



MANUAL BASICO DE GITHUB y Git

Sergio Josue Lopez Chanta
2013-13940

INTRODUCCION A LA PROGRAMACION Y
COMPUNTACION 2.
CURSO DE VACACIONES JUNIO 2015

MANUAL DE GITHUB

Que es Git?



git

Git es un software de **control de versiones**. El control de versiones, resumiéndolo mucho, es la gestión de los diversos cambios que se realizan sobre un repositorio (un repositorio es el nombre que

recibe el lugar donde se aloja el código de un proyecto de desarrollo en algún lenguaje de programación).

Que es GITHUB?



Github es una plataforma de desarrollo colaborativo de software para **alojar proyectos** usando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una

cuenta de pago. También se pueden obtener repositorios privados (de pago) si se es estudiante.

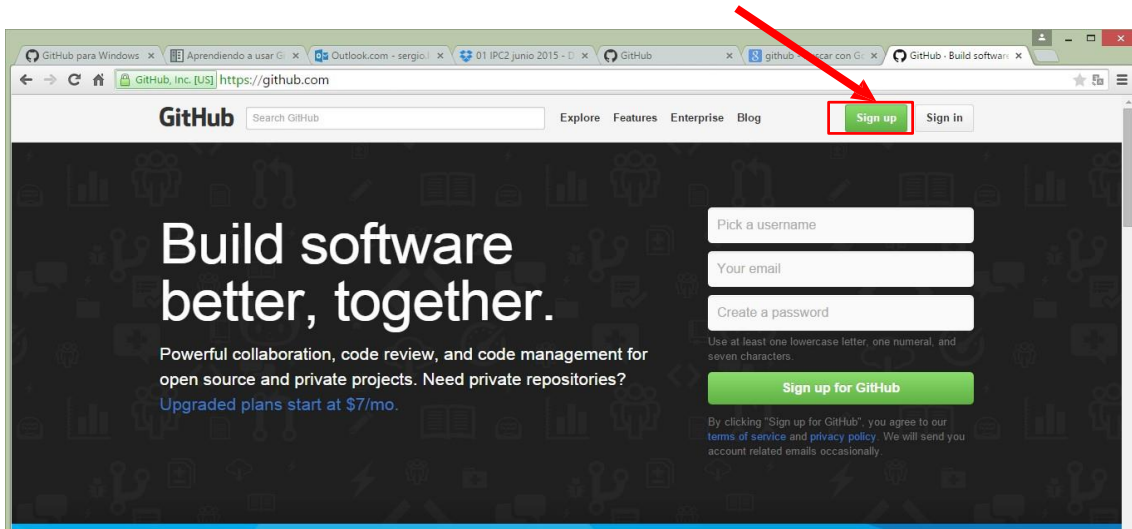
Github no sólo ofrece alojamiento del código si no muchas más posibilidades asociadas a los repos como son, forks, issues, pull requests, diffs, etc. Se verán todos con detalle más adelante.

¿Como crear una cuenta en GitHub?

Paso 1:

Entrar a la página oficial de git.

Clic



Paso 2:

Ingresar llenar el formulario con nuestros datos.

- ☐ Nombre de usuario
- ☐ Dirección de email
- ☐ Contraseña
- ☐ Confirmar contraseña

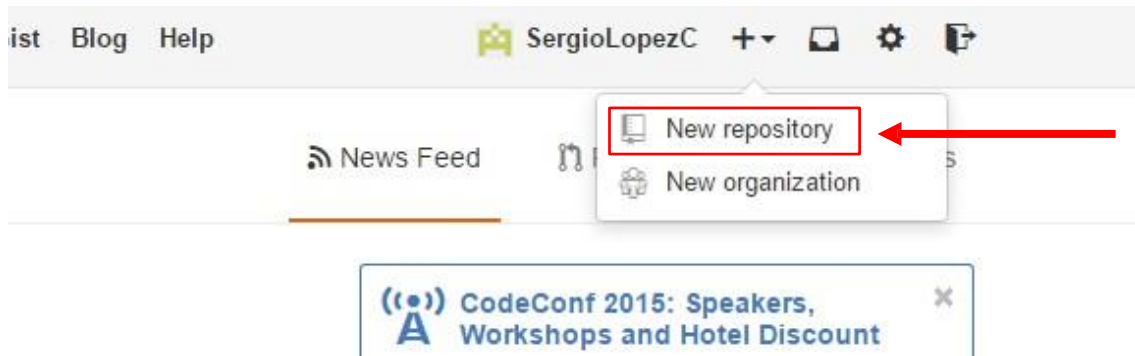
Y por ultimo damos clic en CREATE y listo.

