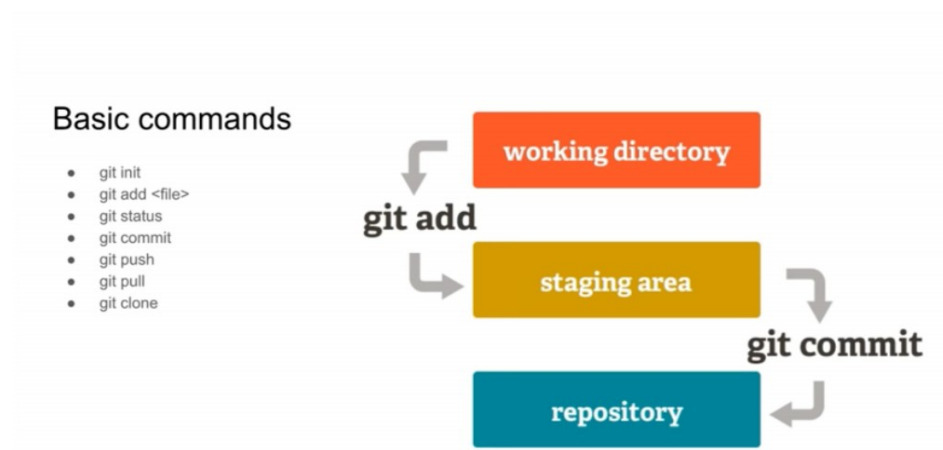


INSTALACIÓN DE GIT EN UBUNTU SERVER 18.04

Git es un sistema de control de versiones distribuido, de código abierto, desarrollado por Linus Torvalds.

Principalmente es utilizado para la gestión de código fuente en el desarrollo de software, pero también puede manejar versionado de cualquier proyecto grande o pequeño que tenga ficheros como markdown, txt, gráficos, binarios y otros tipos de archivo. Es realmente eficiente, escalable y muy rápido, está diseñado para que nada se pierda, registra todos los cambios que se realizan sobre los archivos permitiendo llevar un seguimiento de esas modificaciones, comparar versiones o regresar a versiones anteriores. Gracias a que es un sistema distribuido, múltiples desarrolladores pueden trabajar de forma simultánea sobre un mismo proyecto, informando y manteniendo un historial de quién realizó cada cambio y en qué momento lo hizo.

Git realiza snapshots (fotos) de los cambios realizados.



Instalación

```
sudo apt install git
```

Para verificar que la instalación fue exitosa y qué versión de git está instalada escribimos:

```
$ git --version
usuario@ubuntuserver01:~/PruebaGit$ git --version
git version 2.17.1
```

Configuración

Hay que configurar el usuario y su correo:

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

Si queremos validar lo que hemos configurado ejecutamos el siguiente comando y lo comprobamos:

```
git config --list
```

Lista básica de comandos de Git

Ahora Git ya está instalado y configurado correctamente, para ver una lista completa de todos los comandos que se pueden usar:

```
git --help
```

Uso Básico

Creación de un repositorio local

Lo primero es crear una carpeta en nuestro ordenador, que servirá como un repositorio local. Para ello, basta ejecutar el siguiente comando:

```
git init PruebaGit
```

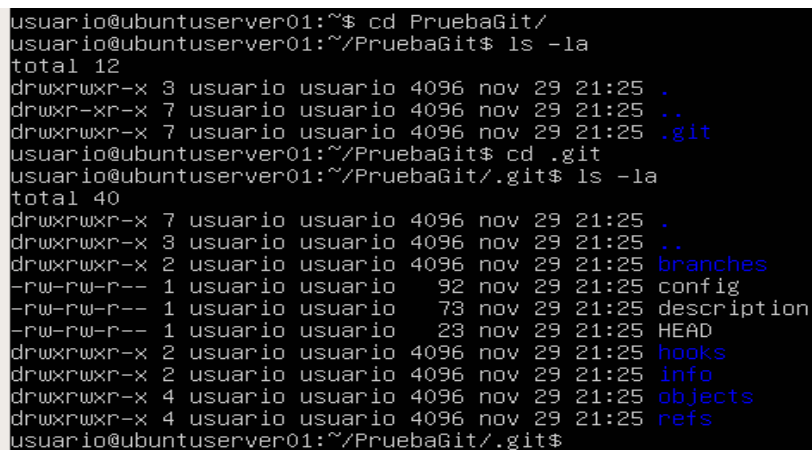
Este comando crea la carpeta PruebaGit. A su vez, la sub-carpeta .init hace que PruebaGit sea reconocido como un repositorio local de Git.

