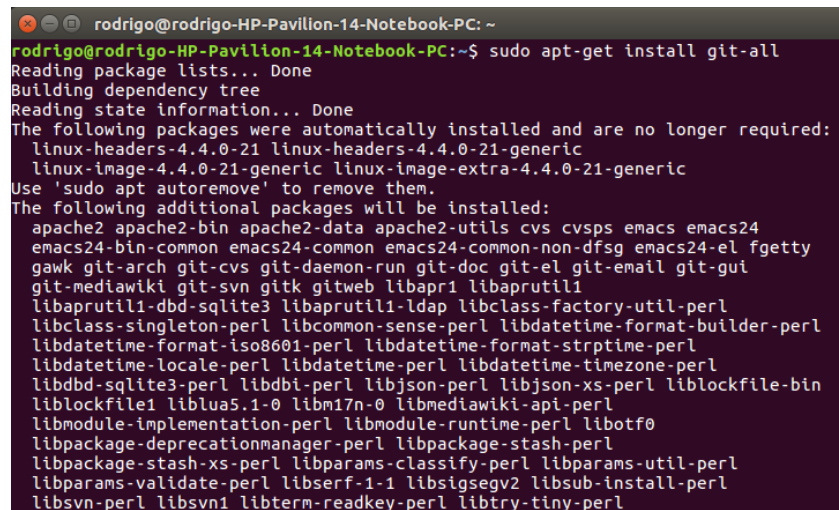


Manual para la creación de un repositorio de GitHub

Antes de comenzar con la creación de algún repositorio, es necesario instalar Git en nuestra computadora. Para eso, debemos abrir la consola y ejecutar el siguiente comando (Para Ubuntu):

```
sudo apt-get install git-all
```

Con ese comando empezarán a descargarse todos los paquetes y aparecerá una pantalla como la siguiente:



```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC: ~  
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~$ sudo apt-get install git-all  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following packages were automatically installed and are no longer required:  
  linux-headers-4.4.0-21 linux-headers-4.4.0-21-generic  
  linux-image-4.4.0-21-generic linux-image-extra-4.4.0-21-generic  
Use 'sudo apt autoremove' to remove them.  
The following additional packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils cvs cvspc emacs emacs24  
  emacs24-bin-common emacs24-common emacs24-common-non-dfsg emacs24-el fgetty  
  gawk git-arch git-cvs git-daemon-run git-doc git-el git-email git-gui  
  git-mediawiki git-svn gitk gitweb libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libclass-factory-util-perl  
  libclass-singleton-perl libcommon-sense-perl libdatetime-format-builder-perl  
  libdatetime-format-iso8601-perl libdatetime-format-strptime-perl  
  libdatetime-locale-perl libdatetime-perl libdatetime-timezone-perl  
  libdbd-sqlite3-perl libdbi-perl libjson-perl libjson-xs-perl liblockfile-bin  
  liblockfile1 liblua5.1-0 libm17n-0 libmediawiki-api-perl  
  libmodule-implementation-perl libmodule-runtime-perl libotf0  
  libpackage-deprecationmanager-perl libpackage-stash-perl  
  libpackage-stash-xs-perl libparams-classify-perl libparams-util-perl  
  libparams-validate-perl libserf-1-1 libsigsegv2 libsub-install-perl  
  libsvn-perl libsvn1 libterm-readkey-perl libtry-tiny-perl
```

Si no se tiene alguna cuenta de GitHub es necesario que creemos alguna para poder utilizar la plataforma y poder trabajar con los archivos que queramos. Debemos de dar un nombre de usuario y un correo electrónico. Los comandos son los siguientes:

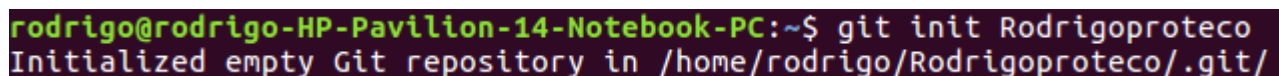
```
git config - -global user.name "nombre_usuario"  
git config - -global user.email "email_id"
```

Creación de un repositorio

Antes de crear el repositorio en sí, debemos de crear una carpeta en nuestra computadora, el cual nos servirá como un repositorio local. Para ello, es necesario poner el siguiente comando:

```
git init Carpeta
```

El nombre "Carpeta" puede ser sustituido por cualquier otro como por ejemplo "Rodrigoproteco" así como se muestra en la siguiente imagen:



```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~$ git init Rodrigoproteco  
Initialized empty Git repository in /home/rodrigo/Rodrigoproteco/.git/
```

El mensaje que aparece después del comando, significa que el repositorio se creó con éxito.

