

Git y uso de Github

Herramienta para el desarrollo y entrega de prácticas

¿Por qué usar Git y Github?

Cuando programamos es muy importante:

- ▶ Mantener un código ordenado
- ▶ Hacer backups para no perder el trabajo

Cuando queremos compartir los codigos y/o queremos trabajar desde más de una computadora es util:

- ▶ Contar con una versión actualizada de los códigos en la nube

Git y Github permite escencialmente resolver estas tres necesidades.

¿Qué es Git?

Git es un sistema de control de versiones distribuido.

El control de versiones es la práctica de encontrar y administrar los cambios en el código. Permite mantener registro de los cambios realizados y un orden entre múltiples versiones del mismo código. (Queremos evitar el clásico de tener la carpeta llena de proyecto-final-version-2 o proyecto-final-version-2-final-final)

Distribuido hace referencia a que los códigos no se encuentran guardados únicamente en nuestra computadora, sino que existen copias en más de una computadora.

Estructura de Git

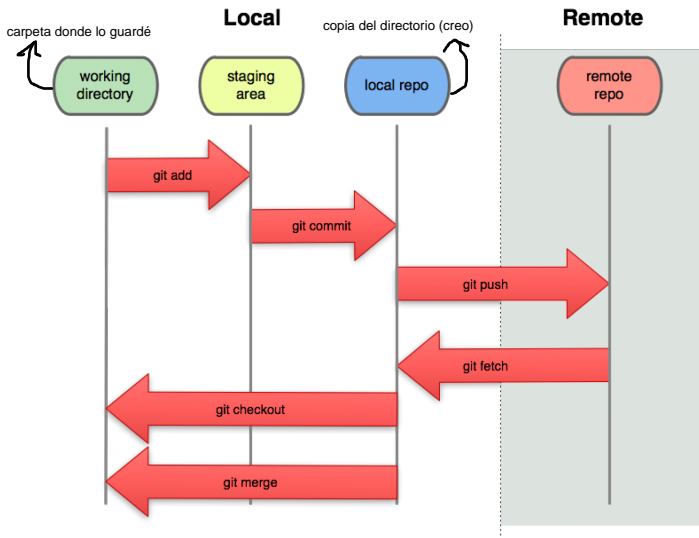


Figure 1: Estructura general de Git

a traves de la terminal nos "comunicamos" con el servidor

Comandos de configuración Git

todos los movimientos van a ser bajo un usuario, por eso el nombre y mail

La configuración global de git la vamos a hacer una única vez:

```
$ git config  
$ git config --global user.name <nombre>  
$ git config --global user.email <email>
```

Este paso es necesario para que todas las acciones que se realicen en el sistema estén asociadas a un autor.

Adicionalmente, vamos a configurar la rama master con el nombre main, el motivo de esto va a quedar claro más adelante pero es únicamente por practicidad:

```
$ git config --global init.defaultBranch main
```

Comandos para la creación de un repositorio

Generamos una carpeta donde guardar los archivos de la Práctica 1 y creamos el repositorio local:

```
$ mkdir Practica1 mkdir --> comando para crear carpetas  
$ cd Practica1 cd --> para entrar en la carpeta  
Practica1$ git init
```

El repositorio local está en un archivo oculto '.git'. Para corroborar que se inició correctamente podemos ejecutar los comandos:

```
Practica1$ ls -a  
.git
```

dado que .git es un archivo oculto, si ejecutamos

```
Practica1$ ls
```

no vamos a ver nada.

Comandos para agregar archivos al repositorio local

Supongamos que ya realizamos los códigos de la práctica (ejercicio1.R, ejercicio2.R). Lo que vamos a hacer *siempre* antes de enviar archivos al repositorio Git es revisar el estatus del proyecto:

```
Practical1$ git status -s
```

```
?? ejercicio1.R      ?? --> los codigos existen en mi carpeta pero el repositorio local no sabe que existen
```

```
?? ejercicio2.R
```

Aquí vemos una lista de los archivos en el directorio de trabajo y el estado de seguimiento. Si se han seguido, se mostrarán como cambios listos para commit en el área de Stage (A). Si no se han seguido, se mostrarán como cambios no rastreados (??). Para agregar al área de stage utilizamos git add

```
Practical1$ git add ejercicio1.R
```

para que el repositorio local haga seguimiento de esos cod

```
Practical1$ git status -s
```

```
A ejercicio1.R
```

```
?? ejercicio2.R
```

Comandos para agregar archivos al repositorio local

Si ya sabemos que queremos agregar todos los archivos podemos usar:

```
Practical1$ git add .
```

Ahora todos los archivos se van a 'seguir' por el sistema de versiones de git. Después de hacer la última revisión y comprobar que todos los códigos funcionan, lo enviamos de la 'staging area' al 'repositorio local' así: commit es como "sacar una foto"

```
$ git commit -m <mensaje de referencia>  
$ git status -s
```

Si todo está al día con el repositorio local, no aparecerá nada luego de git status.

