Gitlab – Informática I – R1051

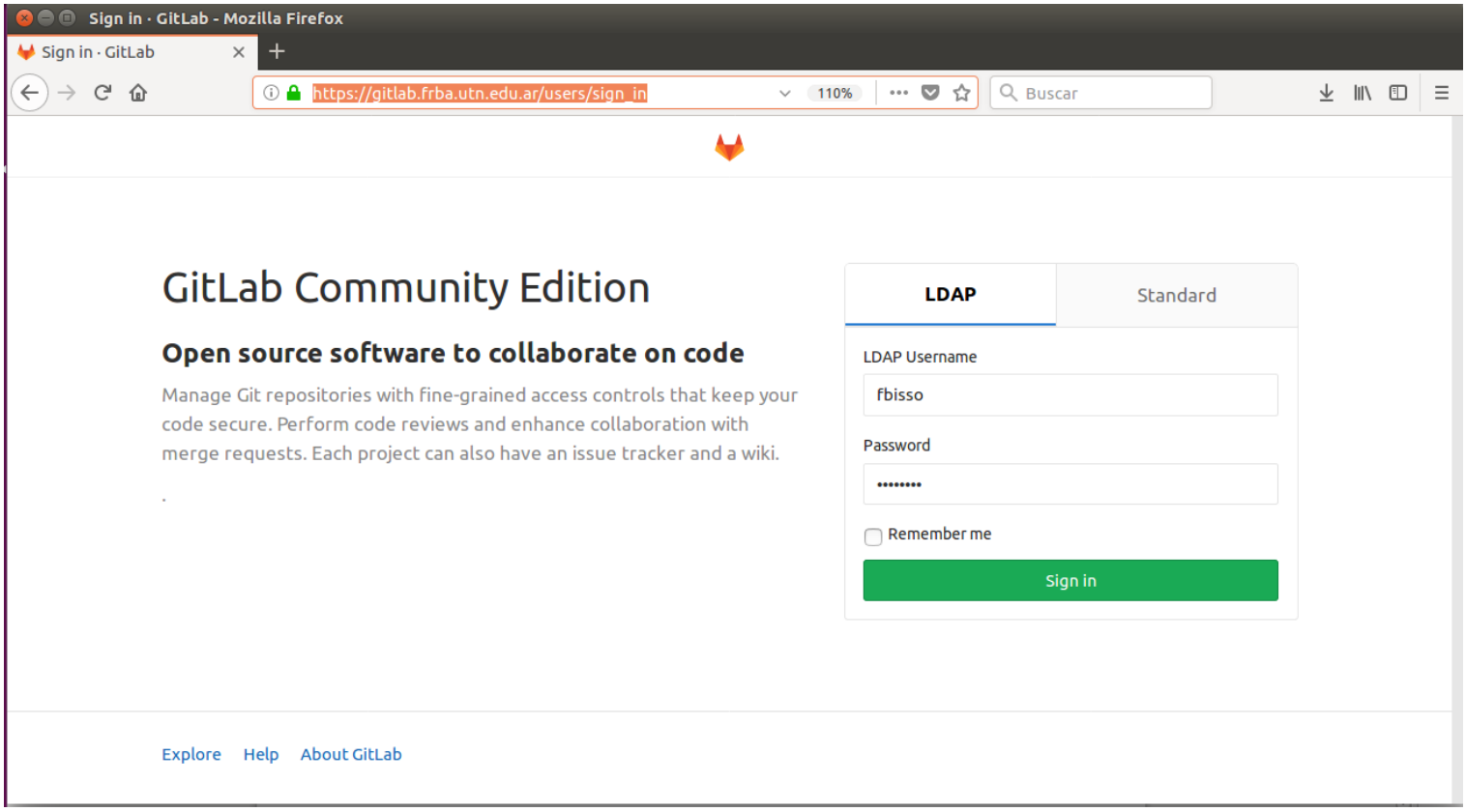
Requisitos previos:

- Conocer usuario y password SinAp.
- Informar usuario SinAp al docente para su alta en Gitlab – FRBA y creación de Proyecto.
- Instalación y configuración de git (Ver anexo I).

