

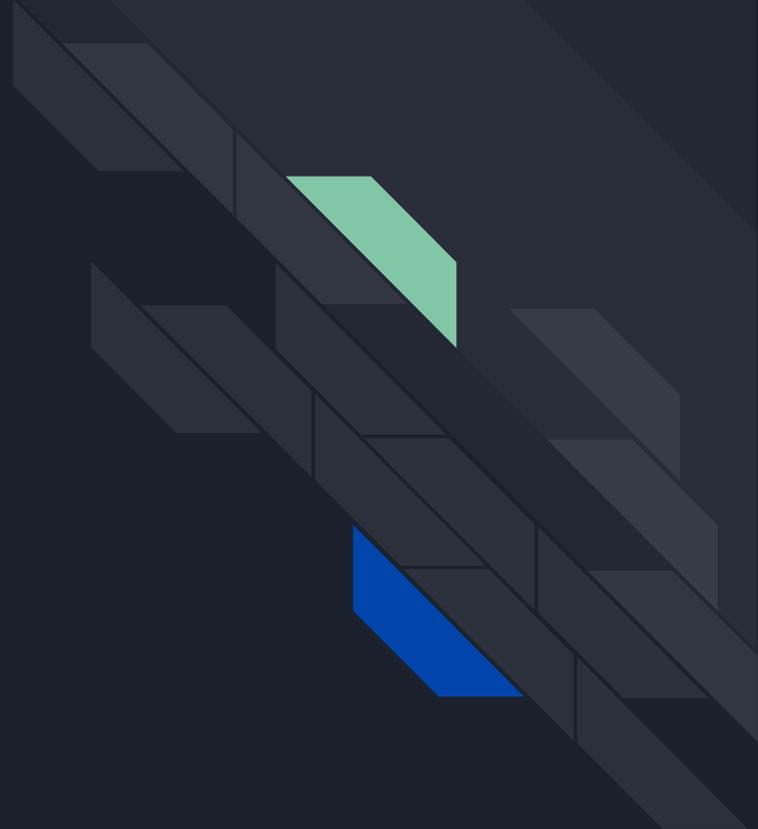


Introducción a Git y Gitlab en VSCode

Realizado por Alejandro Ruiz Becerra



Crear y configurar un repositorio





Crear e inicializar un proyecto

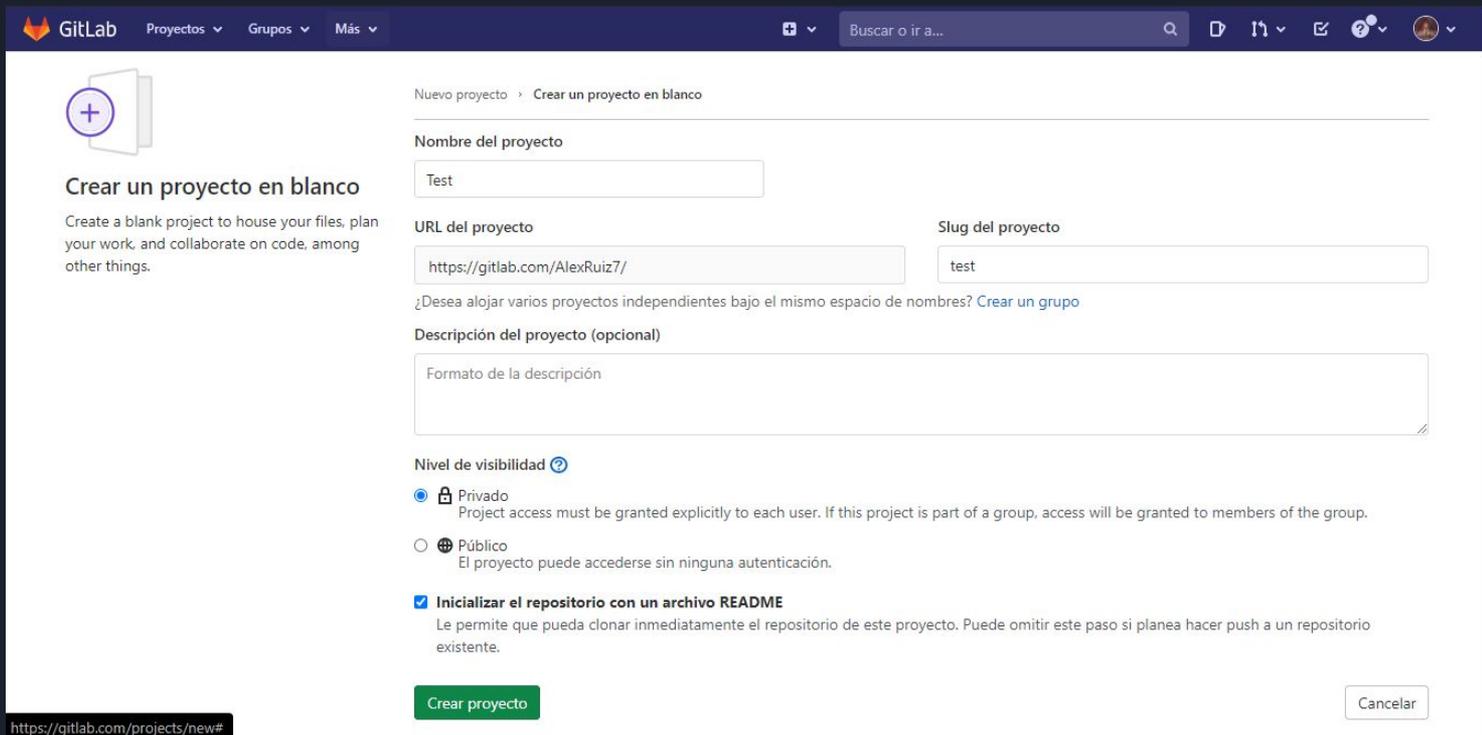
Esta parte está dedicada a los usuarios con permisos de tipo Maintainer y Owner.

Puedes saltar esta sección si no es tu caso.

Para crear e inicializar un repositorio correctamente seguimos los siguientes pasos:

1. Hacemos click en “Nuevo proyecto”
2. Seleccionamos “Crear un proyecto en blanco” en la nueva ventana que nos aparece.
3. Configuramos el nombre y la visibilidad del proyecto.
4. Activamos la casilla “Inicializar el repositorio con un archivo README”

Inicializa el repositorio con un README



Crear un proyecto en blanco

Create a blank project to house your files, plan your work, and collaborate on code, among other things.

Nuevo proyecto > Crear un proyecto en blanco

Nombre del proyecto

URL del proyecto

Slug del proyecto

¿Desea alojar varios proyectos independientes bajo el mismo espacio de nombres? [Crear un grupo](#)

Descripción del proyecto (opcional)

Nivel de visibilidad 

Privado
Project access must be granted explicitly to each user. If this project is part of a group, access will be granted to members of the group.

Público
El proyecto puede accederse sin ninguna autenticación.

Inicializar el repositorio con un archivo README
Le permite que pueda clonar inmediatamente el repositorio de este proyecto. Puede omitir este paso si planea hacer push a un repositorio existente.

<https://gitlab.com/projects/new#>



Configuración del repositorio

Una vez tenemos el proyecto creado e inicializado con un README es hora de subir código.

Gitlab por defecto solo permite a los usuarios con rol de Maintainer “tocar” el código, por lo que el rol de desarrollador (el rol por defecto cuando se añaden nuevos miembros al proyecto) resulta bastante inútil con esta configuración.

Para cambiar esta configuración, vamos al proyecto, y en la barra de navegación del lateral izquierdo, hacemos click en “Configuración” y luego en “Repositorio”.

