

https://en.wikipedia.org/wiki/File:Ancient_Egypt_rope_usage.jpg

Trabajo en grupo con Unity y GitHub

Sistemas de control de versiones

- En cualquier desarrollo de un proyecto informático colaborativo es esencial usar un sistema de control de versiones
- Un sistema de control de versiones permite almacenar la historia de todos los cambios realizados sobre el código de una aplicación
 - También se pueden almacenar recursos, como imágenes y modelos
- Cualquier miembro del equipo puede obtener los ficheros en el estado actual, realizar cambios y subir dichos cambios al repositorio
 - A partir de ese momento, los demás miembros del equipo podrán ver los cambios realizados

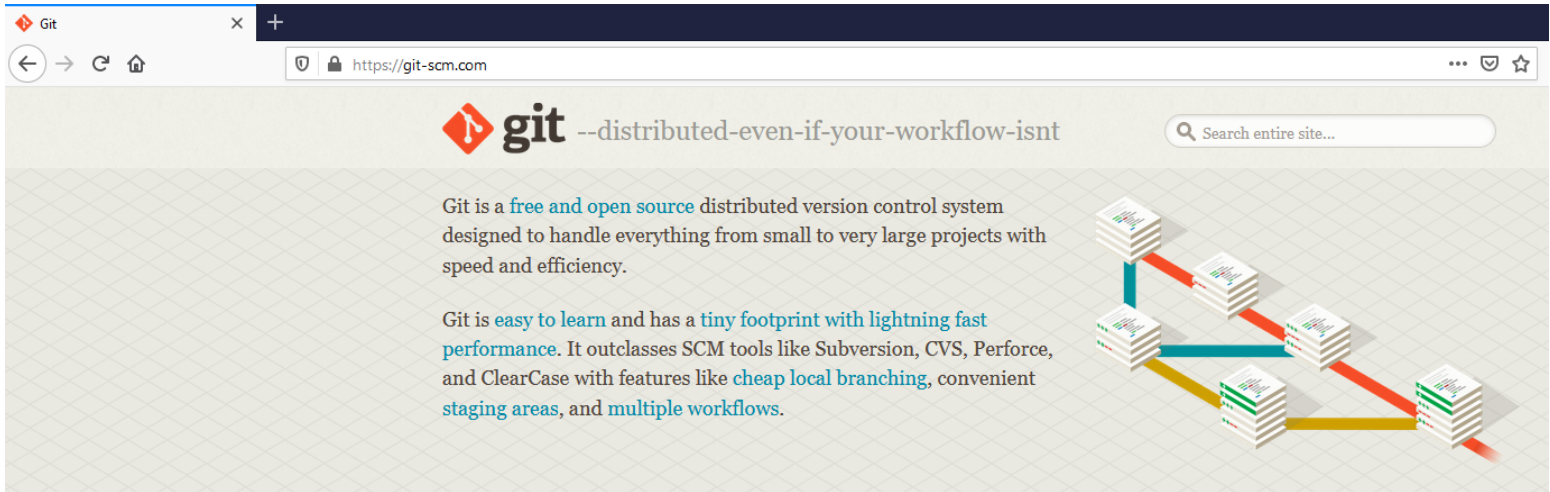
Sistemas de control de versiones

- Las ventajas de un sistema de control de versiones son:
 - El código está almacenado en un servidor, por lo que si un desarrollador pierde su máquina, sólo se pierde los cambios realizados desde la última sincronización
 - Facilita el trabajo en grupo
 - Permite volver a cualquier versión anterior
 - Se pueden crear versiones alternativas, invisibles al resto del equipo, hasta completar una tarea, y luego combinarla con el código común
 - Varios miembros pueden trabajar sobre el mismo fichero
 - Muestra un historial del desarrollo del proyecto

Sistemas de control de versiones

- Casos de uso típicos:
 - Bajarse la versión actual del código a una máquina nueva
 - Subir cambios realizados localmente al servidor
 - Descartar cambios locales
 - Volver a una versión indicada (por fecha o identificador)
 - Empezar un subproyecto para probar nuevo código, sin afectar al general
 - Etiquetar el estado actual del proyecto

Git



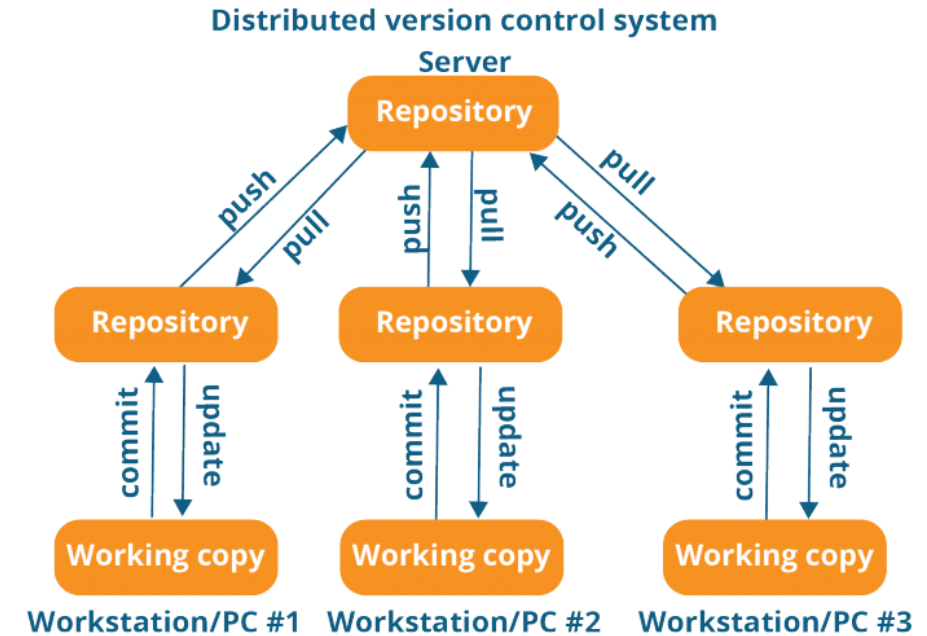
- Creado en 2005 por Linus Torvalds, creador de Linux,
 - Salió de la comunidad de desarrollo de Linux
 - Diseñado para hacer control de versiones en el kernel de Linux
- Metas de Git:
 - Velocidad
 - Soporte para desarrollo no lineal (miles de ramas paralelas)
 - Completamente distribuido
 - Capaz de manejar grandes proyectos de manera eficiente

Git

- Sitio web de Git: <http://git-scm.com/>
- Libro en línea gratuito: <http://git-scm.com/book>
- Prontuario de Git en <https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf>
- GUIs para Windows: <https://git-scm.com/download/gui/windows>

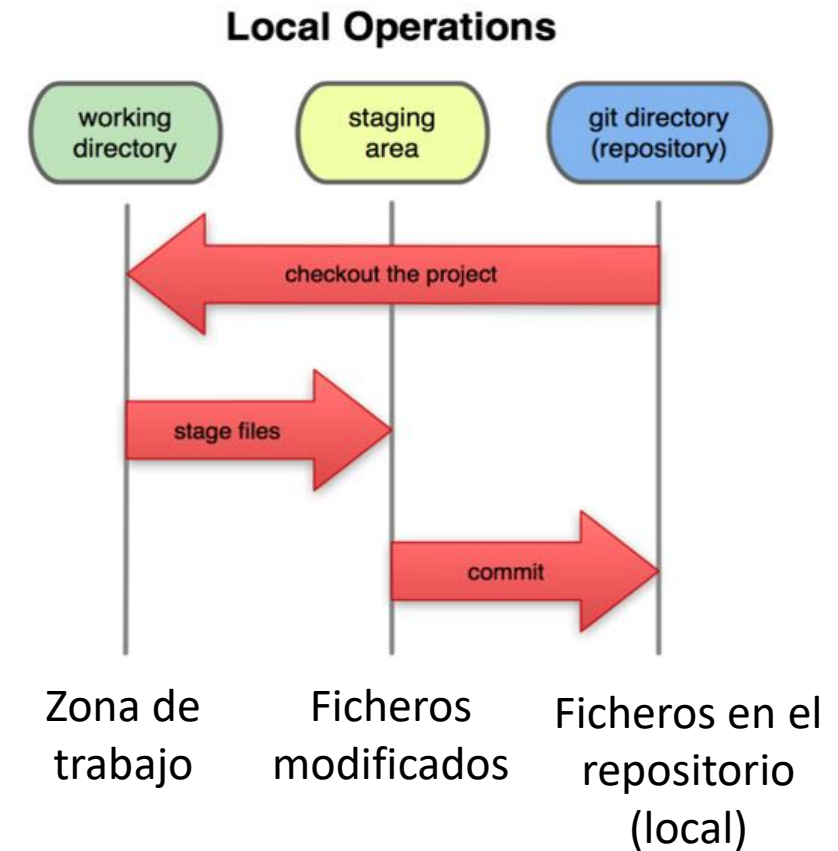
Git

- Flujo de trabajo
 - Inicialmente, se clona/copia la versión deseada a un directorio local desde el repositorio remoto
 - El repositorio local es una copia completa de todo en el servidor remoto
 - La copia local es "tan buena" como cualquier otra



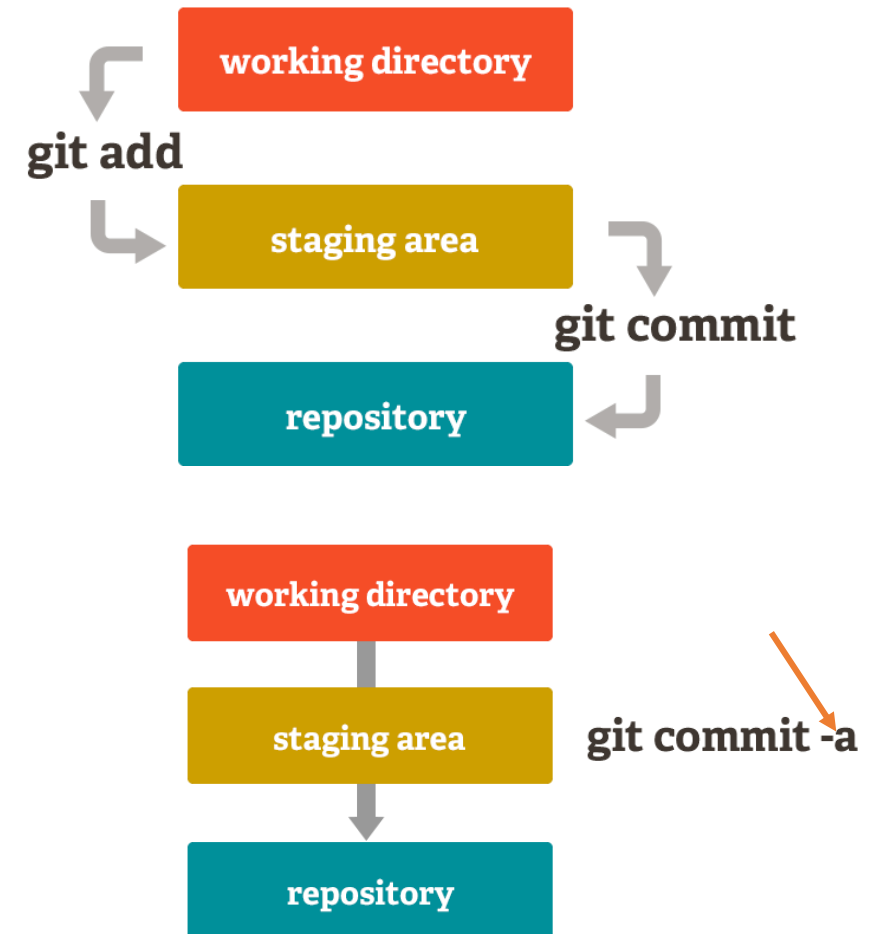
Git

- Flujo de trabajo
 - Mucho del trabajo se hace sobre la copia local:



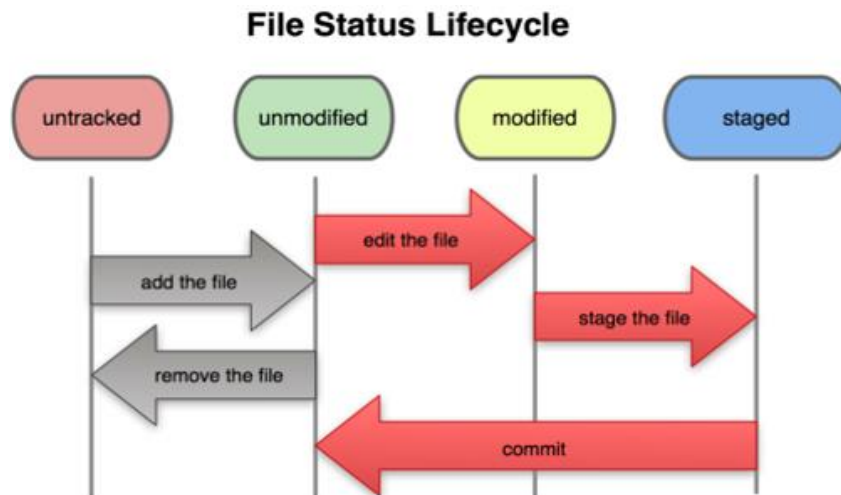
Git

- Área de ensayo (staging)
 - Área intermedia donde los ficheros pueden revisarse antes de completar el commit
 - Así se puede determinar qué cambios nos interesa guardar el repositorio
 - Se puede ignorar esta característica simplemente agregando una '-a' a la línea de comando de confirmación para agregar todos los cambios a todos los archivos



Git

- En su copia local en git, los usuarios pueden:
 - Modificar los ficheros en su directorio de trabajo
 - Organizar los ficheros, agregando instantáneas de ellos al área de ensayo
 - Hacer *commit*, que toma los archivos en el área de ensayo y almacena esa instantánea permanentemente en el directorio Git local



- *untracked*: un fichero ajeno al sistema de versiones
- *unmodified*: un fichero bajo revisión, sin modificar
- *modified*: un fichero bajo revisión, modificado
- *staged*: un fichero bajo revisión modificado, listo para subir al repositorio

Git

Git Cheat Sheet



GIT BASICS

<code>git init <directory></code>	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
<code>git clone <repo></code>	Clone repo located at <repo> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
<code>git config user.name <name></code>	Define author name to be used for all commits in current repo. Devs commonly use <code>—global</code> flag to set config options for current user.
<code>git add <directory></code>	Stage all changes in <directory> for the next commit. Replace <directory> with a <file> to change a specific file.
<code>git commit -m "message"</code>	Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.
<code>git status</code>	List which files are staged, unstaged, and untracked.
<code>git log</code>	Display the entire commit history using the default format. For customization see additional options.
<code>git diff</code>	Show unstaged changes between your index and working directory.

UNDOING CHANGES

<code>git revert <commit></code>	Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch.
<code>git reset <file></code>	Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
<code>git clean -n</code>	Shows which files would be removed from working directory. Use the <code>-f</code> flag in place of the <code>-n</code> flag to execute the clean.

REWRITING GIT HISTORY

<code>git commit —amend</code>	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
<code>git rebase <base></code>	Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
<code>git reflog</code>	Show a log of changes to the local repository's HEAD. Add <code>—relative-date</code> flag to show date info or <code>—all</code> to show all refs.

GIT BRANCHES

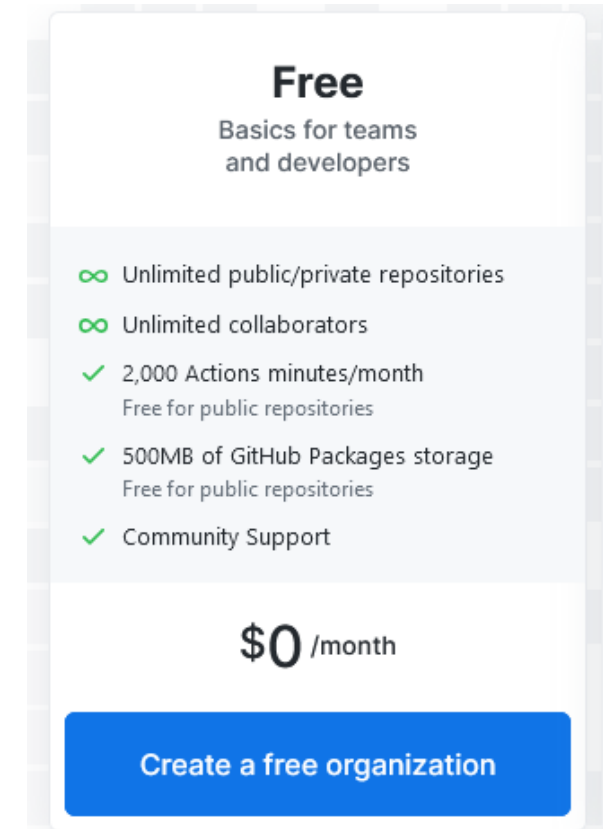
<code>git branch</code>	List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.
<code>git checkout -b <branch></code>	Create and check out a new branch named <branch>. Drop the <code>-b</code> flag to checkout an existing branch.
<code>git merge <branch></code>	Merge <branch> into the current branch.

REMOTE REPOSITORIES

<code>git remote add <name> <url></code>	Create a new connection to a remote repo. After adding a remote, you can use <name> as a shortcut for <url> in other commands.
<code>git fetch <remote> <branch></code>	Fetches a specific <branch>, from the repo. Leave off <branch> to fetch all remote refs.
<code>git pull <remote></code>	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
<code>git push <remote> <branch></code>	Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

GitHub

- github.com es un sitio para el almacenamiento en línea de repositorios Git
 - github ≠ git
- Muchos proyectos de código abierto lo usan, como el kernel de Linux
- Ofrece bastante funcionalidad en la versión gratuita
- Dispone de una versión para estudiantes, con más funcionalidad:
 - <https://education.github.com/students>



The image shows a screenshot of the GitHub Free tier pricing page. It features a white card with a light gray border. At the top, the word "Free" is in bold, followed by "Basics for teams and developers". Below this, a list of features is shown with green checkmarks and infinity symbols. At the bottom, the price "\$0 /month" is displayed, and a blue button says "Create a free organization".