Clonación de repositorios git

Paso I: Login en Gitlab

- Ingresar utilizando cualquier navegador a https://gitlab.frba.utn.edu.ar/users/sign_in
- En la pestaña LDAP (no Standard) ingresar como usuario y password los de SinAp.
- Presionar Sign in.



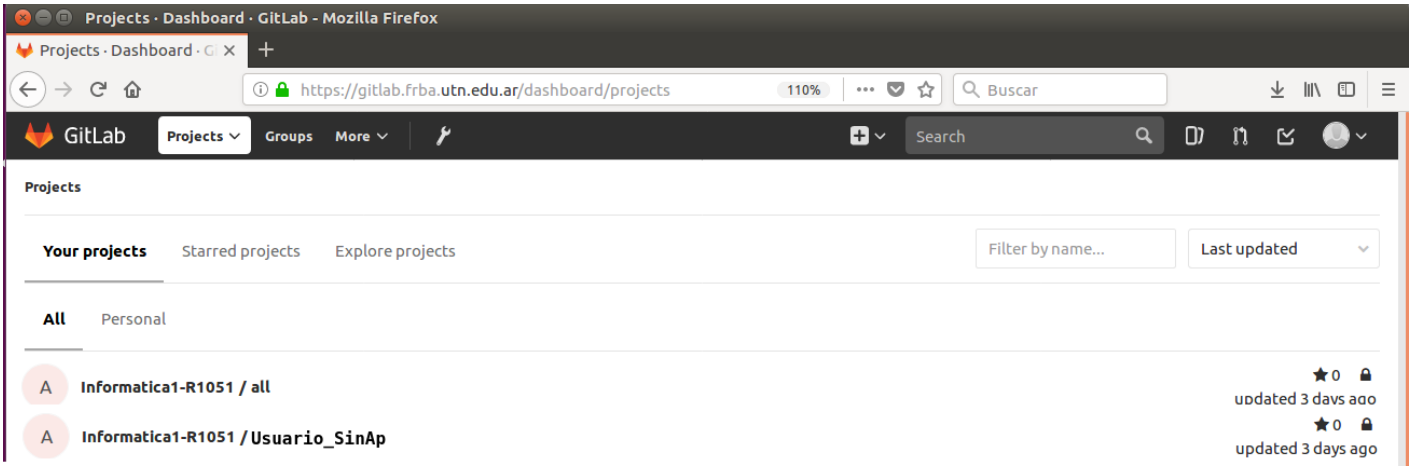
Nota: Si falla el login, notificarlo al docente para que gestione una solución.

Paso II: Acceso al proyecto y elección del protocolo a utilizar en la clonación

- Dentro de Gitlab, en la pestaña “Projects”, deben encontrar 2 proyectos:

  1. Informatica1-R1051/UsuarioSinAp
  2. Informatica1-R1051/all

  El primero se utilizará para gestionar el código realizado por el alumno, por lo tanto, se tendrán permisos de “lectura/escritura”. Es privado para cada alumno.
  El segundo en cambio, será para descargar archivos brindados por los docentes, por lo tanto, solo se tendrá permiso de “lectura”. Es común a todos los alumnos.