Después sigue entrar nuestra carpeta, en este caso “Rodrigoproteco” con el comando:

```
cd Rodrigoproteco
```

Ya tenemos nuestra carpeta donde se guardarán todos nuestros archivos para nuestro repositorio, lo que sigue entonces es la adición de los archivos del repositorio a un índice. Es un paso importante que se debe de hacer antes de poder subir los cambios a Github; 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.

Para agregar archivos al índice se hará con el comando add, de la siguiente manera:

```
git add archivo
```

El comando “git add” se utiliza para agregar cualquier número de archivos y carpetas al índice.

Es necesario poner un git add . (con punto final) para agregar todos los cambios, sin especificar el nombre de los archivos.

```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git add .  
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$
```

Una vez que se añadan todos los archivos,, es posible dejar un registro de los cambios que se hicieron, y esto se logra con un “commit”. Eso es para cuando ya se han terminado de agregar o modificar los archivos y que los cambios pueden ser subidos al repositorio de Github. El comando es el siguiente:

```
git commit -m 'mensaje'
```

‘mensaje’ puede ser cualquier mensaje que nosotros queramos y que describa las acciones que realizamos como “Agregué una modificación”.

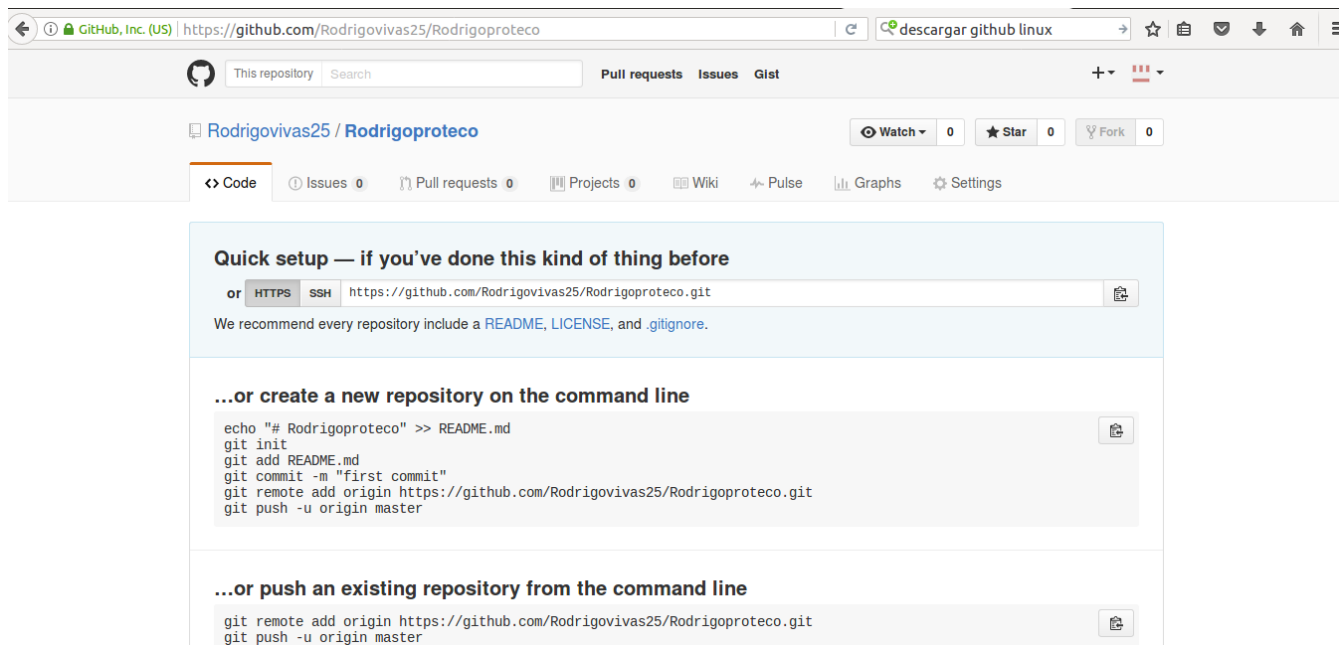
```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git add .  
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git commit -m 'Inic  
iacion de repositorio'  
On branch master  
  
Initial commit  
  
nothing to commit
```

En la imagen nos muestra un mensaje que nos dice que no fue agregado nada pero es es porque, efectivamente, no se agregó ningún archivo. Si se agrega algún archivo como algún programa hola.c, no nos dará ningún problema.

