

## Tutorial para instalación Git

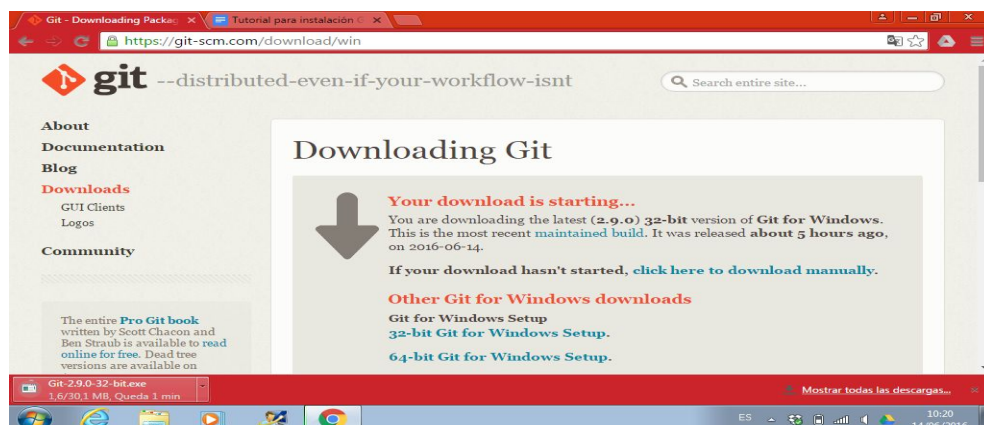
git es una herramienta de control de versiones por medio de la cual se pueden gestionar proyectos, en ella se pueden crear branches donde cada uno es una versión del proyecto que podemos modificar y si nos satisfacen los cambios realizar merge con el master, esto nos garantiza que cualquier cambio siempre podamos volver a un punto donde nuestro proyecto era totalmente funcional.

para instalacion debemos seguir los siguientes pasos

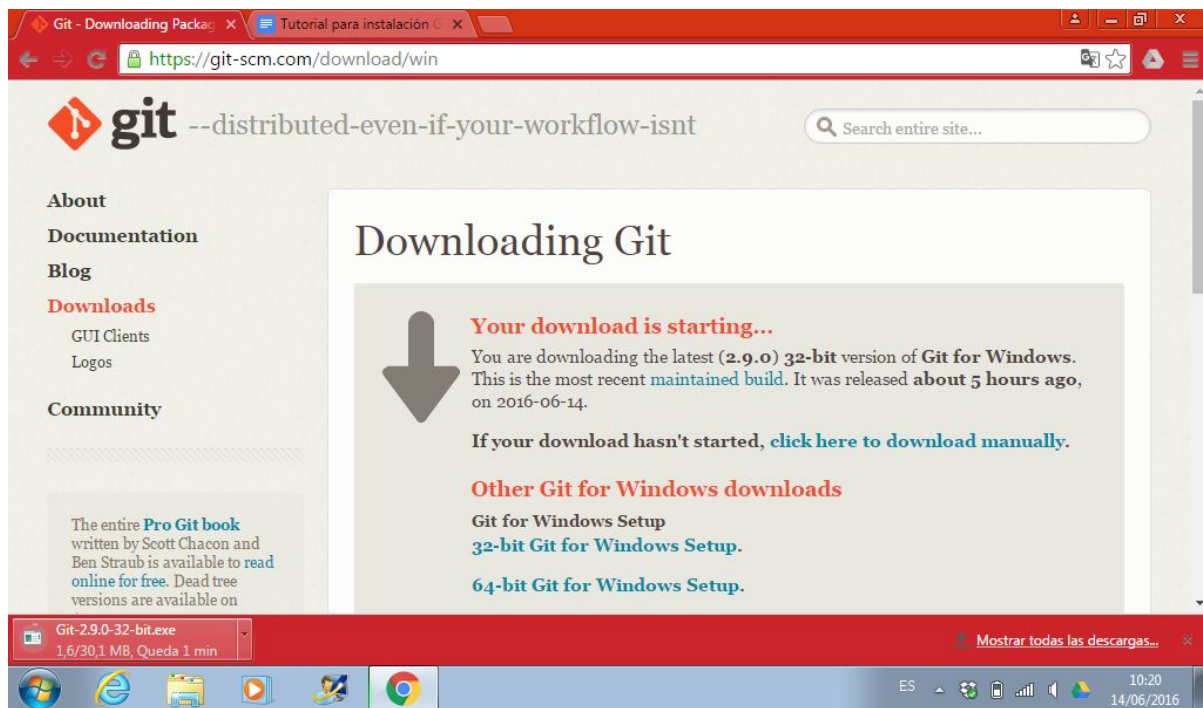
1. primero debemos realizar la descarga de git, esto lo encontraremos en el siguiente link <https://git-scm.com/> ya en esta pagina localizamos el instalador en mi caso lo instalare sobre windows 7



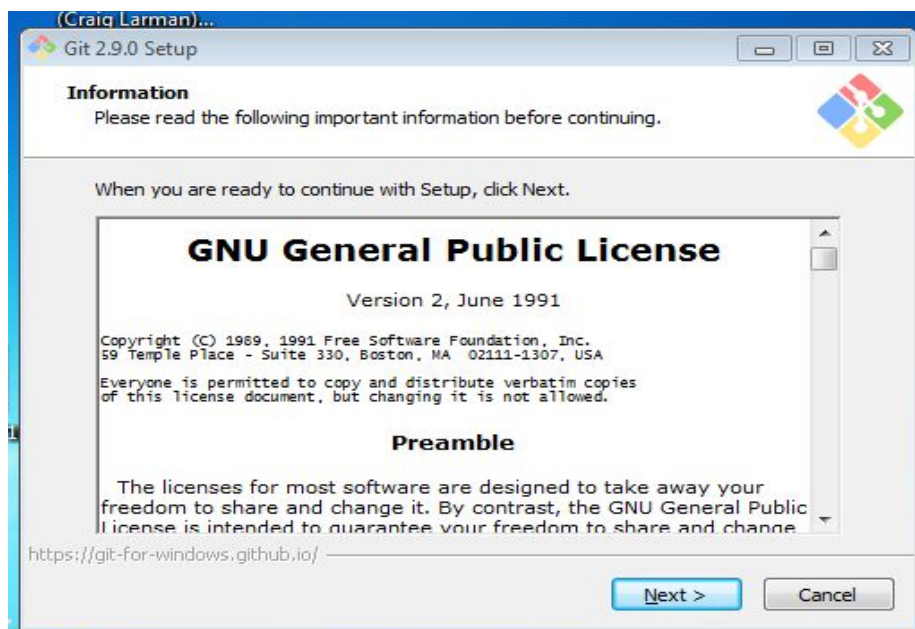
2. la descarga comienza



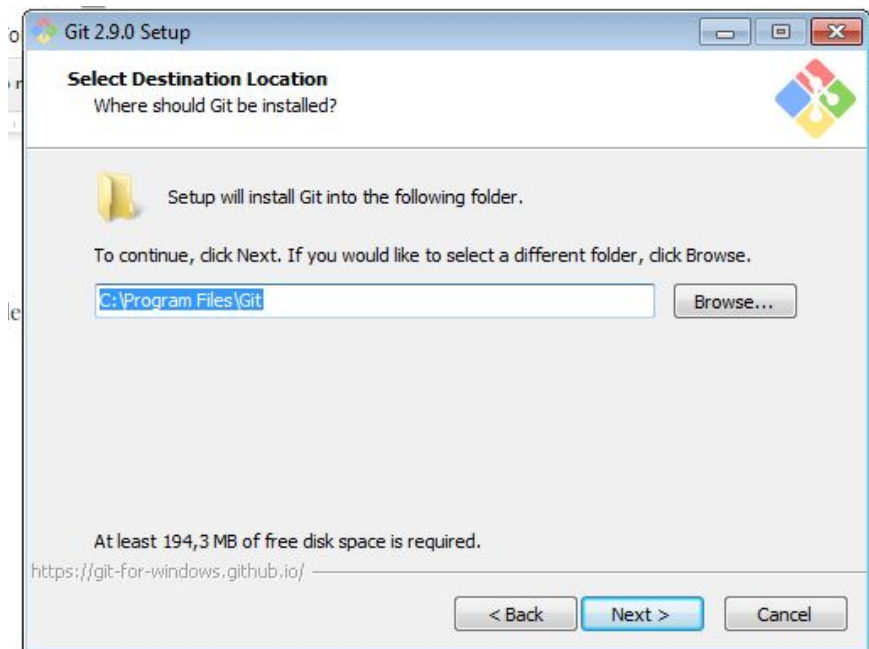
3. ejecutamos el archivo .exe descargado