Veremos la siguiente configuración:

Configuración del repositorio

The screenshot shows the GitLab repository configuration page. The left sidebar contains navigation options: Paquetes y registros, Analíticas, Wiki, Fragmentos de código, Miembros, Configuración (General, Integraciones, Webhooks, Repositorio, CI / CD, Operaciones, Páginas), and a button to collapse the sidebar. The main content area is divided into two sections: 'Rama por defecto' and 'Protected branches'. The 'Rama por defecto' section has a 'Default branch' dropdown set to 'master' and a checked option for 'Auto-close referenced issues on default branch'. The 'Protected branches' section lists rules for the 'master' branch, including permissions for merge and push. A table at the bottom summarizes these permissions for the 'master' branch, showing 'Maintainers' for both merge and push, and an 'Unprotect' button. On the right side, there are 'Contraer' buttons for each section and explanatory text in Spanish.

Rama por defecto Contraer

Set the default branch for this project. All merge requests and commits are made against this branch unless you specify a different one.

Default branch

master

Auto-close referenced issues on default branch
When merge requests and commits in the default branch close, any issues they reference also close. ?

Guardar los cambios

Protected branches Contraer

Keep stable branches secure, and force developers to use merge requests. [What are protected branches?](#)

By default, protected branches protect your code and:

- Allow only users with Maintainer [permissions](#) to create new protected branches.
- Allow only users with Maintainer permissions to push code.
- Prevent **anyone** from force-pushing to the branch.
- Prevent **anyone** from deleting the branch.

Rama	Permitido merge	Permitido push	
master <small>default</small>	Maintainers	Maintainers	Unprotect

Debe existir una rama por defecto.

La rama por defecto solo se crea si el repositorio ha sido inicializado, es decir, contiene archivos (por eso hemos creado el README)

La rama principal (master) queda protegida por defecto, de forma que solo los Maintainers pueden subir archivos.

Esto no es lo que queremos.



Configuración del repositorio

Como vemos la rama master queda protegida de forma que solo los Maintainers pueden realizar push y merge.

Tenemos 3 opciones para solventar este problema:

1. Dar permisos a los desarrolladores.
2. Desproteger la rama master haciendo click en el botón “Unprotect”.
3. Ascender a todos los miembros a Maintainers.

Configuración del repositorio

Opción 1: Dar permisos a los desarrolladores.

Protected branches

Keep stable branches secure, and force developers to use merge requests. [What are protected branches?](#)

By default, protected branches protect your code and:

- Allow only users with Maintainer [permissions](#) to create new protected branches.
- Allow only users with Maintainer permissions to push code.
- Prevent **anyone** from force-pushing to the branch.
- Prevent **anyone** from deleting the branch.

Rama	Permitido merge	Permitido push	
master <small>default</small>	Developers + Mai... ▾	Developers + Mai... ▾	<button>Unprotect</button>
		<div><p>Roles</p><ul style="list-style-type: none">Maintainers✓ Developers + MaintainersNo one</div>	