Free
Basics for teams and developers

- ∞ Unlimited public/private repositories
- ∞ Unlimited collaborators
- ✓ 2,000 Actions minutes/month
Free for public repositories
- ✓ 500MB of GitHub Packages storage
Free for public repositories
- ✓ Community Support

\$0 /month

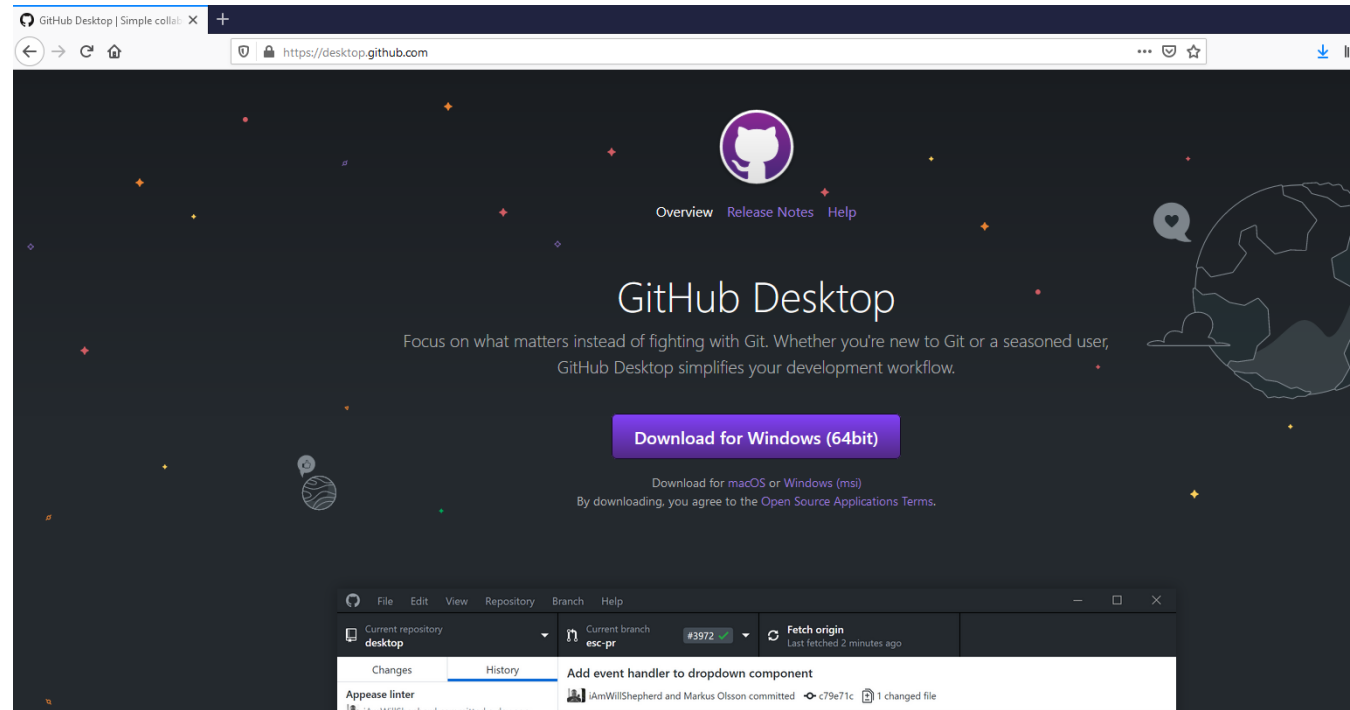
Create a free organization

GitHub Desktop

- Es una aplicación de escritorio para interactuar con GitHub (y, en general, con cualquier servidor git)
- Ofrece una interfaz gráfica amigable para realizar todas las tareas habituales que se realizan durante la construcción de un proyecto

GitHub Desktop

- Instalación:
 - <https://desktop.github.com>



GitHub Desktop

Welcome to GitHub Desktop

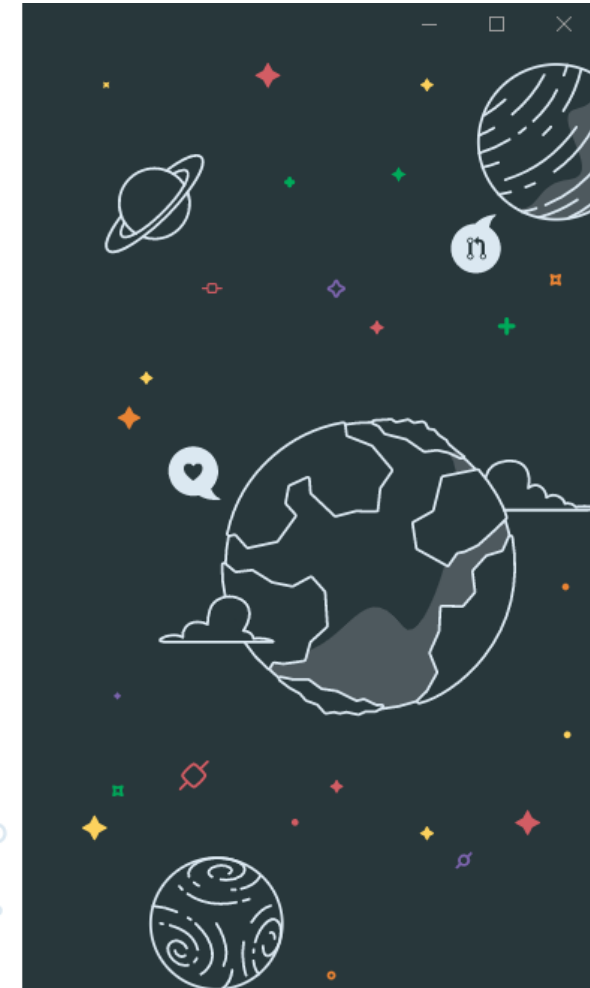
GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise Server. Sign in below to get started with your existing projects.

Crear un nuevo usuario → New to GitHub? [Create your free account.](#)

Acceder mediante el navegador → [Sign in to GitHub.com](#)

Acceder mediante login/password → [Sign in to GitHub.com using your username and password](#)

[Skip this step](#)



GitHub Desktop

- Crear un nuevo usuario

Nombre de usuario (único)

Correo electrónico

Clave

Setting up GitHub Desktop - G...Join GitHub · GitHub

https://github.com/join?source=github-desktop

Search GitHubSign in

Join GitHub

Create your account

Username *

TheRealVideoGameProgrammer

Email address *

trvgp@upv.es

Password *

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.[Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

Verify your account

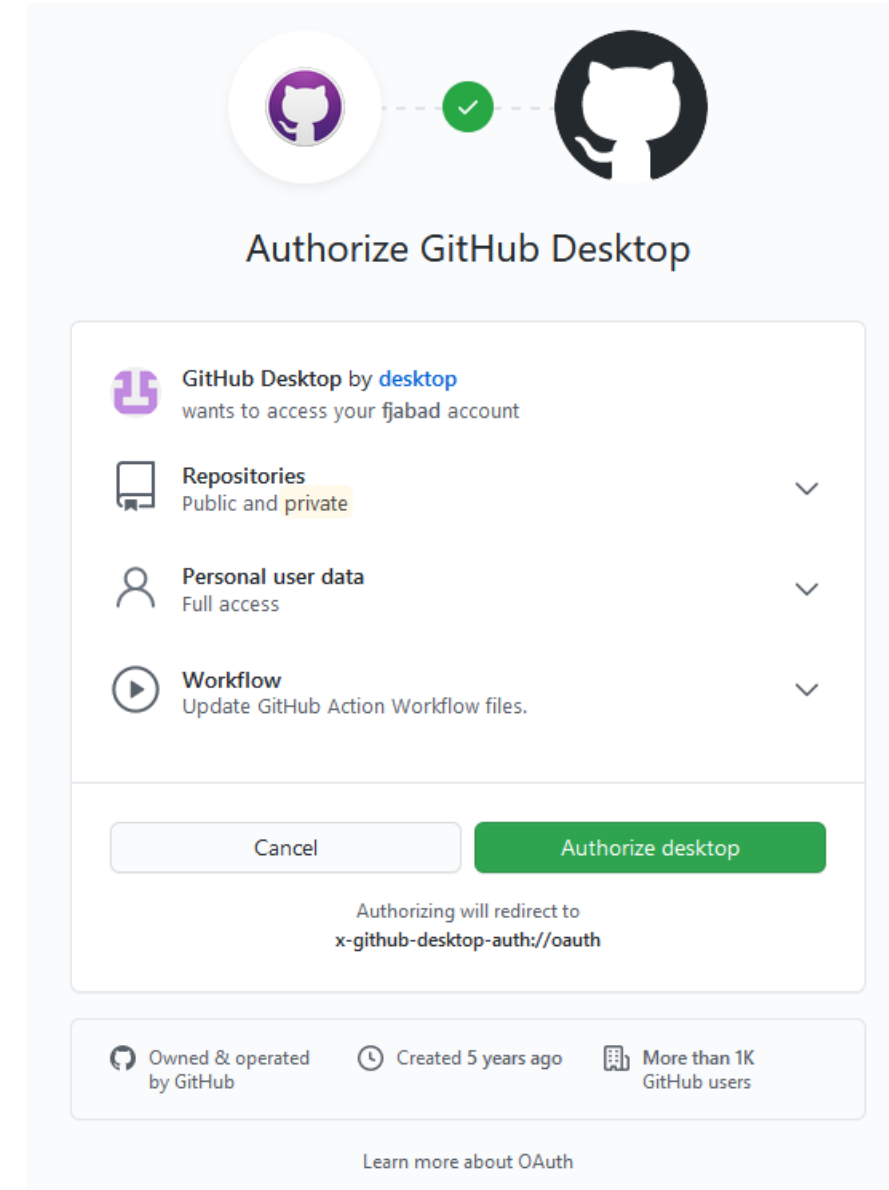
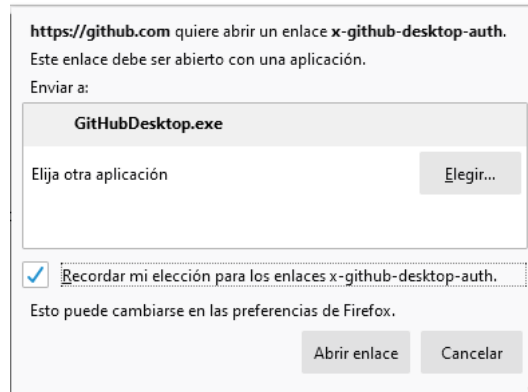
Elija la galaxia en espiral

?
↺
🔊

Create account

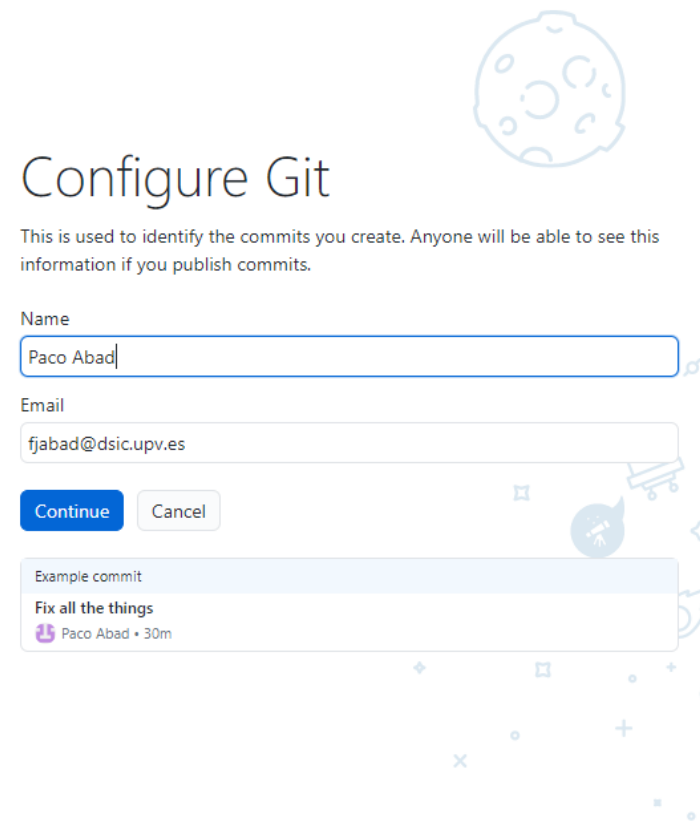
GitHub Desktop

- Al entrar en GitHub, se puede mostrar la siguiente página que pide permisos de acceso para GitHub Desktop
 - Pulsar *Authorize desktop*
- Y luego, seleccionar la aplicación para gestionar los enlaces a *github-desktop*



GitHub Desktop

- La primera vez que entramos en GitHub Desktop, nos pregunta qué nombre y qué correo queremos que aparezca en los commits



The image shows the 'Configure Git' window in GitHub Desktop. At the top, there is a light blue illustration of the Moon. Below it, the title 'Configure Git' is displayed. A subtitle explains: 'This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.' There are two input fields: 'Name' with the text 'Paco Abad' and 'Email' with the text 'fjabad@dsic.upv.es'. Below these fields are two buttons: 'Continue' (in blue) and 'Cancel' (in light gray). At the bottom, there is a section titled 'Example commit' showing a commit message 'Fix all the things' and a commit by 'Paco Abad' 30 minutes ago, accompanied by a small GitHub logo.

Configure Git

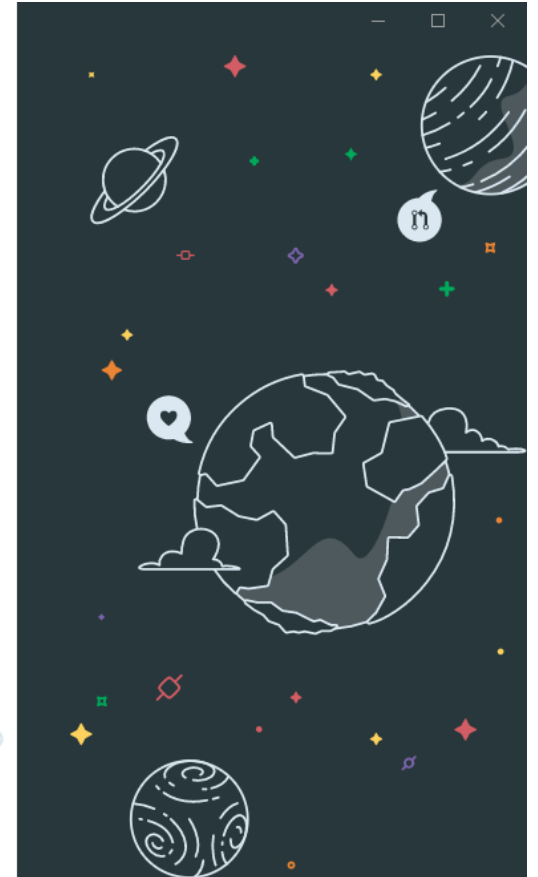
This is used to identify the commits you create. Anyone will be able to see this information if you publish commits.

Name
Paco Abad

Email
fjabad@dsic.upv.es

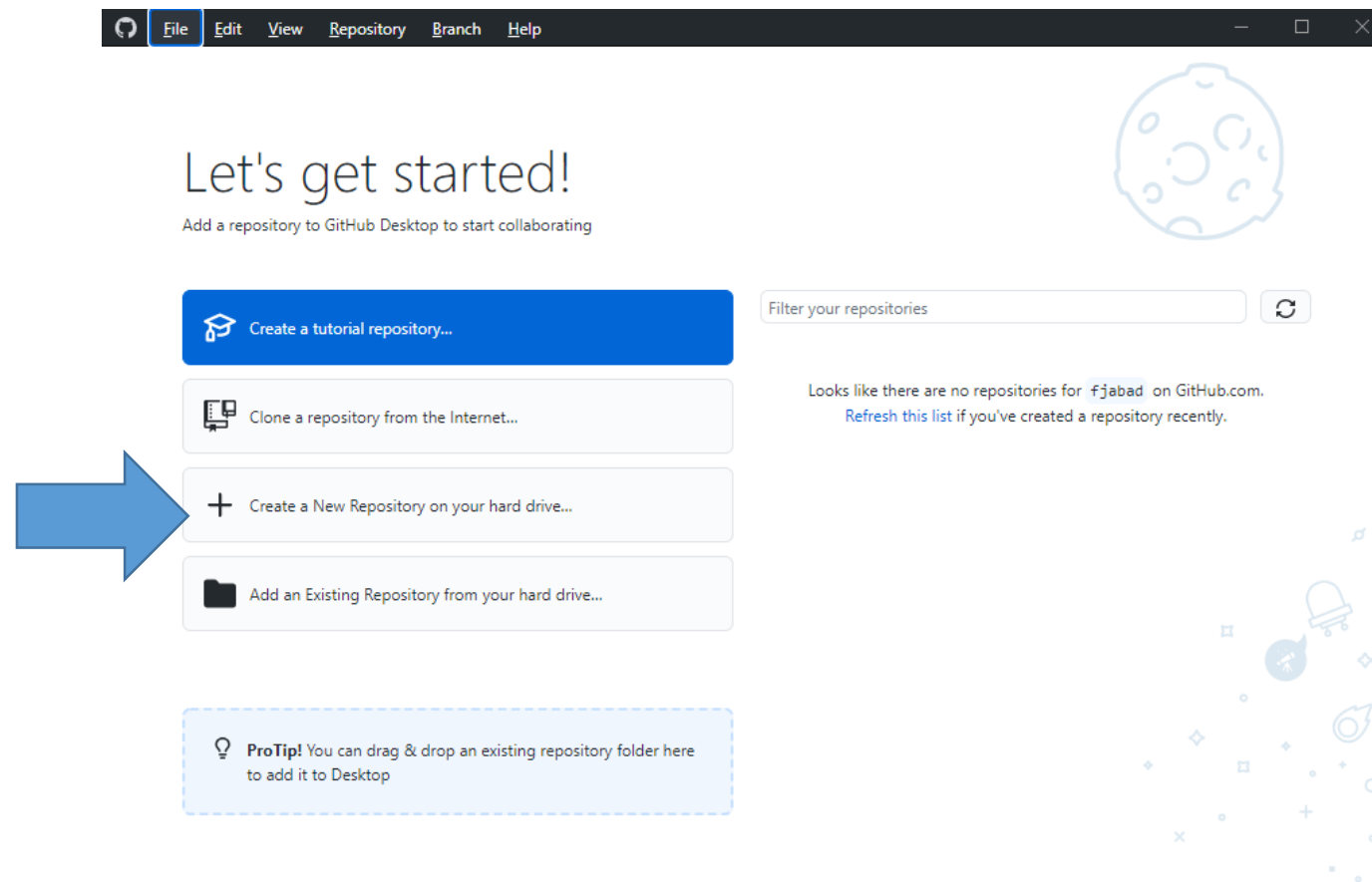
Continue Cancel

Example commit
Fix all the things
Paco Abad • 30m



GitHub Desktop

- Primeros pasos: creando un nuevo repositorio (proyecto)



GitHub Desktop

- Primeros pasos

Create a new repository

Name
my-awesome-game

Description
An FPS game where a farmer has to chase the evil Lord Potato

Local path
C:\Users\fjabad\Documents\GitHub Choose...