Control de versiones

Luego de aplicar el commit, la versión actual de los archivos estará guardada para siempre en el repositorio local. Un 'commit' es un 'punto de guardado', las versiones asociadas a cada 'punto de guardado' o 'commit' que hagamos van a quedar guardadas y vamos a poder acceder a ellas a través de su identificador o 'hash'.

Pueden pensar esto como tomar una foto del estado actual de los códigos para luego volver si es necesario.

Git checkout y reset

Supongamos que acabamos de hacer algunos cambios en un único archivo y deseamos volver al estado anterior sin guardar las modificaciones porque logramos que el código funcione mejor resolviendo un error en el código. El siguiente código nos lleva hacia la forma anterior del archivo:

porque lo que habia hecho estaba bien y lo modifiqué mal.Vuelve a

```
$ git checkout <nombre de archivo>
```

Si queremos volver al estado previo de todo el repositorio, es decir, a un commit anterior usamos el siguiente comando:

```
$ git reset
```

porque lo que habia hecho estaba bien y lo modifiqué mal.Vuelve a la "

Viajar entre versiones del repositorio

Para ver todo el historial de cambios del proyecto usamos:

```
$ git log
```

veo todas las versiones previas. todas las "fotos que fui sacando"

Si realizamos varios commits, lo que vamos a ver son todos los commits con sus respectivos mensajes:

```
((base) juliamindlin@MacBook-Air-Emanuele Practica1 % git log
commit 75bd6fed5db2ea471b9f7df8d431f19ee7679ae7 (HEAD -> main)
Author: Julia Mindlin <juliamindlin@MacBook-Air-Emanuele.local>
Date: Mon Aug 14 15:01:44 2023 -0300

    agregó todo

commit cfafba911827b477f8a94a88151b04708b9b7d47
Author: Julia Mindlin <juliamindlin@MacBook-Air-Emanuele.local>
Date: Mon Aug 14 14:57:48 2023 -0300

    ejercicio 4
```

Para recuperar el estado del repositorio en un estadio anterior, hacemos o siguiente:

```
$ git reset --hard <numero identificador del commit>
```

Github

Github es independiente de Git. Lo que provee es un servidor para hacer respaldos de repositorios Git. Así como existe Github, también existen otros. En esta materia vamos a usar Github porque es el más usado.