Si se crea el repositorio con éxito, aparecerá una línea similar a la siguiente:

```
Initialized empty Git repository in /home/tu_usuario/PruebaGit/.git/
```

Entramos a la carpeta PruebaGit:

```
cd PruebaGit
```



```
usuario@ubuntuserver01:~$ cd PruebaGit/
usuario@ubuntuserver01:~/PruebaGit$ ls -la
total 12
drwxrwxr-x 3 usuario usuario 4096 nov 29 21:25 .
drwxr-xr-x 7 usuario usuario 4096 nov 29 21:25 ..
drwxrwxr-x 7 usuario usuario 4096 nov 29 21:25 .git
usuario@ubuntuserver01:~/PruebaGit$ cd .git
usuario@ubuntuserver01:~/PruebaGit/.git$ ls -la
total 40
drwxrwxr-x 7 usuario usuario 4096 nov 29 21:25 .
drwxrwxr-x 3 usuario usuario 4096 nov 29 21:25 ..
drwxrwxr-x 2 usuario usuario 4096 nov 29 21:25 branches
-rw-rw-r-- 1 usuario usuario  92 nov 29 21:25 config
-rw-rw-r-- 1 usuario usuario  73 nov 29 21:25 description
-rw-rw-r-- 1 usuario usuario  23 nov 29 21:25 HEAD
drwxrwxr-x 2 usuario usuario 4096 nov 29 21:25 hooks
drwxrwxr-x 2 usuario usuario 4096 nov 29 21:25 info
drwxrwxr-x 4 usuario usuario 4096 nov 29 21:25 objects
drwxrwxr-x 4 usuario usuario 4096 nov 29 21:25 refs
usuario@ubuntuserver01:~/PruebaGit/.git$
```

Creación de un archivo README para describir el repositorio

El archivo README se utiliza generalmente para describir lo que el repositorio contiene o lo que el proyecto se trata. Para crear uno, basta ejecutar:

```
gedit README
```

Adición de los archivos del repositorio a un índice

Este es un paso importante. Antes de poder subir los cambios a Github u otro servidor compatible con Git, hay que indexar todos los archivos contenidos en el repositorio local. Este índice contendrá los archivos nuevos así como los cambios a los archivos existentes en el repositorio local.

En nuestro caso, nuestro repositorio local ya contiene un nuevo archivo: el README. Por lo tanto, vamos a crear otro archivo con un programa C simple y al cual llamaremos ejemplo.c. Los contenidos del mismo serán:

```
#include <stdio.h>
int main()
{
    printf("hello world");
    return 0;
}
```

Así que, ahora tenemos 2 archivos en nuestro repositorio local: README y ejemplo.c.

El siguiente paso es agregar estos archivos al índice:

```
git add README
```

```
git add ejemplo.c
```

El comando “git add” se puede utilizar para agregar cualquier número de archivos y carpetas al índice. Para agregar todos los cambios, sin especificar el nombre de los archivos, es posible ejecutar “git add .” (con un punto al final).

Al ejecutar este comando, nuestro archivo ha pasado del **working directory**, o directorio de trabajo, al depósito intermedio, o **Stage**. Si queremos hacer marcha atrás y sacarlo de *Stage*, ejecutaremos la orden:

```
git reset mi_archivo
```

Guardar los cambios realizados en el índice

Una vez añadidos todos los archivos, es posible dejar un registro de estos cambios haciendo lo que se llama un “commit”. Esto significa que ya se ha terminado de agregar o modificar archivos y que los cambios pueden ser subidos al repositorio remoto de Github. Para ello, hay que ejecutar el siguiente comando:

```
git commit -m "mensaje"
```

“mensaje” puede ser cualquier mensaje que describa brevemente los cambios en cuestión, por ejemplo: “agregué tal funcionalidad” o “corregí tal cosa”, etc.

Ahora modificamos el fichero ejemplo.c y observamos si git ha captado los cambios, mediante el comando:

```
git status
```

```
usuario@ubuntu01:~/PruebaGit$ gcc -o ejemplo ejemplo.c
usuario@ubuntu01:~/PruebaGit$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   ejemplo
        modified:   ejemplo.c

no changes added to commit (use "git add" and/or "git commit -a")
usuario@ubuntu01:~/PruebaGit$
```

Con git diff vemos las diferencias con (\pm)

Con git log vemos todos los commit que hemos hecho

Podríamos descartar los cambios realizados en git checkout --nombreFichero

Podríamos querer ignorar un directorio de nuestro proyecto esto lo conseguiríamos mediante un fichero .gitignore en él escribiríamos el nombre de la carpeta o fichero/s a ignorar.

