

Manual GitHub con Eclipse

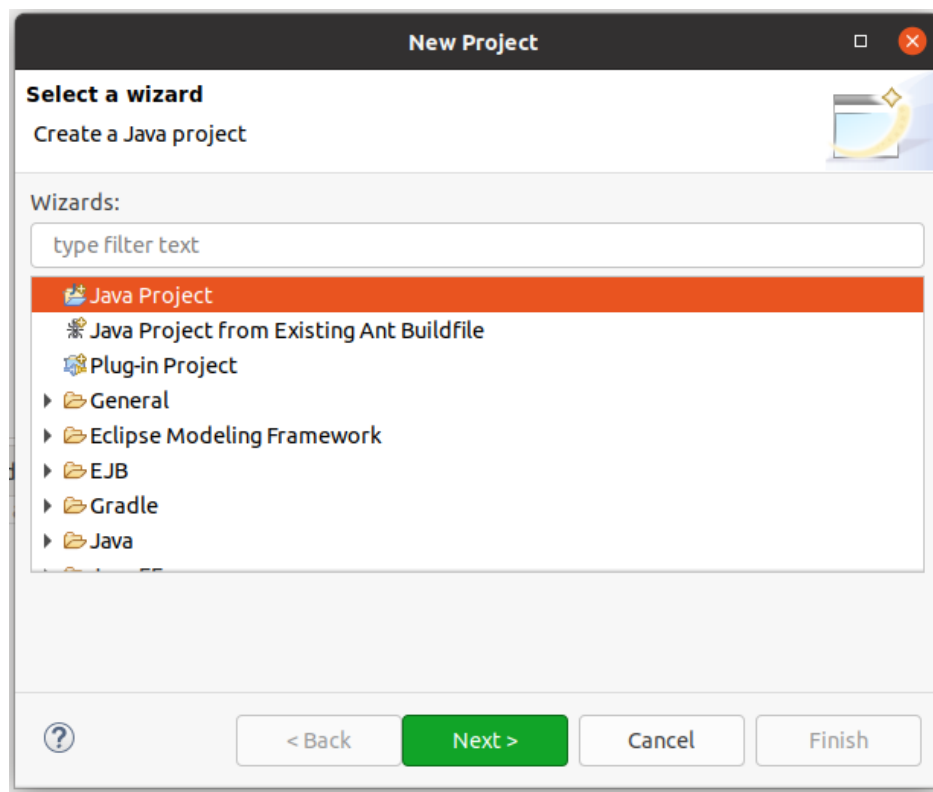
Crear un repositorio

1.- Antes de nada, instalamos Git en nuestra máquina con el comando:

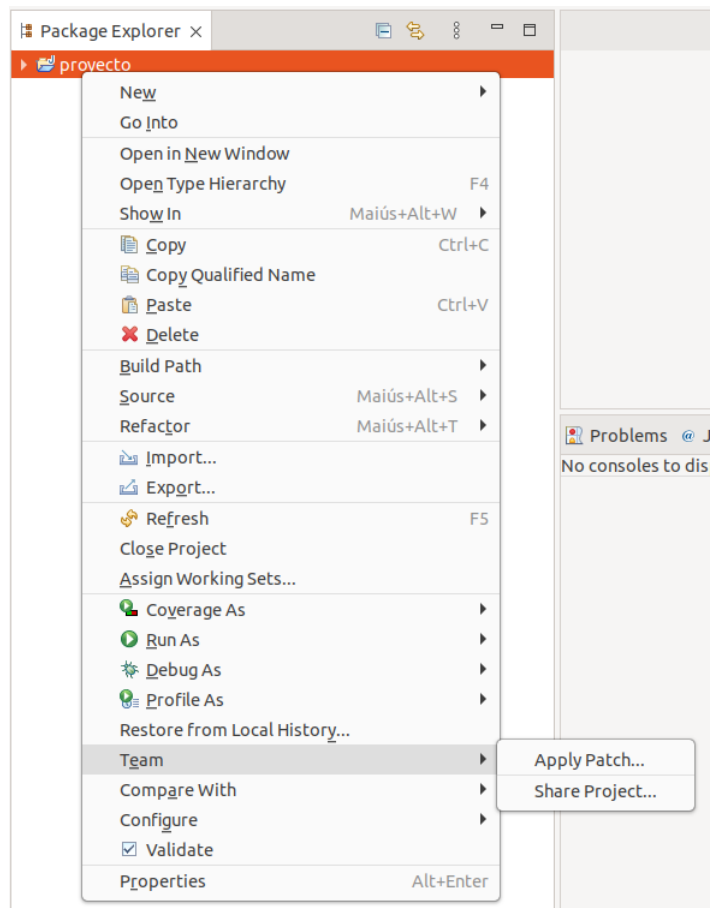
sudo apt install git

2.- Abrimos Eclipse y creamos nuestro nuevo proyecto.

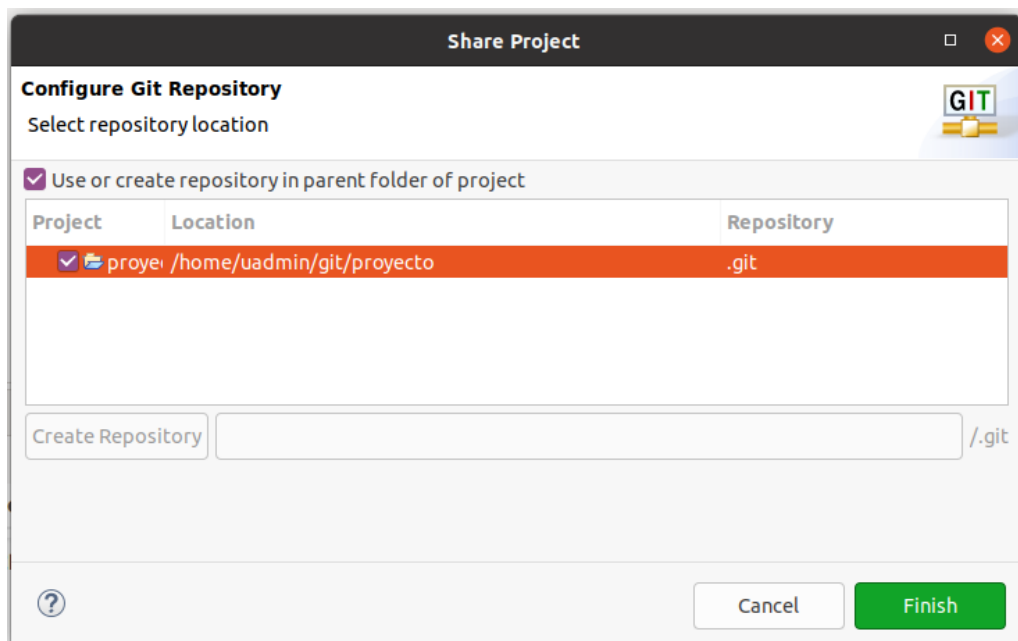
***NOTA: Por defecto, los proyectos se crean en el directorio *eclipse-workspace*. Si en nuestro workspace tenemos otros proyectos que no estén relacionados o que no usen Git, lo mejor es que utilicemos un directorio diferente para nuestros proyectos de Git. Una buena práctica es crear un directorio *git* en nuestro perfil de usuario (por ejemplo, */home/uadmin/git*) y gestionarlos desde ahí.



3.- Creamos un repositorio Git para el proyecto. Para eso, hacemos click derecho sobre el proyecto y seleccionamos *Team > Share Project...*



Seleccionamos el directorio en el que queremos crear el repositorio:



4.- Ya tenemos creado nuestro repositorio. Podemos comprobarlo desde la terminal:

A terminal window titled 'uadmin@base: ~/git/proyecto'. The user has entered 'git status'. The output shows the current branch is 'master', there are no commits yet, and a list of files not being tracked: .classpath, .gitignore, .project, .settings/, and src/. A message at the bottom suggests using 'git add' to track these files.