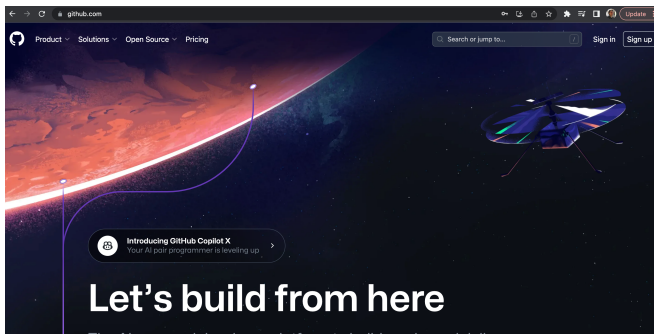


Figure 2: Página de inicio de Github

Github - Generar un usuario

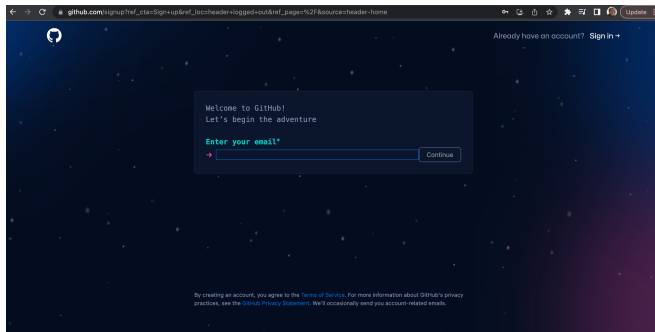


Figure 3: Github paso 1

Github - Página de trabajo

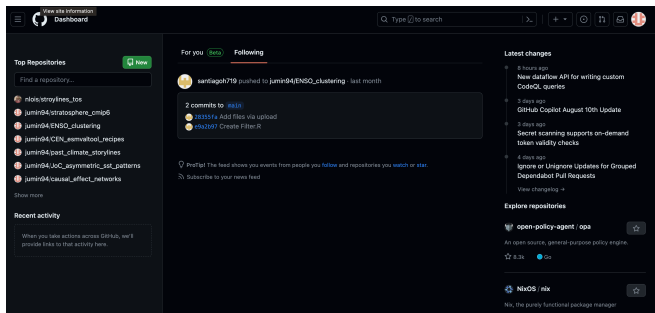
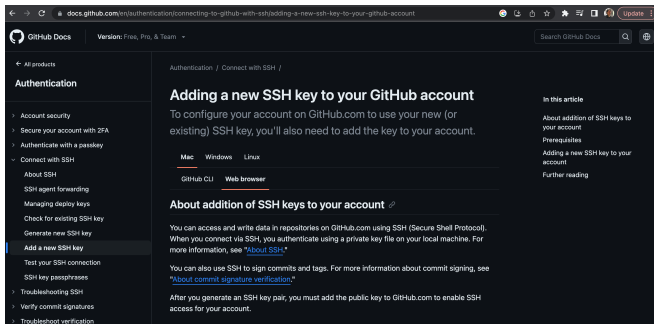


Figure 4: Github - dashboard

Github - Vincular la computadora con Github para hacer envíos y descargas

Una forma de enviar y descargar archivos desde un servidor es mediante el protocolo SSH (Secure Shell Protocol). Para hacer uso de este protocolo tenemos que configurar una 'cerradura y llave' (actúa como una especie de contraseña) en nuestra computadora local y asociar la 'llave' a nuestro usuario de Github.

Las instrucciones para esto están en la página de Github (lo vamos a hacer en la práctica):



The screenshot shows the GitHub documentation page for adding an SSH key. The browser address bar shows the URL: docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account. The page has a dark theme. On the left is a sidebar with the 'Authentication' section expanded, showing a list of topics including 'Connect with SSH' and 'Add a new SSH key'. The main content area has the title 'Adding a new SSH key to your GitHub account' and a sub-header 'About addition of SSH keys to your account'. The text explains that SSH is used to access and write data in repositories and that a private key file is used for authentication. It also mentions that SSH can be used to sign commits and tags. The page includes tabs for 'Mac', 'Windows', 'Linux', 'GitHub CLI', and 'Web browser', with 'Web browser' currently selected. On the right side, there is a section titled 'In this article' with links to 'About addition of SSH keys to your account', 'Prerequisites', 'Adding a new SSH key to your account', and 'Further reading'.

docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

GitHub Docs Version: Free, Pro, & Team

Search GitHub Docs

Authentication / Connect with SSH /

Adding a new SSH key to your GitHub account

To configure your account on GitHub.com to use your new (or existing) SSH key, you'll also need to add the key to your account.

Mac Windows Linux

GitHub CLI Web browser

About addition of SSH keys to your account

You can access and write data in repositories on GitHub.com using SSH (Secure Shell Protocol). When you connect via SSH, you authenticate using a private key file on your local machine. For more information, see ["About SSH."](#)

You can also use SSH to sign commits and tags. For more information about commit signing, see ["About commit signature verification."](#)

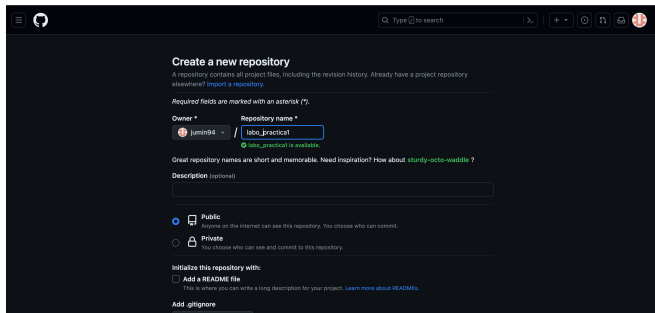
After you generate an SSH key pair, you must add the public key to GitHub.com to enable SSH access for your account.

In this article

- About addition of SSH keys to your account
- Prerequisites
- Adding a new SSH key to your account
- Further reading

Github - Crear un repositorio I

En el dashboar o página de inicio encontramos un botón de 'Nuevo repositorio'. Al clickearlo vamos a a lla siguiente página. Sólo tenemos que elegir el nombre del repositorio.