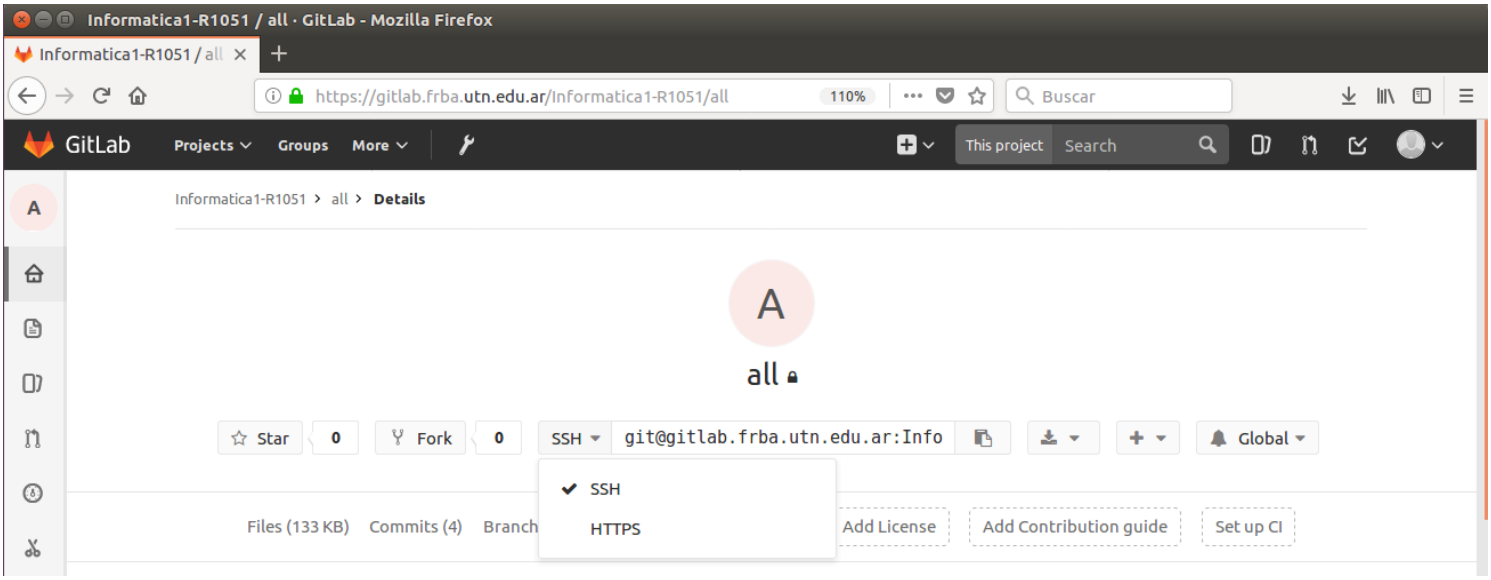


- Seleccionamos el proyecto que queremos clonar.

- Aquí tendremos que elegir que protocolo utilizaremos. Dependiento de nuestra selección, aparecerá un “link” que utilizaremos para clonar el proyecto.

  Si vamos a conectarnos desde una PC propia, utilizaremos el protocolo SSH y el “link” tendrá la siguiente forma:
  git@gitlab.frba.utn.edu.ar:Informatica1-R1051/nombre_del_proyecto.git

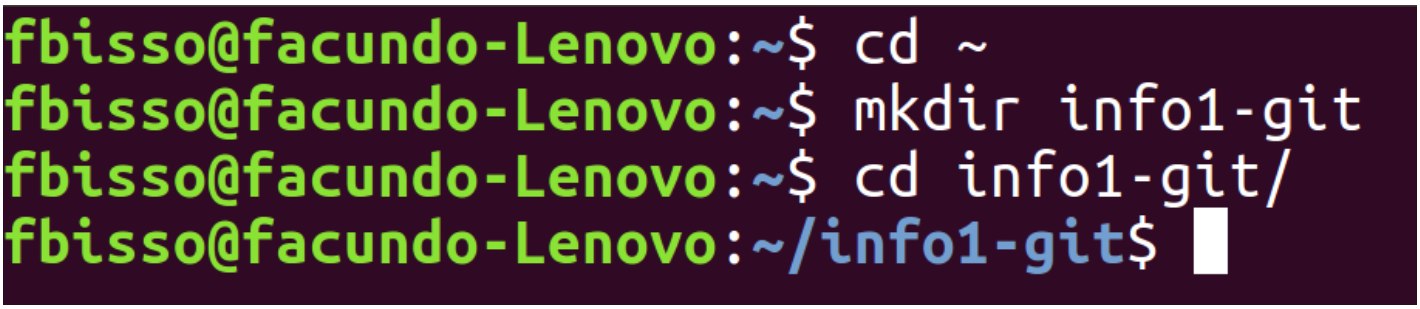
  Si nos conectaremos desde una PC pública (por ejemplo una pc de la facultad) utilizaremos el protocolo HTTPS y el “link” tendrá la siguiente forma:
  https://gitlab.frba.utn.edu.ar/Informatica1-R1051/nombre_del_proyecto.git