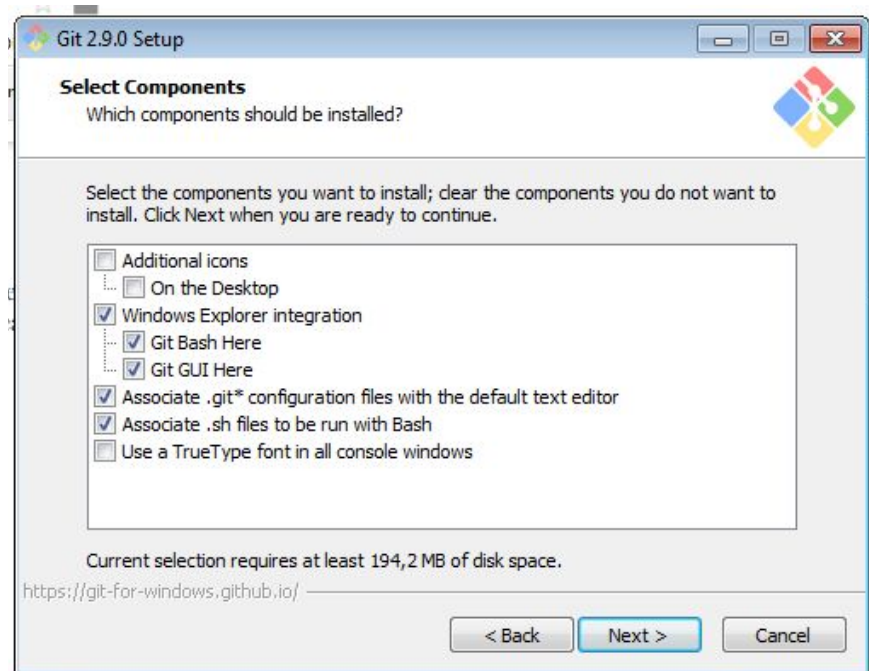
4. aceptamos licencia y clickeamos siguiente



5. seleccionamos carpeta donde lo instalaremos por simplicidad lo dejare en la de default clickeamos siguiente



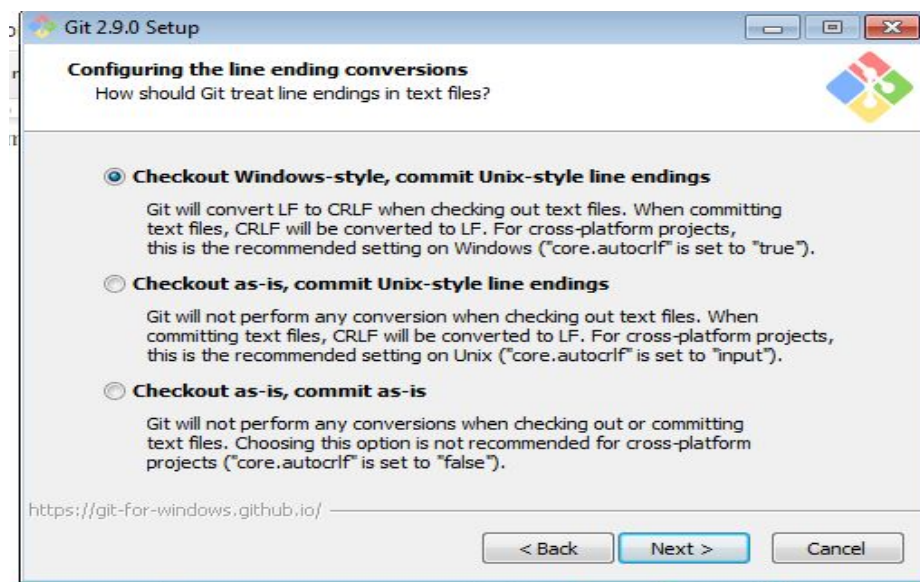
6. seleccionamos los componenetes a instalar en mi caso instalare los accesos directos a la linea de comandos de git



7. en esta parte elegire la primera opcion debido a que no deseo poder utilizar los comandos de git directamente desde el prompt de windows ya que en el paso anterior elegimos crear un acceso directo para manejar los comandos desde alli



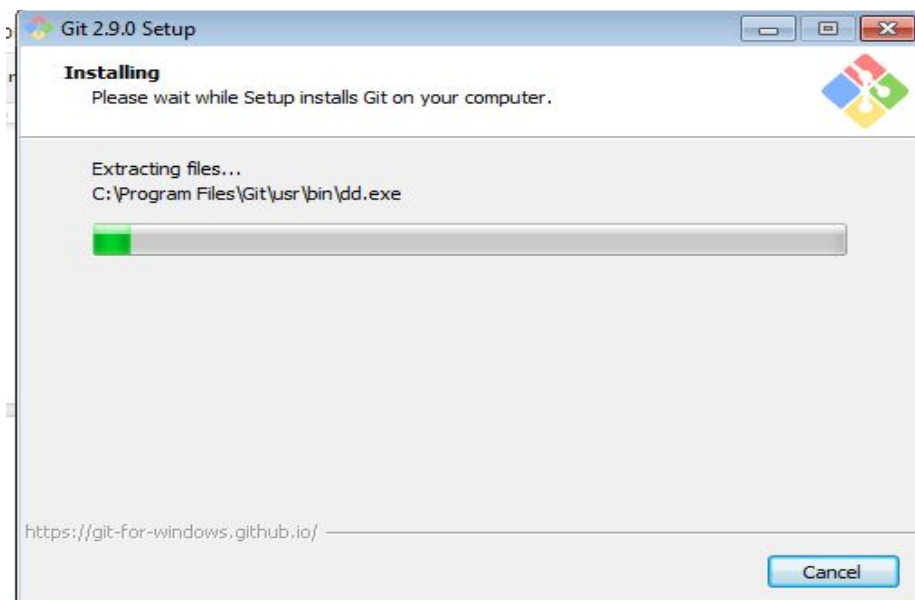
8. seleccionamos el estilo de linea, en esta ocasión dejare las siguientes opciones





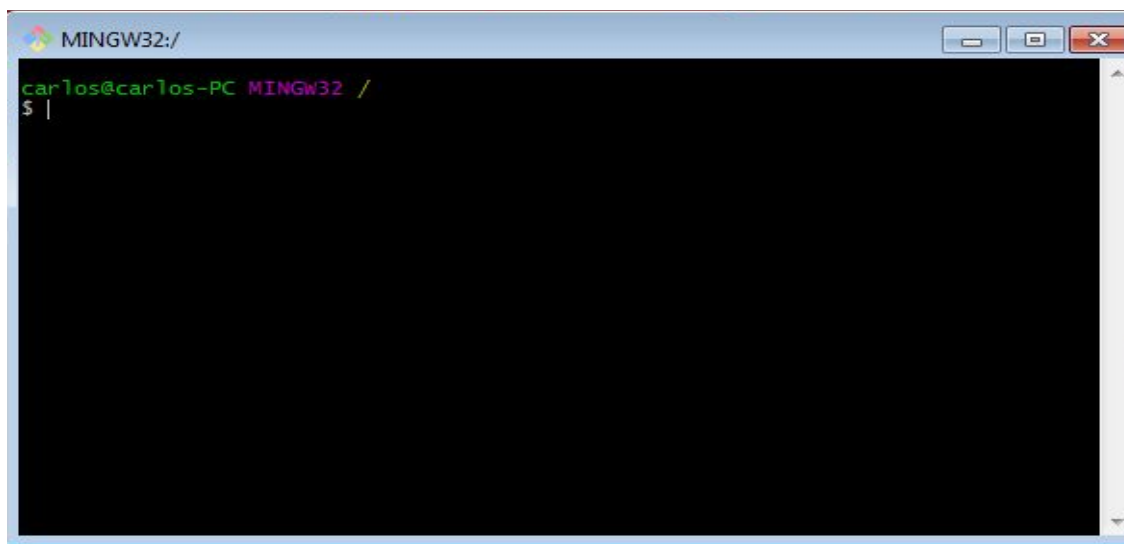


9. clickeamos siguiente e instalamos la aplicación





10. después si la instalacion fue exitosa nos mostrara la entrada de comandos de git