A screenshot of the GitHub account creation form. The browser's address bar shows 'conociendogithub.readthedocs.org/en/latest/_images/gi'. The form is titled 'Create your free personal account' and contains four input fields: 'Username', 'Email Address', 'Password', and 'Confirm Password'. Below the 'Email Address' field, there is a note: 'We promise we won't share your email with anyone.' Below the 'Password' field, there is a note: 'Must contain one lowercase letter, one number, and be at least 7 characters long.' At the bottom of the form, there is a 'Create an account' button, which is circled in red. A red arrow points from the word 'CREATE' in the next block to this button.

CREATE

¿Como crear un repositorio?

Para crear un repositorio en GitHub, solo hay que seleccionar el boton “New Repository”, de la barra de herramientas, habiendo entrada a GitHub con nuestra cuenta.

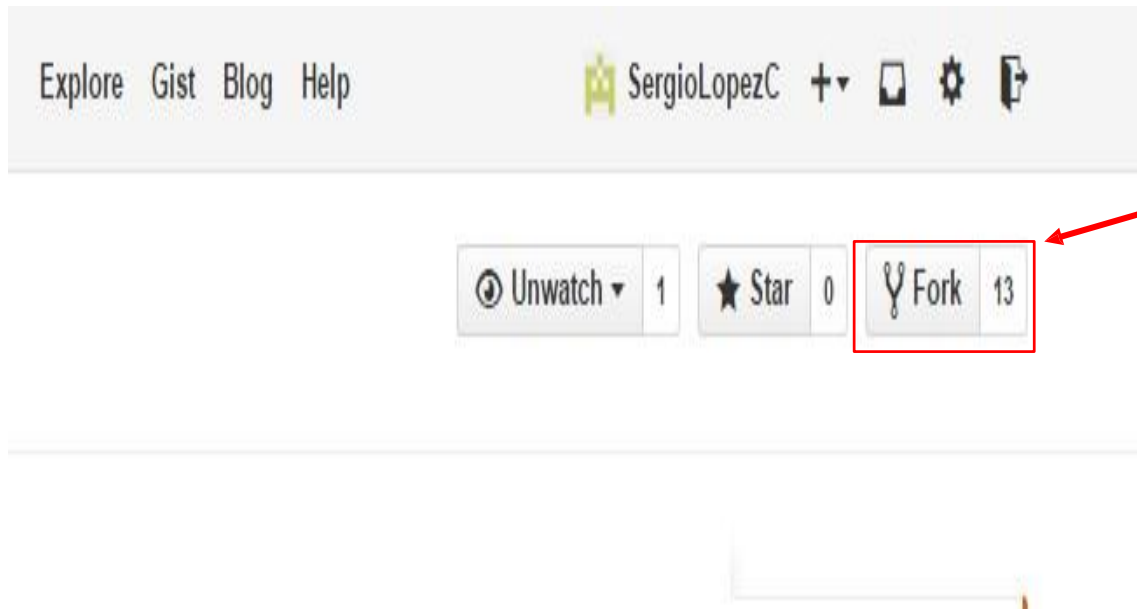


Después solo agregamos el nombre que le queremos dar a nuestro repositorio y si queremos que nuestro repositorio público o privado, y damos clic en “Create Repository”

A screenshot of the GitHub 'Create new repository' form. The form is divided into several sections. The first section is 'Owner', which shows 'SergioLopezC' with a dropdown arrow. The second section is 'Repository name', which has a text input field containing 'myNewRepository' and a green checkmark icon to its right. Below this, there's a text prompt: 'Great repository names are short and memorable. Need inspiration? How about'. The third section is 'Description (optional)', which has a large empty text area. The fourth section is 'Visibility', with two radio buttons: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.' The fifth section is 'Initialize this repository with a README', which has a checkbox that is not checked. Below this, there's a text prompt: 'This will let you immediately clone the repository to your computer. Skip this step'. The sixth section is 'Add .gitignore: None' and 'Add a license: None', both with dropdown arrows. At the bottom, there's a green button labeled 'Create repository', which is highlighted with a red box and a red arrow points to it from the right.