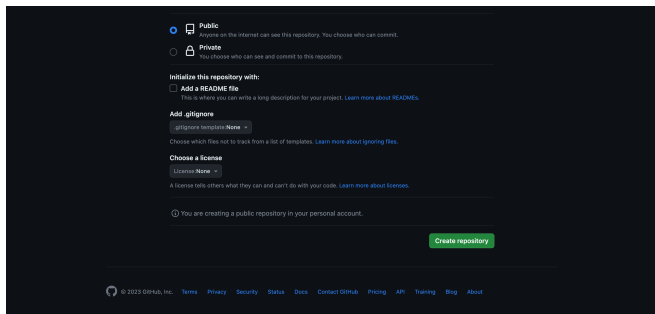


The screenshot shows the GitHub 'Create a new repository' interface. At the top, there's a search bar and navigation icons. The main heading is 'Create a new repository', followed by a subtext explaining that a repository contains project files and revision history. Below this, there's a note about required fields marked with an asterisk. The 'Owner' field is set to 'jumin94'. The 'Repository name' field contains 'labo_practical', with a green checkmark indicating it's available. A hint suggests great repository names are short and memorable, with an example 'sturdy-octo-waddle'. There's an optional 'Description' field. Under the 'Visibility' section, 'Public' is selected, with a note that anyone on the internet can see the repository. 'Private' is also an option. The 'Initialize this repository with:' section has a checkbox for 'Add a README file' (checked) and a link to learn more about READMEs. At the bottom, there's a link to 'Add .gitignore'.

Figure 6: Github paso 2

Github - Crear un repositorio II

A fines de la entrega de prácticas, vamos a dejarlo público.



The screenshot shows the GitHub 'Create repository' form. At the top, there are two radio buttons: 'Public' (selected) and 'Private'. Below this is the 'Initialize this repository with:' section, which includes a checkbox for 'Add a README file' and a dropdown menu for 'Add .gitignore' (set to 'None'). There is also a 'Choose a license' dropdown menu (set to 'None'). A green 'Create repository' button is located at the bottom right. The footer contains the GitHub logo, copyright information, and links to Terms, Privacy, Security, Status, Docs, Contact GitHub, Pricing, API, Training, Blog, and About.

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
`.gitignore` template: `None`

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
License: `None`

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

☐ You are creating a public repository in your personal account.

[Create repository](#)

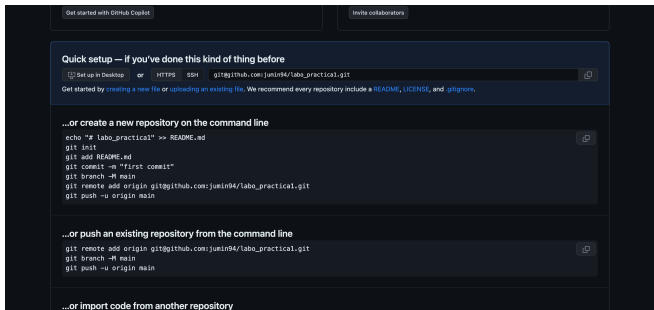
© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

Figure 7: Github paso 2

Github - Vincular el repositorio local con el repositorio en el servidor

Para vincular el repositorio local vamos a copiar los comandos de la segunda caja, es decir desde:

```
$ git remote add origin git@github.com:jumin94/labo...
```



estos comandos de aca

Figure 8: Github paso 3

Procedimiento para la entrega de prácticas

- ▶ Crear una carpeta que contenga los códigos a entregar
- ▶ Crear un repositorio Git
- ▶ Vincular el repositorio local con el repositorio en Github
- ▶ Hacer un 'push' de los códigos a entregar
- ▶ Compartir el link al repositorio

Instalación de Git en Windows

Windows - la forma más fácil de instalar Git es a través de la página de descargas de Git para Windows:

git-scm.com/download/win

Una vez descargado el archivo de instalación hay que seguir las instrucciones como con cualquier otro programa. Durante la instalación, se instalará automáticamente una terminal compatible con Git, llamada Git Bash.

Instalación de Git en Linux

Instalación en Linux/Unix La mayoría de las distribuciones de Linux ya incluyen Git en sus repositorios:

git-scm.com/download/linux

Por ejemplo, para instalar Git en Ubuntu o Debian, abrimos la terminal y escribimos:

```
usuario@nombre_de_maquina:~$ sudo apt-get install git
```

En caso de Fedora, abrimos la terminal y escribimos:

```
usuario@nombre_de_maquina:~$ sudo dnf install git
```

Corroboramos a correcta instalación con:

```
usuario@nombre_de_maquina:~$ git --version o git -v.
```

Documentación oficial

La información más actualizada y las mejores instrucciones las pueden encontrar siempre en la documentación oficial de Git (git-scm.com/downloads5)