- Creación de un repositorio en GitHub

Ya que se siguieron los pasos anteriores, tenemos que iniciar sesión en la plágina de GitHub y cuando estemos ahí, debemos crear un nuevo 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/user_name/Carpeta.git`

```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git remote add origin https://github.com/Rodrigovivas25/Rodrigoproteco.git
```

El paso final es empujar el contenido del repositorio local hacia el repositorio remoto, y eso se hace con el comando:

`git push origin master`

Esto subirá todo el contenido de la carpeta “Carpeta” (repositorio local) a GitHub (repositorio externo). Para los archivos que queramos subir después, no será necesario hacer todo lo anterior, sino que basta con empezar desde el paso de creación de archivos.

Para que funcione lo anterior, es necesario hacer algún tipo de archivo, ya que de no hacerlo, nos mandará un error, debido a que no estamos subiendo nada. Si se tiene, por ejemplo, un archivo llamado holapruebas.txt, se verá lo siguiente:

```

rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ echo "holamundo" >
holapruebas.txt
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git add holapruebas
.txt
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git commit -m 'Inic
ializacion de repo'
[master (root-commit) 27ca114] Inicializacion de repo
 1 file changed, 1 insertion(+)
 create mode 100644 holapruebas.txt
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ git push origin mas
ter
Username for 'https://github.com': Rodrigovivas25
Password for 'https://Rodrigovivas25@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 244 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Rodrigovivas25/Rodrigoproteco.git
 * [new branch]      master -> master

```

Como se puede ver, al momento de hacer el push, pedirá el nombre y la contraseña de tu usuario de GitHub para poder mandarlo. Al finalizar, se verá lo siguiente en la página de GitHub:

The screenshot shows the GitHub interface for a repository named 'Rodrigoproteco' by user 'Rodrigovivas25'. At the top, there are tabs for 'Pull requests', 'Issues', and 'Gist'. Below the repository name, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. A navigation bar includes 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main content area shows 'Archivos para Proteco' with a 'New' button and 'Add topics'. It displays '1 commit', '1 branch', '0 releases', and '1 contributor'. A 'Branch: master' dropdown is next to a 'New pull request' button. Below this are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A commit history table shows the latest commit 'Inicializacion de repo' by 'Rodrigovivas25' 4 minutes ago, with a file 'holapruebas.txt'. At the bottom, there is a prompt to 'Add a README' to help people understand the project.

Con esto, la creación de nuestro repositorio, quedaría terminada.

Creación de una llave SSH

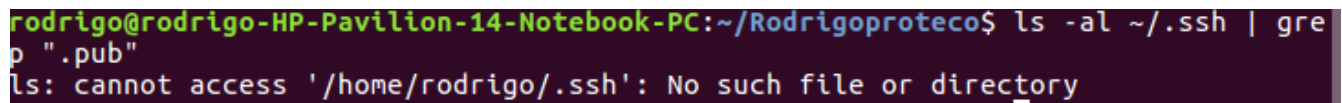
Las claves SSH también son conocidas como llaves SSH, estas son una manera de identificar las computadoras de confianza, sin tener que ingresar una contraseña.

Estas se deben de generar para cada usuario. Luego de realizar la generación, se obtendrá una clave privada y una clave pública.

Primero que nada, debemos verificar si tenemos alguna llave existente, para eso debemos de poner esto en nuestra terminal:

```
ls -al ~/.ssh | grep ".pub"
```

Como no tenemos ninguna, nos aparecerá el siguiente mensaje:

A terminal window with a dark background. The prompt is 'rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco\$'. The command entered is 'ls -al ~/.ssh | gre'. The output shows the first line of the command being executed, 'p ".pub"', and the second line showing an error: 'ls: cannot access '/home/rodrigo/.ssh': No such file or directory'.

```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ ls -al ~/.ssh | gre
p ".pub"
ls: cannot access '/home/rodrigo/.ssh': No such file or directory
```

Para poder generarla lo que haremos es escribir lo siguiente en la terminal, colocando nuestro verdadero correo electrónico:

```
ssh-keygen -t rsa -C "tu\_correo@tu\_dominio.com"
```

Nos pedirá algunos datos, pero es mejor dejarlos con los ajuste que viene por defecto, así que sólo queda pulsar Enter.

Hecho eso, aparecerá lo siguiente:

```

rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ ssh-keygen -t rsa -
C "rodrigo.vivas.25@hotmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rodrigo/.ssh/id_rsa):
Created directory '/home/rodrigo/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rodrigo/.ssh/id_rsa.
Your public key has been saved in /home/rodrigo/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vtV70hg55UCJirnYe7auHRnjyM4qtQIp04YXMdgiIsQ rodrigo.vivas.25@hotmail.com
The key's randomart image is:
+---[RSA 2048]---+
|+=                .  |
|*E+               . 0  |
|o o o . . .       |
| . o . . .       |
| + . o .S =       |
|* + o.o+ + .+ .   |
|. = . .o. = . . =  |
| o .o..o+ o.o     |
| o..+==. .o       |
+---[SHA256]---+

```

Esto nos dice que creamos con éxito nuestra llave SSH, y nos dice en qué carpeta se guardaron tanto la llave privada, como la llave pública.

