Clonar el repositorio.

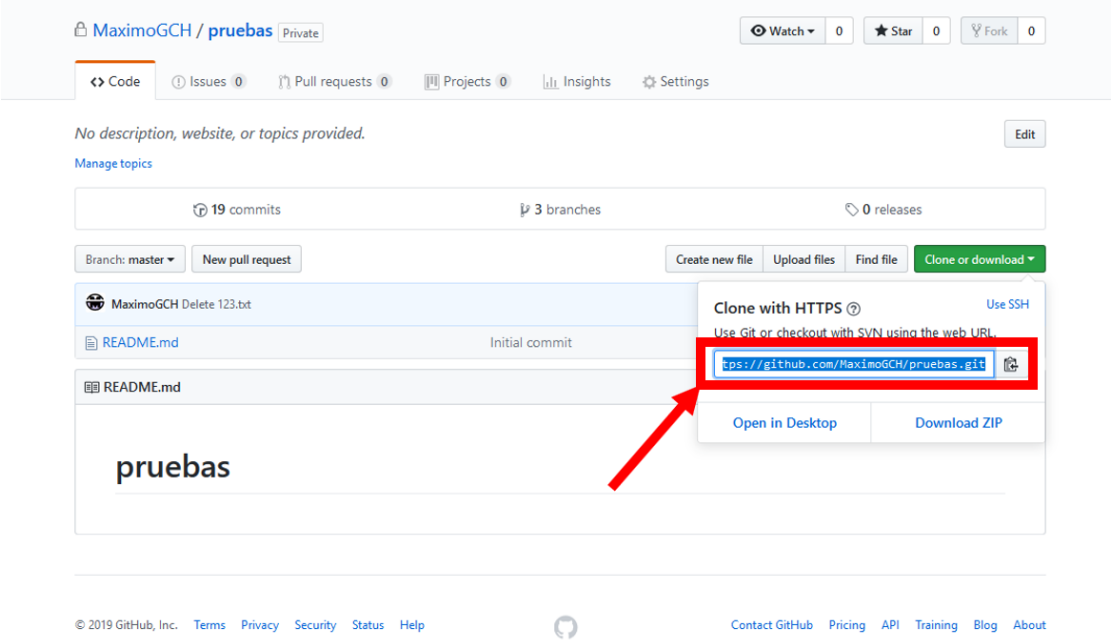
Lo primero que tenemos que hacer es clonar en nuestro ordenador el repositorio en la nube con el que vamos a trabajar.

Abrimos una consola y usamos cd para desplazarnos al directorio de nuestro ordenador en el que queramos clonar el repositorio.

Usamos el comando:

```
git clone URL_REPOSITORIO
```

La url es la de nuestro proyecto en github




```
Microsoft Windows [Versión 10.0.17134.590]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\SirMaximo>E:

E:\>cd E:\git\git

E:\git\git>git clone https://github.com/MaximoGCH/pruebas.git
Cloning into 'pruebas'...
fatal: AggregateException encountered.
   Se han producido uno o varios errores.
Username for 'https://github.com': MaximoGCH
Password for 'https://MaximoGCH@github.com':
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 47 (delta 11), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (47/47), done.

E:\git\git>
```

Como resultado de esto habremos clonado el repositorio en nuestro ordenador. En la carpeta en la que nos encontrábamos se habrá creado una carpeta con el nombre del repositorio.

En mi caso, he clonado el repositorio pruebas en la carpeta E:git\git\pruebas.

Dentro de esta carpeta tendremos la carpeta. git, que es la carpeta en la que git almacena datos del repositorio y lo que estuviera metido en el repositorio de github en el momento en el que lo clonamos.

En mi caso, como era un proyecto vacío, solo se encuentra un archivo README.md.

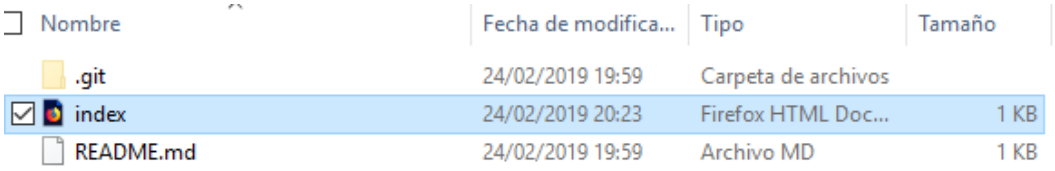
En la consola, usamos cd para meternos en la carpeta nueva.