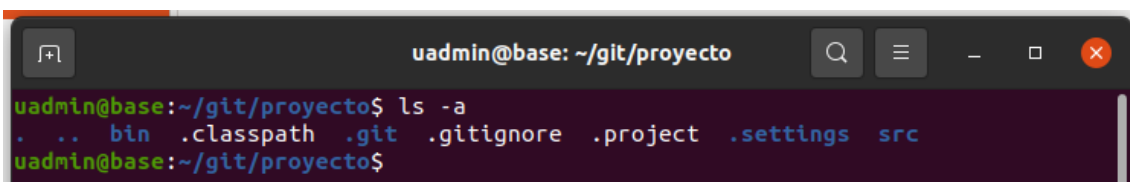
```
uadmin@base:~/git/proyecto$ git status
En la rama master

No hay commits todavía

Archivos sin seguimiento:
(usa "git add <archivo>..." para incluirlo a lo que se será confirmado)
.classpath
.gitignore
.project
.settings/
src/

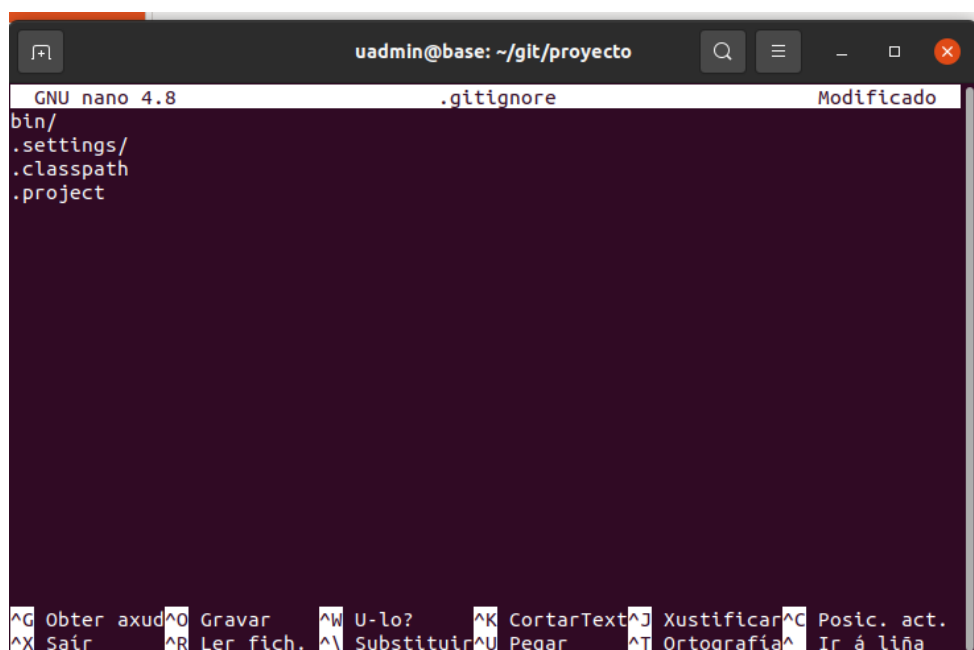
no hay nada agregado al commit pero hay archivos sin seguimiento presentes (usa
"git add" para hacerles seguimiento)
uadmin@base:~/git/proyecto$
```

5.- Podemos ver todos los archivos del proyecto con el comando **ls -a**:

A terminal window titled 'uadmin@base: ~/git/proyecto'. The user has entered 'ls -a'. The output lists all files and directories, including hidden ones: ., .., bin, .classpath, .git, .gitignore, .project, .settings, and src.

```
uadmin@base:~/git/proyecto$ ls -a
.  .. bin .classpath .git .gitignore .project .settings src
uadmin@base:~/git/proyecto$
```

6.- Editamos el archivo **.gitignore** para que solo tenga en cuenta el directorio **src** a la hora de hacer los commits, e ignore los demás. Lo podemos hacer con el comando **nano** **.gitignore** :