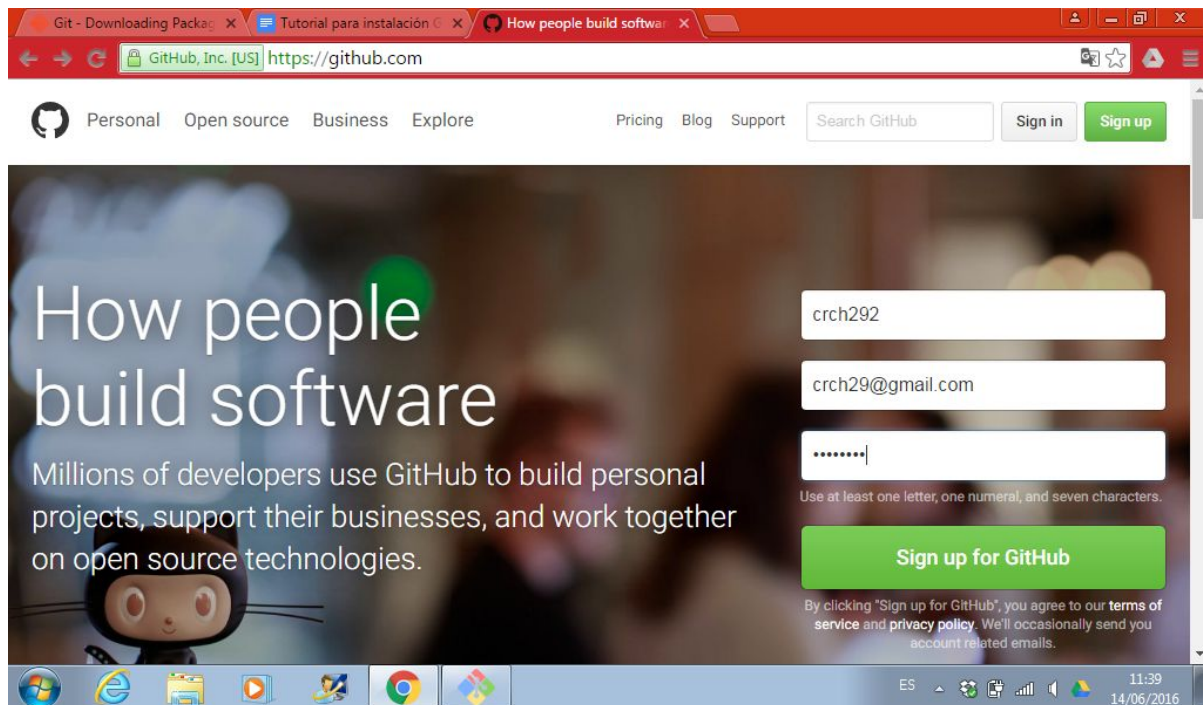


## Utilización de Git

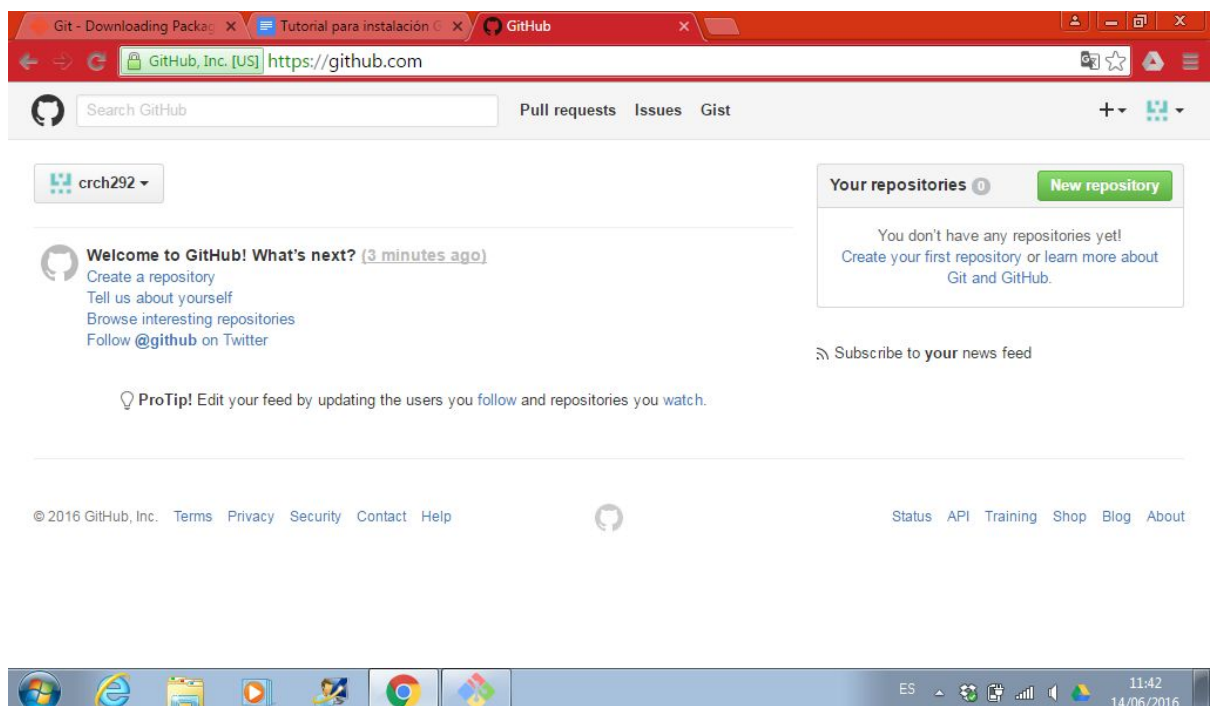
Ahora veremos como utilizar git, esto lo podemos realizar de 2 formas, la primera utilizando la linea de comandos de git y la otra por medio de un cliente que podemos descargar dependiendo del sistema operativo que utilicemos, en este tutorial utilizaremos la linea de comandos

1. lo primero que haremos sera crear una cuenta en github que es donde crearemos nuestro repositorio remoto el cual estaremos modificando con nuestro proyecto,