Crear un archivo y subirlo al git

Creamos un archivo que vamos a subir al repositorio. Por ejemplo, creamos un archivo index.html que contiene una pequeña página web en la que estamos trabajando:

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
  </body>
</html>
```

Este archivo lo metemos en la carpeta que habíamos creado en el paso anterior (con el clone), en mi caso la carpeta E:git\git\pruebas.



Si usamos el comando

```
 “git status”
```

comprobamos que git no tiene en cuenta nuestro archivo pues no lo está rastreando.

```
E:\git\git>cd pruebas

E:\git\git\pruebas>git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        index.html

nothing added to commit but untracked files present (use "git add" to track)

E:\git\git\pruebas>
```

Como podemos comprobar, el propio git nos muestra que deberíamos usar el comando

```
git add NOMBRE_ARCHIVO
```

parar que empiece a rastrear nuestro archivo.

Si tenemos muchos archivos que queremos subir al repositorio podemos usar

```
git add -A
```

Este comando añade todos los archivos modificados o nuevos a la lista de archivos que se subirán cuando hagamos un commit.

Si hacemos un “git status” ahora, podemos comprobar el archivo “index.html” han sido añadidos a la lista de cambios que se guardarán en el próximo commit.

```
E:\git\git\pruebas>git add index.html

E:\git\git\pruebas>git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   index.html


E:\git\git\pruebas>
```

Tenemos el archivo index.html preparado para subirse, pero todavía falta un paso, hacer el commit. Una vez echo el commit, se creará una nueva versión del programa guardando todos los cambios subidos con add. En este caso, nuestro único cambio es añadir el archivo “index.html”.

Usamos el comando:

```
git commit -m “NOMBRE DEL CAMBIO” -m “MENSAJE MAS LARGO”
```

El mensaje es opcional, en el caso del nombre podéis no ponerlo, pero git os lo preguntará luego. Intentar poner un nombre que identifique bien el cambio, para que luego, si surge cualquier error, podáis volver atrás sabiendo en que estado estaba el programa.

```
E:\git\git\pruebas>git commit -m "Añadido fichero index.html"
[master e22d246] Añadido fichero index.html
 1 file changed, 10 insertions(+)
 create mode 100644 index.html

E:\git\git\pruebas>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean

E:\git\git\pruebas>
```

Si hacéis ahora un “git status”, git os pondrá que no hay ningún cambio ni ningún elemento para hacer commit. Como veis, git os muestra todos los elementos que se han modificado en el cambio y en que rama se ha realizado el cambio.

Modificar un archivo

El proceso al modificar un archivo es muy parecido que el de añadirlo.

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
  <p>Hola caracola</p>
  </body>
</html>
```

Voy a añadir una línea al archivo “index.html”

Si hacemos ahora un “git status”, vemos que git nos marca que un archivo ha sido modificado.

```
E:\git\git\pruebas>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
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:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

E:\git\git\pruebas>
```

Al igual que cuando incluimos el archivo, tenemos que hacer un “git add” para añadir el archivo al repositorio. Sin embargo, que pasaría si antes de hacer el “git commit” modificáramos otra vez el archivo e hiciéramos un “git status”.

En mi caso voy a modificar la nueva línea.

```
<p>Hola caracola, esto es nuevo</p>
```

```
E:\git\git\pruebas>git add index.html

E:\git\git\pruebas>git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html

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:   index.html


E:\git\git\pruebas>
```

Como podéis ver git nos marca que index.html está incluido en los cambios que se van a subir con el próximo comit pero a la vez, index.html ha sido modificado y hay que hacer un “git add” para que se suba en el próximo commit.

Esto ocurre porque el “index.html” que se va a subir con el próximo commit es el de antes que modificáramos la línea. En cambio, el que no se va a subir es el que tiene la modificación “, esto es nuevo”.