Paso III: Creación de carpeta donde se guardarán todos los proyectos.

- Para fácil acceso, la crearemos en la carpeta 'home' del usuario. Puede optar por crearla en otro lado, siempre y cuando recuerde donde.
  Para esto, abrimos una terminal y ejecutamos:

```
cd ~                //Nos mueve al directorio home del usuario
mkdir info1-git     //Crea el directorio 'info1-git'
cd info1-git        //Ingresa al directorio recien creado
```



Paso IV – HTTPS: Clonar proyecto en PC pública.

- Si elegimos protocolo HTTPS, copiamos el "link" y volvemos a la terminal.
- Ejecutamos: git clone "link" (sin las comillas)
- Cuando solicite usuario y password, ingresamos los de SinAp.

```
fbisso@facundo-Lenovo:~$ cd ~
fbisso@facundo-Lenovo:~$ mkdir info1-git
fbisso@facundo-Lenovo:~/info1-git$ git clone https://gitlab.frba.utn.edu.ar/Informatica1-R1051/all.git
Clonar en «all»...
Username for 'https://gitlab.frba.utn.edu.ar': fbisso
Password for 'https://fbisso@gitlab.frba.utn.edu.ar':
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 12 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (12/12), done.
Comprobando la conectividad… hecho.
fbisso@facundo-Lenovo:~/info1-git$
```

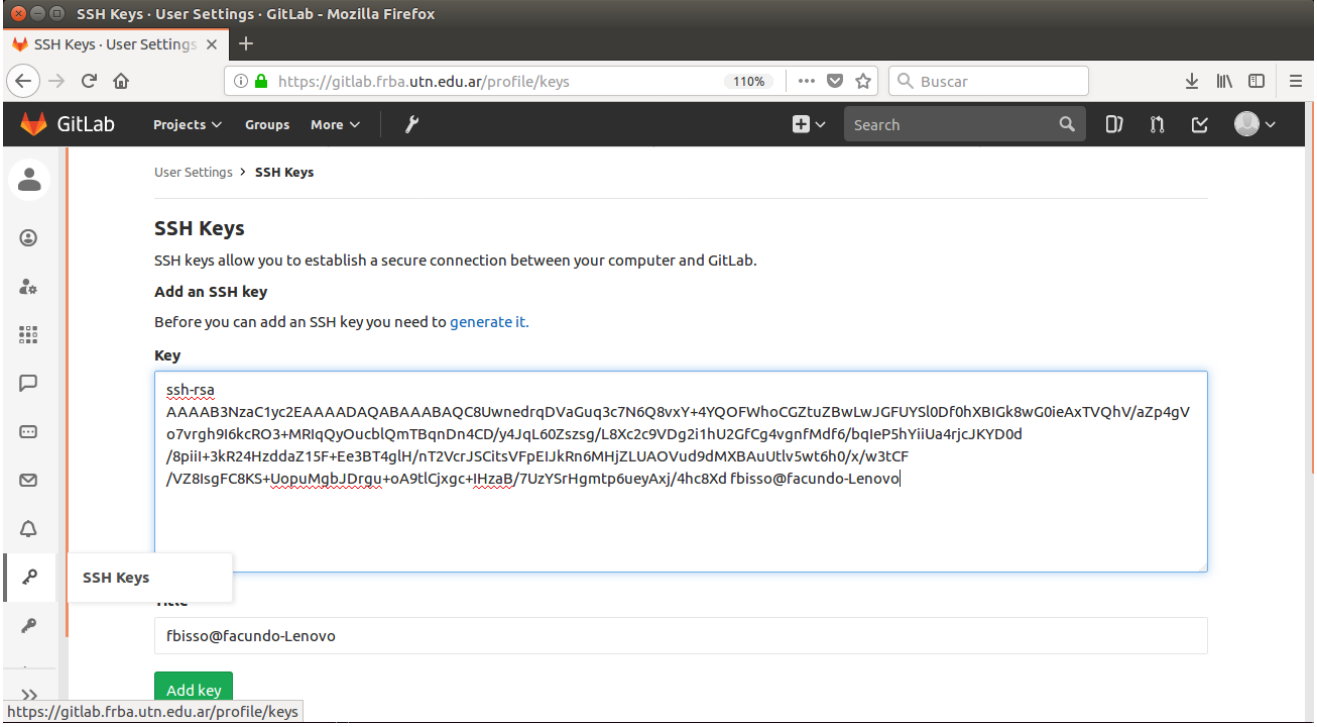
Paso IV – SSH: Agregar clave pública.