Ahora se debe de agregar nuestra nueva clave SSH al ssh-agent, escribiendo lo siguiente:

```
eval "$(ssh-agent -s)" && ssh-add ~/.ssh/id_rsa
```

Con esto ya se agrega la clave al programa para que administre nuestras llaves en vez de nosotros, es decir que la clave se mantendrá en memoria para que nosotros no lo tengamos que estar ingresando. Debe salir algo como lo siguiente:

```

rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ eval "$(ssh-agent -
s)" && ssh-add ~/.ssh/id_rsa
Agent pid 14596
Identity added: /home/rodrigo/.ssh/id_rsa (/home/rodrigo/.ssh/id_rsa)

```

La parte importante de esto es agregar nuestra clave pública a GitHub

Antes que daba se debe de instalar xclip, para poder copiar el contenido de nuestra clave pública por consola. Eso se hace con el comando:

```
sudo apt-get install xclip
```

Luego debemos seleccionar y copiar el contenido de nuestra clave pública, con:

```
xclip -sel clip < ~/.ssh/id_rsa.pub
```

```

rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ xclip -sel clip < ~
/.ssh/id_rsa.pub

```

Con eso tendremos nuestra clave pública en el portapapeles, así que lo que queda es entrar a GitHub. En la sección de **Setting**, se debe seleccionar la opción **SSH Keys** e ingresamos.

Se hace click en el botón **Add SSH Key**

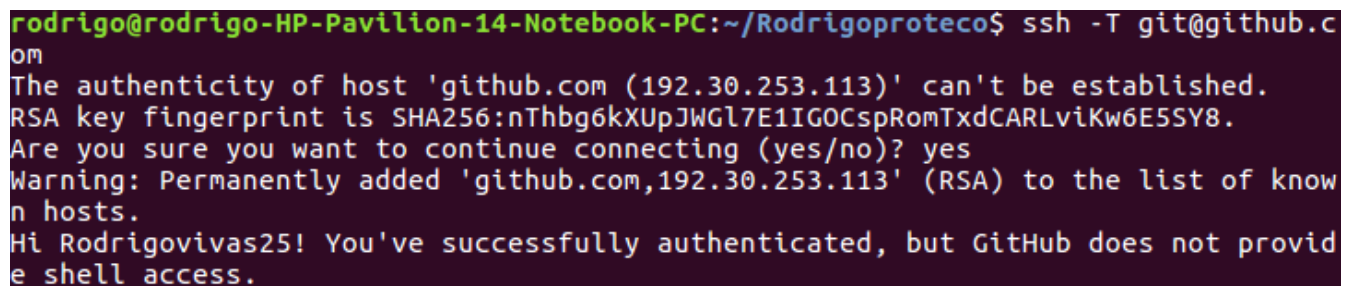
En **Title** escribimos algo sobre la clave y en **Key** se pega el contenido del archivo `~/.ssh/id_rsa.pub` (lo que tenemos en el portapapeles)

Luego se hace click en el botón **Add Key**, y se nos pedirá nuestra contraseña de GitHub para validar y listo.

Por último debemos de validar nuestra clave SSH con GitHub. Debemos escribir en la terminal:

`ssh -T git@github.com`

Si salió bien todo, aparecerá un mensaje como el siguiente:



```
rodrigo@rodrigo-HP-Pavilion-14-Notebook-PC:~/Rodrigoproteco$ ssh -T git@github.com
The authenticity of host 'github.com (192.30.253.113)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGL7E1IGOCspRomTxdCARLviKw6E5SY8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,192.30.253.113' (RSA) to the list of known hosts.
Hi Rodrigovivas25! You've successfully authenticated, but GitHub does not provide shell access.
```

Con esto ya estará registrada nuestra clave pública con GitHub, lo cual nos permitirá subir cambios al servidor de GitHub creando una conexión segura.

Función Hash

Una función digestora puede definirse como $H: U \rightarrow M$, una función que mapea o proyecta al conjunto U en un conjunto M mucho menor; una característica muy deseable de toda función hash es que la distribución resultante en M sea homogénea y tan poco dependiente de la secuencialidad de la entrada como sea posible.

La idea básica de un valor hash es que sirva como una representación compacta de la cadena de entrada. Por esta razón se dice que estas funciones resumen datos del conjunto dominio.