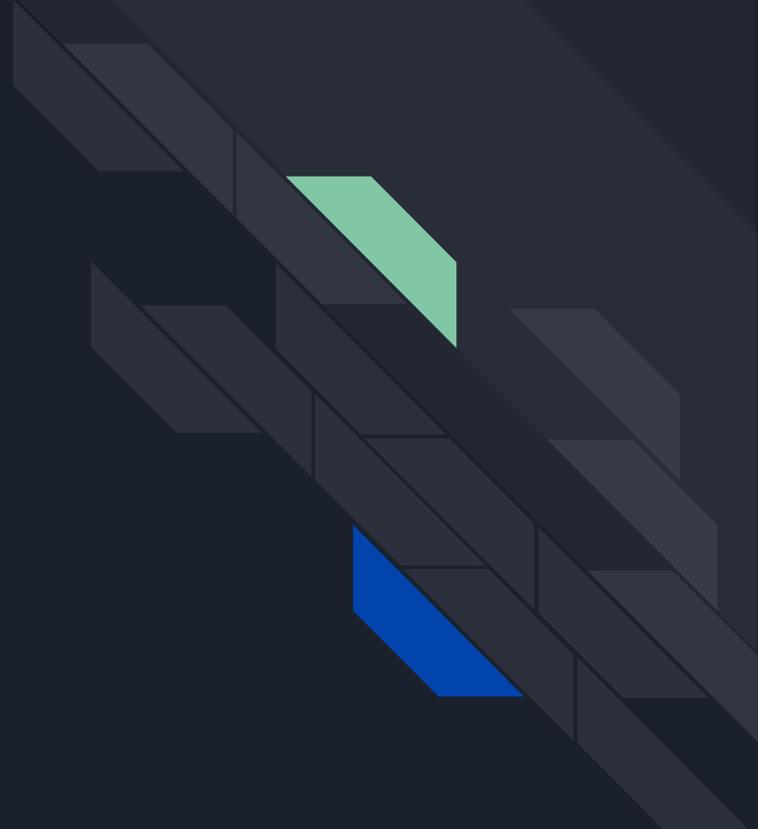
Configuración del repositorio

Opción 3: Ascender miembros a Maintainers

- En la barra lateral cambiamos a la sección “Members”
- Cambiamos los roles de los miembros a Maintainers



Configuración de SSH





ssh-keygen

La mejor forma de comunicarnos con *git* es usando *ssh*.

La herramienta *ssh-keygen* nos genera un par de claves únicas y relacionadas.

- La clave **privada** (por defecto *id_rsa*) **no debe compartirse con nadie**.
- La clave pública (por defecto *id_rsa.pub*) es la que debemos proporcionar a Gitlab.

Cada vez que nos conectemos a Gitlab (pull, push...), se comprueba que la clave pública proporcionada es correcta, comparándola con la clave privada de nuestro ordenador.

Para generar el par de claves escribimos en una terminal *ssh-keygen* (*ssh-keygen.exe* en Windows) y pulsamos Intro hasta completar el proceso.



ssh-keygen

Al completar el proceso el par de claves se habrá generado en la ruta por defecto. *ssh-keygen* también nos dice dónde ha almacenado las claves.

La ruta por defecto es:

- Linux:
 - `~/.ssh/`
- Windows:
 - `X:\Users\<<usuario>\.ssh`



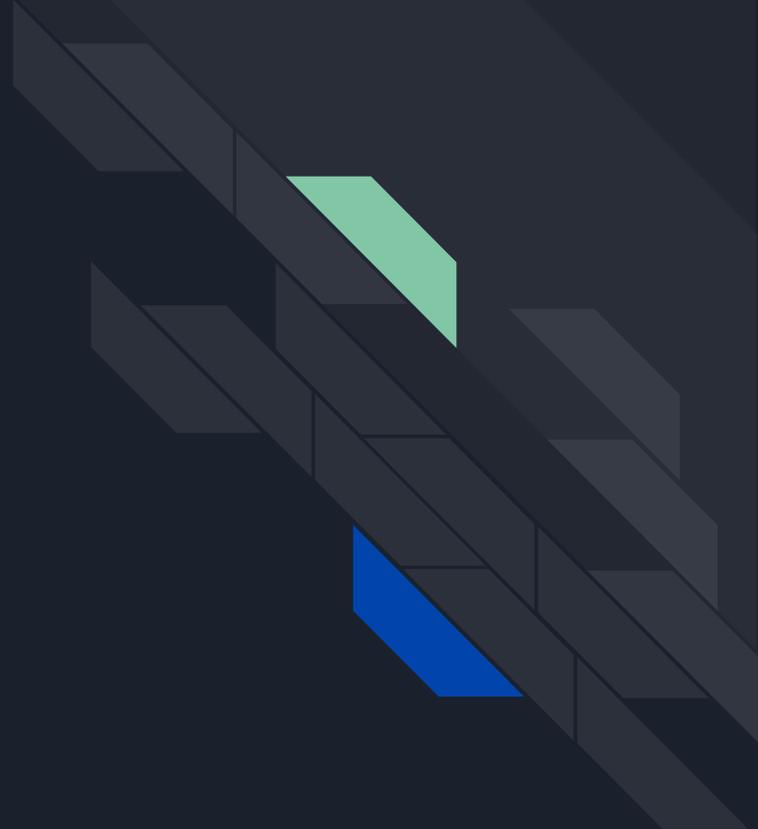
Añadir la clave pública en Gitlab

- Abrimos la clave pública (*id_rsa.pub*) con cualquier editor de texto y copiamos todo su contenido.
- En Gitlab, vamos a nuestra configuración de usuario, haciendo click en nuestro icono en la esquina superior derecha.
- Vamos a la sección de “Claves SSH”
- Pegamos nuestra **clave pública** y la añadimos.

La primera vez que nos comuniquemos con Git mediante SSH nos pedirá si queremos añadirlo a la lista de hosts conocidos, indicamos que sí escribiendo YES.



Visual Studio Code





VS Code: clonar un repositorio

Para clonar un repositorio usando VS Code hacemos lo siguiente:

- En Gitlab:
 - Vamos a la página del repositorio en Gitlab y hacemos click en “Clonar”.
 - Clonamos mediante SSH, así que hacemos click en copiar la URL SSH.
- En VS Code:
 - Opción 1: ‘Ctrl + Shift + P’ y escribimos “clone”, click en git:clone y pegamos la URL.
 - Opción 2: abrimos la sección de “Source Control” en la barra izquierda y hacemos click en “Clone Repository”. Pegamos la URL.

Copiar URL del repositorio (SSH)

The screenshot shows the GitLab interface for a repository named 'test'. The left sidebar contains navigation options like 'Resumen del proyecto', 'Detalles', 'Actividad', 'Versiones', 'Repositorio', 'Incidencias', 'Merge requests', 'CI / CD', 'Seguridad y cumplimiento', 'Operaciones', 'Paquetes y registros', and 'Analíticas'. The main content area displays the repository details, including the commit history and a table of files. A dropdown menu is open over the 'Clonar' button, showing options for cloning via SSH, HTTPS, and opening in an IDE. The SSH URL is highlighted with a tooltip that says 'Copiar la URL'.