Si hacemos un “git add”, subiremos la última versión del archivo, y por último podremos ejecutar “git commit”.

```
E:\git\git\pruebas>git add index.html

E:\git\git\pruebas>git commit -"modificado index.html"
[master c8e6cd0] odificado index.html
 1 file changed, 1 insertion(+)
```

Subiendo los cambios a github.

Ya hemos modificado nuestro repositorio, pero estos cambios solo han ocurrido en nuestro ordenador, y, por lo tanto, solo los tenemos nosotros. Lo normal es que queramos subir estos cambios al github para que nuestros compañeros los tengan.

Para esto tendremos que saber primero como se llama el git remoto al que vamos a mandar los datos. Esto lo hacemos con:

```
git remote
```

Este comando nos muestra todos los repositorios remotos con los que está conectado nuestro repositorio local (Para nuestros proyectos básicos solo va a haber uno,

nuestro repositorio en github, pero podría haber más si cogiéramos recursos de varios sitos).

```
E:\git\git\pruebas>git remote
origin

E:\git\git\pruebas>
```

Como habíamos comentado, tenemos solo un repositorio remoto.

Si quieres saber cual es la url de este repositorio puedes usar:

```
git remote -v
```

```
E:\git\git\pruebas>git remote -v
origin  https://github.com/MaximoGCH/pruebas.git (fetch)
origin  https://github.com/MaximoGCH/pruebas.git (push)
```

Con esto comprobamos que, efectivamente, origin es nuestro repositorio github.

Ahora, vamos a subir el estado actual de nuestro repositorio local al de github. Esto lo hacemos con el comando:

```
git push NOMBRE_REPOSITORIO RAMA
```

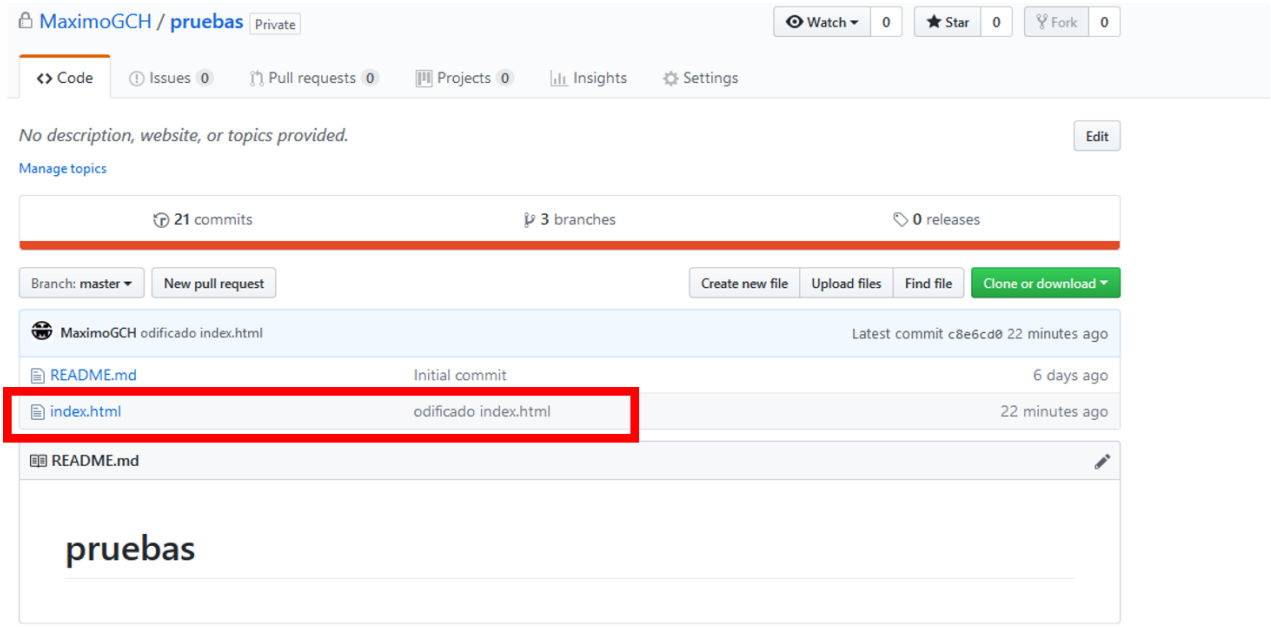
En nuestro caso, el nombre es origin. La rama es opcional, se explicará que es más adelante (Si no se pone se una la rama por defecto, que suele ser la master).