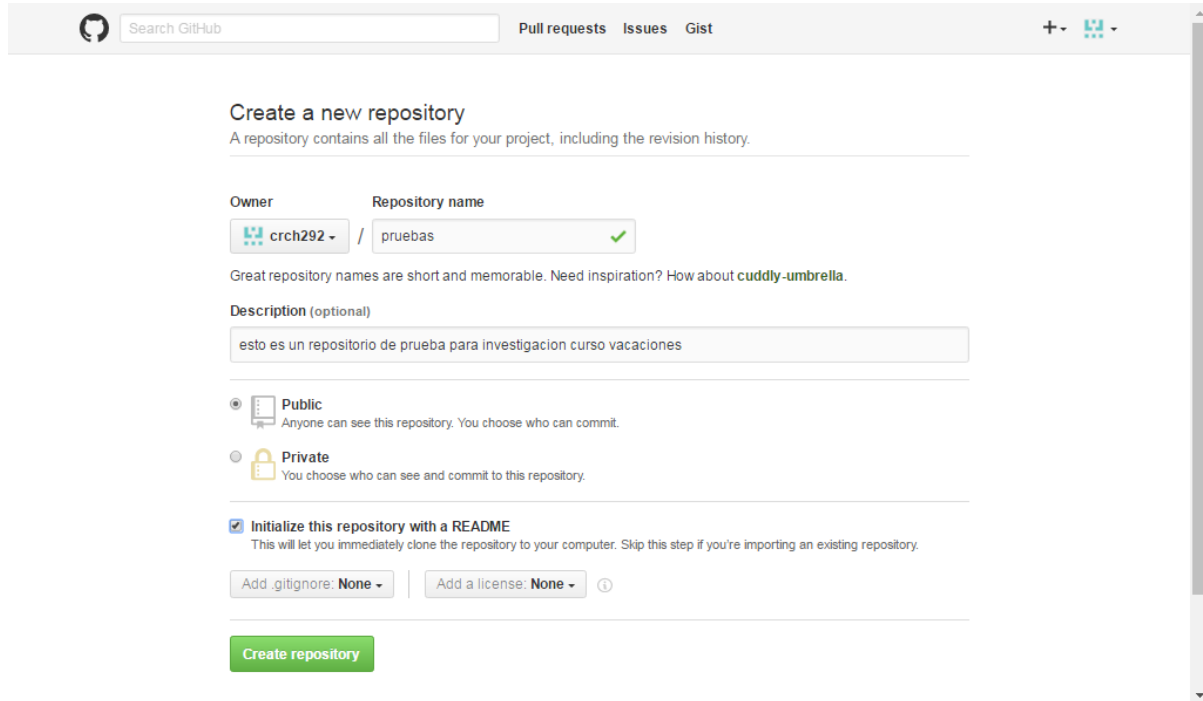
para crear la cuenta ingresamos a <https://github.com/> lo unico que debemos hacer es ingresar nuestro nickname, colocar nuestro correo electronico y creamos nuestra contraseña



2. Ya habiendo colocado los datos creamos nuestra cuenta, tendremos esto donde nos preguntara que tipo de cuenta deseamos, tendremos opcion gratuita y opcion de pago, en mi caso utilizaremos gratuito y terminamos el registro, nos aparecera ya la pagina principal de nuestro perfil

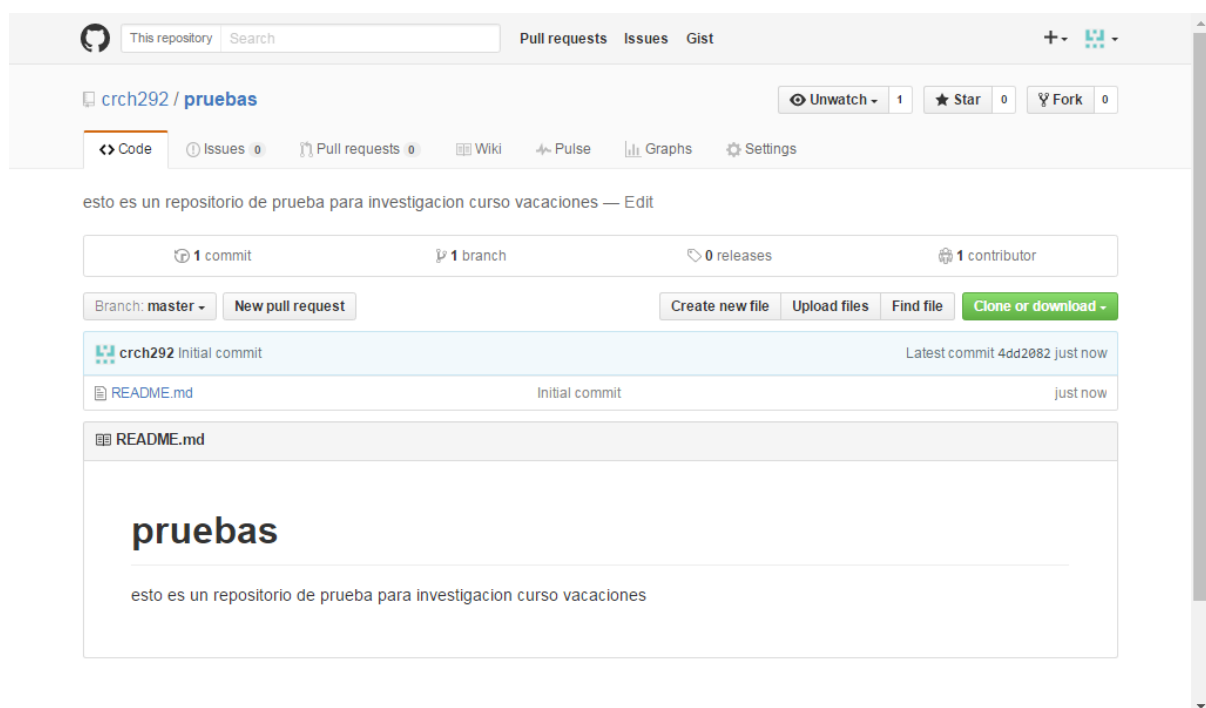


3. ahora crearemos un repositorio que utilizaremos para guardar los cambios de nuestro proyecto, en este ejemplo crearemos el repositorio pruebas, seleccionamos new repository y llenamos los campos, se agrega descripcion y seleccionamos agregar un archivo readme



The screenshot shows the 'Create a new repository' page on GitHub. At the top, there's a search bar and navigation links for 'Pull requests', 'Issues', and 'Gist'. The main heading is 'Create a new repository' with a subtext: 'A repository contains all the files for your project, including the revision history.' Below this, there are two input fields: 'Owner' (set to 'crch292') and 'Repository name' (set to 'pruebas' with a green checkmark). A note says: 'Great repository names are short and memorable. Need inspiration? How about **cuddly-umbrella**.' There's a 'Description (optional)' text area containing 'esto es un repositorio de prueba para investigacion curso vacaciones'. Below the description, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' There's a checked checkbox for 'Initialize this repository with a README' with a subtext: 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' Below this, there are two dropdown menus: 'Add .gitignore: None' and 'Add a license: None'. At the bottom, there's a green 'Create repository' button.