GitLab Proyectos Grupos Más

test

Resumen del proyecto

Detalles

Actividad

Versiones

Repositorio

Incidencias 0

Merge requests 0

CI / CD

Seguridad y cumplimiento

Operaciones

Paquetes y registros

Analíticas

Contraer la barra lateral

test

test ID de proyecto: 24724510

1 Commit 1 Branch 0 Etiquetas 143 KB Archivos 143 KB Almacenamiento

master test / +

Historial Buscar archivo Web IDE Clonar

Initial commit
Álex Autor 1 hour ago

README Añadir LICENSE Añadir CHANGELOG Añadir CONTRIBUTING

Configurar CI/CD

Nombre	Último cambio
README.md	Initial commit 1 hour ago

README.md

test

Clonar con SSH
git@gitlab.com:AlexRuiz7/test

Clonar con HTTPS
https://gitlab.com/AlexRuiz7/

Open in your IDE
Visual Studio Code

Copiar la URL

Clonar repositorio usando VS Code

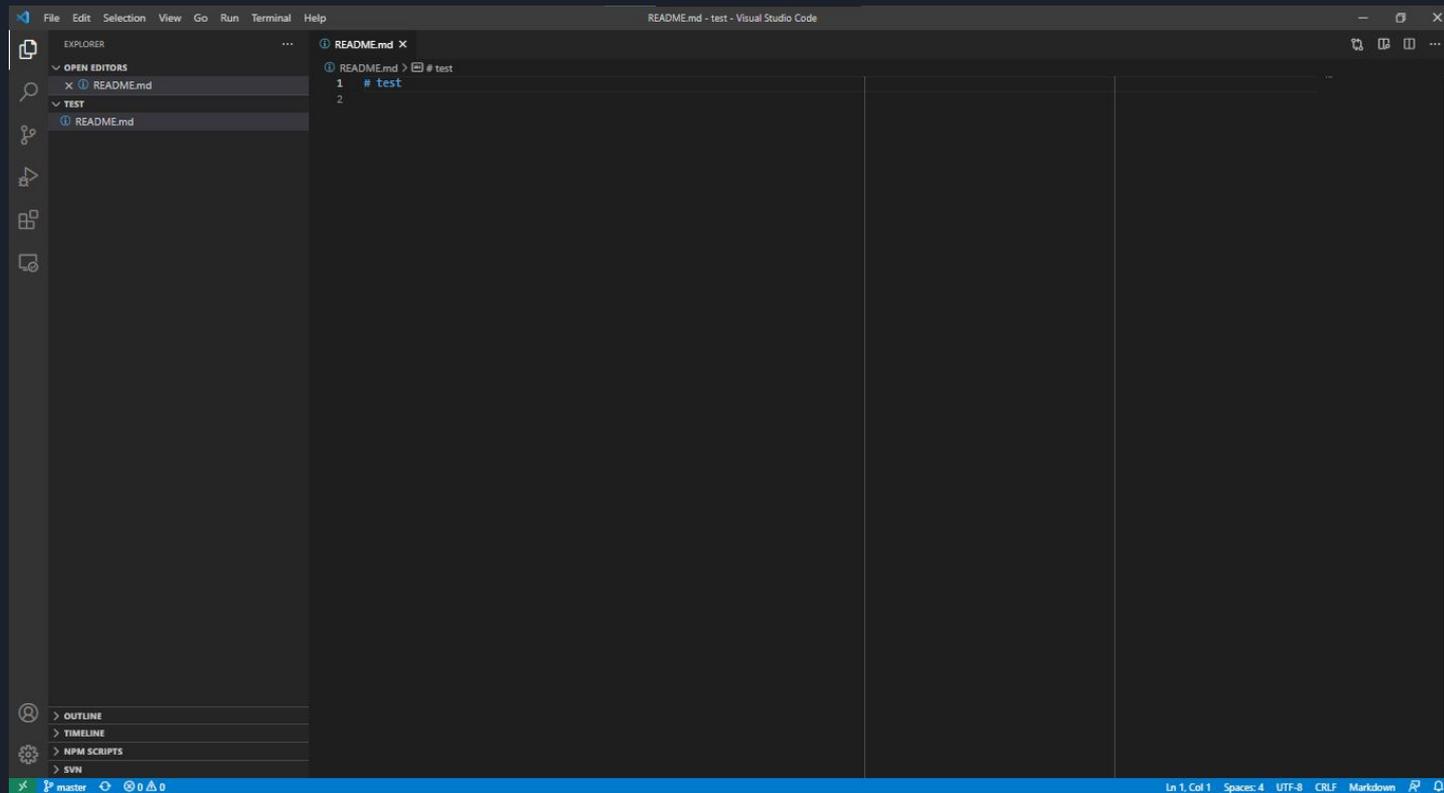
The image shows the Visual Studio Code interface with two methods for cloning a repository highlighted:

- Opción 2:** In the left sidebar, the **Source Control** view is active. The **Clone Repository** button is highlighted with a red box. An arrow points from this button to the terminal window.
- Opción 1:** The **Command Palette** is open, showing the **Show All Commands** option with the keyboard shortcut **Ctrl + Shift + P** highlighted with a red box. An arrow points from this option to the terminal window.