¿Colaborar en un proyecto de otra persona?

Para colaborar en un proyecto ajeno simplemente basta con buscarlo dentro de los repositorios, y luego presionar el botón fork. Esto genera automáticamente una copia del mismo en tu perfil.

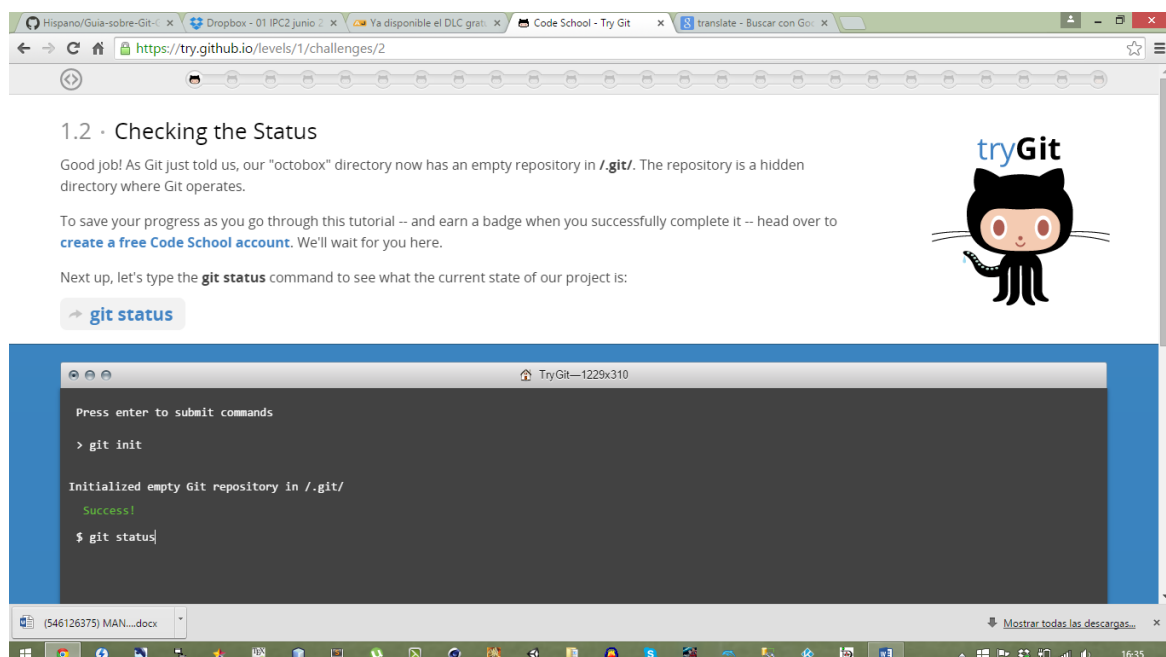
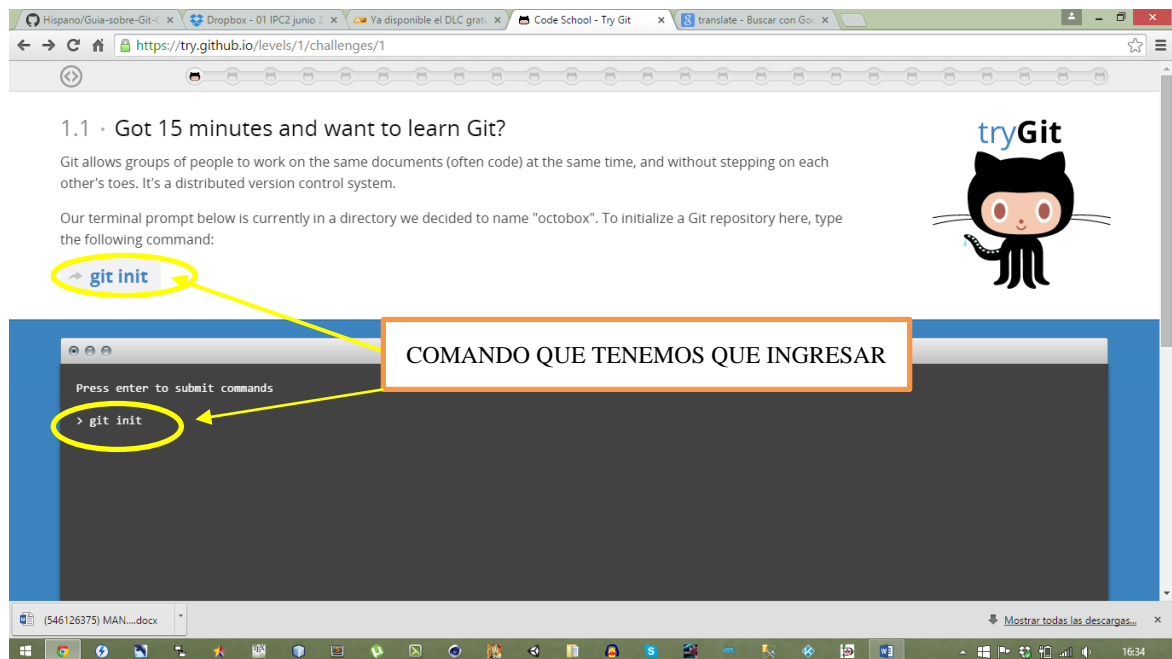


