

Repositorios remotos en GitLab

Cuando se trabaja en colaboración con otras personas es cuando el sistema de control de versiones alcanza su pleno potencial. Ya hemos visto la mayor parte de las herramientas que necesitamos para ello, tan sólo nos falta ver cómo copiar los cambios realizados de un repositorio a otro.

Sistemas como Git ya nos permiten mover el trabajo realizado entre dos repositorios cualesquiera. Sin embargo, en la práctica es más sencillo establecer uno de ellos como repositorio central y tenerlo en la red en lugar de tu computadora particular. La mayoría de desarrolladores usan servicios de alojamiento en la red, tales como [GitHub](#), [BitBucket](#) o [GitLab](#), para alojar ese repositorio central; en la última sección de esta lección exploraremos los pros y los contras de cada uno de ellos.

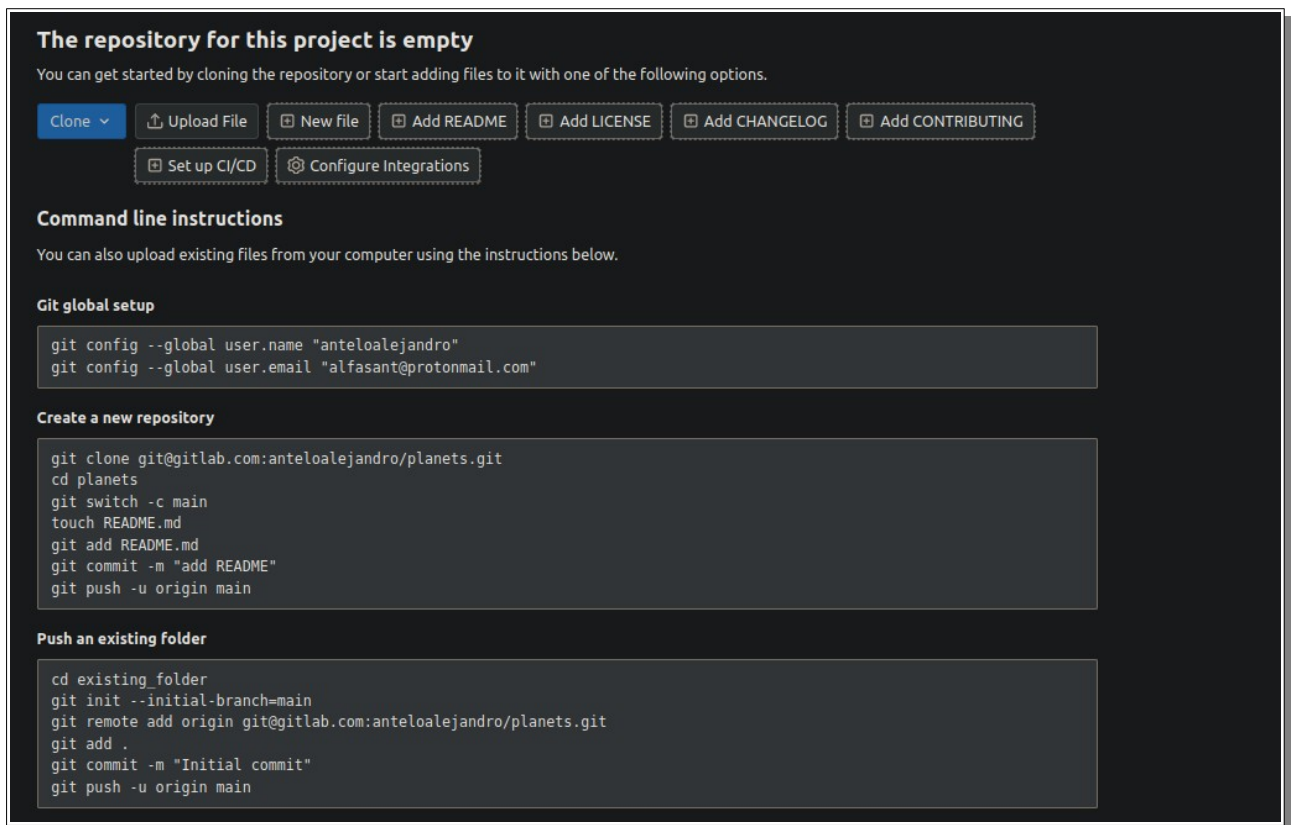
Empecemos por compartir con todos los demás los cambios que hemos realizado en nuestro proyecto actual. Para ello, ingresa en [tu cuenta de GitLab](#) y haz click en el icono que hay en la esquina superior derecha para crear un nuevo repositorio llamado planets:



Dale a tu repositorio el nombre “planets” y haz click en “Create project”:

A screenshot of the "Create project" form in GitLab. The form is dark-themed and contains several sections. The "Project name" section has a text input field with "planets" entered. The "Project URL" section has a text input field with "https://gitlab.com/anteloalejandro/" entered. The "Project slug" section has a text input field with "planets" entered. Below these, there's a link "Want to house several dependent projects under the same namespace? Create a group." The "Project description (optional)" section has a large text area with "Description format" as a placeholder. The "Visibility Level" section has two radio buttons: "Private" (selected) and "Public". The "Project Configuration" section has two checkboxes: "Initialize repository with a README" (checked) and "Enable Static Application Security Testing (SAST)" (unchecked). At the bottom, there are two buttons: "Create project" and "Cancel".

Tan pronto es creado el repositorio, GitLab muestra una página con una URL y algo de información sobre cómo configurar tu repositorio local.



The repository for this project is empty

You can get started by cloning the repository or start adding files to it with one of the following options.

[Clone](#) [Upload File](#) [New file](#) [Add README](#) [Add LICENSE](#) [Add CHANGELOG](#) [Add CONTRIBUTING](#)

[Set up CI/CD](#) [Configure Integrations](#)

Command line instructions

You can also upload existing files from your computer using the instructions below.

Git global setup

```
git config --global user.name "anteloalejandro"
git config --global user.email "alfasant@protonmail.com"
```

Create a new repository

```
git clone git@gitlab.com:anteloalejandro/planets.git
cd planets
git switch -c main
touch README.md
git add README.md
git commit -m "add README"
git push -u origin main
```

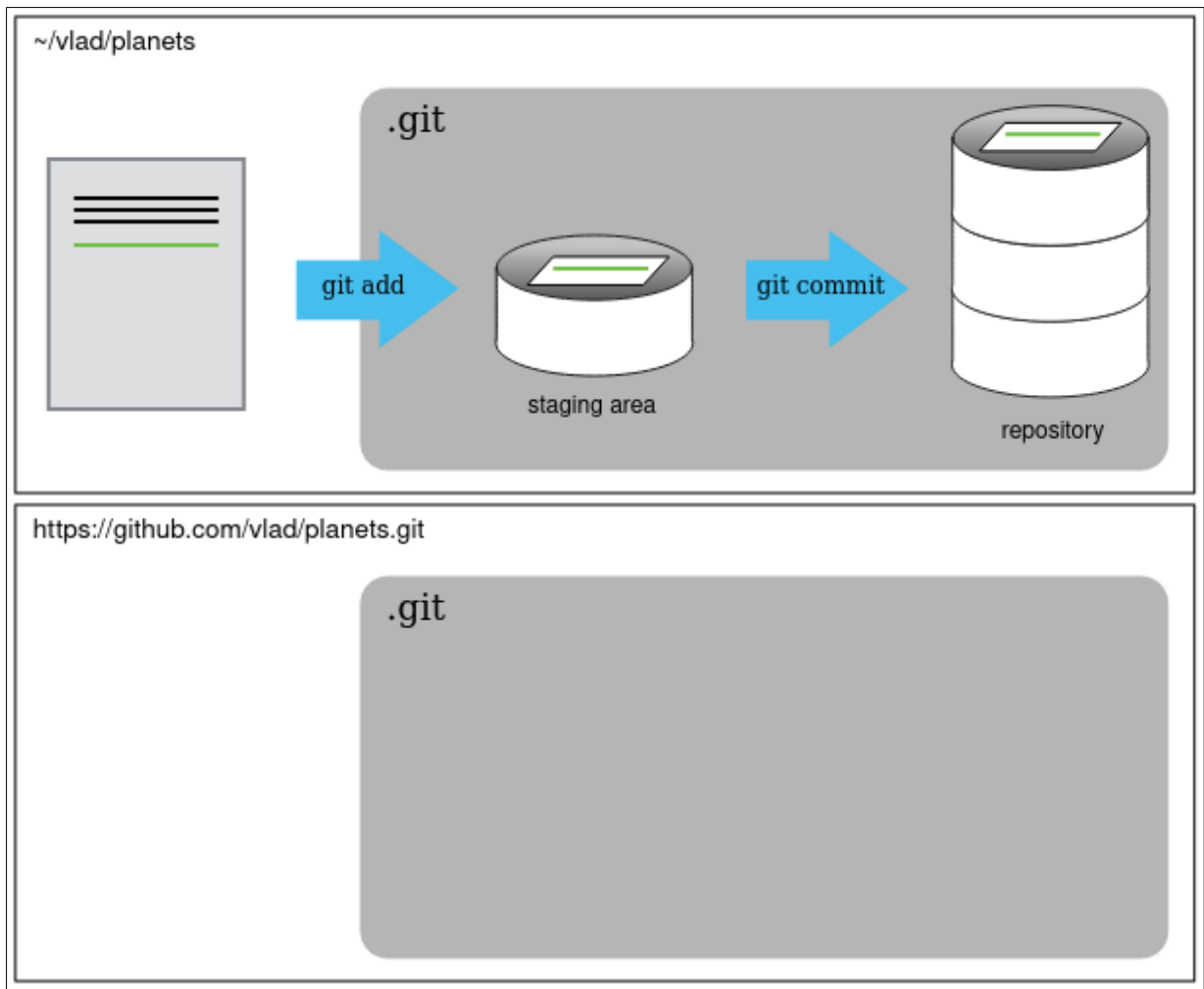
Push an existing folder

```
cd existing_folder
git init --initial-branch=main
git remote add origin git@gitlab.com:anteloalejandro/planets.git
git add .
git commit -m "Initial commit"
git push -u origin main
```