Sudo nano .gitignore

test

fichero1.txt

Para saber en qué rama estamos:

git branch

Para crear una nueva rama (versión) se utiliza:

git branch nombreRama.

Para movernos a esa nueva rama:

git checkout login

Dentro de esta nueva rama vamos a hacer algunos cambios, creamos ficheros autentificacion, pruebaLogin, etc. (Si una carpeta está vacía no la muestra el git status)

Para obtener la documentación de git:

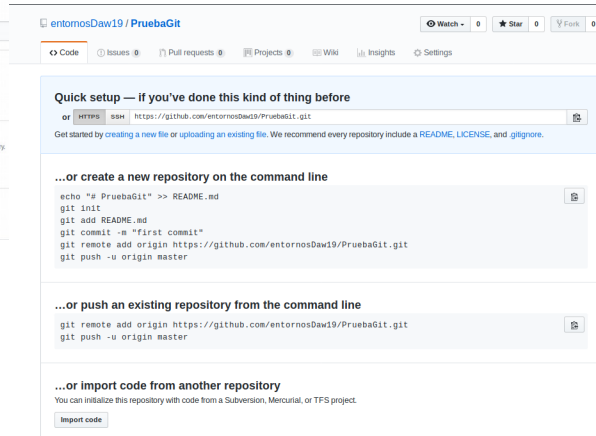
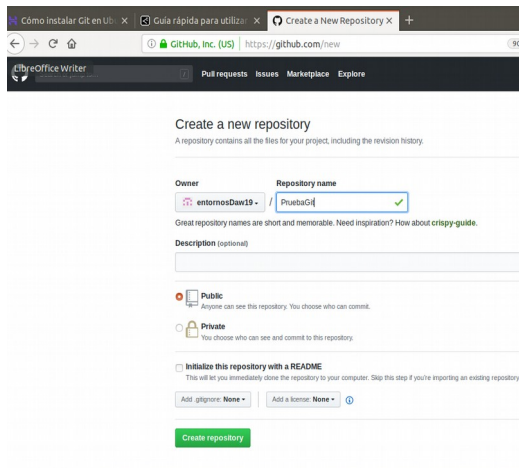
git help

git help status

(Para salir q)

Creación de un repositorio en GitHub

El nombre del repositorio debe ser el mismo que el repositorio del sistema local. En este caso, será “PruebaGit”. Para ello, antes que nada, hay que iniciar sesión en [Github](#). Luego, hay que hacer clic en el signo más (+) en la esquina superior derecha de la página y seleccionar la opción “crear nuevo repositorio”. Finalmente, hay que rellenar los datos y hacer clic en el botón “crear repositorio”.



Una vez hecho esto se creará el repositorio y será posible subir el contenido del repositorio local en el repositorio GitHub. Para conectarse al repositorio remoto en GitHub hay que ejecutar el comando:

```
git remote add origin https://github.com/nombre\_usuario/PruebaGit.git
```

(Reemplazad 'nombre_usuario' y 'PruebaGit' con el nombre de usuario y carpeta correspondientes.

Empujar archivos del repositorio local al repositorio GitHub

El paso final es empujar el contenido del repositorio local hacia el repositorio remoto, mediante el comando:

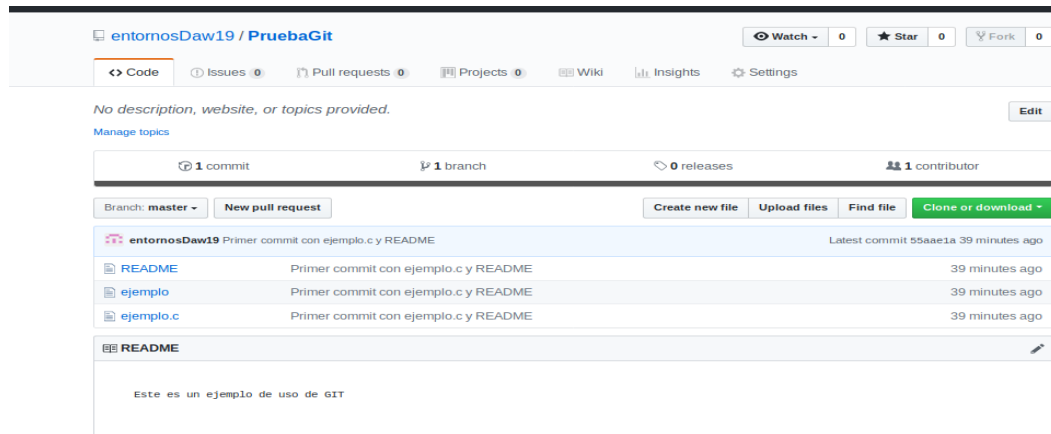
```
git push -u origin master
```

Sólo queda introducir las credenciales de inicio de sesión (nombre de usuario y contraseña).