Al terminar las modificaciones se podrá presionar Pull Request para enviárselo al creador del mismo.

COMO UTILIZAR GIT

Para utilizar git basta con seguir los pasos de un tutorial que encontraremos en la página de git <https://try.github.io/levels/1/challenges/1> (25 pasos) los cuales iremos haciendo uno a uno con los cuales aprenderemos cuales son los comandos que utiliza git y cómo utilizar los mismos.

Imagen de algunos de los pasos que tendremos que hacer en el tutorial.



Hispano/Guia-sobre-Git- x Dropbox - 01 IPC2 junio 2 x Ya disponible el DLC grat. x Code School - Try Git x translate - Buscar con Go x


https://try.github.io/levels/1/challenges/3

1.3 · Adding & Committing

I created a file called **octocat.txt** in the octobox repository for you (as you can see in the browser below).

You should run the **git status** command again to see how the repository status has changed:

[git status](#)



```
TryGit—1229x310
Initialized empty Git repository in .git/
Success!
$ git status

# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
Success!
$
```

(546126375) MAN...docx

Mostrar todas las descargas...

16:41

Hispano/Guia-sobre-Git- x Dropbox - 01 IPC2 junio 2 x Ya disponible el DLC grat. x Code School - Try Git x translate - Buscar con Go x


https://try.github.io/levels/1/challenges/4

1.4 · Adding Changes

Good, it looks like our Git repository is working properly. Notice how Git says **octocat.txt** is "untracked"? That means Git sees that **octocat.txt** is a new file.

To tell Git to start tracking changes made to **octocat.txt**, we first need to add it to the staging area by using **git add**.

[git add octocat.txt](#)



```
TryGit—1229x310
Success!
$ git status

# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   octocat.txt
#
nothing added to commit but untracked files present (use "git add" to track)
Success!
$ |
```

(546126375) MAN...docx

Mostrar todas las descargas...

16:41


Hispano/Guia-sobre-Git- x Dropbox - 01 IPC2 junio 2 x Ya disponible el DLC grat. x Code School - Try Git x translate - Buscar con Go x

https://try.github.io/levels/1/challenges/5

1.5 · Checking for Changes

Good job! Git is now tracking our **octocat.txt** file. Let's run **git status** again to see where we stand:

[git status](#)



```
TryGit—1229x310
$ git add
Nothing specified, nothing added.
Maybe you wanted to say "git add ."?
Did not add octocat.txt
$ git add octocat.txt

Nice job, you've added octocat.txt to the Staging Area
$
```

(546126375) MAN...docx

Mostrar todas las descargas...