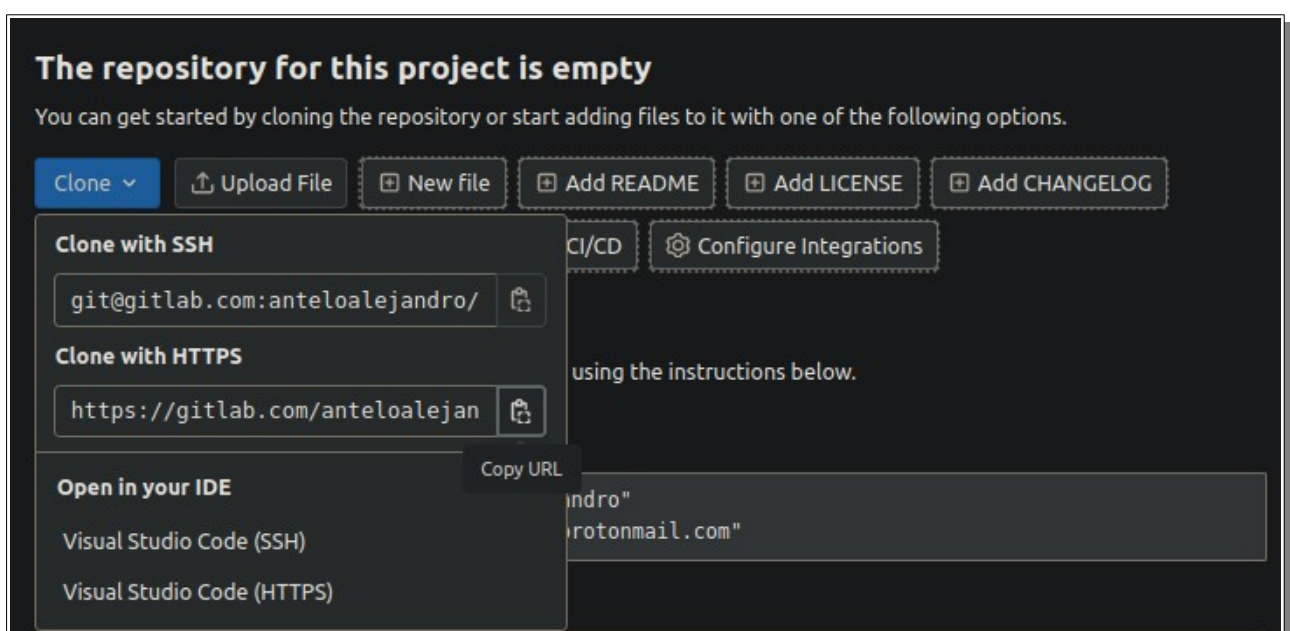
Esto en realidad ejecuta lo siguiente en los servidores de GitLab:

1. `$ mkdir planets`
2. `$ cd planets`
3. `$ git init`

Nuestro repositorio local contiene nuestro trabajo previo en `mars.txt`, pero el repositorio remoto en GitLab todavía no contiene ningún archivo:



El siguiente paso es conectar los dos repositorios. Ello se consigue convirtiendo al repositorio en GitLab en un [repositorio remoto](#) del repositorio local. La página de inicio del repositorio en GitLab incluye la secuencia de caracteres que necesitamos para identificarlo:



Copia dicha URL desde el navegador, ve al repositorio local planets y ejecuta allí este comando:

```
~/repo master > git remote add origin https://gitlab.com/anteloalejandro/planets.git
```

Asegúrate de usar la URL de tu repositorio en lugar de la de anteloalejandro: la única diferencia debería ser tu nombre de usuario en lugar de anteloalejandro.