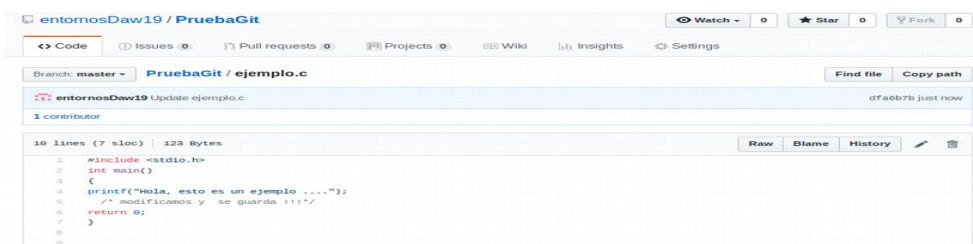
Esto subirá todo el contenido de la carpeta PruebaGit (repositorio local) a GitHub (repositorio externo). Para los proyectos subsiguientes ya no hará falta seguir estos pasos desde cero. Se empezará desde el paso 3 directamente.

```
usuario@ubuntu01:~/PruebaGit$ git remote add origin https://github.com/entornosDaw19/PruebaGit.git
usuario@ubuntu01:~/PruebaGit$ git push -u origin master
Username for 'https://github.com': entornosDaw19
Password for 'https://entornosDaw19@github.com':
Counting objects: 5, done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 2.70 KiB | 1.35 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/entornosDaw19/PruebaGit/pull/new/master
remote:
To https://github.com/entornosDaw19/PruebaGit.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
usuario@ubuntu01:~/PruebaGit$
```

Ahora, vamos a la página de github y comprobamos que se hayan subido nuestros ficheros:

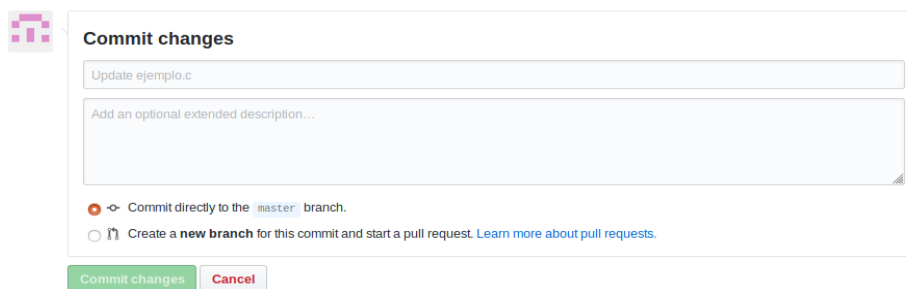


Podemos editar, modificar y guardar los cambios:



Al final de la página, guardamos

los cambios:



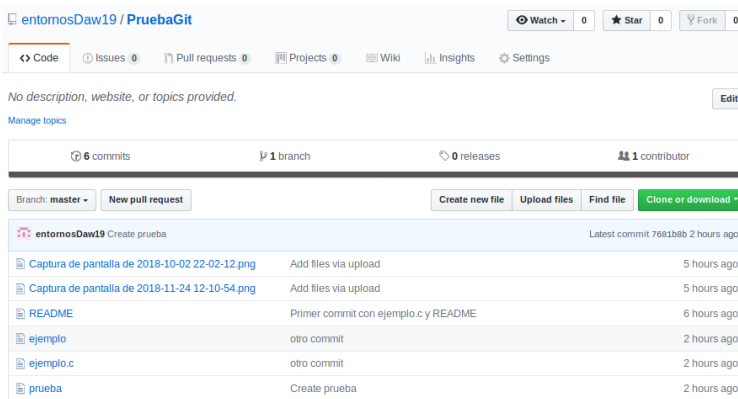
Ahora vamos a borrar todo lo que tenemos en el equipo local para probar cómo clonar el proyecto desde github. Pulsamos en **clone** or downloads y copiamos la URL para pegarla en nuestra terminal utilizando el comando: git clone. Previamente hemos creado un directorio en nuestro equipo.

```
usuario@ubuntuserver01:~$ pwd
/home/usuario
usuario@ubuntuserver01:~$ sudo rm -R PruebaGit/
[sudo] password for usuario:
usuario@ubuntuserver01:~$ _
```

