GitHub Básico

Outer games Studios 🛰️

Introducción

Hasta ahora hemos visto cómo tomar screenshots de nuestro código mediante Git. Esta es una función muy útil. En esta guía veremos otra función muy útil, compartir nuestro código y colaborar con otros.

Recordemos que en la guía pasada creamos un repositorio local. Ahora buscamos poder subirlo a una plataforma (GitHub) para que otros puedan ver nuestro código.

GitHub

Cómo vimos previamente, GitHub funciona como un tipo de Google Drive, en dónde podemos hostear archivos. Más concretamente, es una plataforma en dónde se pueden hostear repositorios. Pero estos repositorios tienen algo especial, no están en tu computadora, sino que están en algún servidor. A estos repositorios los llamamos **repositorios remotos**.

En esta guía vamos a ver cómo se crea un repositorio remoto, y cómo sincronizamos nuestro repositorio local con el remoto y viceversa.

Crear

Primero debemos de crear un repositorio remoto. Esto es realmente sencillo. Solo navega a la pestaña de repositorios y haz click en el símbolo de más. Para crear un repositorio GitHub te pedirá varias cosas, pero por ahora solo ingresa el nombre y créalo.

¡Listo! Tenemos un repositorio remoto.

Conectar

Ahora lo que necesitamos en conectar nuestro repositorio local con nuestro repositorio remoto. Si observas, GitHub te muestra una página con varios comandos. Algunos de ellos ya los conoces, como `git init`, `git add` y `git commit`. Ahora nos vamos a enfocar en los últimos dos.

```
git branch -M main
```

Este comando habla de crear ramas. Este es un concepto muy importante en programación colaborativa, y la siguiente guía hablará de esto. Por ahora solo ejecutemos el comando.

Origin

El siguiente comando guarda nuestro repositorio local en el remoto. Este comando depende de lo que usas (HTTP o SSH), entonces es importante elegir la opción correcta. Yo utilizo SSH, por lo que mi comando es:

```
git remote add origin git@github.com:alexxbez/temp.git
```

Este añade nuestro repositorio remoto a nuestro Git local. Esencialmente toma el HTTP o SSH y lo guarda en una variable, `origin`. En este caso estamos diciendo que para un repositorio remoto, agregue la variable `origin` y guarde dentro de esta la dirección de nuestro repo remoto.

En este paso todavía no conectamos los dos repositorios, solo guardamos la dirección del repo remoto dentro de `origin`.

Upstream

El siguiente comando es el que finalmente conecta los dos repositorios.

```
git push -u origin main
```

Este comando hace dos cosas, conecta ambos repositorios y envía todo lo del repositorio local al repositorio remoto.

Lo veremos más adelante, pero el comando `push` manda, o empuja, todo lo que está en nuestro local al remoto.

La parte `-u` es un poco más complicada, pero es un shorthand para `--set-upstream`. Es justo esta instrucción la que conecta los dos repos, le dice a Git que todo va a venir de la siguiente dirección.

Upstream

Después sigue `origin`. Nuevamente, `origin` es una variable que contiene la dirección de nuestro repositorio remoto. Esto quiere decir que le estamos diciendo a `push` a dónde mandar los archivos, y a `-u` con quién conectar.

Finalmente tenemos `main`. Esto es nuevamente una rama, pero ahora del repositorio remoto, no el local. Hablaremos más sobre estas ramas en la siguiente guía.

Conectar

Entonces, para conectar un repo local a uno remoto, se siguen estos pasos.

- Crear una rama local `main` (lo veremos después)
- Guardar la dirección del repo remoto en `origin`
- Conectar el repo local a la rama `main` en el repo remoto y empujar todo

Si algo no queda claro o no hace sentido, puedes volver a revisar este contenido. No es tan fácil, así que no te estreses si no lo entiendes a la primera.

Workflow

Okay, hemos visto varias cosas, ahora ¿cómo juntamos todo? A continuación vamos a ver como se junta todo.

Pull

Primero vamos a introducir in último comando. Cuando estas trabajando colaborativamente, tus compañeros pudieron haber hecho cambios al código, y tú quieres poder traer esos cambios a tu computadora. ¡No quieres trabajar en código viejo!

Para esto se utiliza el siguiente comando:

```
git pull origin main
```

Este comando se traerá todo lo que se encuentra en el repositorio remoto dentro de la rama `main`.

Git

Ahora ejecutamos todo lo que sabemos de Git.

- Modificamos
- Preparamos con `git add .`
- Screenshot con `git commit -m "message"`

Push

Finalmente, pusheamos al repo remoto. Esto quiere decir, mandamos todos nuestros cambios locales al repositorio remoto.

```
git push origin main
```

Nota que ahora no estamos usando `-u`. Como ya conectamos los dos repositorios, ya no es necesario volver a conectarlos. Simplemene pusheamos a la rama main del repo remoto.

Workflow

Entonces obtenemos un workflow como este:

- Traemos los cambios del repo remoto `git pull origin main`
- Git normal `mod > stage > commit`
- Pusheamos al repo remoto `git push origin main`

Clone

Antes de irnos, es importante ver otra funcionalidad de GitHub. Hasta ahora hemos visto cómo crear tu propio repositorio, pero si trabajas en un proyecto grande lo más probable es que no sea tu repositorio. Para esto Git tiene una función muy útil `clone`.

Clonar un repositorio es esencialmente descargarlo en tu computadora, y se hace con el siguiente comando:

```
git clone <HTTP or SSH>
```

Una vez que hagas esto, si tienes permisos para editar el repositorio, ya puedes empezar a hacer cambios. Recuerda, no necesitas realizar un `git init` porque el repositorio ya contiene un `.git/`.

Trabajar en nuestro proyecto

Hasta ahora hemos visto y analizado varias funcionalidades de Git y GitHub. Sin embargo, es importante aclarar que **no es necesario hacer todo esto cuando estes trabajando en el proyecto**.

El repositorio ya está hecho y lo puedes encontrar en GitHub. Lo único que debes hacer es clonarlo desde tu computadora. Con esto habrás "descargado" el proyecto de Godot. Ahora lo único que deberías hacer es abrir ese proyecto dentro de Godot y ¡PAM!, estás listo para empezar a desarrollar.

En las siguientes guías veremos funcionalidades un poco más avanzadas de Git y GitHub, pero que son fundamentales para el desarrollo colaborativo.

Recap

- Un repo remoto se crea desde GitHub
- GitHub nos da instrucciones para no tener que memorizarlas
- `git pull` realiza `local <= remoto`
- `git push` realiza `local => remoto`
- `git clone` descarga un repositorio desde GitHub

¡Gracias!

Siguiente parte: Ramas Parte 1