4. listo ahora ya tenemos nuestro repositorio en github creado



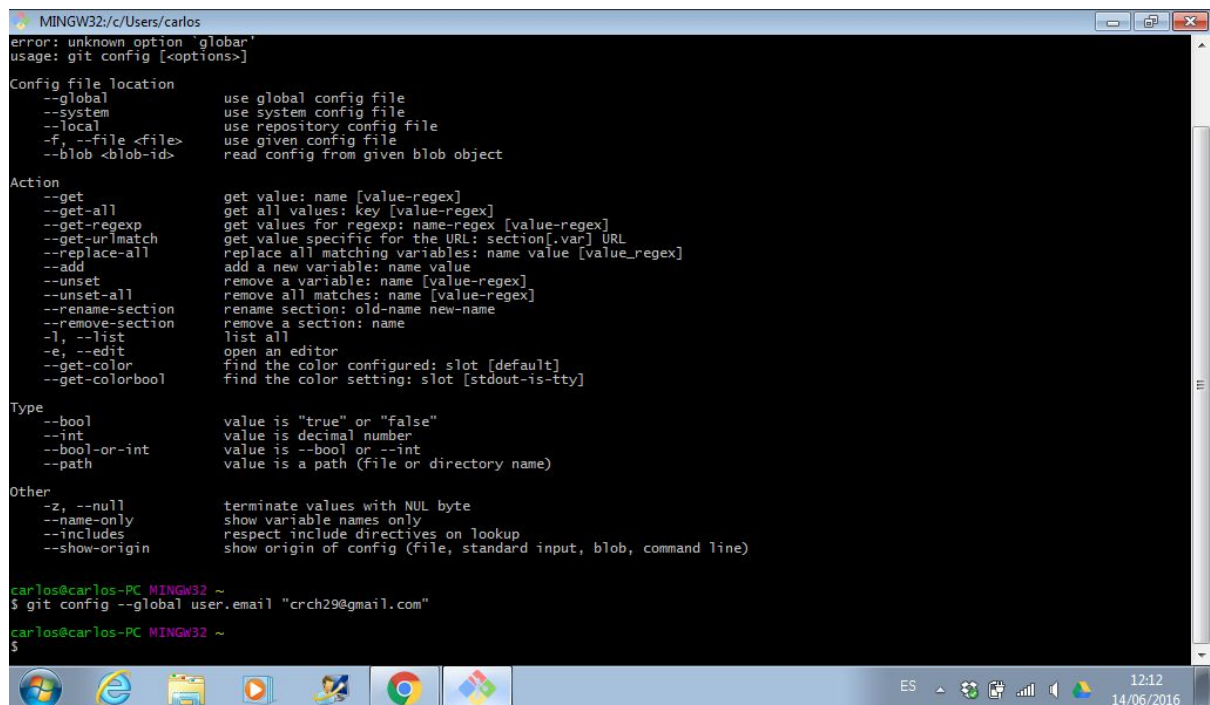
The screenshot shows the GitHub repository page for 'crch292 / pruebas'. At the top, there's a search bar and navigation links for 'Pull requests', 'Issues', and 'Gist'. Below the repository name, there are buttons for 'Unwatch', 'Star', and 'Fork'. There are also buttons for 'Code', 'Issues', 'Pull requests', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The repository description is 'esto es un repositorio de prueba para investigacion curso vacaciones'. Below this, there are statistics: '1 commit', '1 branch', '0 releases', and '1 contributor'. There are also buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. Below the statistics, there's a table showing the repository's history. The table has two columns: 'Commit' and 'Date'. The first row shows 'crch292 Initial commit' and 'Latest commit 4dd2082 just now'. Below the table, there's a section for the 'README.md' file. The file content is 'pruebas' followed by 'esto es un repositorio de prueba para investigacion curso vacaciones'.



5. ahora creamos una carpeta en nuestra computadora preferiblemente con el nombre de nuestro repositorio en este caso pruebas, y ejecutamos git, ya estando en ventana de comandos, y configuramos las variables globales con los siguientes comandos

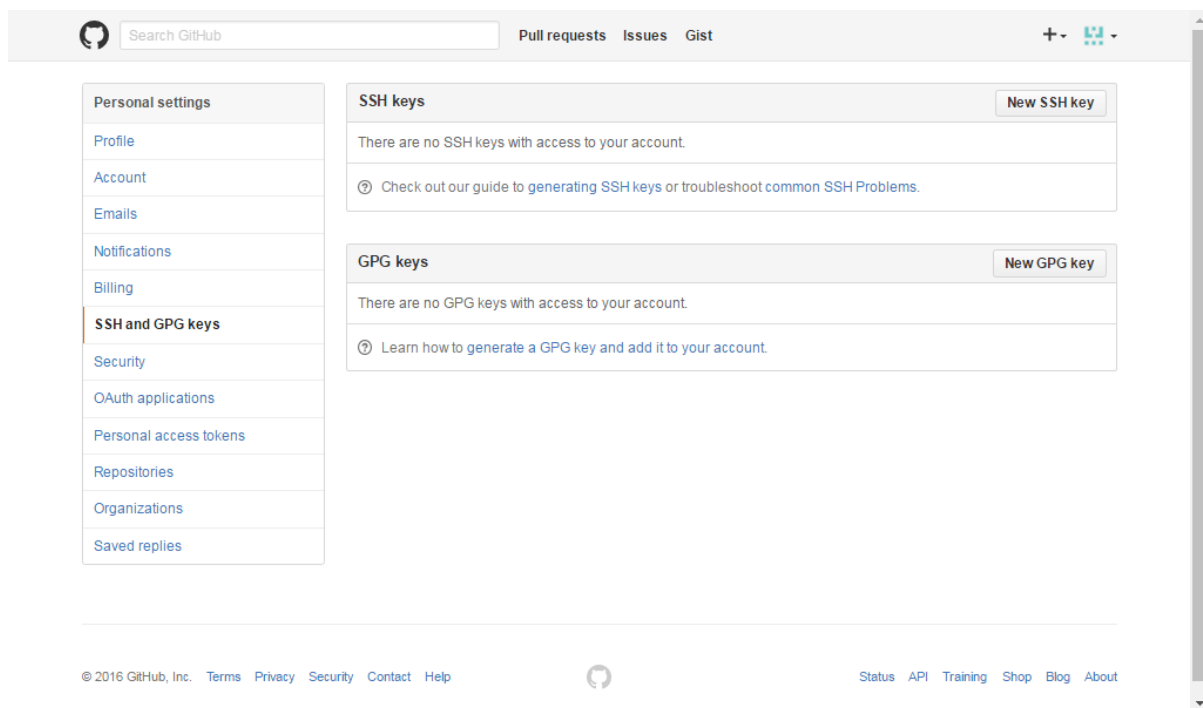
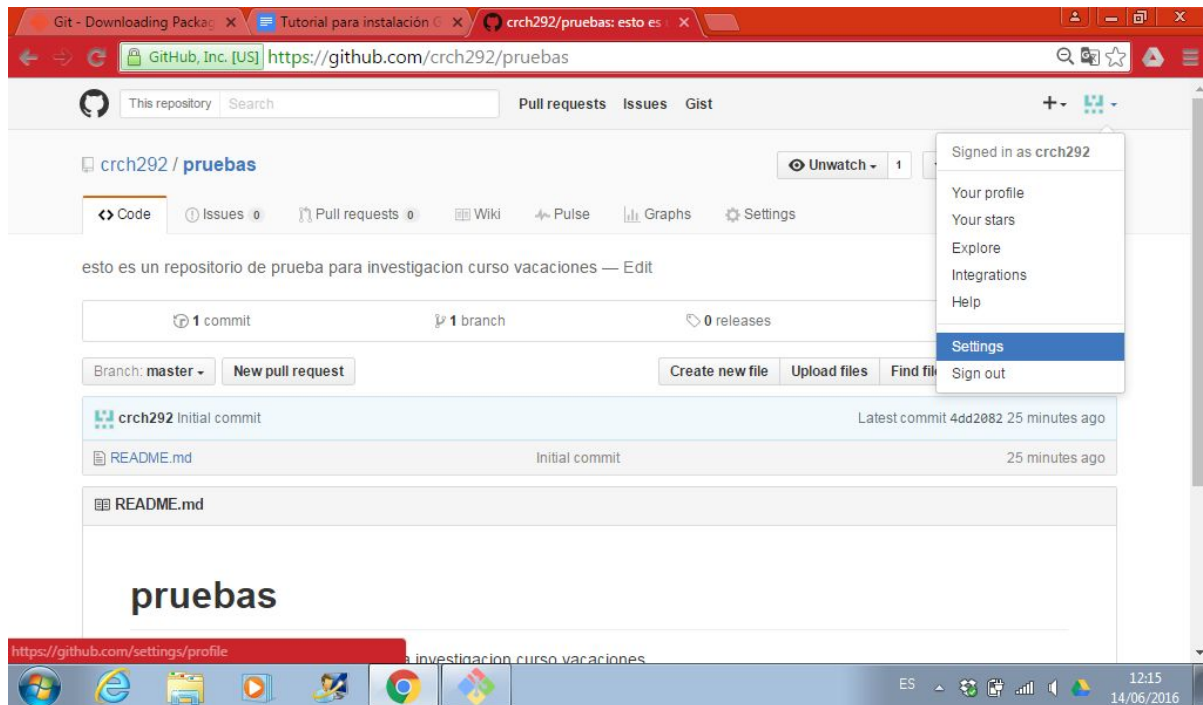
```
git config --global user.name "crch292"  
git config --global user.email "crch29@gmail.com"
```