- Si elegimos protocolo SSH, antes de clonar el proyecto debemos agregar nuestra clave pública a Gitlab. La clave pública es propia de cada PC y de cada usuario, es decir que si queremos trabajar con el mismo proyecto en mas de una PC utilizando el protocolo SSH, debemos agregar las claves públicas de todas ellas.
  Para conocer nuestra clave pública seguimos los siguientes pasos:
  - cd ~ //Nos lleva al home del usuario
  - cd .ssh //Ingresamos al directorio que contiene las claves SSH
  - cat id_rsa.pub //Imprimimos en pantalla nuestra clave pública

```
fbisso@facundo-Lenovo:~/info1-git$ cd ~
fbisso@facundo-Lenovo:~$ cd .ssh/
fbisso@facundo-Lenovo:~/.ssh$ ls
id_rsa  id_rsa.pub  known_hosts
fbisso@facundo-Lenovo:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC8UwnedrqDVaGuq3c7N6Q8vxY+4YQOFWhoCGZtuZBwLwJGFUYSl0Df0hXBIGk8wG0ieAxTVQhV/aZp4
gVo7vrgh9I6kcRO3+MRIqQyOucblQmTBqnDn4CD/y4JqL60Zszsg/L8Xc2c9VDg2i1hU2GfCg4vgnfMdf6/bqIeP5hYiiUa4rjcJKYD0d/8piiI+3kR24
HzddaZ15F+Ee3BT4glH/nT2VcrJSCitsVFpEIJkRn6MHjZLUAOVud9dMXBAuUtlv5wt6h0/x/w3tCF/VZ8IsgFC8KS+UopuMgbJDrgu+oA9tlCjxgc+IH
zaB/7UzYSrHgmtp6ueyAxj/4hc8Xd fbisso@facundo-Lenovo
fbisso@facundo-Lenovo:~/.ssh$
```

Nota: Si no existe el directorio ‘.ssh’ o el archivo id_rsa.pub debemos generar la clave pública (Ver Anexo II)

- Copiamos toda la clave pública. Recuerde que el atajo para copiar en la terminal es Ctrl + Shift + C.
- Volvemos al navegador donde nos logueamos a gitlab e ingresamos a la configuración oprimiendo en el botón arriba a la derecha y seleccionando “Settings”.
- Luego a la izquierda clickeamos el ícono de la llave que dice ‘SSH Keys’.
- Pegamos toda la clave en el campo’Key’
- Oprimimos el botón ‘Add key’



Paso IVbis – SSH: Clonar proyecto en PC privada.