Podemos comprobar que el comando ha funcionado bien ejecutando git remote -v:

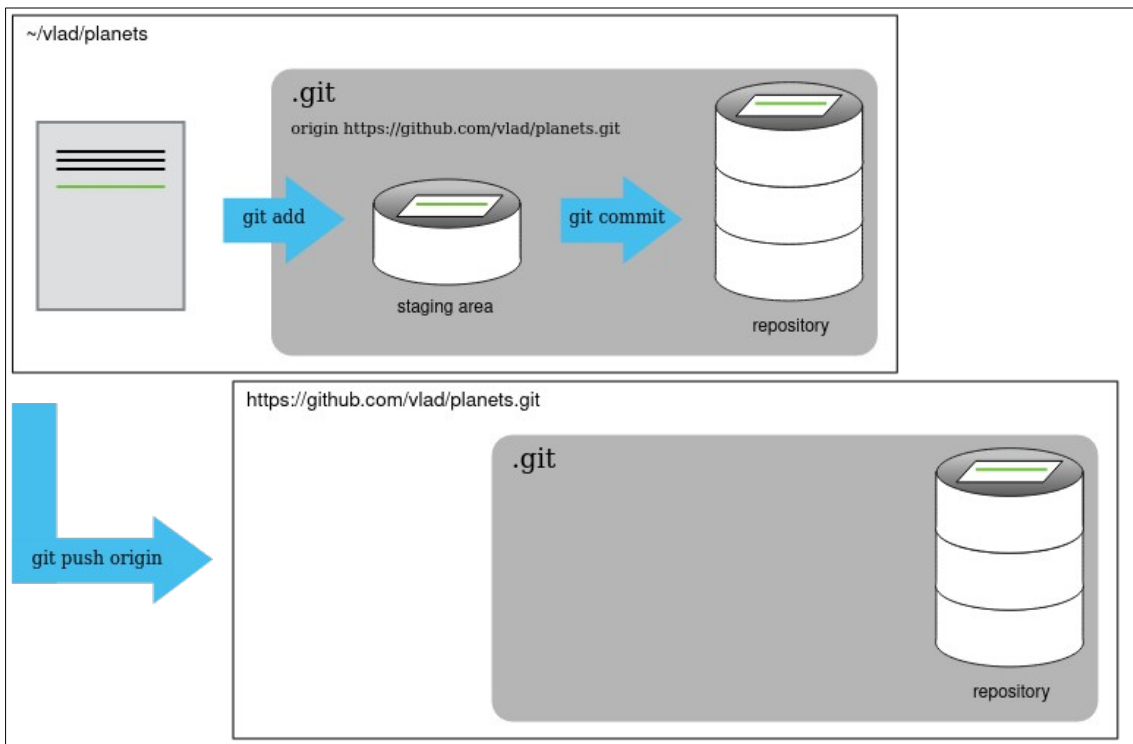
```
~/repo master > git remote -v
origin https://gitlab.com/anteloalejandro/planets.git (fetch)
origin https://gitlab.com/anteloalejandro/planets.git (push)
```

El nombre origin es un apodo local para tu repositorio remoto. Se puede usar cualquier otro nombre si se desea, pero origin es la elección más habitual.

Una vez seleccionado el apodo local origin, el siguiente comando enviará los cambios realizados en nuestro repositorio local al repositorio en GitLab:

```
~/repo master > git push origin master
Username for 'https://gitlab.com': anteloalejandro
Password for 'https://anteloalejandro@gitlab.com':
Enumerando objetos: 3, listo.
Contando objetos: 100% (3/3), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (2/2), listo.
Escribiendo objetos: 100% (3/3), 231 bytes | 231.00 KiB/s, listo.
Total 3 (delta 0), reusado 0 (delta 0)
To https://gitlab.com/anteloalejandro/planets.git
* [new branch]      master -> master
```

Nuestros repositorios local y remoto se encuentran ahora en el siguiente estado:



También podemos hacer **pull**, es decir, traernos cambios desde el repositorio remoto al repositorio local:

```
~/repo master > git pull origin master
Username for 'https://gitlab.com': anteloalejandro
Password for 'https://anteloalejandro@gitlab.com':
Desde https://gitlab.com/anteloalejandro/planets
* branch          master      -> FETCH_HEAD
Ya está actualizado.
```

En este caso, hacer **pull** no ha tenido ningún efecto porque los dos repositorios están ya sincronizados. Por el contrario, si alguien antes hubiera subido con **push** algunos cambios al repositorio en GitLab, este comando los habría incorporado a nuestro repositorio local.