☒ Initialize this repository with a README

Git ignore
Unity

License
The Unlicense

Create repository Cancel

Nombre del repositorio

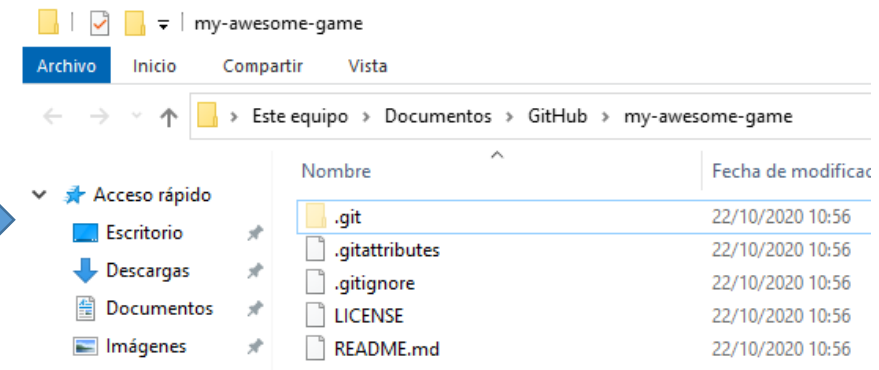
Descripción

Ruta en el disco local

Crear un fichero README

Seleccionar “Unity” para
no subir ficheros
innecesarios

Seleccionar la licencia



Markdown

- El fichero README.md usa el formato Markdown
 - <https://guides.github.com/features/mastering-markdown/>
 - Markdown permite especificar el formato deseado del texto con marcas especiales:

Escribimos en el
readme así:

```
It's very easy to make some words bold and other words  
italic with Markdown. You can even  
[link to Google!] (http://google.com)
```

Y en la página de
GitHub se ve así:

It's very easy to make some words **bold** and other words *italic* with
Markdown. You can even [link to Google!](http://google.com)

Markdown

Sometimes you want numbered lists:

1. One
2. Two
3. Three

Sometimes you want bullet points:

- * Start a line with a star
- * Profit!

Alternatively,

- Dashes work just as well
- And if you have sub points, put two spaces before the dash or star:
 - Like this
 - And this



Sometimes you want numbered lists:

1. One
2. Two
3. Three

Sometimes you want bullet points:

- Start a line with a star
- Profit!

Alternatively,

- Dashes work just as well
- And if you have sub points, put two spaces before the dash or star:
 - Like this
 - And this

Markdown

If you want to embed images, this is how you do it:

```
![Image of Yaktocat](https://octodex.github.com/images/yaktocat.png)
```

If you want to embed images, this is how you do it:



Markdown

Structured documents

Sometimes it's useful to have different levels of headings to structure your documents. Start lines with a `#` to create headings. Multiple `##` in a row denote smaller heading sizes.

This is a third-tier heading

You can use one `#` all the way up to `#####` six for different heading sizes.

If you'd like to quote someone, use the > character before the line:

```
> Coffee. The finest organic suspension ever
devised... I beat the Borg with it.
> - Captain Janeway
```

Structured documents

Sometimes it's useful to have different levels of headings to structure your documents. Start lines with a `#` to create headings. Multiple `##` in a row denote smaller heading sizes.

This is a third-tier heading

You can use one `#` all the way up to `#####` six for different heading sizes.

If you'd like to quote someone, use the > character before the line:

```
> Coffee. The finest organic suspension ever devised... I beat the Borg
with it. - Captain Janeway
```


Markdown

There are many different ways to style code with GitHub's markdown. If you have inline code blocks, wrap them in backticks: ``var example = true``. If you've got a longer block of code, you can indent with four spaces:

```
    if (isAwesome){  
      return true  
    }
```

GitHub also supports something called code fencing, which allows for multiple lines without indentation:

```
```\nif (isAwesome){  
 return true
}\n```
```

And if you'd like to use syntax highlighting, include the language:

```
```javascript\nif (isAwesome){  
  return true  
}\n```
```



There are many different ways to style code with GitHub's markdown. If you have inline code blocks, wrap them in backticks: `var example = true`. If you've got a longer block of code, you can indent with four spaces:

```
    if (isAwesome){  
      return true  
    }
```

GitHub also supports something called code fencing, which allows for multiple lines without indentation:

```
```\nif (isAwesome){  
 return true
}\n```
```

And if you'd like to use syntax highlighting, include the language:

```
```javascript\nif (isAwesome){  
  return true  
}\n```
```

Markdown

GitHub supports many extras in Markdown that help you reference and link to people. If you ever want to direct a comment at someone, you can prefix their name with an @ symbol: Hey @kneath — love your sweater!

But I have to admit, tasks lists are my favorite:

- [x] This is a complete item
- [] This is an incomplete item

When you include a task list in the first comment of an Issue, you will see a helpful progress bar in your list of issues. It works in Pull Requests, too!

And, of course emoji!



GitHub supports many extras in Markdown that help you reference and link to people. If you ever want to direct a comment at someone, you can prefix their name with an @ symbol: Hey @kneath — love your sweater!

But I have to admit, tasks lists are my favorite:

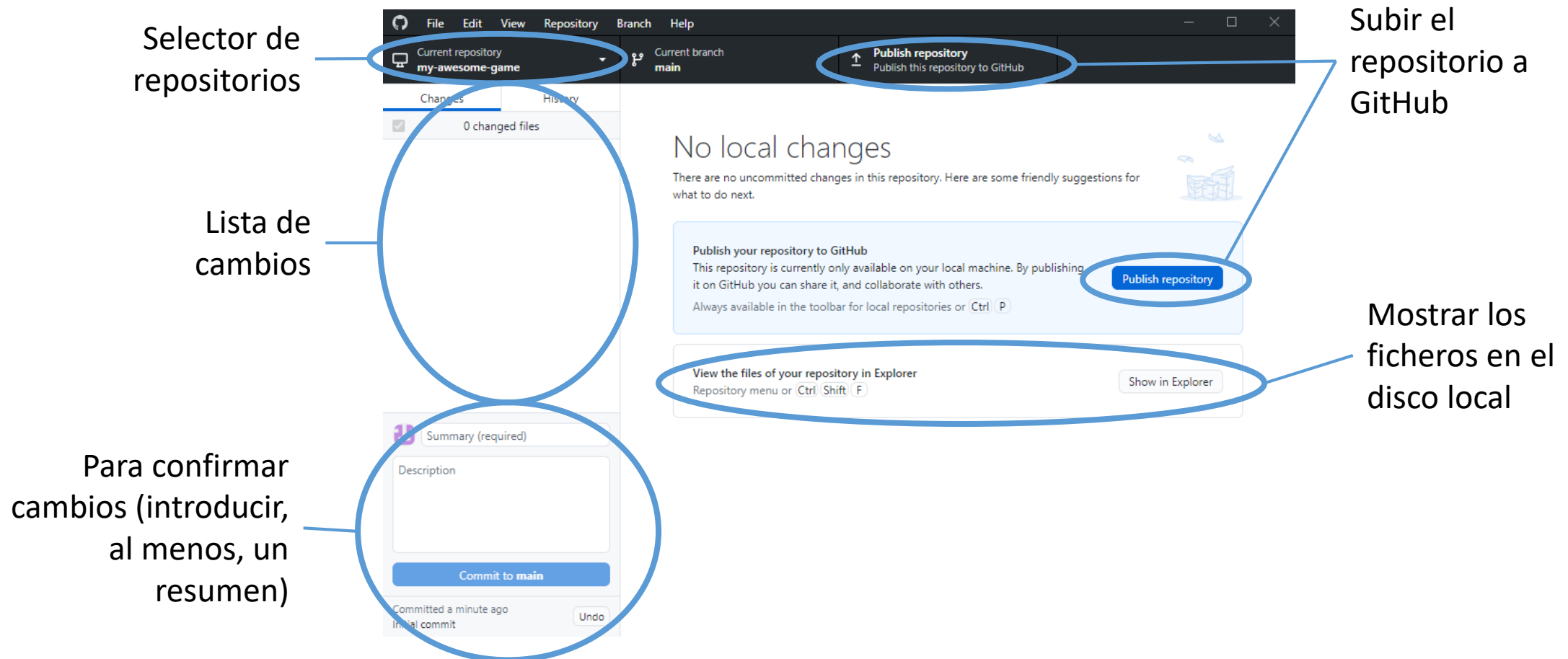
- ☒ This is a complete item
- ☐ This is an incomplete item

When you include a task list in the first comment of an Issue, you will see a helpful progress bar in your list of issues. It works in Pull Requests, too!

And, of course emoji!

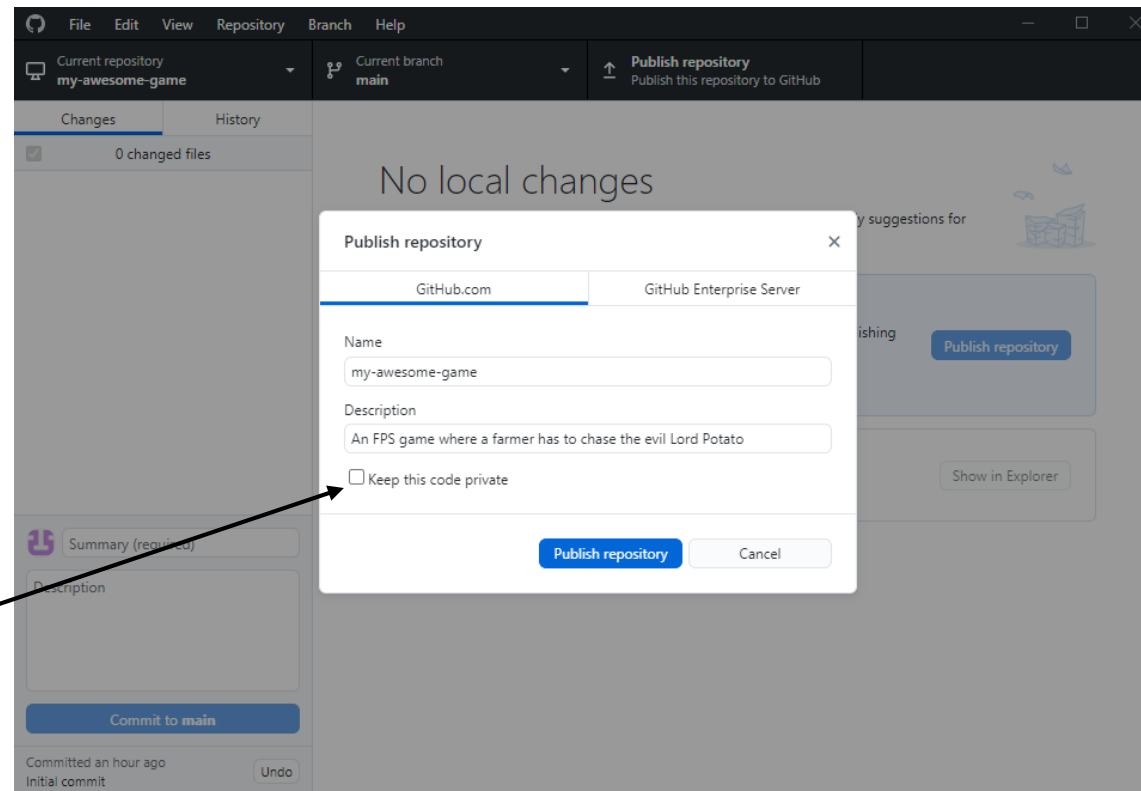
GitHub Desktop

- Ventana principal



GitHub Desktop

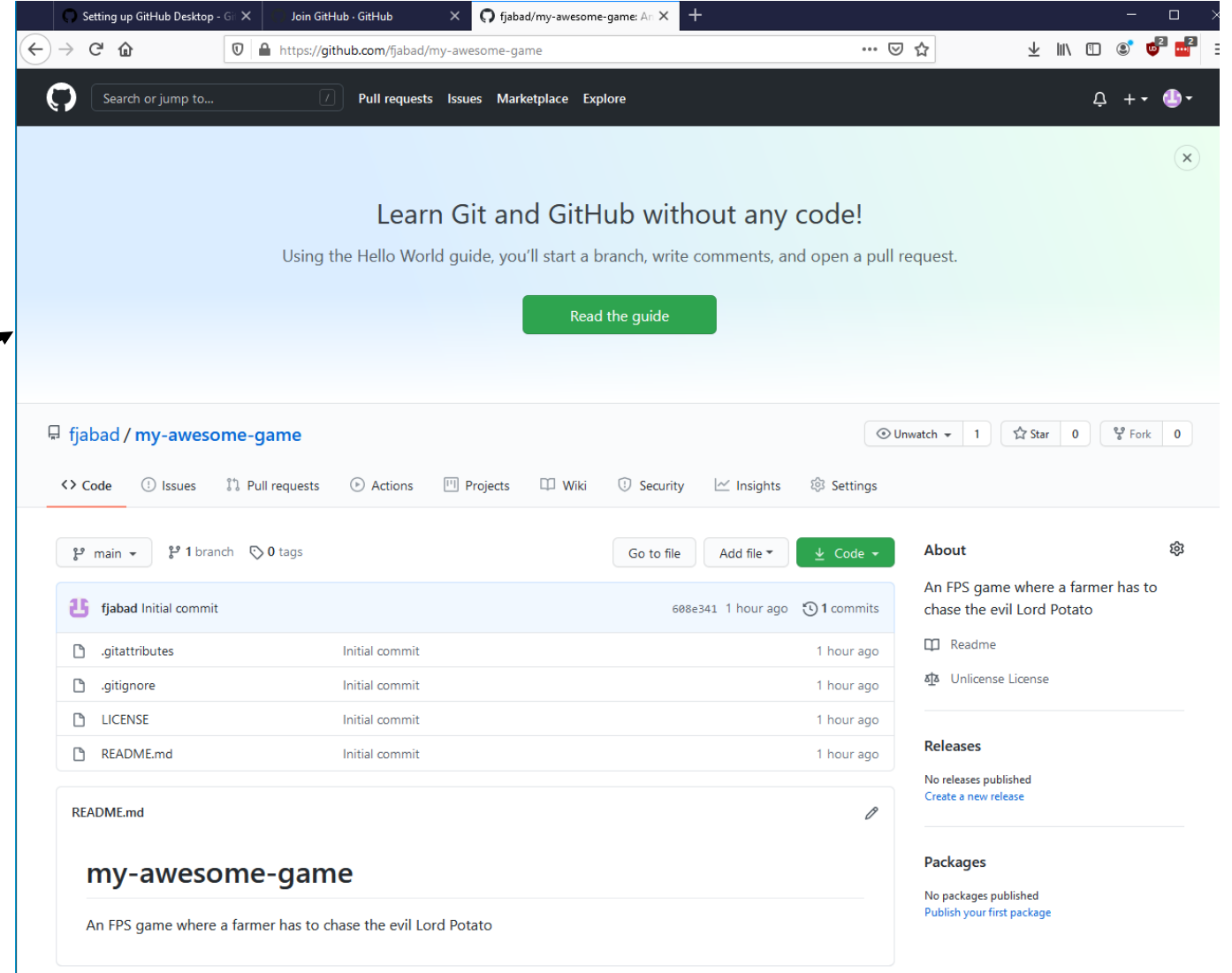
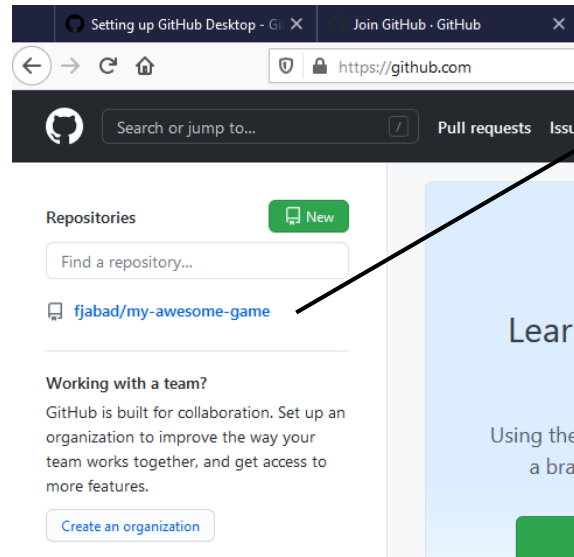
- Podemos publicar el proyecto en GitHub



Deshabilita la opción para que el proyecto sea público

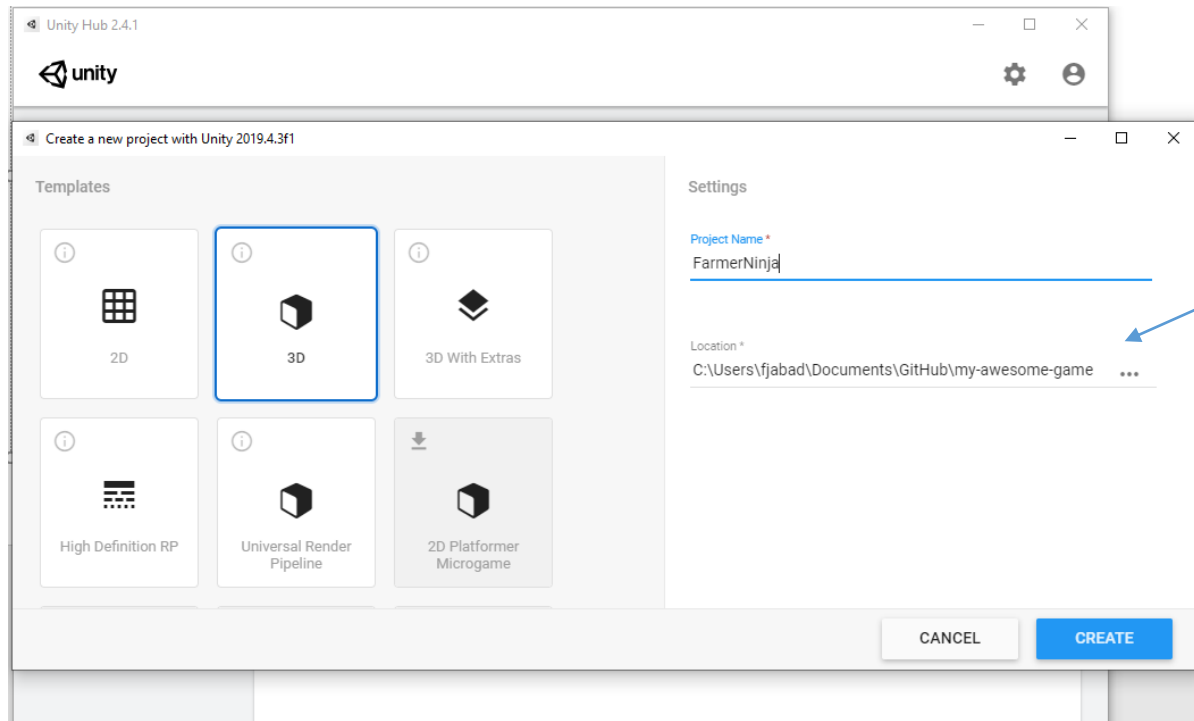
GitHub Desktop

- Y en GitHub...