- Elegimos protocolo SSH en la página del proyecto (ver Paso II), copiamos el “link” y volvemos a la terminal.
- Vamos al directorio creado anteriormente: cd ~/info1-git
- Ejecutamos: git clone “link” (sin las comillas)
- No debe pedirnos usuario y password ya que realiza la autenticación utilizando la clave pública.

```
fbisso@facundo-Lenovo:~/info1-git$ cd ~/info1-git/
fbisso@facundo-Lenovo:~/info1-git$ git clone git@gitlab.frba.utn.edu.ar:Informatica1-R1051/all.git
Clonar en «all»...
Warning: Permanently added the ECDSA host key for IP address '                ' to the list of known hosts.
remote: Counting objects: 12, done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 12 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (12/12), done.
Comprobando la conectividad… hecho.
fbisso@facundo-Lenovo:~/info1-git$
```

Paso V – Verificar Clonación.

- Independientemente de que protocolo hayamos utilizado, si listamos el contenido de la carpeta ‘info1-git’ luego de la clonación, debemos ver una carpeta con el nombre del proyecto.

```
fbisso@facundo-Lenovo:~/info1-git$ ls
all
fbisso@facundo-Lenovo:~/info1-git$
```

Nota: Hemos finalizado la clonación del proyecto ‘all’. Para realizar la clonación de otros proyectos repetimos el paso ‘IVbis – SSH’ o ‘IV – HTTPS’ según el protocolo a utilizar. Vale la pena aclarar que no debe copiarse nuevamente la clave pública a menos que se cambie de usuario y/o PC.
Nota II: Es recomendable clonar todos los proyectos en el mismo directorio, en nuestro caso ‘info1-git’.

Anexo I: Instalación y configuración mínima de git.

- Aquí se expondrán aspectos básicos sobre la instalación y configuración de git. Para mas información consultar la documentación: https://git-scm.com/book/es/v1/Empezando

- Para instalar git en Distribuciones basadas en Debian:

  sudo apt-get install git

- Cofiguración de identidad:

  git config --global user.name "Nombre y Apellido"
  git config --global user.email tu_email

- Configuración de editor:

  git config --global core.editor emacs

- Configuración de herramienta de diferencias:

  git config --global merge.tool vimdiff

- Comprobar configuración:

  git config --list

Anexo II: Generación de clave pública

- Para generar la clave pública ejecutamos el comando:

  sudo ssh-keygen -t rsa

- Cuando nos pida el directorio donde se guardará la clave pública apretamos la tecla enter sin especificar nada. De esta manera utilizará por defecto '/home/usuario/.ssh', carpeta utilizada en este tutorial.
- Cuando nos pida ingresar una frase de contraseña o 'passphrase' volvemos a ingresar enter sin especificar nada. Esta opción se utiliza para agregar mayor seguridad, en este caso lo omitimos.
- Pide reingresar la frase, nuevamente presionamos enter.
- Listo, ya podremos encontrar la clave pública utilizada en el paso IV – SSH.

Crear nueva "revisión" en git

Como ya sabemos, git nos permite guardar diferentes versiones de nuestro código a medida que lo vamos desarrollando para que, en caso de ser necesario, volvamos a una versión anterior o simplemente para llevar un control de las diferentes instancias por las que pasó nuestro proyecto. A cada uno de estos estados le llamaremos "revisión". Como para crear una nueva revisión utilizamos el comando "commit" es común referirse a las revisiones como "commits".

Antes de crear un nuevo commit, tendremos que seleccionar los archivos que van a formar parte del mismo. Para saber que archivos sufrieron cambios desde el último commit (o desde el que estemos parados en el momento) utilizamos el comando "git status".

```
fbisso@facundo-Lenovo:~/info1-git/all$ touch pepe2.txt
fbisso@facundo-Lenovo:~/info1-git/all$ git status
En la rama master
Su rama está actualizada con «origin/master».
Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir en lo que se ha de confirmar)

        pepe.txt
        pepe2.txt

no se ha agregado nada al commit pero existen archivos sin seguimiento (use «git add» para darle seguimiento)
fbisso@facundo-Lenovo:~/info1-git/all$
```