Git nos pedirá que nos registremos (la primera vez) y, una vez registrados, subirá los cambios.

```
E:\git\git\pruebas>git push origin
fatal: AggregateException encountered.
   Se han producido uno o varios errores.
Username for 'https://github.com': MaximoGCH
Password for 'https://MaximoGCH@github.com':
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 714 bytes | 0 bytes/s, done.
Total 6 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/MaximoGCH/pruebas.git
   07460b2..c8e6cd0  master -> master

E:\git\git\pruebas>
```

Como podemos comprobar, los cambios han sido realizados:



¿Qué pasa si otro compañero está trabajando en otra parte del proyecto y quiere actualizar su repositorio para tener los cambios que has realizado tu?

Tu compañero podría clonar el repositorio todo el rato, pero no es la solución, pues ya tiene un repositorio git con el proyecto, solo que de una versión antigua.

Para esto existe el comando:

```
git pull NOMBRE_REPOSITORIO RAMA
```

La rama es opcional, si no se pone se usa la rama por defecto, que suele ser la master (Esto se explica más adelante).

Conflictos.

Imagínate que dos personas están trabajando sobre el mismo archivo. Cuando la primera persona suba el archivo, no pasará nada, simple. Pero la segunda persona no está trabajando con los cambios que ha subido la primera persona, a si que cuando suba sus cambios, git se encontrará con dos archivos distintos. Evidentemente, git no sabe cual de estos dos archivos es correcto, o que hay que hacer para unirlos, a si que se produce un conflicto.

Un colaborador ha modificado la página index.html mientras yo estaba trabajando en esa misma página, y ha hecho un push con sus modificaciones.

Su archivo se ve así:

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
  <p>Hola caracola, esto no es nuevo</p>
  <h2> Esto es un título </h2>
  <p> Parrafo </p>
  </body>
</html>
```

Yo he estado trabajando en el mismo archivo y he incluido unos agradecimientos

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
  <p>Hola caracola, esto es nuevo</p>
  <p>Un saludo</p>
  <p>A mis padres, a Juan y a Castro</p>
  </body>
```

```
</html>
```

Como se que mi compañero ha modificado el proyecto, hago un pull para actualizar mi repositorio con los nuevos cambios. Como ambos hemos trabajado en el mismo archivo, git no sabe que cambios hay que conservar y cuales no, a si que se genera un conflicto que hay que resolver.

```
E:\git\git\pruebas>git pull origin master
fatal: AggregateException encountered.
   Se han producido uno o varios errores.
Username for 'https://github.com': MaximoGCH
Password for 'https://MaximoGCH@github.com':
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/MaximoGCH/pruebas
 * branch            master     -> FETCH_HEAD
   b28b15c..8657fec  master     -> origin/master
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

Si abrimos nuestro fichero index.html, se muestra lo siguiente:

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
<<<<<<< HEAD
  <p>Hola caracola, esto es nuevo</p>
  <p>Un saludo</p>
  <p>A mis padres, a Juan y a Castro</p>
=======
  <p>Hola caracola, esto no es nuevo</p>
  <h2> Esto es un título </h2>
  <p> Parrafo </p>
>>>>>>> 8657fec7c4e208a86bc9e6dbe5ed9c09aab497e7
  </body>
</html>
```

Nos muestra el fichero con las líneas en las que ha ocurrido el conflicto.

Podemos borrar estas líneas y dejar el código como nosotros quedamos:

En mi caso, voy a conservar las líneas de los dos archivos.

```
<!DOCTYPE html>
<html>
<head>
<title>Página de inicio</title>
</head>
  <body>
  <h1>Mi página web</h1>
  <p>Es muy bonita</p>
  <p>Hola caracola, esto es nuevo</p>
  <p>Un saludo</p>
  <p>A mis padres, a Juan y a Castro</p>
  <p>Hola caracola, esto no es nuevo</p>
  <h2> Esto es un título </h2>
  <p> Parrafo </p>
  </body>
</html>
```

Guardamos el archivo, ejecutamos un add, un commit y un pull y podemos realizar un push para guardar nuestros cambios en el servidor.