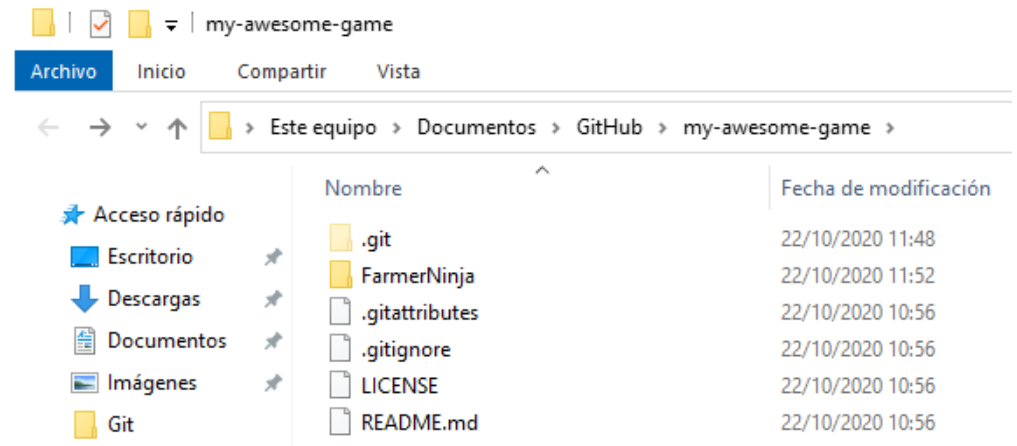
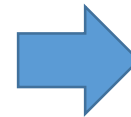


GitHub Desktop

- A continuación creamos el proyecto en Unity:

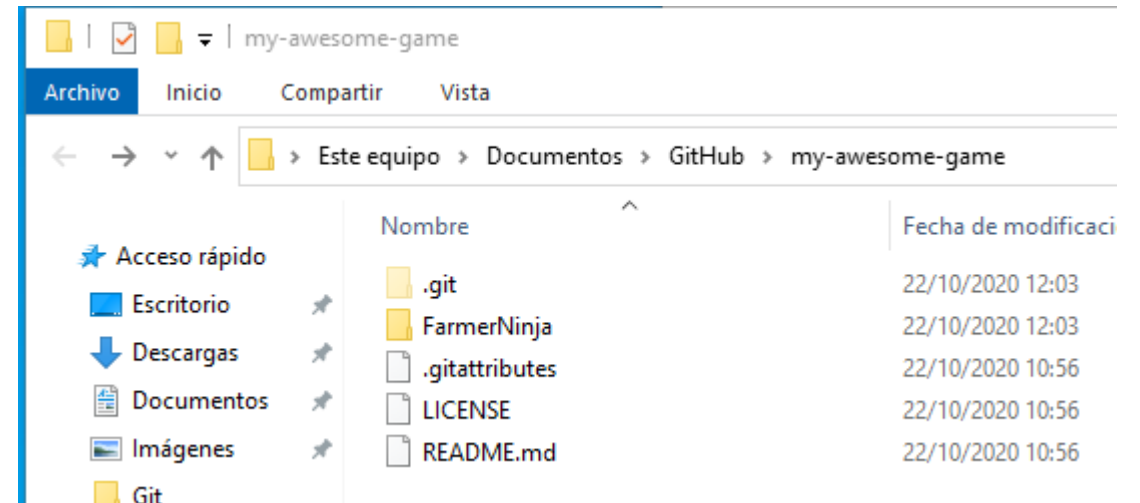
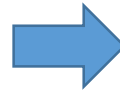
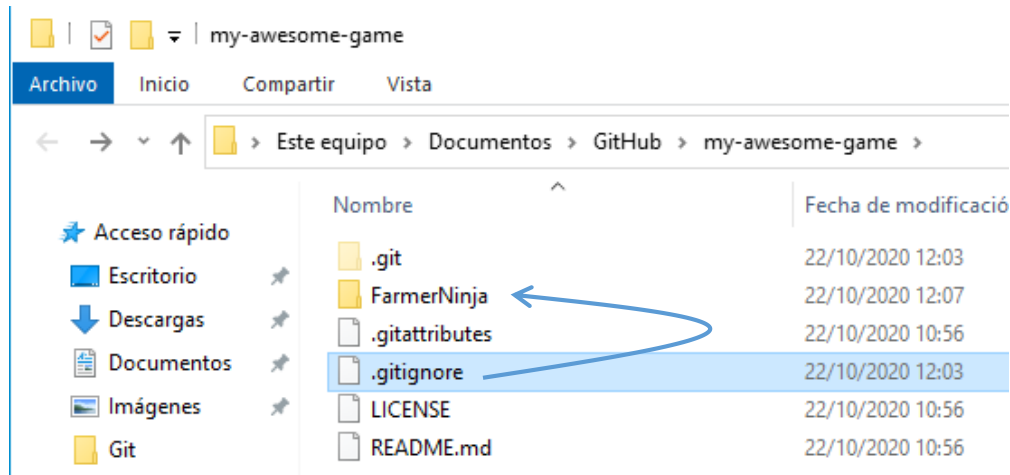


Elige aquí el directorio de tu disco duro que contiene el repositorio



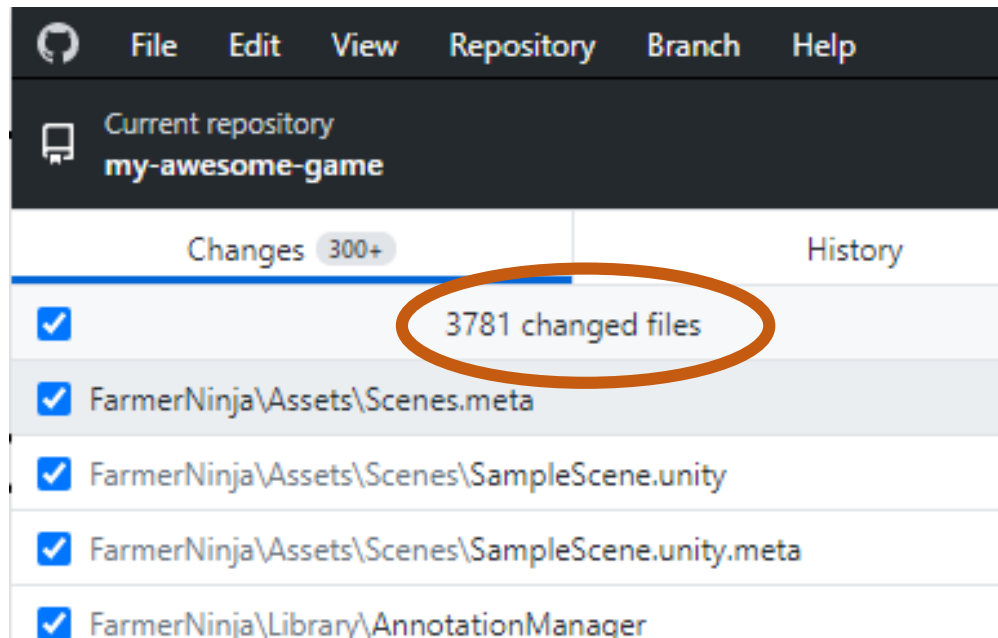
GitHub Desktop

- Mueve el fichero .gitignore al directorio del juego:

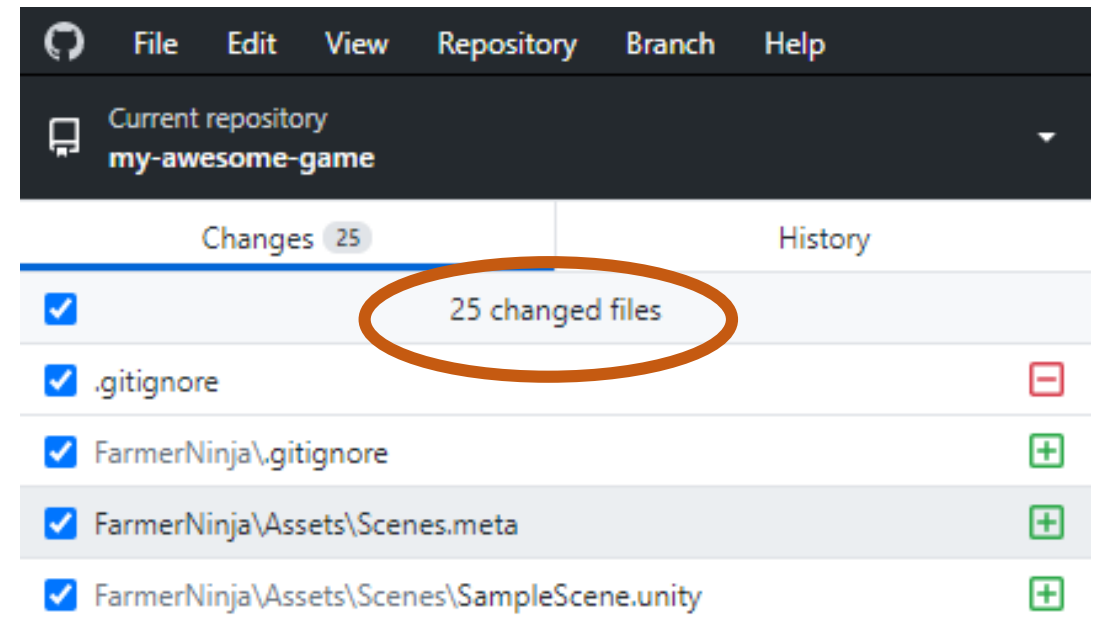
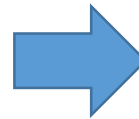


GitHub Desktop

- Mueve el fichero .gitignore al directorio del juego:



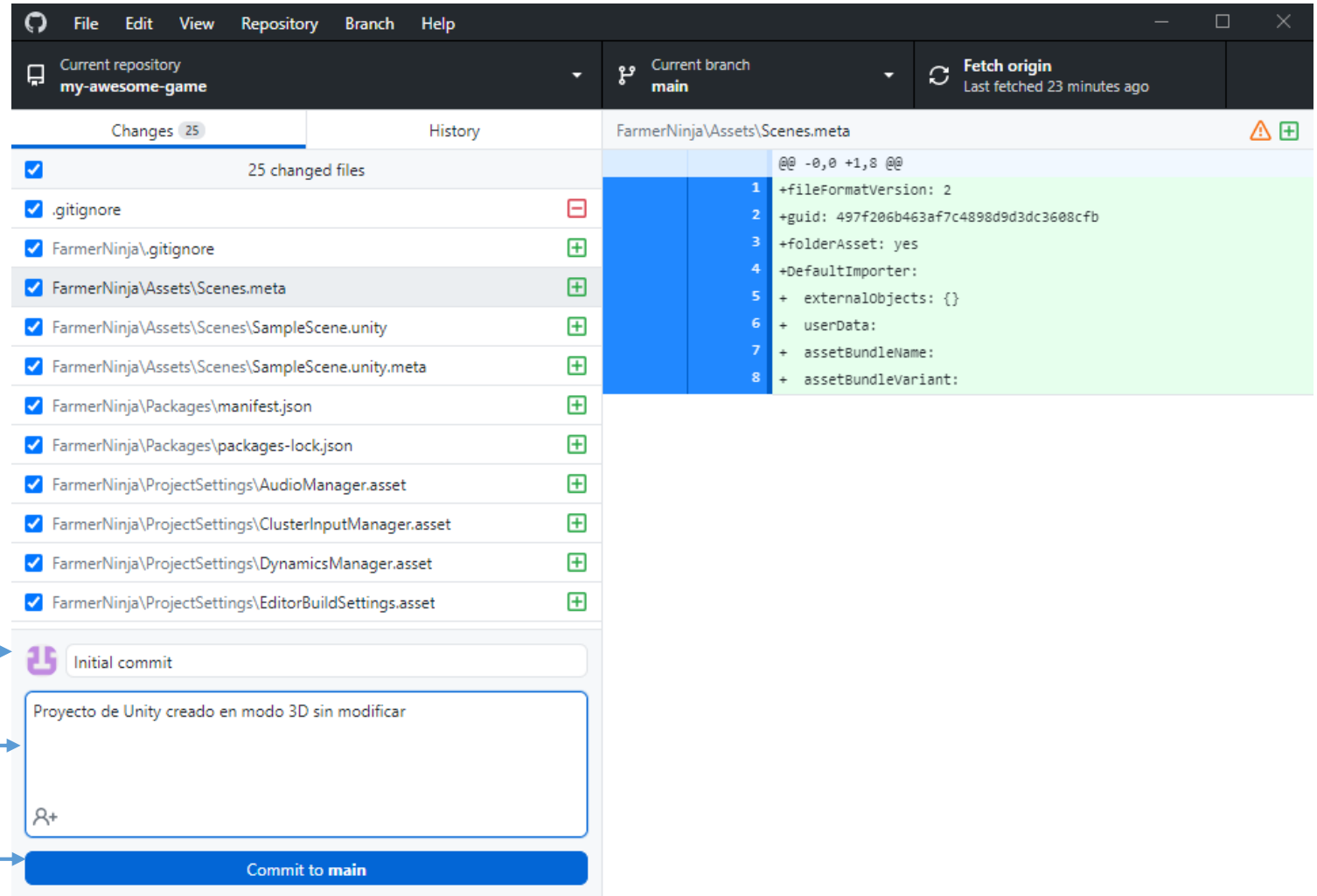
Antes de mover .gitignore



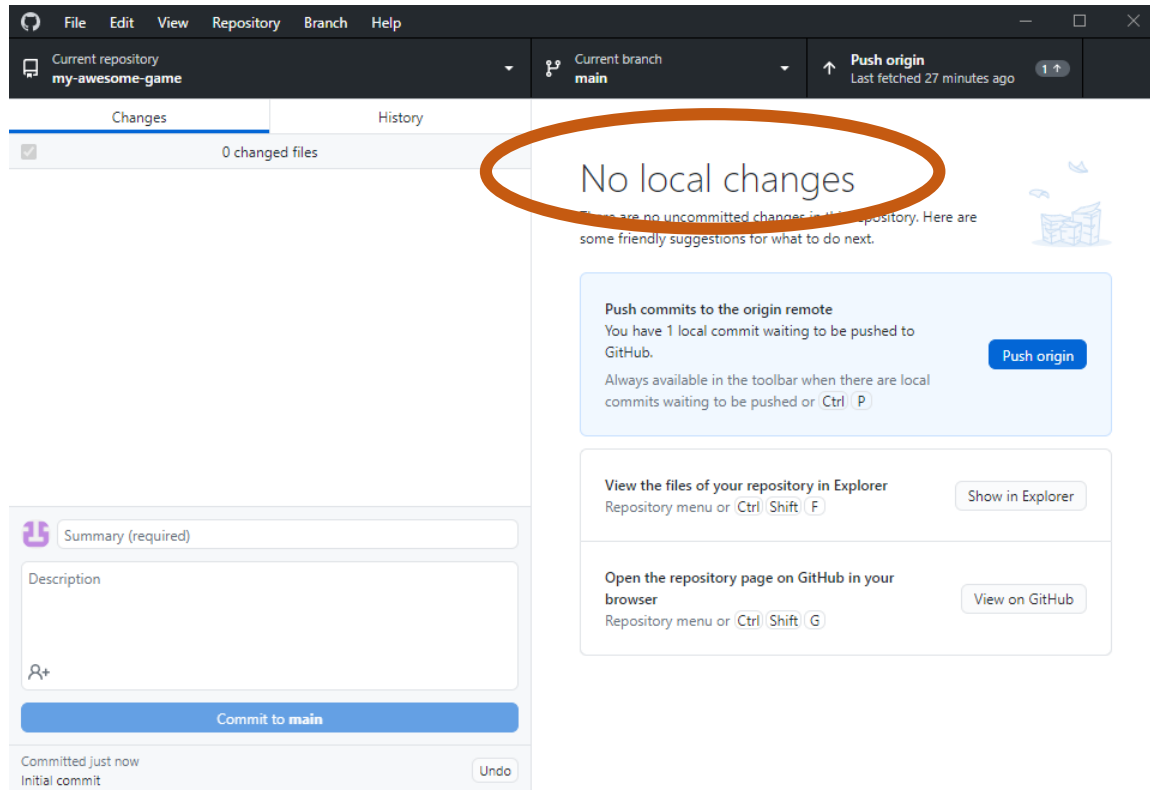
Después de mover .gitignore

GitHub Desktop

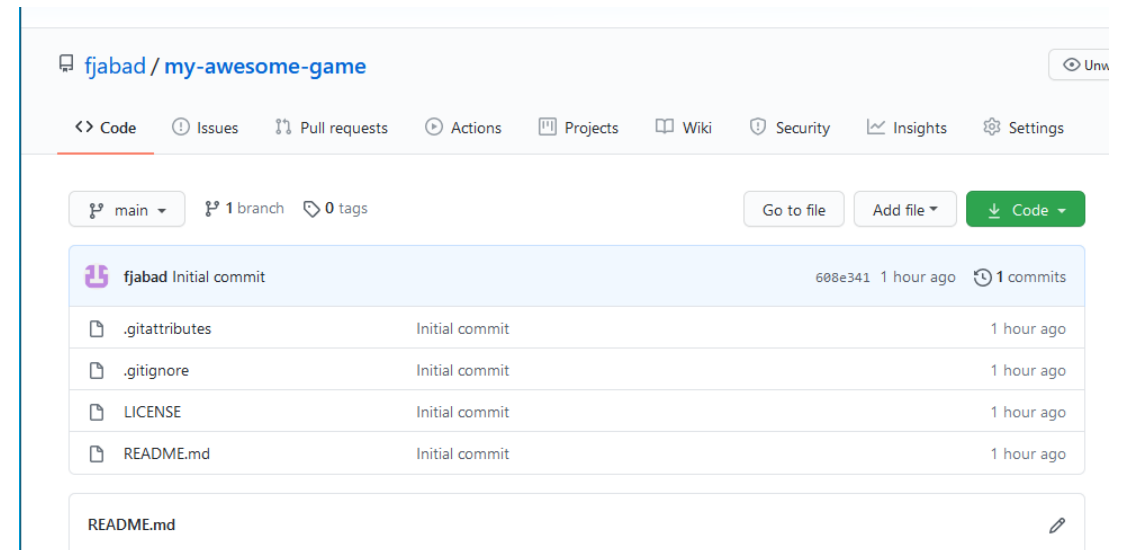
- Vamos a sincronizar los cambios



GitHub Desktop

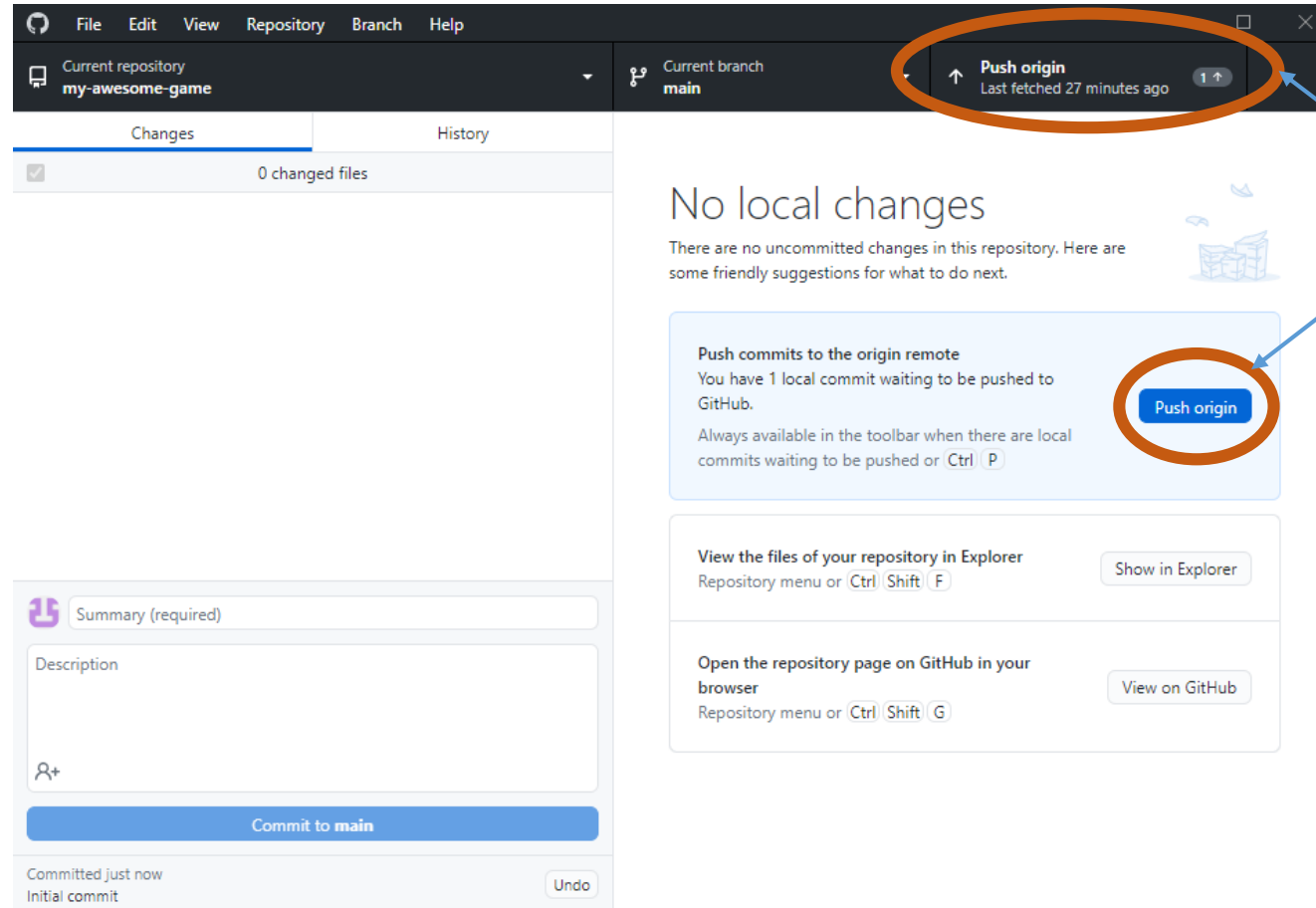


Pero, en GitHub:



¿dónde se han guardado los cambios?

GitHub Desktop



Sube los
cambios a
GitHub

GitHub Desktop

- Los cambios ya son visibles para el resto del equipo

The screenshot shows the GitHub web interface for a repository named 'my-awesome-game' by user 'fjabad'. The repository is in the 'main' branch, which has 1 branch and 0 tags. The commit history shows an 'Initial commit' by 'fjabad' 5 minutes ago, with a commit hash of '5a90e61'. The commit message is 'Initial commit'. The commit includes four files: 'FarmerNinja', '.gitattributes', 'LICENSE', and 'README.md', all of which were added in the initial commit 1 hour ago.

fjabad / my-awesome-game

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

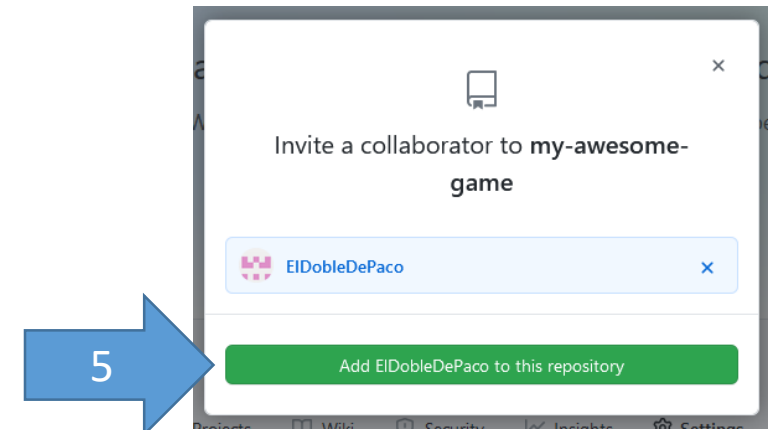
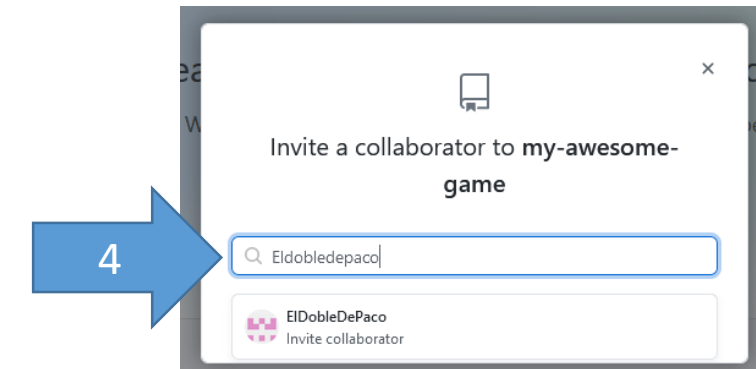
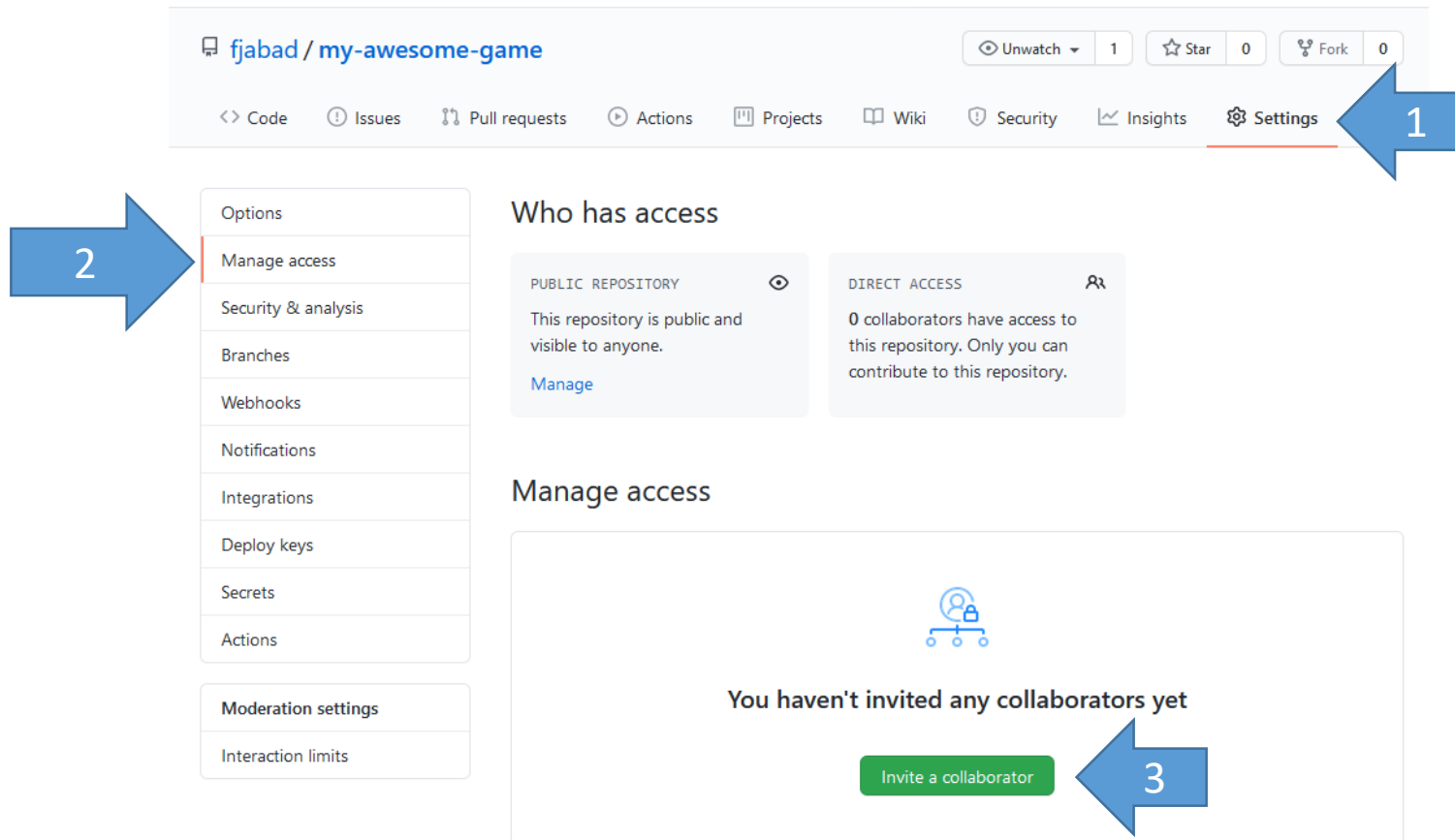
Go to file Add file Code

fjabad Initial commit 5a90e61 5 minutes ago 2 commits

FarmerNinja	Initial commit	5 minutes ago
.gitattributes	Initial commit	1 hour ago
LICENSE	Initial commit	1 hour ago
README.md	Initial commit	1 hour ago

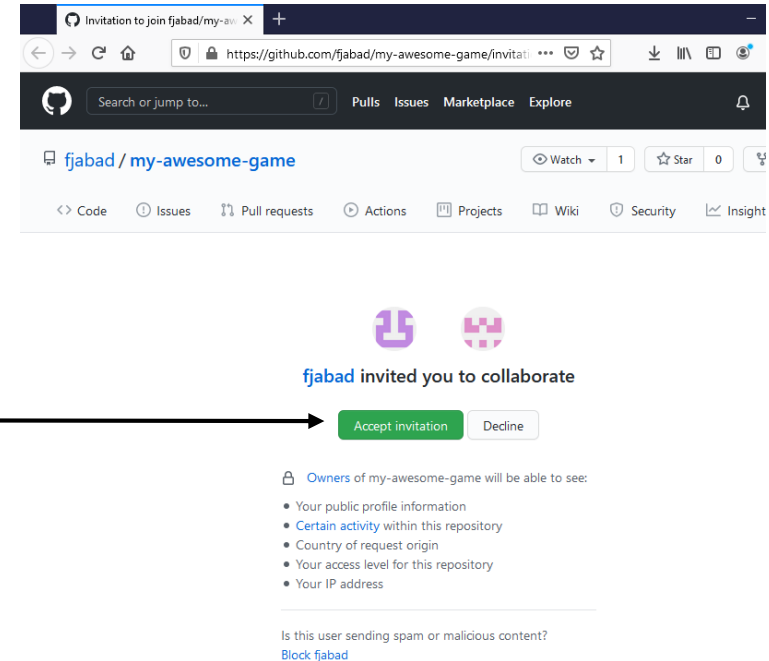
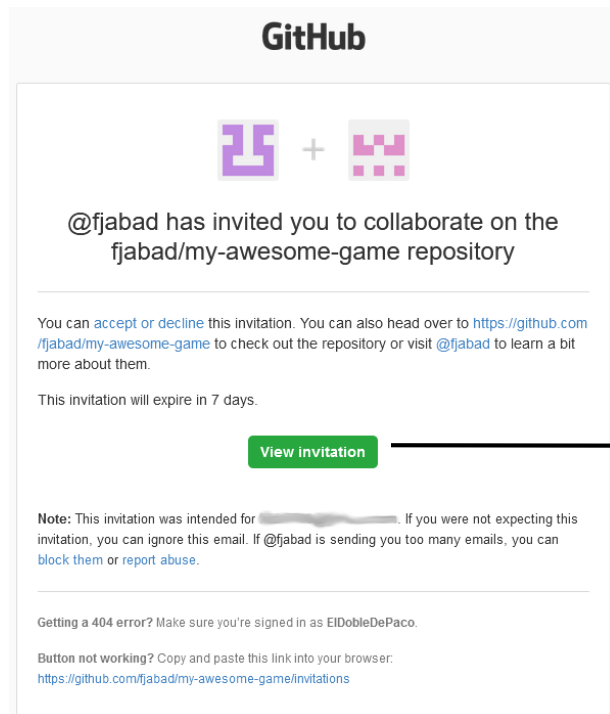
GitHub Desktop