Creamos una carpeta en el home del usuario:

```
usuario@ubuntuserver01:~$ mkdir PruebaClonacion
usuario@ubuntuserver01:~$ cd PruebaClonacion/
usuario@ubuntuserver01:~/PruebaClonacion$ _
```

Y finalmente clonamos el proyecto de github a nuestro equipo local:



git clone https://github.com/entornosDaw19/PruebaGit.git

```
usuario@ubuntuserver01: ~/PruebaClonacion$ git clone https://github.com/entornosDaw19/PruebaGit.git
Cloning into 'PruebaGit'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 22 (delta 9), reused 7 (delta 2), pack-reused 0
Unpacking objects: 100% (22/22), done.
usuario@ubuntuserver01:~/PruebaClonacion$
```

Con git log veremos todos los cambios que se han realizado en nuestro proyecto.

```
usuario@ubuntuserver01:~/PruebaClonacion/PruebaGit$ git log
commit 7681b8b4fa40987e6d361c97a1bcd3aa79c4e6c6 (HEAD -> master, origin/master, origin/HEAD)
Author: entornosDaw19 <45696683+entornosDaw19@users.noreply.github.com>
Date: Sun Dec 9 17:25:23 2018 +0100

    Create prueba

commit 8cac88ed804e17d17efbea0d0270a9ffb8939ee4
Author: entornosDaw19 <ciclos201819@gmail.com>
Date: Sun Dec 9 16:18:56 2018 +0000

    otro commit

commit 706af6c22fe9e15c7dd96f7a9fa08a7461ea4d70
Author: entornosDaw19 <45696683+entornosDaw19@users.noreply.github.com>
Date: Sun Dec 9 14:15:39 2018 +0100

    Add files via upload

commit dfa6b7bcdef714a6b487d47f620f3c5cd5d760db
Author: entornosDaw19 <45696683+entornosDaw19@users.noreply.github.com>
Date: Sun Dec 9 14:01:37 2018 +0100

    Update ejemplo.c

commit 53c361d225caa3bba29f9f96152792099f3f99b0
Author: entornosDaw19 <45696683+entornosDaw19@users.noreply.github.com>
Date: Sun Dec 9 14:00:49 2018 +0100

    Update ejemplo.c

commit 55aae1a7c3f4149057605a8e60a0ab0db8e95266
Author: entornosDaw19 <ciclos201819@gmail.com>
Date: Sun Dec 9 12:17:51 2018 +0000

    Primer commit con ejemplo.c y README
usuario@ubuntuserver01:~/PruebaClonacion/PruebaGit$ _
```

Repositorios remotos

Para ver qué repositorios remotos tienes configurados, puedes ejecutar el comando `git remote`. Si se ha clonado un repositorio, deberías ver por lo menos "origin" —es el nombre predeterminado que le da Git al servidor del que clonaste—:

```
usuario@ubuntuserver01:~/PruebaClonacion$ cd PruebaGit/  
usuario@ubuntuserver01:~/PruebaClonacion/PruebaGit$ git remote -v  
origin https://github.com/entornosDaw19/PruebaGit.git (fetch)  
origin https://github.com/entornosDaw19/PruebaGit.git (push)  
usuario@ubuntuserver01:~/PruebaClonacion/PruebaGit$ _
```

Para recuperar datos de tus repositorios remotos se puede ejecutar:

```
$ git fetch [remote-name]
```

Este comando recupera todos los datos del proyecto remoto que no tengas todavía. Después de hacer esto, deberías tener referencias a todas las ramas del repositorio remoto, que puedes unir o inspeccionar en cualquier momento.

Introducción:

<https://www.youtube.com/watch?v=zH3I1DZNovk>

Curso de github:

<https://www.youtube.com/watch?v=HiXLkL42tMU>

Curso Git - Nuestro primer proyecto

<https://www.youtube.com/watch?v=vH9pkFf1D7M>

<https://www.youtube.com/watch?v=3X1ZWpLwvvo>

<https://git-scm.com/book/es/v1/Fundamentos-de-Git-Trabajando-con-repositorios-remotos>

<https://blog.desdelinux.net/guia-rapida-para-utilizar-github/>