The terminal window at the top shows the following text:

```
git@gitlab.com:AlexRuiz7/test.git
Clone from URL git@gitlab.com:AlexRuiz7/test.git
Clone from GitHub
```

Resultado



The image shows a screenshot of the Visual Studio Code editor interface. The title bar indicates the file is 'README.md - test' and the application is 'Visual Studio Code'. The Explorer sidebar on the left shows a project structure with 'README.md' under a 'TEST' folder. The main editor area displays the content of the selected file, which is a single line of text: '# test'. The status bar at the bottom shows the current file is on 'Ln 1, Col 1', using 'UTF-8' encoding with 'CRLF' line endings, and is in 'Markdown' mode. The bottom-left corner of the status bar shows the current branch is 'master'.

```
README.md > # test
1 # test
2
```

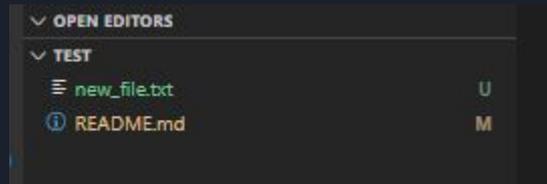
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Markdown



Subir cambios a Git usando VS Code git commit & git push

Visual Studio Code nos irá coloreando los ficheros en distintos colores según su estado comprado con lo que hay subido en Git:

- Verde Untracked ficheros nuevos
- Naranja claro Modified ficheros modificados
- Rojo Deleted ficheros eliminados

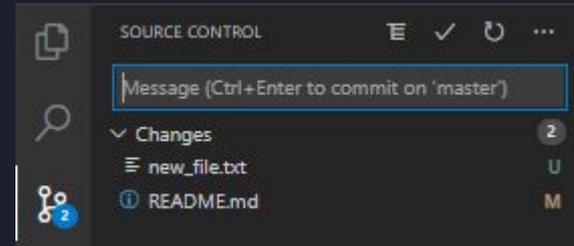


Subir cambios a Git usando VS Code

git commit & git push

Para los subir los cambios a git hay que realizar 3 pasos:

1. Añadir los cambios (stash - git add)
2. Guardar los cambios (commit - git commit)
3. Subir los cambios (push - git push)



En VS Code vamos a la sección de “Source Control” y hacemos click en el tick para realizar un commit.

Nos preguntará si queremos añadir todos los cambios detectados al commit, le decimos que sí. Luego, nos pedirá un mensaje para el commit.

Alternativamente podemos acceder a estas y a más opciones haciendo click en los 3 puntos suspensivos.



Subir cambios a Git usando VS Code git commit & git push

Una vez hemos realizado el *commit* podemos subirlo a Gitlab realizando un *push*.

Para ello hacemos click en los puntos suspensivos y luego en Push.

Listo, nuestros cambios ya se han subido a Gitlab.

Podemos hacer cuantos commits queramos, sin necesidad de subirlos uno a uno. Cuando hagamos push se subirán todos los commits que haya hasta ese momento.

Resultado en Gitlab

The screenshot shows the GitLab web interface for a repository named 'test'. The top navigation bar includes the GitLab logo, 'Proyectos', 'Grupos', and 'Más' menus, along with a search bar and user profile icons. The left sidebar contains navigation options: 'Resumen del proyecto', 'Repositorio', 'Archivos', 'Commits', 'Ramas', 'Etiquetas', 'Contribuidores', 'Gráfico', 'Comparar', 'Incidencias', 'Merge requests', 'CI / CD', and 'Seguridad y cumplimiento'. The main content area displays the repository path 'Álex > test > Repositorio' and a commit history table. The commit history table has three columns: 'Nombre', 'Último cambio', and 'Última actualización'. Below the table, the content of the 'README.md' file is shown, which contains the text 'Hello World!'.

GitLab Proyectos Grupos Más

Buscar o ir a...

Álex > test > Repositorio

master test / +

Historial Buscar archivo Web IDE Clonar

01a6b352

Nombre	Último cambio	Última actualización
README.md	Add Readme and New File	3 minutos ago
new_file.txt	Add Readme and New File	3 minutos ago

README.md

```
Hello World!
```

https://gitlab.com/AlexRuiz7/test