- Vamos a dar permisos al resto de miembros del equipo



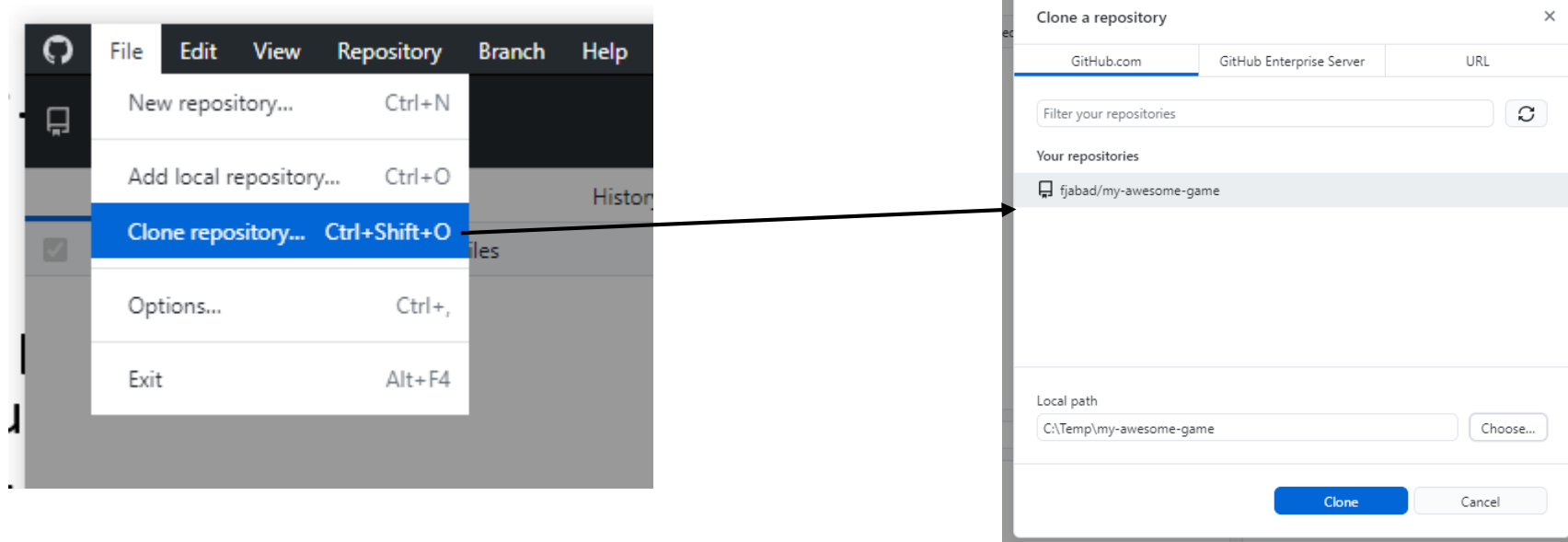
GitHub Desktop

- El invitado tiene que aceptar la invitación



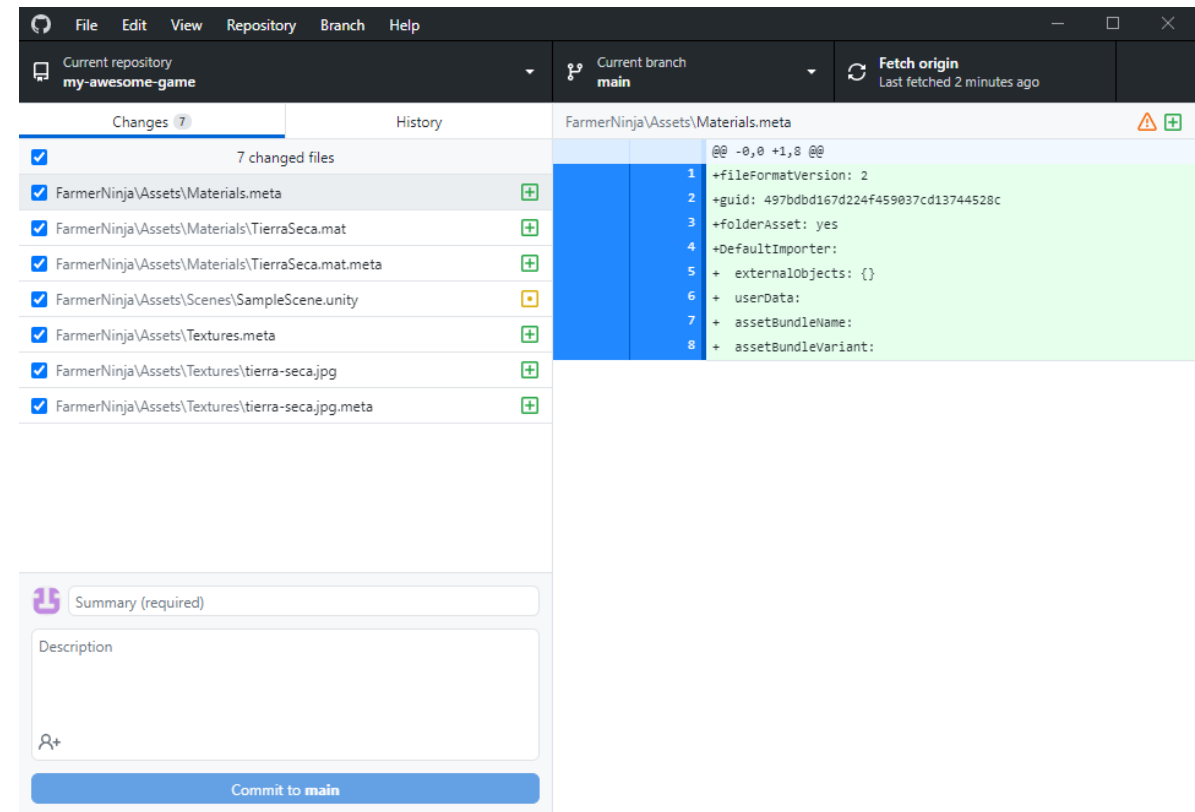
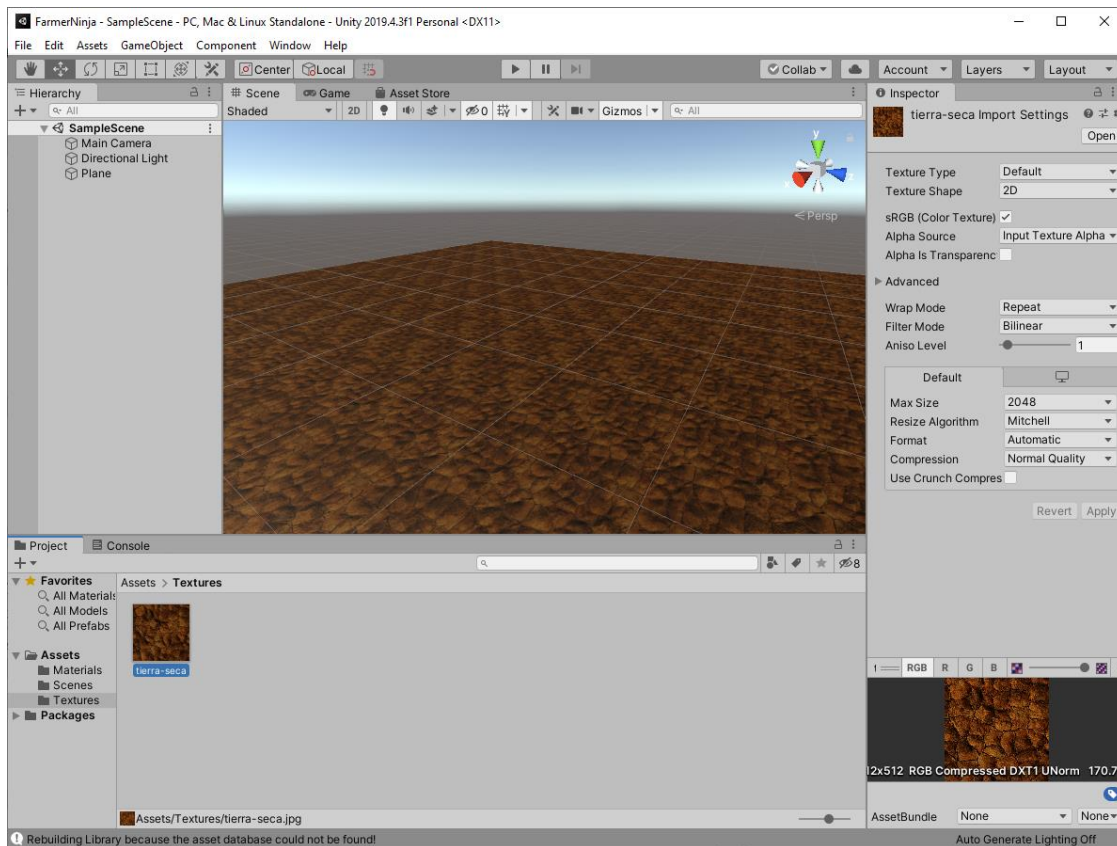
GitHub Desktop

- A partir de ese momento el miembro invitado puede participar en el desarrollo y subir cambios
- Al entrar en GitHub Desktop e identificarse, ya podrá clonar el repositorio y empezar a trabajar:

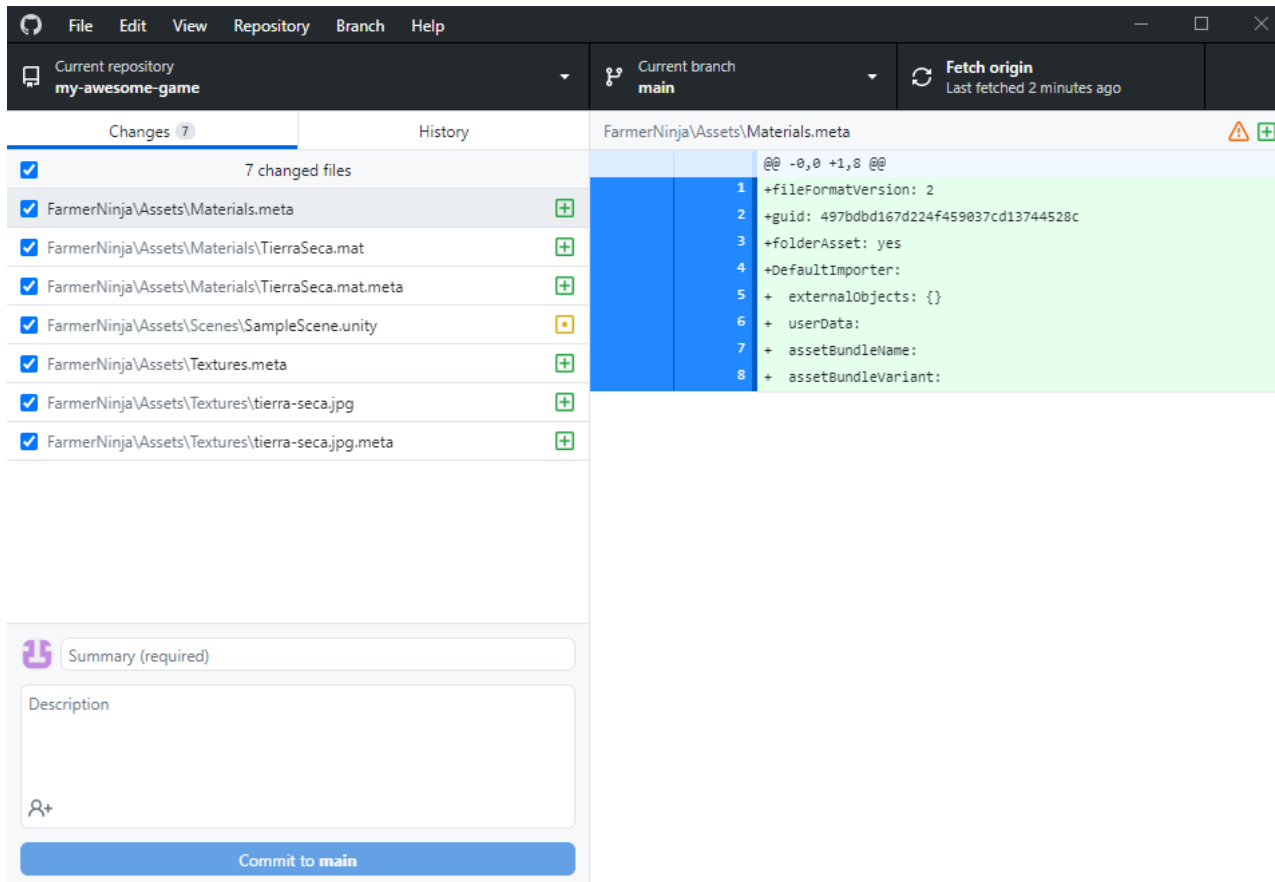


GitHub Desktop

- Uno de los usuarios hace un cambio en la escena:



GitHub Desktop

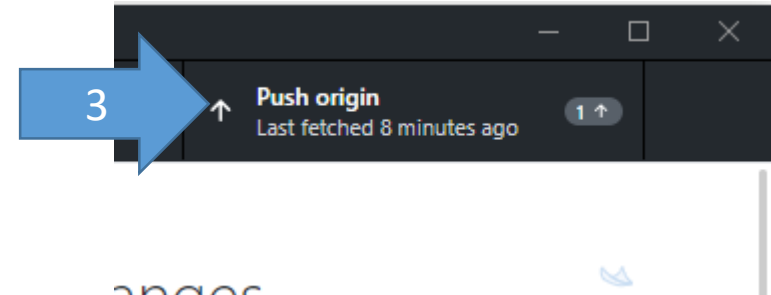
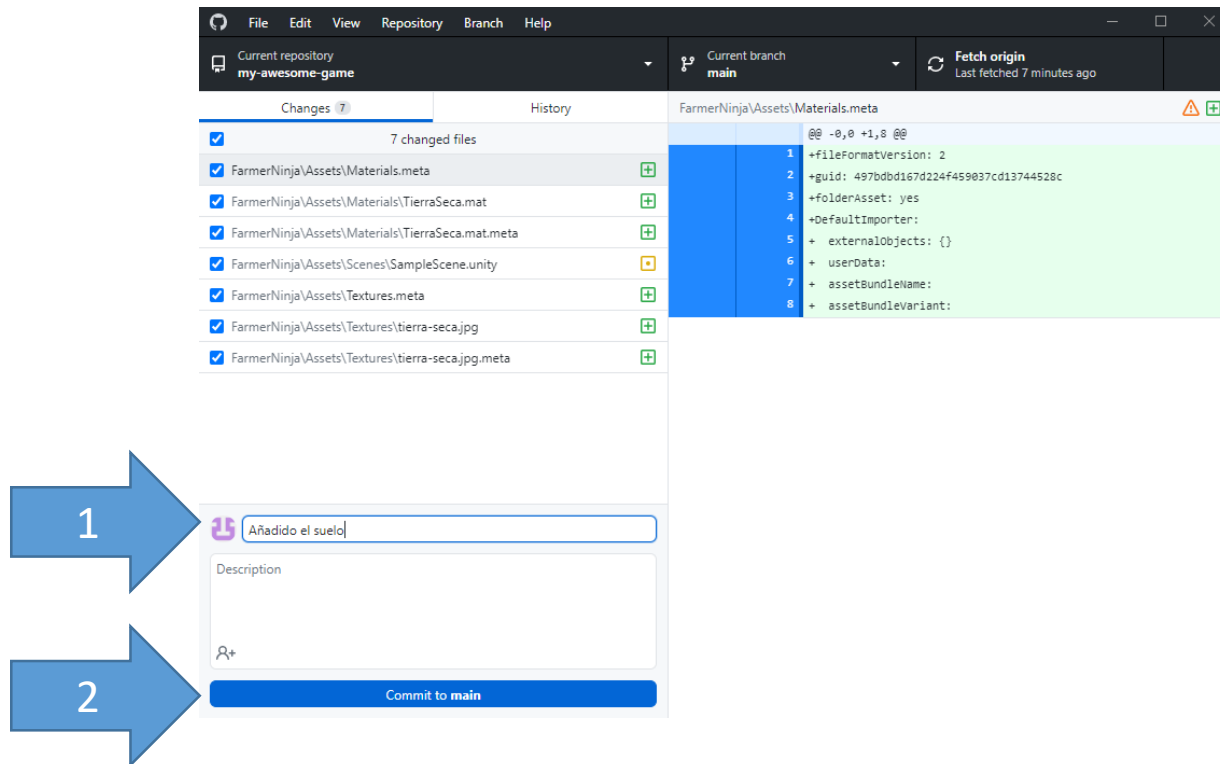


Changes ●	History
3 changed files	
✓ bad-idea.md	✖
✓ idea.rb	+
✓ README.md	●

Eliminado
Nuevo
Modificado

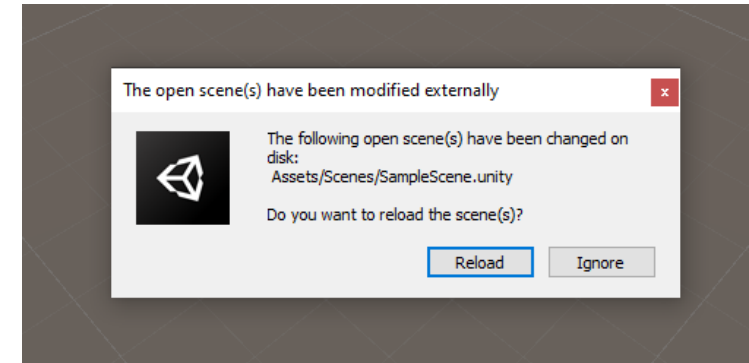
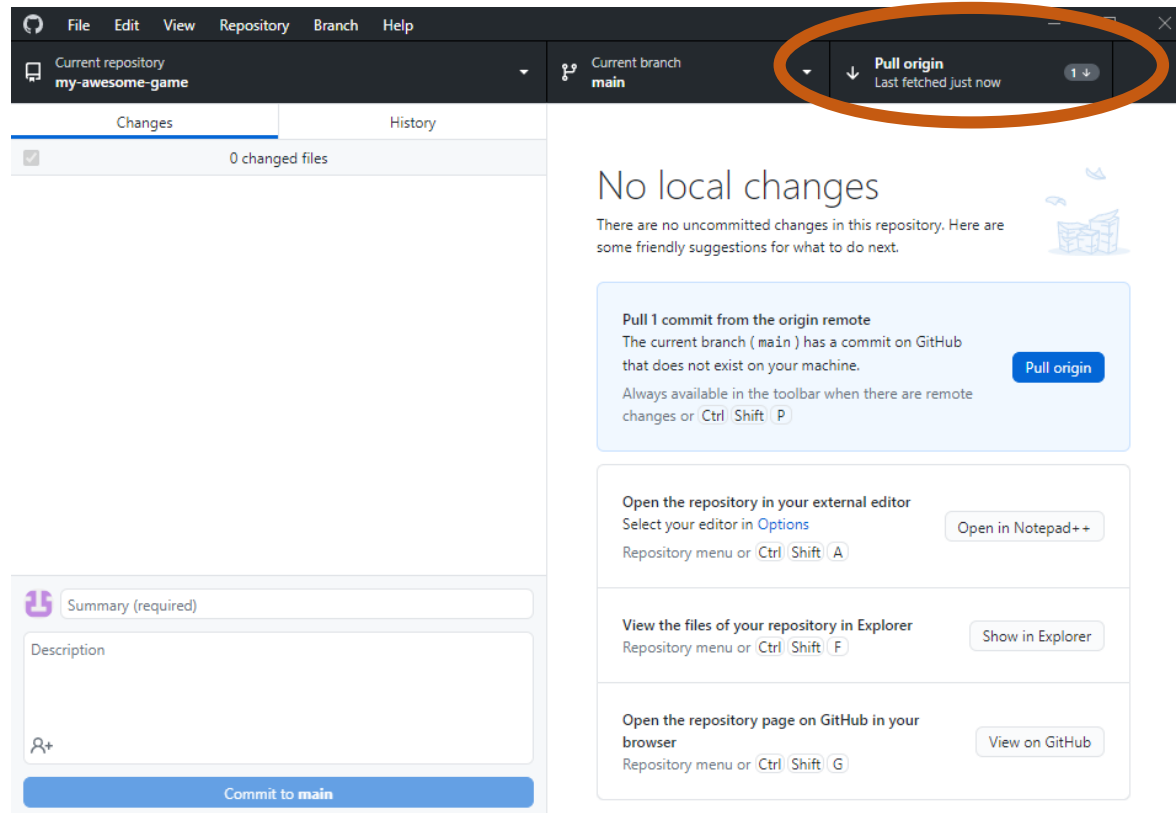
GitHub Desktop

