierto que se mantiene activo y lo utilizan miles de desarrolladores en todo el mundo.

Sistema de control de versione, administrar mis cambios.

Control de versiones: antes, ahora, futuro

Multiusuario

Control de cambios por usuario

Regresar a versiones anteriores cuando quiera

Repositorios Locales y remotos

Yo decido cuando cambio

Control de versiones con GIT

Git es un Sistema de Control de Versiones Distribuido (DVCS) utilizado para guardar diferentes versiones de un archivo (o conjunto de archivos) para que cualquier versión sea recuperable cuando lo desee.

Git también facilita el registro y comparación de diferentes versiones de un archivo. Esto significa que los detalles sobre qué cambió, quién cambió qué, o quién ha iniciado una propuesta, se pueden revisar en cualquier momento.

Estados de un archivo en git

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged).

Estado modificado

Un archivo en el estado modificado es un archivo revisado – pero no acometido (sin registrar).

En otras palabras, archivos en el estado modificado son archivos que has modificado, pero no le has instruido explícitamente a Git que controle.

Estado preparado

Archivos en la etapa preparado son archivos modificados que han sido seleccionados – en su estado (versión) actual – y están siendo preparados para ser guardados (acometidos) al repositorio .git durante la próxima instantánea de confirmación.

Una vez que el archivo está preparado implica que has explícitamente autorizado a Git que controle la versión de ese archivo.

Estado confirmado

Archivos en el estado confirmado son archivos que se guardaron en el repositorio .git exitosamente.

Por lo tanto, un archivo confirmado es un archivo en el cual has registrado su versión preparada en el directorio (carpeta) Git.

Como se configura un repositorio

Creamos una cuenta.



Configuramos repositorio.

Owner: Mballina42 / Repository name: EjemploSimu ✓

Great repository names are short and memorable. Need inspiration? How about [legendary-ecto-dollop?](#)

Description (optional):

☐ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None Add a license: None ⓘ

Create repository

Creamos un nuevo repositorio.



Quick setup — if you've done this kind of thing before

or **HTTPS** **SSH** `https://github.com/Mballina42/EjemploSimu.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# EjemploSimu" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Mballina42/EjemploSimu.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Mballina42/EjemploSimu.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Comandos en GIT

- \$git log
- git checkout master
- vemos los elementos de nuestro commit junto a su hash
- cambiamos a la rama master
- git init
- git status
- git add file1
- git status
- git add file2
- git status
- git commit
- iniciamos git, carpeta se convierte en Working directory
- revisamos el contenido
- agregamos file1 a staging area
- muestra estados de los archivos
- agregamos file2 a staging area
- muestra estados de los archivos
- abre su editor de texto, siempre agregar un comentario. Se mueve todo al repositorio
- git status
- echo nueva linea >> file1
- git status
- git checkout -- file1
- git status
- echo 42 u infidel >> file1
- cat file1
- git diff file1
- git add file1
- git commit
- muestra que no hay nada a que hacer commit
- escribimos en file1
- muestra que hay un archivo en el WD
- deshace la modificación, reemplazando con lo del ultimo commit
- muestra que no hay modificaciones
- modificamos file1
- muestra diferencia entre file1 del WD y el Rep
- agregamos file1 a staging area
- abre su editor de texto, agregar un comentario. Se mueve todo al repositorio

- mkdir nover
- touch nover.txt
- touch .gitignore
- echo nover >> .gitignore
- echo nover.c >> .gitignore
- git status
- touch nover/otro.txt
- git status
- git add .gitignore
- git commit -m "comentario"

- creamos una carpeta
- creamos un archivo
- necesitamos un archivo de nombre .gitignore
- escribimos el nombre de la carpeta y del archivo dentro del archivo .gitignore
- muestra que solo existe .gitignore
- creo un archivo dentro de la carpeta ignorada
- muestra que solo existe .gitignore
- agregamos .gitignore a staging area
- se mueve todo al repositorio con la instrucción (-m) podemos agregar un comentario en línea

- git branch
- git branch OTRAR
- git branch
- git checkout OTRAR
- git branch
- touch f1 f2 f3 f4 f5
- git add .
- git status
- git commit -m "comentar"
- git log

- muestra que solo existe la rama master
- creamos otra rama en nuestro proyecto
- muestra las ramas existentes
- cambiamos a la rama OTRAR
- muestra que estamos en rama OTRAR
- creamos varios archivos
- agregar los archivos del proyecto usamos (.)
- muestra los files en staging area
- se mueve todo al repositorio.
- vemos los elementos de nuestros commits junto a cada hash