En nuestro ejemplo, agregamos 2 archivos, pepe.txt y pepe2.txt al directorio de nuestro proyecto. Al ejecutar "git status", nos informa que tenemos dos archivos nuevos o "sin seguimiento". Para agregar uno de los archivos a nuestro nuevo commit utilizamos el comando "git add"

```
fbisso@facundo-Lenovo:~/info1-git/all$ git add pepe.txt
fbisso@facundo-Lenovo:~/info1-git/all$ git status
En la rama master
Su rama está actualizada con «origin/master».
Cambios para hacer commit:
  (use «git reset HEAD <archivo>...» para sacar del stage)

        nuevo archivo: pepe.txt

Archivos sin seguimiento:
  (use «git add <archivo>...» para incluir en lo que se ha de confirmar)

        pepe2.txt

fbisso@facundo-Lenovo:~/info1-git/all$
```

Al volver a ejecutar “git status”, nos informa que el archivo pepe.txt va a ser “commiteado” pero que “pepe2.txt” aún sigue sin revisión. Siempre podemos optar por commitear solo algunos de nuestros archivos, o agregarlos en commits distinto. Esto depende de como organicemos nuestro proyecto y va a estar directamente relacionado con el comentario que agreguemos al commit.

Si queremos agregar todo el directorio donde estamos parados al siguiente commit, ejecutamos “git add .”

```
fbisso@facundo-Lenovo:~/info1-git/all$ git add .
fbisso@facundo-Lenovo:~/info1-git/all$ git status
En la rama master
Su rama está actualizada con «origin/master».
Cambios para hacer commit:
  (use «git reset HEAD <archivo>...» para sacar del stage)

        nuevo archivo: pepe.txt
        nuevo archivo: pepe2.txt

fbisso@facundo-Lenovo:~/info1-git/all$
```

Como puede apreciarse, esto agregó el archivo que nos había quedado sin revisión.

Una vez seleccionados los archivos deseados, procedemos a realizar el commit. Para esto ejecutamos: git commit -m“mensaje que explica las modificaciones”

```
fbisso@facundo-Lenovo:~/info1-git/all$ git commit -m"Este es un commit de prueba"
[master 837e1d8] Este es un commit de prueba
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 pepe.txt
 create mode 100644 pepe2.txt
fbisso@facundo-Lenovo:~/info1-git/all$ git status
En la rama master
Su rama está delante de «origin/master» para 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working directory clean
fbisso@facundo-Lenovo:~/info1-git/all$
```

Aquí podemos realizar 2 observaciones. Primero, al realizar el commit, nos muestra las modificaciones que hicimos y nos da el identificador del mismo. En este caso es 837e1d8 de la rama master. Git permite tener varios “hilos” dentro de un mismo proyecto, que se denominan branches, pero no vamos a utilizarlos por ahora.
Segundo, al ver el nuevo status, nos indica que diferimos de la rama master por 1 commit. Esto significa que aún no subimos nuestros cambios al repositorio de la facultad (y por ende los profes no pueden verlo!). Para subir todos los commits realizados al repositorio de la facultad utilizamos el comando “git push”

```
fbisso@facundo-Lenovo:~/info1-git/all$ git status
En la rama master
Su rama está actualizada con «origin/master».
nothing to commit, working directory clean
fbisso@facundo-Lenovo:~/info1-git/all$
```

Luego de esto podemos observar que estamos “actualizados con respecto a la rama master”, lo que significa que subimos todos los cambios al repositorio de la facultad.

Nota: Este procedimiento, podremos realizarlo unicamente en el proyecto con nuestro usuario sinap, en el cual tenemos permisos de “lectura/escritura”

Como descargar los nuevos archivos subidos por los docentes al proyecto “all”

Una vez clonado el repositorio como vimos en la primer parte de esta guía, podemos proceder a descargar los cambios ejecutando el comando “git pull”