- Haciendo commit de los cambios:



GitHub Desktop

- El resto de miembros verán que hay cambios pendientes



GitHub Desktop

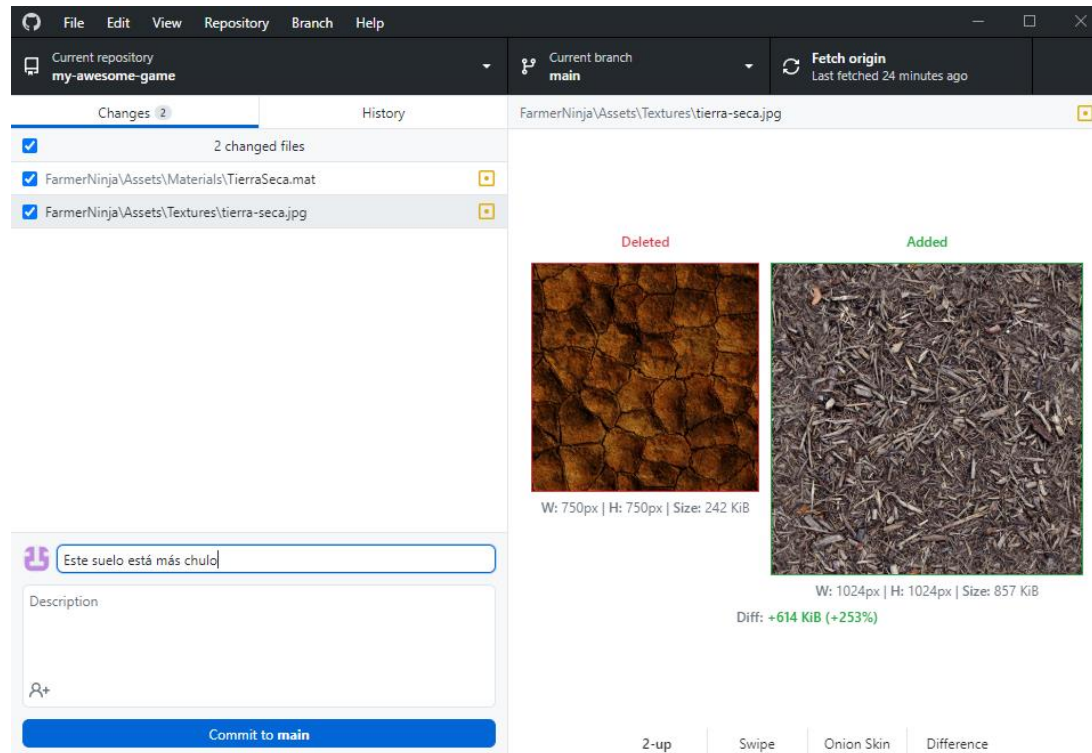
Conflictos

- Resolución de conflictos
 - En ficheros de código (scripts de C#), normalmente git puede combinar los cambios sin problema
 - Sin embargo, si dos diseñadores modifican, por ejemplo, una imagen, git no tiene forma de saber cuál es la buena

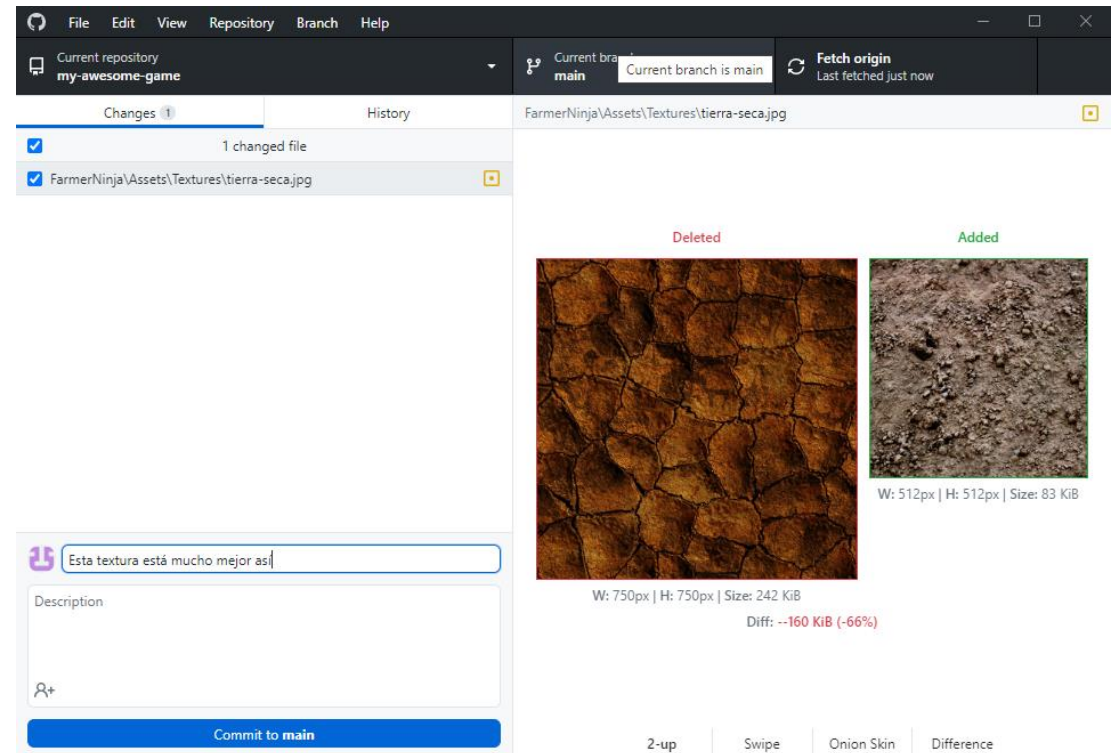
GitHub Desktop

Conflictos

El diseñador A hace commit y push de una imagen



El diseñador B hace commit de la misma imagen



GitHub Desktop

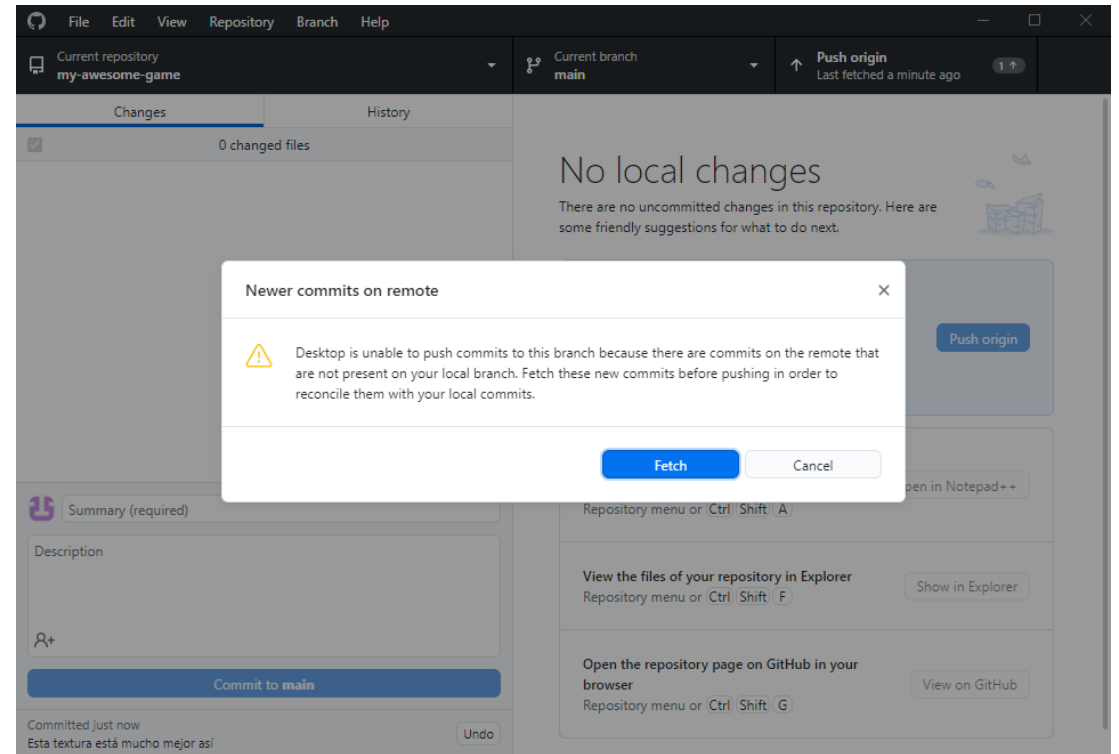
Conflictos

Git dice que en el servidor hay cambios que no tenemos. Antes de subir nuestros cambios, debemos traer los cambios del servidor...

Normalmente no habrá conflicto (se han modificado ficheros que no hemos tocado, o git es capaz de mezclar ambas versiones).

Pero en este caso...

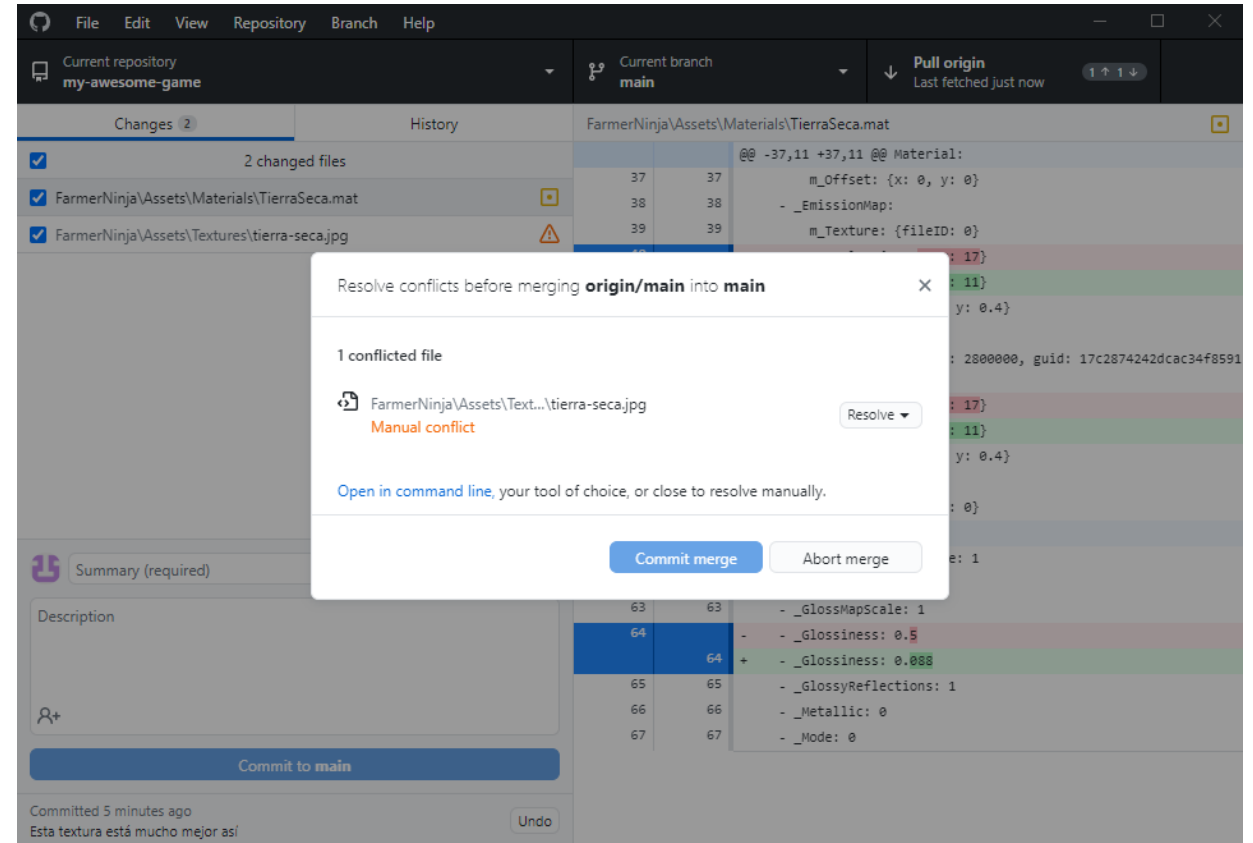
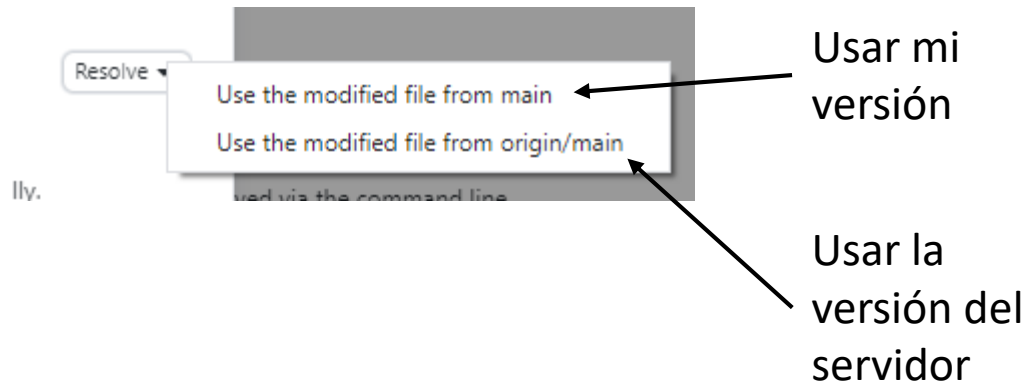
El diseñador B hace push



GitHub Desktop

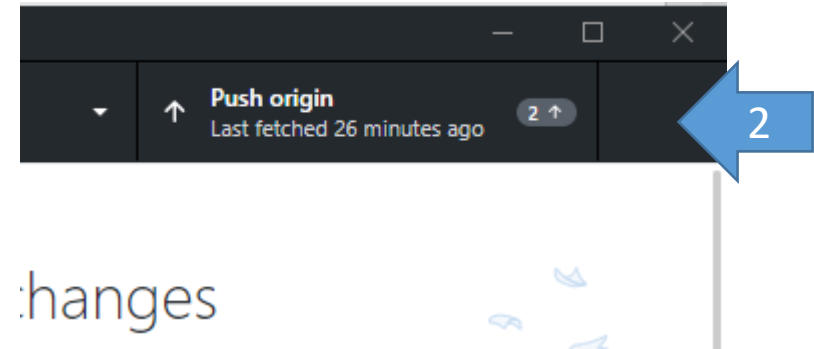
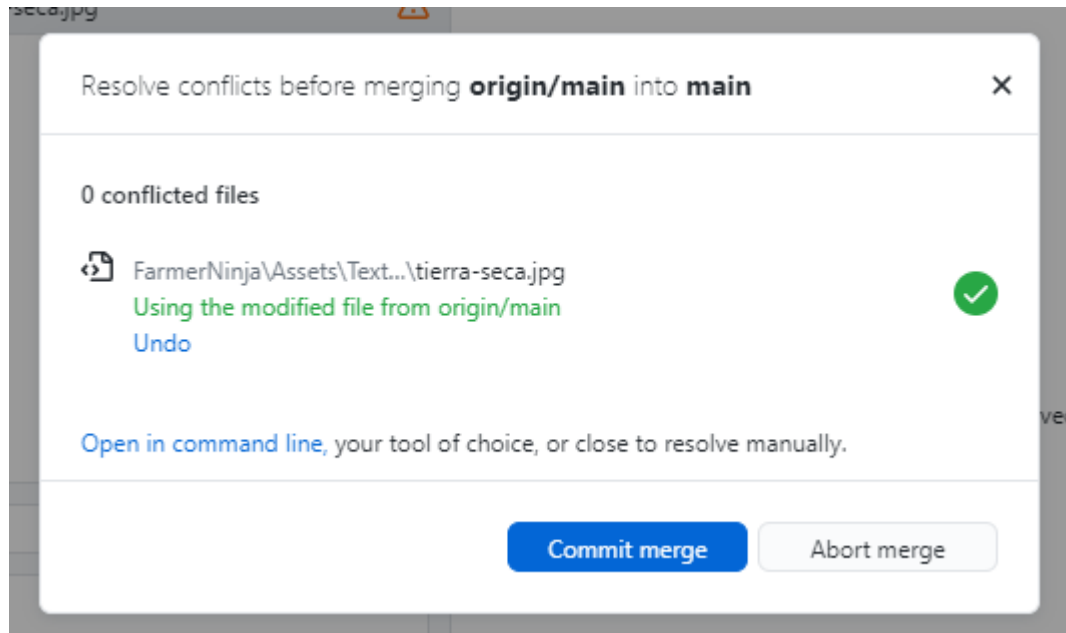
Conflictos

- Es el diseñador B el que tiene que solucionar el conflicto localmente
 - Normalmente tendrá que decidir si se usa la versión local o la del servidor:



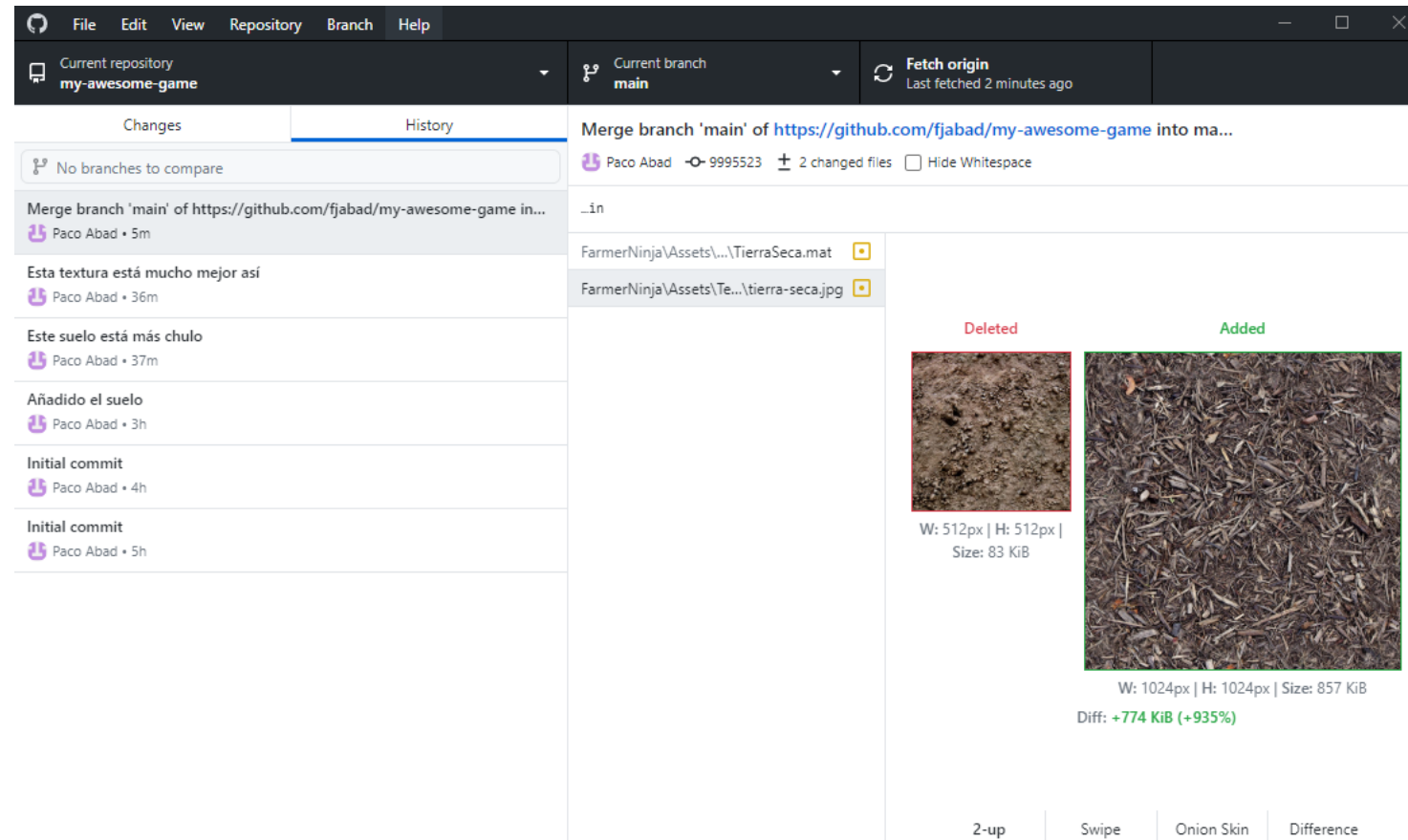
GitHub Desktop

Conflictos



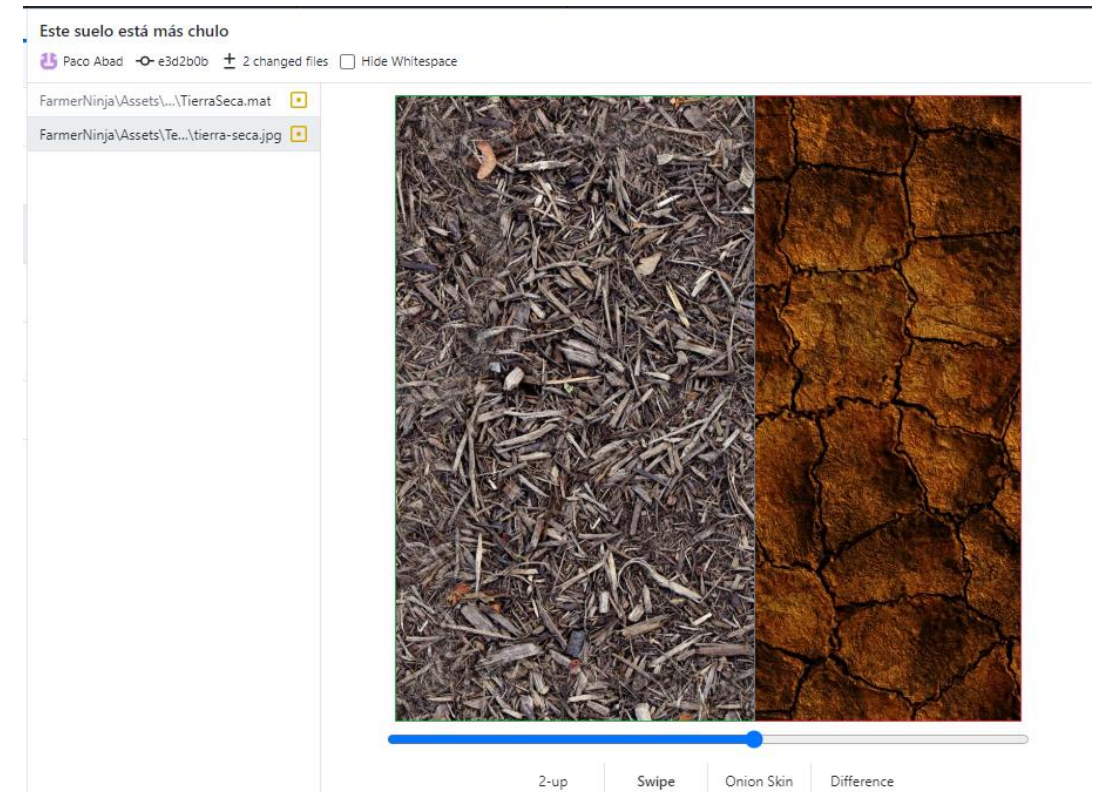
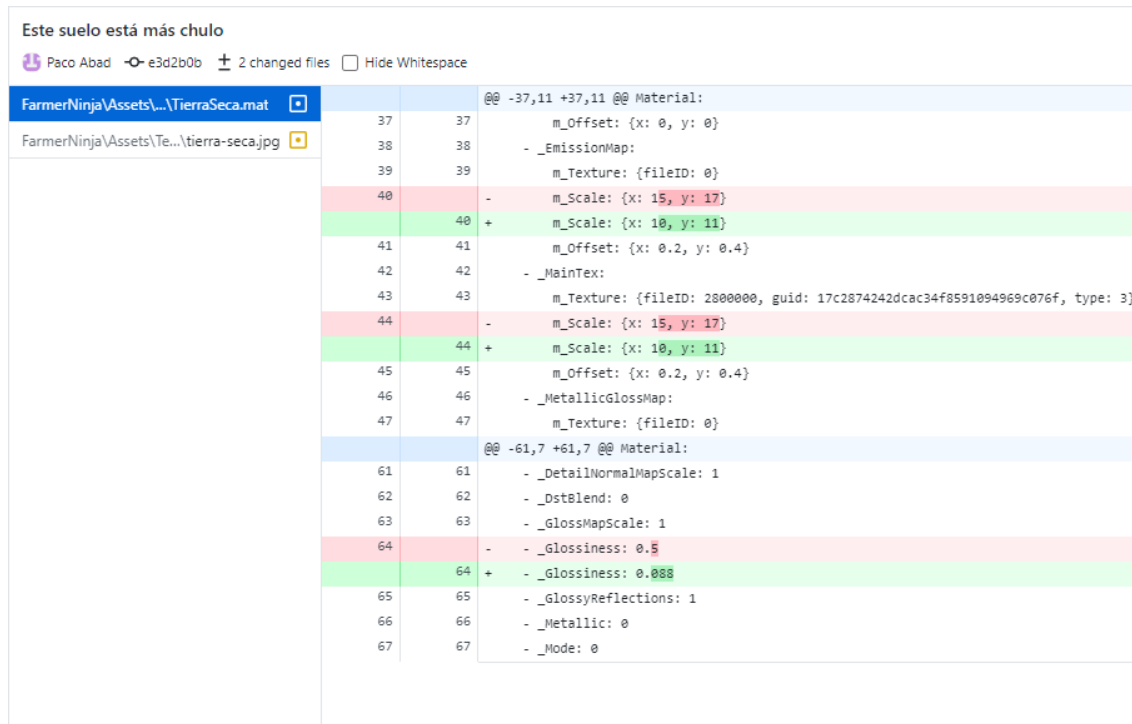
GitHub Desktop

- La solapa History muestra toda la historia del desarrollo del proyecto



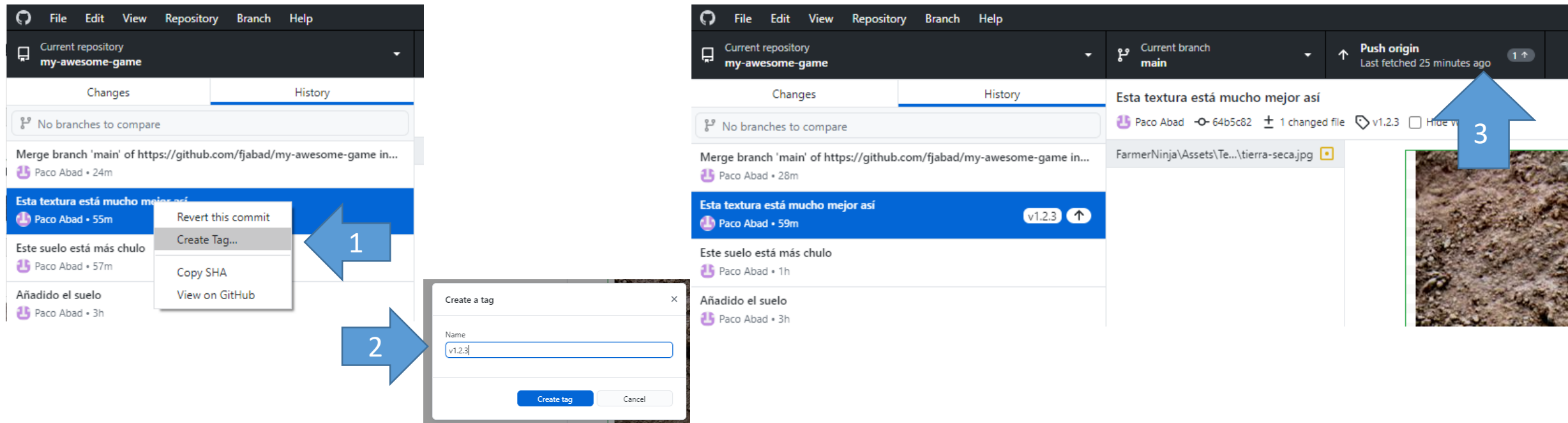
GitHub Desktop

- Según el tipo de fichero, se muestran las diferencias introducidas



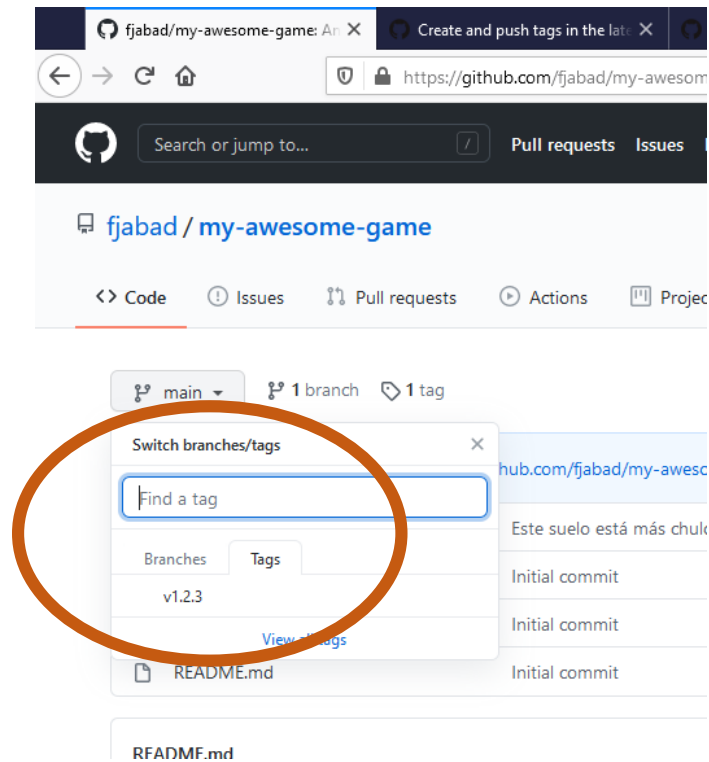
GitHub Desktop

- También podemos marcar un commit con una etiqueta (*tag*)
 - Esto nos permite referirnos a momentos específicos del desarrollo (por ejemplo, v1.0, o demo-2009)



GitHub Desktop

- Y en github...



Unity y Git

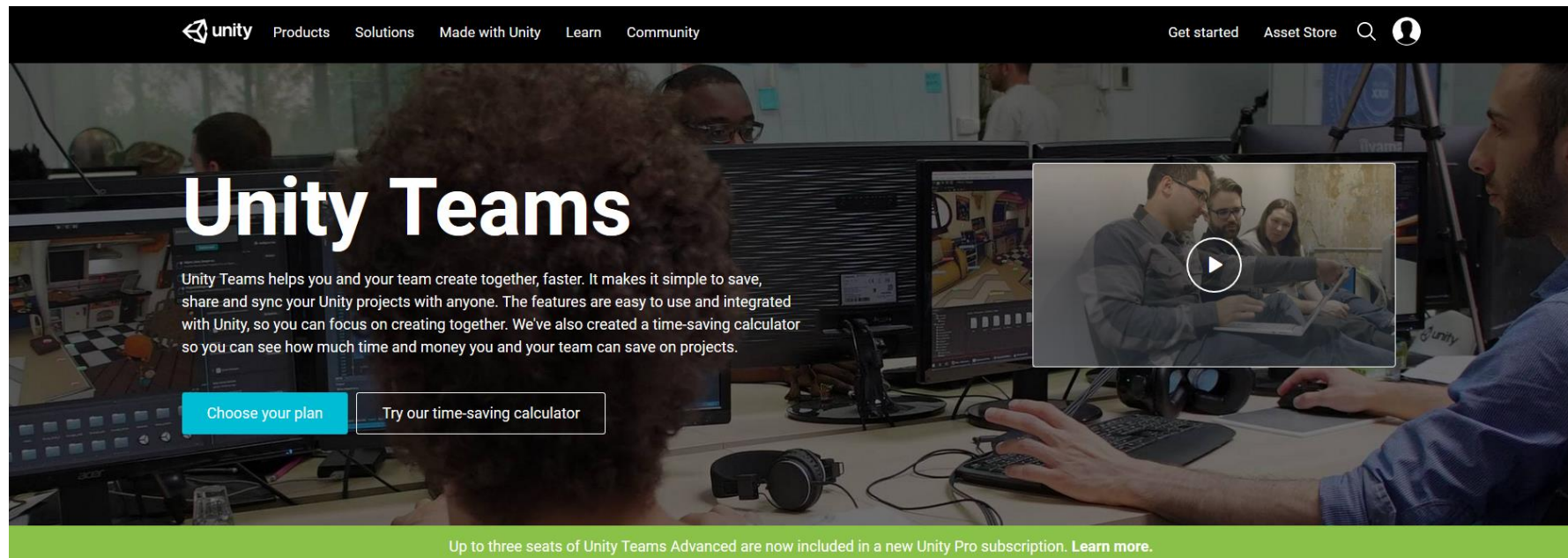
- Como hemos visto, Unity permite mantener el proyecto en un sistema de control de versiones
- Consejos:
 - Que todos los participantes usen exactamente la misma versión de Unity
 - Actualizar a menudo
 - No subir ficheros innecesarios
 - No subir compilaciones rotas al repositorio remoto

Unity y Git

- Para añadir un proyecto Unity a un sistema de control de versiones:
 - Hacer visibles los ficheros .meta
 - Edit\Project Settings\Editor. Selecciona Visible Meta Files en Version Control
 - Cada fichero y carpeta en el proyecto tiene un fichero de texto con el mismo nombre y extensión .meta
 - Los ficheros .meta son necesarios. Mantenlos junto a su fichero
 - Sólo son necesarias las carpetas Assets, Packages y ProjectSettings.
 - El resto de carpetas se reconstruyen a partir de estas y no deberían almacenarse en el sistema de control de versiones.
 - Ver las instrucciones paso a paso en:
 - <https://docs.unity3d.com/Manual/ExternalVersionControlSystemSupport.html>

Unity Teams

- Unity tiene integrado un sistema de control de versiones (Unity Teams)
- Es gratuito hasta para 3 desarrolladores y 1 GB de espacio



<https://unity.com/products/unity-teams>

Unity Teams

- Unity Teams tiene:
 - Herramientas para almacenar, compartir y sincronizar proyectos entre los miembros del equipo
 - Almacenamiento del proyecto en la nube
 - Mantiene el historial completo del proyecto, por lo que se pueden deshacer cambios y volver a una versión anterior en cualquier momento
 - Compilación en la nube: después de un cambio, manda un correo con el enlace a la nueva versión ya compilada

Consejos finales

- *Always playable*
 - Objetivo: siempre hay disponible una versión jugable
 - Esto permite a los desarrolladores añadir y probar elementos rápidamente (ciclos rápidos de prototipado)
 - Siempre se puede enseñar el juego en el estado en el que se encuentre
 - Se está probando el juego continuamente, lo que lo hace más robusto
 - Es más fácil estimar el estado actual del proyecto

Consejos finales

- Actualizar la copia local a menudo
- Subir código a menudo*
 - No fichero a fichero. Cambios con una razón
- Sólo subir al repositorio remoto proyectos funcionales
- Cuidado al subir sólo parte de los cambios, o no subir ficheros nuevos
- En el mensaje de commit, explicar brevemente el objetivo de los cambios realizados
- Establecer una política común y conocida con el uso del repositorio

Consejos finales

- Actualizar a la última versión (pull) antes de realizar un push
- Prueba el juego tras la actualización y verifica que funciona
- Si el programa se comporta correctamente, aunque no realice toda la funcionalidad requerida, entonces consolidar la nueva versión sin errores en el repositorio. Realizar entonces el push