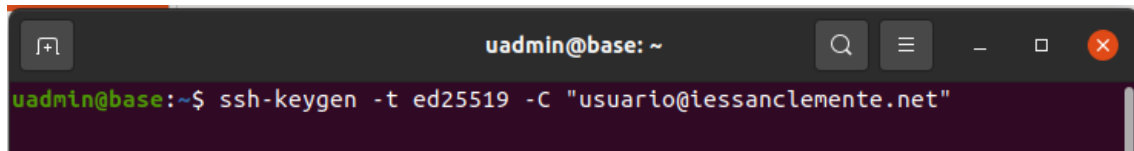
A terminal window titled 'uadmin@base: ~/git/proyecto' showing the nano text editor editing the .gitignore file. The editor title bar says 'GNU nano 4.8 .gitignore Modificado'. The content of the file shows 'bin/' and '.settings/' being ignored, while '.classpath' and '.project' are not. The bottom of the screen shows nano's command shortcuts.

```
GNU nano 4.8 .gitignore Modificado
bin/
.settings/
.classpath
.project

^G Obter axud ^O Gravar ^W U-lo? ^K CortarText ^J Xustificar ^C Posic. act.
^X Saír ^R Ler fich. ^\ Substituir ^U Pegar ^T Ortografia ^_ Ir á liña
```

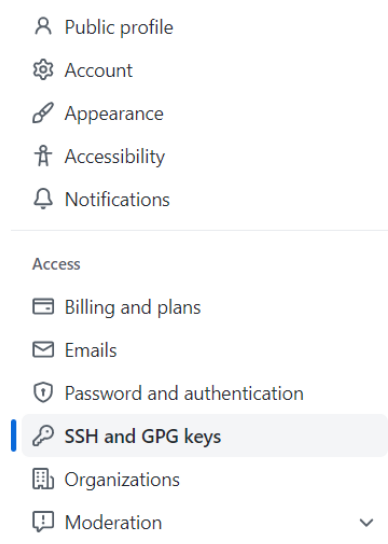
7.- Ya tenemos configurado nuestro repositorio local. Ahora, vamos a sincronizarlo con GitHub. Para eso, primero tenemos que configurar una clave SSH que nos permita autenticarnos con GitHub. Desde terminal, generamos la clave en nuestra máquina con el siguiente comando:

***NOTA: Tendrías que poner vuestro correo, este es solo para el ejemplo.

A terminal window with a dark background. The prompt is 'uadmin@base: ~'. The command entered is 'ssh-keygen -t ed25519 -C "usuario@iessanclemente.net"'.

```
uadmin@base:~$ ssh-keygen -t ed25519 -C "usuario@iessanclemente.net"
```

Esto os generará un directorio `.ssh` en vuestro perfil (por ejemplo `/home/uadmin/.ssh`). Dentro de ese directorio, se generará un archivo `.pub` con vuestra clave. Tenéis que copiar esa clave en vuestro perfil de GitHub. Para eso, vais a vuestros ajustes de cuenta en GitHub y creáis una nueva clave SSH con el contenido del archivo `.pub`:

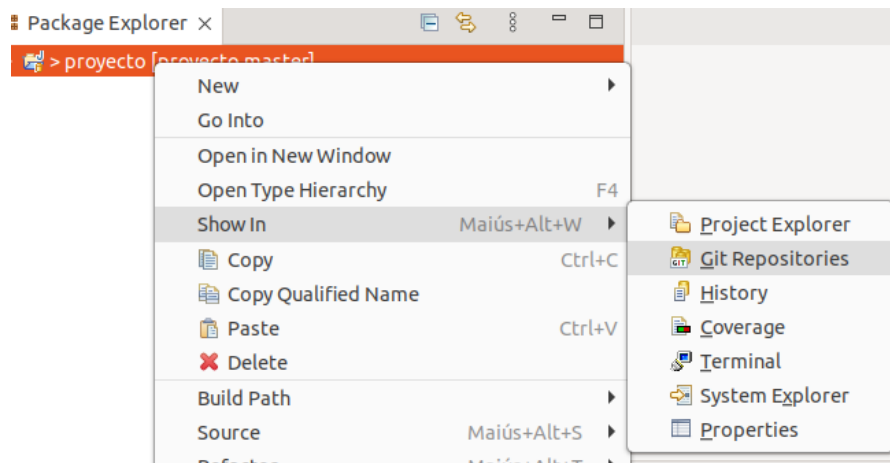


Tenéis información más detallada sobre cómo realizar este paso en el siguiente enlace:

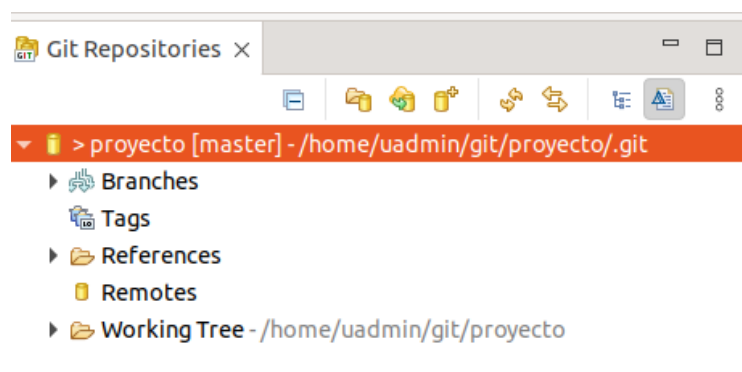
<https://docs.github.com/es/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account>

8.- Cuando ya tenemos configurada nuestra clave, creamos un nuevo repositorio vacío en GitHub.

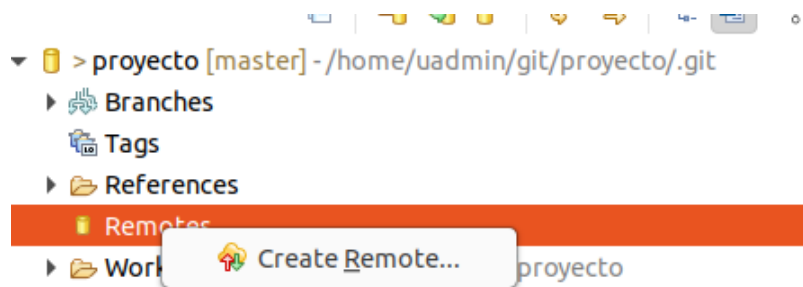
9.- Una vez está creado el repositorio en GitHub, ya podemos sincronizarlo con nuestro repositorio local. Para eso mostramos en Eclipse nuestro repositorio del proyecto (*Click derecho > Show In > Git Repositories*):



Se nos abrirá un panel como el siguiente:



Creamos un nuevo repositorio remoto (Click derecho en *Remotes* > *Create Remote...*):



Y le indicamos la URL SSH de nuestro repositorio de GitHub:

Select a URI

Source Git Repository
Enter the location of the source repository.

Location

URI: Local Folder... Local Bundle File...

Host:

Repository path:

Connection

Protocol:

Port:

Authentication

User:

Password:

☐ Store in Secure Store

Y por último, lo guardamos con el botón *Save*.

10.- Ya está listo el repositorio, solo nos quedaría hacer un commit (Click derecho > *Team* > *Commit...*) y comprobar el repositorio en GitHub:

> proyecto [master]

Unstaged Changes (4)

- .classpath
- project
- org.eclipse.core.resources.prefs - .settings
- org.eclipse.jdt.core.prefs - .settings

Staged Changes (2)

- .gitignore
- module-info.java - src

Commit Message

Unborn branch: this commit will create the branch 'master'.

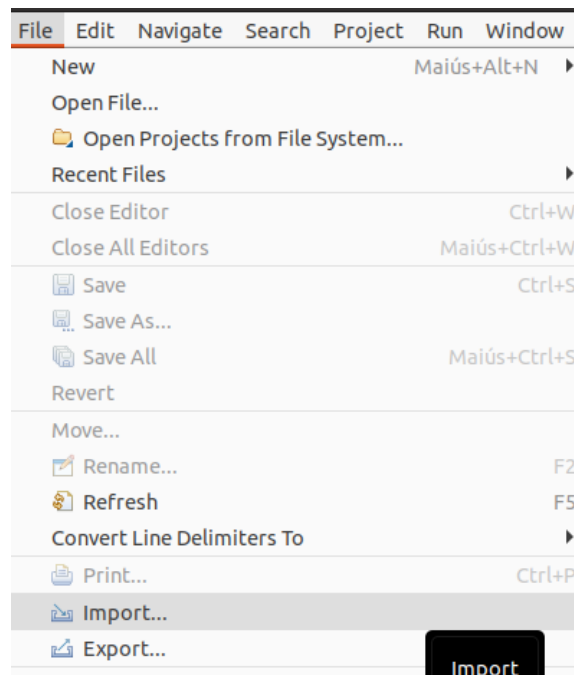
Commit inicial

Author:

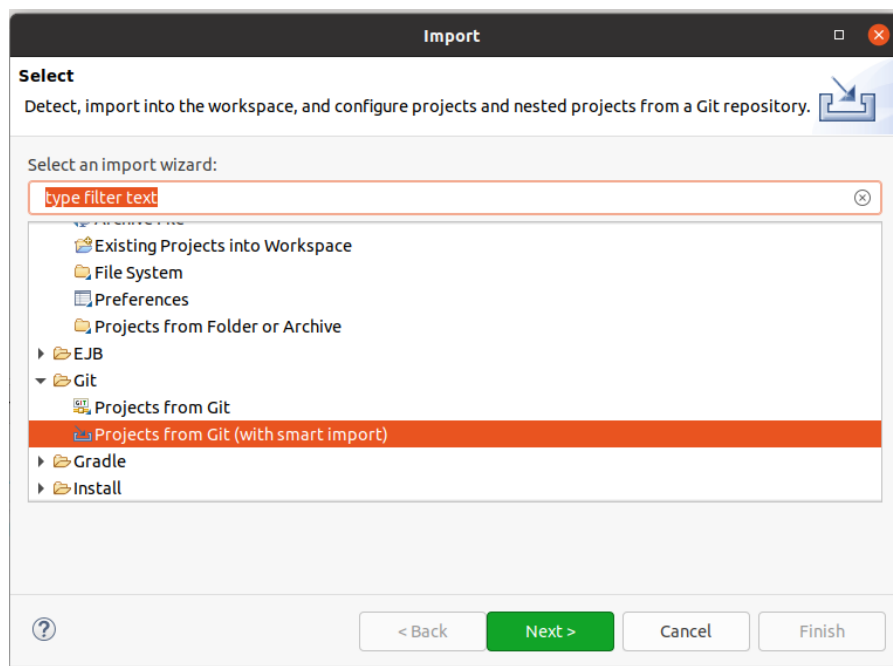
Committer:

Clonar un repositorio

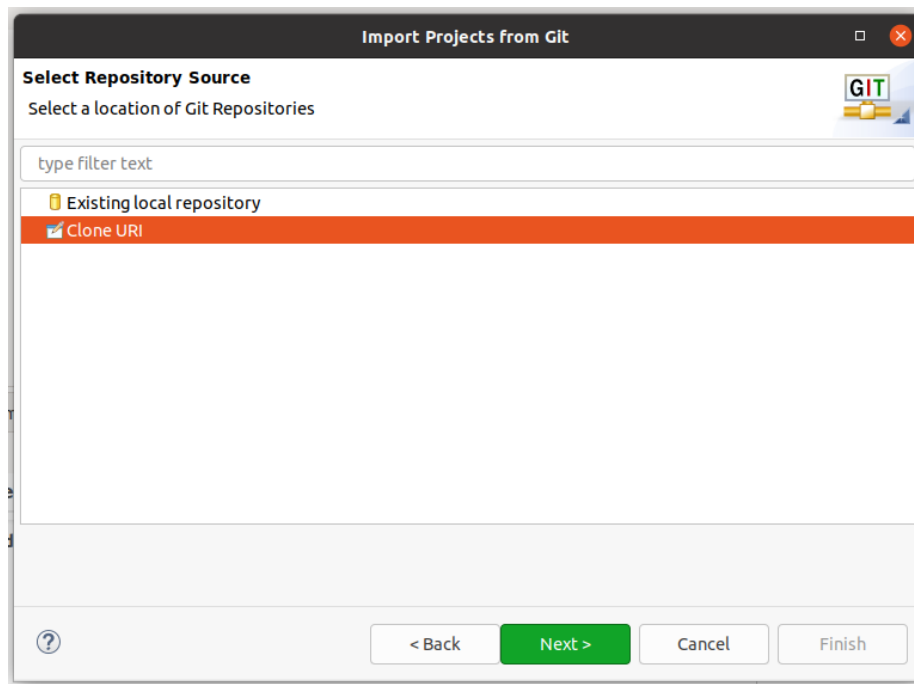
1.- Para clonar un proyecto de GitHub, simplemente lo importamos desde *File > Import...*:



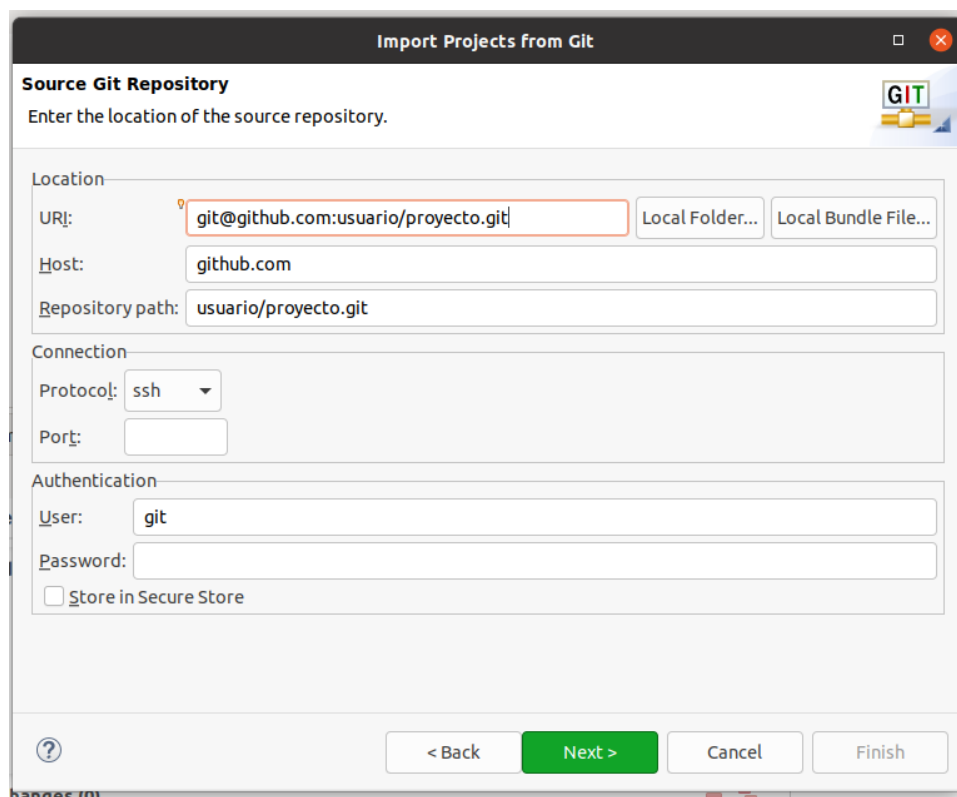
2.- Seleccionamos la importación desde Git de forma inteligente:



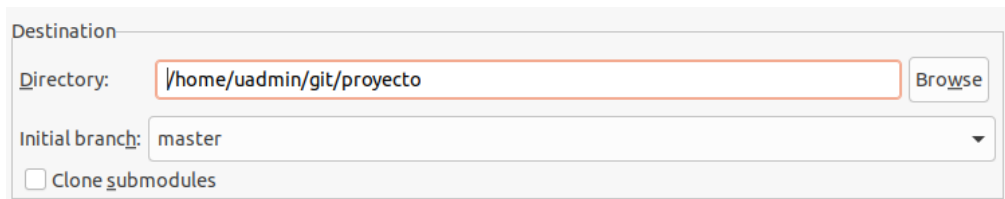
3.- Indicamos que queremos importar un repositorio externo:



4.- Indicamos la URL SSH de nuestro repositorio:



5.- Y, por último, indicamos el directorio local en el que lo queremos clonar:



Destination

Directory: [Browse](#)

Initial branch: master ▼

☐ Clone submodules

6.- Le damos a Finalizar y ya estaría clonado nuestro repositorio.