esto configurara git con nuestro nombre de usuario y correo electronico



```
error: unknown option 'global'  
usage: git config [<options>]  
  
Config file location  
--global          use global config file  
--system          use system config file  
--local           use repository config file  
-f, --file <file> use given config file  
--blob <blob-id>  read config from given blob object  
  
Action  
--get             get value: name [value-regex]  
--get-all        get all values: key [value-regex]  
--get-regexp      get values for regexp: name-regex [value-regex]  
--get-urlmatch    get value specific for the URL: section[.var] URL  
--replace-all    replace all matching variables: name value [value_regex]  
--add            add a new variable: name value  
--unset          remove a variable: name [value-regex]  
--unset-all      remove all matches: name [value-regex]  
--rename-section  rename section: old-name new-name  
--remove-section remove a section: name  
-l, --list        list all  
-e, --edit        open an editor  
--get-color       find the color configured: slot [default]  
--get-colorbool   find the color setting: slot [stdout-is-tty]  
  
Type  
--bool            value is "true" or "false"  
--int             value is decimal number  
--bool-or-int     value is --bool or --int  
--path            value is a path (file or directory name)  
  
Other  
-z, --null        terminate values with NUL byte  
--name-only       show variable names only  
--includes        respect include directives on lookup  
--show-origin     show origin of config (file, standard input, blob, command line)  
  
carlos@carlos-PC MINGW32 ~  
$ git config --global user.email "crch29@gmail.com"  
carlos@carlos-PC MINGW32 ~  
$
```

6. ya configurado correo y usuario, crearemos la conexion de nuestra carpeta con nuestro repositorio remoto, crearemos nuestra llave ssh, ingresamos a settings de nuestro perfil de github y aparecera la siguiente pantalla



7. regresamos a nuestro terminal e ingresamos el siguiente comando  
ssh-keygen

nos genera el archivo y nos pide donde guardar la llave, simplemente damos enter, luego nos pide una contraseña, la colocamos y la confirmamos y listo ya tenemos nuestra llave

```
MINGW32/c/Users/carlos/desktop/pruebas
carlos@carlos-PC MINGW32 ~
$ cd desktop
carlos@carlos-PC MINGW32 ~/desktop
$ cd pruebas
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/carlos/.ssh/id_rsa):
Created directory '/c/Users/carlos/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/carlos/.ssh/id_rsa.
Your public key has been saved in /c/Users/carlos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HE6ppvZc/5/y72Q1oHkbeEYJ57XKaplyTE34a4HR3Bc carlos@carlos-PC
The key's randomart image is:
+--[RSA 2048]-----+
|      .+o.. E.      |
|      +^oo...       |
|      +.+  ..       |
|      =oo~oo        |
|      o S+ Bo.       |
|      o o Bo. . .    |
|      o ..=. . .o    |
|      . o ... . +    |
|      o ... ..+oo    |
|      o ... ..+oo    |
+-----[SHA256]-----+
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ |
```

8. ya tenemos una llave que identifica a nuestro equipo ahora se la daremos a github para generar la conexion, agregamos el siguiente comando

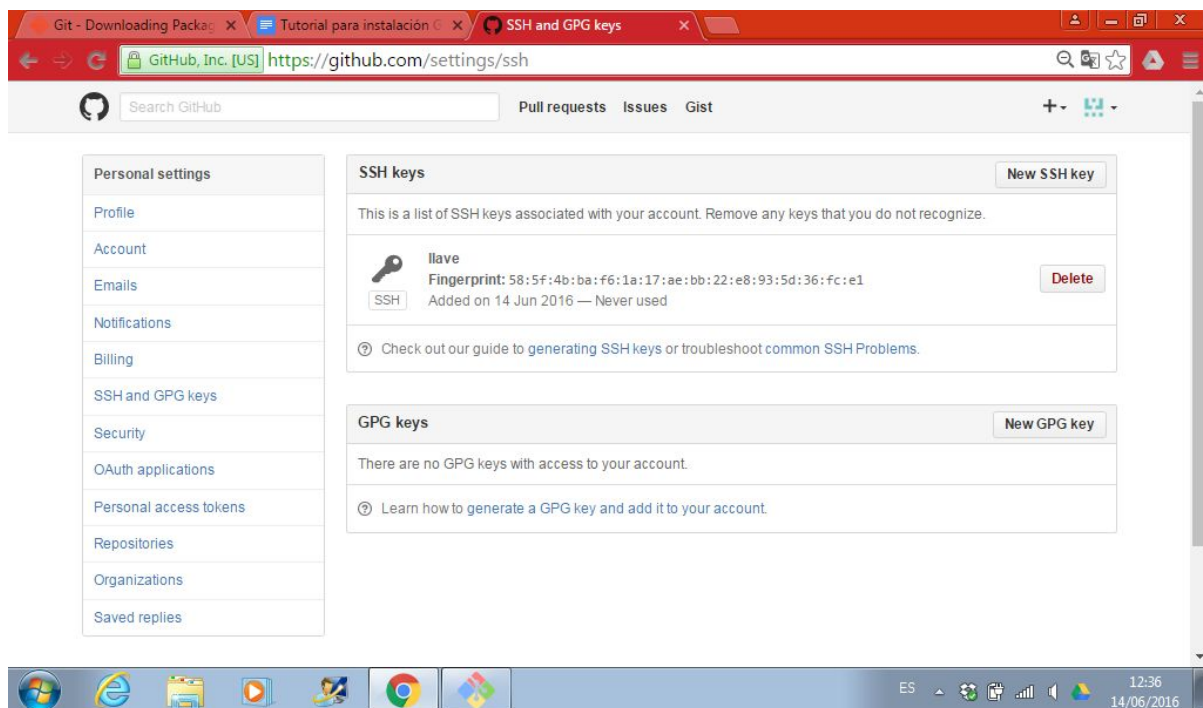
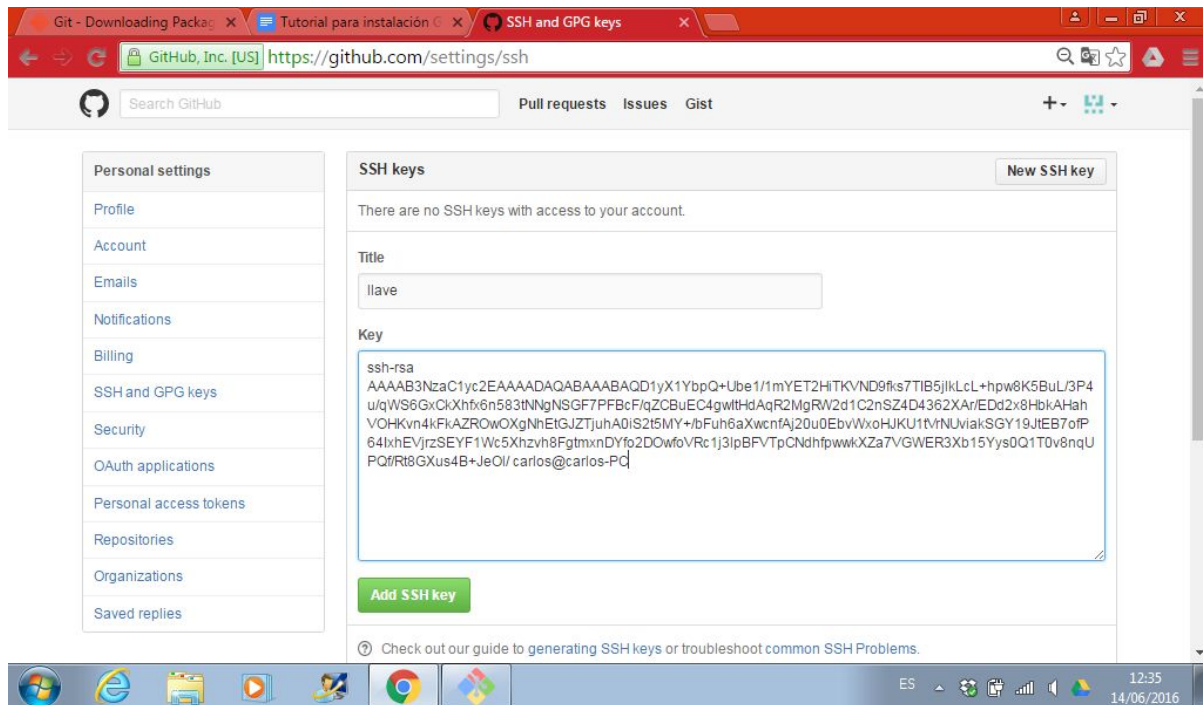
```
cat ~/.ssh/id_rsa.pub
```