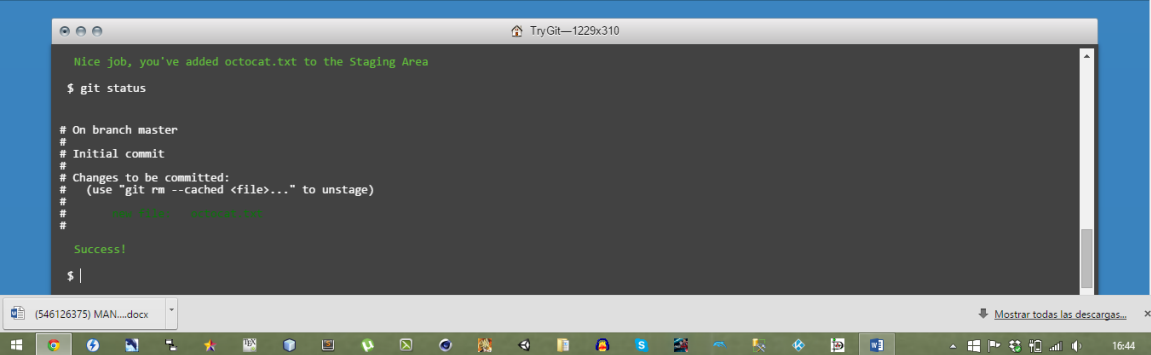

16:43

1.6 · Committing

Notice how Git says **changes to be committed**? The files listed here are in the **Staging Area**, and they are not in our repository yet. We could add or remove files from the stage before we store them in the repository.

To store our staged changes we run the **commit** command with a message describing what we've changed. Let's do that now by typing:

```
git commit -m "Add cute octocat story"
```



```
Nice job, you've added octocat.txt to the Staging Area
$ git status

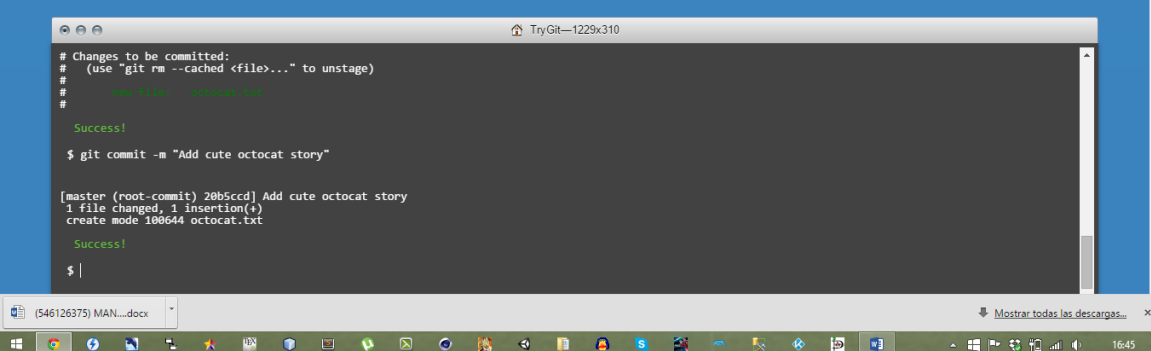

# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   octocat.txt
#
Success!
$ |
```

1.7 · Adding All Changes

Great! You also can use wildcards if you want to add many files of the same type. Notice that I've added a bunch of .txt files into your directory below.

I put some in a directory named "octofamily" and some others ended up in the root of our "octobox" directory. Luckily, we can add all the new files using a wildcard with **git add**. Don't forget the quotes!

```
git add '*.txt'
```



```
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   octocat.txt
#
Success!
$ git commit -m "Add cute octocat story"

[master (root-commit) 20b5ccd] Add cute octocat story
1 file changed, 1 insertion(+)
create mode 100644 octocat.txt
Success!
$ |
```

Cada uno de los pasos viene con el procedimiento que se debe de seguir, al acabar sabremos como utilizar cada comando y empezar a trabajar con ellos.

Ventajas de utilizar un control de Versiones

- Comparar el código de un archivo, de modo que podamos ver las diferencias entre versiones
- Restaurar versiones antiguas
- Fusionar cambios entre distintas versiones
- Trabajar con distintas ramas de un proyecto, por ejemplo la de producción y desarrollo