esto nos generara la llave que es un conjunto de valores alfanumericos que copiaremos

```
MINGW32:/c/Users/carlos/desktop/pruebas
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/carlos/.ssh/id_rsa):
Created directory '/c/Users/carlos/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/carlos/.ssh/id_rsa.
Your public key has been saved in /c/Users/carlos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HE6ppvZC/5/y72QJ0hKbeEYJ57XKaplYTE34a4HR3Bc carlos@carlos-PC
The key's randomart image is:
+-----[RSA 2048]-----+
|      .+o.. E.      |
|      +*oo...      |
|      +.*+ ..      |
|      =oo*oo        |
|      o S* Bo. .    |
|      o o Boo . .   |
|      o ..... .o    |
|      .o .... .+    |
|      o ....+oo     |
+-----[SHA256]-----+
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ ^C
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa
cat: /c/Users/carlos/.ssh/id_rsa: No such file or directory
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa.
cat: /c/Users/carlos/.ssh/id_rsa.: No such file or directory
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQD1yX1YbpQ+Ube1/1mYET2HiTKVND9Fks7TI85j1kLcL+hpw8K5BuL/3P4u/qwS6GxCkXhfx6n583tNNgNSGF7PF8cF/qZCBuEC4gwItHd
AqR2MgRw2d1C2nS24D4362XAr/EDd2x8HbkAHahVOHKvn4kFkAZR0wOxgNhEtGJ2TjuhA0iS2t5MY+/bFuh6aXwcnfAj20u0EbvWxoHJKU1tVrNUViak5GY19JtEB7ofP64IxEVjr2SEYF
1Wc5Xhzv8FgtmxnDYfo2D0wFovRc1j31pBFVTpCNDhfpwmkXZa7VGWER3Xb15Yys0Q1T0v8nqUPQf/Rt8GXus4B+Je01/ carlos@carlos-PC
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$
```

```
MINGW32:/c/Users/carlos/desktop/pruebas
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/carlos/.ssh/id_rsa):
Created directory '/c/Users/carlos/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/carlos/.ssh/id_rsa.
Your public key has been saved in /c/Users/carlos/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:HE6ppvZC/5/y72QJ0hKbeEYJ57XKaplYTE34a4HR3Bc carlos@carlos-PC
The key's randomart image is:
+-----[RSA 2048]-----+
|      .+o.. E.      |
|      +*oo...      |
|      +.*+ ..      |
|      =oo*oo        |
|      o S* Bo. .    |
|      o o Boo . .   |
|      o ..... .o    |
|      .o .... .+    |
|      o ....+oo     |
+-----[SHA256]-----+
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ ^C
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa
cat: /c/Users/carlos/.ssh/id_rsa: No such file or directory
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa.
cat: /c/Users/carlos/.ssh/id_rsa.: No such file or directory
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQD1yX1YbpQ+Ube1/1mYET2HiTKVND9Fks7TI85j1kLcL+hpw8K5BuL/3P4u/qwS6GxCkXhfx6n583tNNgNSGF7PF8cF/qZCBuEC4gwItHd
AqR2MgRw2d1C2nS24D4362XAr/EDd2x8HbkAHahVOHKvn4kFkAZR0wOxgNhEtGJ2TjuhA0iS2t5MY+/bFuh6aXwcnfAj20u0EbvWxoHJKU1tVrNUViak5GY19JtEB7ofP64IxEVjr2SEYF
1Wc5Xhzv8FgtmxnDYfo2D0wFovRc1j31pBFVTpCNDhfpwmkXZa7VGWER3Xb15Yys0Q1T0v8nqUPQf/Rt8GXus4B+Je01/ carlos@carlos-PC
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$
```

9. ya copiada la llave nos dirigimos a nuestro perfil de github y a la opcion de ssh key en settings



10. ya creada la llave ya podemos trabajar y agregar archivos, crearemos un archivo de texto en blanco con nombre uno y crearemos el repositorio, para realizar repositorio en la carpeta colocamos comandos

git init

el cual crea el repositorio



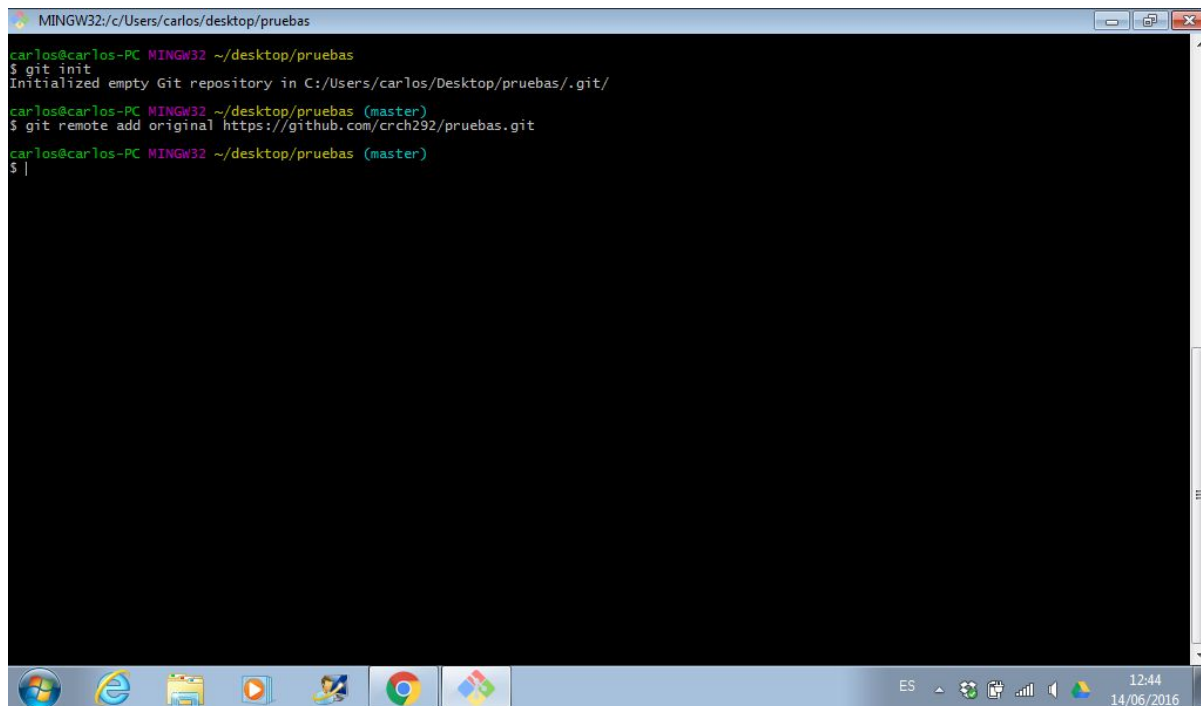
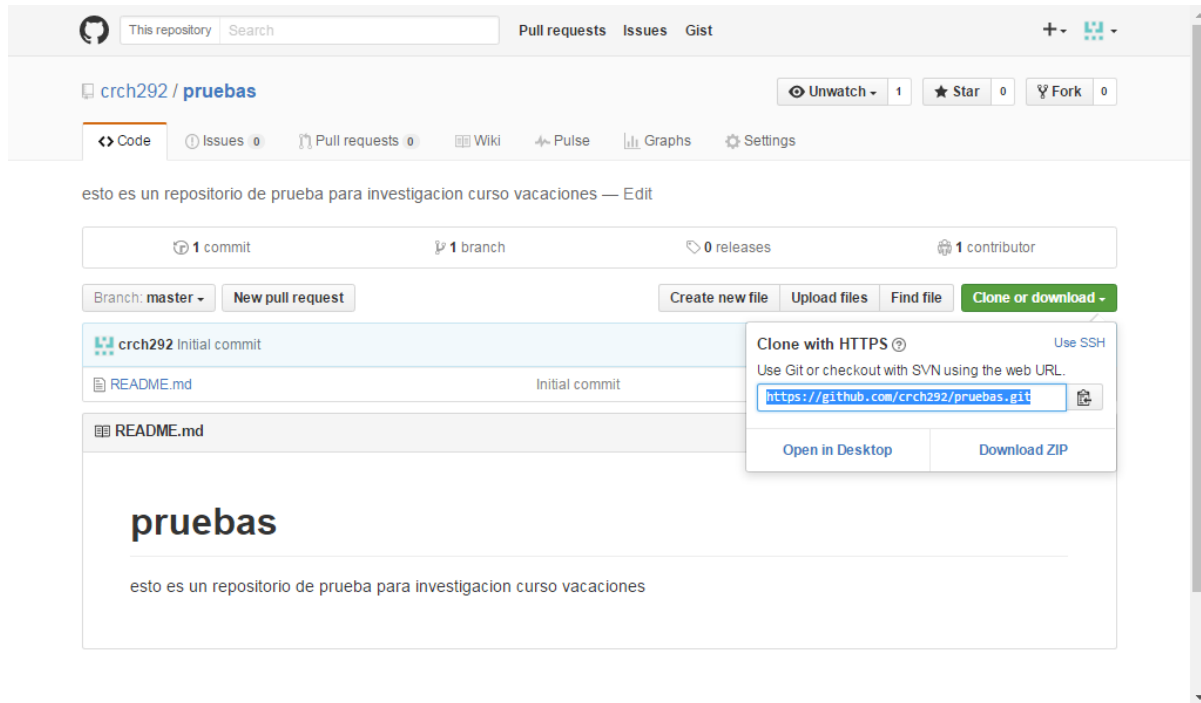
Procederemos a clonar nuestro repositorio que tenemos en github nos dirigimos a nuestro perfil seleccionamos clone or download, lo copiamos y ya en nuestra ventana de git colocamos

Git clone <https://github.com/crch292/pruebas.git> con esto traeremos todo lo que ya tenemos en github a nuestra carpeta

y agregaremos la conexión remota con el siguiente comando

git add remote original <https://github.com/crch292/pruebas.git>

donde la dirección la obtenemos de nuestro repositorio seleccionando opción clone or download



11. ahora crearemos nuestro archivo, lo agregamos a git, realizamos commit y enviamos los cambios con los siguientes comandos

touch archivo

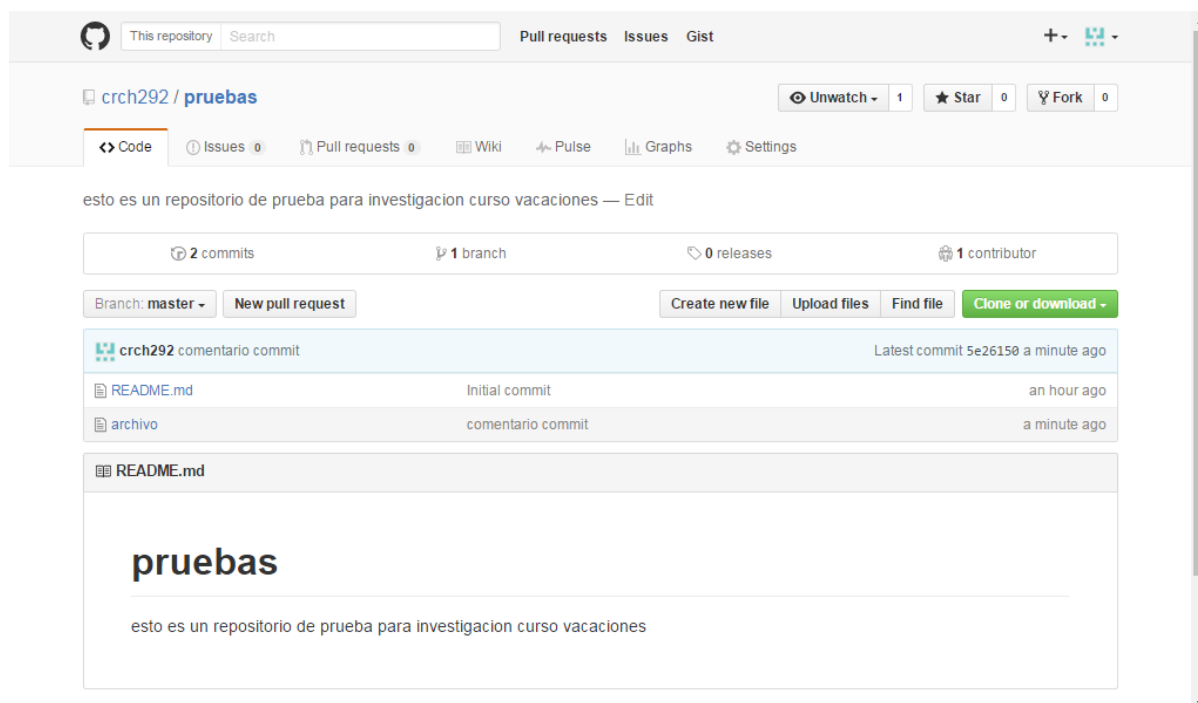
git add archivo

git commit -m "este es el comentario del commit"

git push original master

```
MINGW32: c:/Users/carlos/desktop/pruebas
carlos@carlos-PC MINGW32 ~/desktop/pruebas
$ git init
Initialized empty Git repository in C:/Users/carlos/Desktop/pruebas/.git/
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ git remote add original https://github.com/crch292/pruebas.git
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ touch archivo
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ git add archivo
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ git status
On branch master
Initial commit
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   archivo
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ git commit -m "este es el comentario del commit"
[master (root-commit) 512b356] este es el comentario del commit
1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 archivo
carlos@carlos-PC MINGW32 ~/desktop/pruebas (master)
$ git push original master
```

ya nos aparece cambio



ademas de esto tenemos con git la posibilidad de manejar branch que no es mas que una rama dentro de nuestro proyecto en la cual podemos realizar los cambios necesarios sin comprometer el proyecto en si, ya que cualquier cambio sera aplicado al branch y cuando ya estemos seguros que es correcto realizamos un merge que no es mas que

aplicar los cambios a la rama principal llamada master, ademas de estas ventajas podemos manejar colaboraciones y verificar los cambios dentro los archivos y que usuario las genero.

Existen muchas características mas de git, pero aca solo usamos las principales, ademas tambien podemos mencionar que existen muchas aplicaciones clientes que podemos utilizar para no trabajar directamente desde codigo, github tiene una que se puede utilizar y es gratuita.

## **Comentarios acerca de git**

En mi opinion git es una herramienta muy poderosa para manejar versiones de proyecto ya que es una bitacora donde podemos ver que usuarios realizaron los cambios y hace cuanto tiempo, podemos colaborar con otro proyectos, podemos generar copia de proyectos que alli se publican y realizar cambios, cuenta con una comunidad en crecimiento, podemos crear ramas de desarrollo y ramas de produccion donde las pruebas las realizamos en el branch desarrollo y hacemos merge con branch produccion.

## **Patrones de Diseño**

Los patrones de diseño son soluciones a problemas tipicos y recurrentes al momento de desarrollar una aplicacion.

los tipos de patrones que existen son los siguientes

- patrones creacionales: utilizados para instanciar objetos, y así separar la implementación del cliente de la de los objetos que se utilizan, ejemplos singleton y prototype.
- patrones comportamiento: se utilizan a la hora de definir como las clases y objetos interaccionan entre ellos, ejemplos observer
- Patrones estructurales: utilizados para crear clases u objetos que incluidos dentro de estructuras más complejas ejemplos bridge y Decorator