Contenidos a Trabajar

1. ¿Qué es Github?
2. Crear una cuenta
3. Creación de un repositorio remoto
4. Agregar el repositorio remoto como "origin"
5. Subir cambios al repositorio remoto
6. Bajar cambios desde el repositorio remoto
7. Git Ignore

¿Qué es Github?

Github es una herramienta actualmente propiedad de Microsoft que originalmente fue pensada como una red social para programadores, donde estos pudieran subir y compartir su código, trabajar de forma colaborativa y alojar iniciativas “Open Source”.

Hoy en día se ha vuelto la opción más elegida por empresas y desarrolladores entre las diversas opciones que existen donde podemos destacar además otras alternativas como “Bitbucket” o “Gitlab”.

En particular, Github funciona como un gestor de repositorios en línea o en la “nube” que nos permite subir nuestros repositorios creados localmente con GIT. Para esto se crea un repositorio vacío en Github que luego vincularemos con nuestro repositorio local, con intención de que ambos sean un espejo del estado del proyecto permitiendo que, ya sea nosotros o los miembros del equipo en caso de trabajo colaborativo, podamos tener acceso a la versión más reciente de dicho proyecto.

Crear una Cuenta

Para crear una cuenta en GitHub, sigue estos pasos:

- Ve al sitio web de GitHub en https://github.com/.
- Haz clic en "Sign up" (registrarse) en la esquina superior derecha de la página de inicio.
- Ingresa **la misma dirección de correo electrónico que usaste al configurar GIT en tu computadora** y sigue los pasos.
- Verifica tu dirección de correo electrónico siguiendo las instrucciones en el correo electrónico que te envió GitHub.

Recuerda que si bien Github tiene una versión de pago, la versión gratuita posee todo lo que necesitamos.

Agencia de Aprendizaje a lo largo de la vida

Crear un repositorio en Github

1. Haz clic en el botón verde "New" (nuevo) que se encuentra en la página principal de GitHub.

2. Escribe el nombre que quieres darle a tu repositorio en el campo "Repository name" (nombre del repositorio).

3. Si deseas, puedes agregar una descripción en el campo "Description" (descripción).

4. Elige si deseas que tu repositorio sea público o privado.

5. Haz clic en el botón "Create repository" (crear repositorio).

Agregar el repositorio remoto como "origin"

Para vincular tu repositorio local con el remoto, debes agregar el repositorio remoto como "origin".

Para hacerlo, sigue estos pasos:

1. Abre la terminal o línea de comandos en la carpeta del proyecto que quieres vincular.
2. Escribe el siguiente comando en la terminal:

```
git remote add origin <URL del repositorio remoto>
```

Por ejemplo:

```
git remote add origin https://github.com/tu-usuario/tu-repositorio.git
```

Esto vinculará tu repositorio local con el repositorio remoto.

Subir cambios al repositorio remoto

Una vez que hayas hecho cambios en tu repositorio local, debes subirlos al repositorio remoto para mantenerlos sincronizados. Para hacerlo, sigue estos pasos:

1. Agrega los cambios a tu repositorio local usando el comando **git add**.
2. Crea un commit usando el comando **git commit**.
3. Sube los cambios a tu repositorio remoto usando el comando ***git push origin master***.

Por ejemplo:

```
git add .
git commit -m "Agregado nuevo archivo"
git push origin main
```

Bajar cambios desde el repositorio remoto

Al trabajar de manera colaborativa con otros usuarios, usualmente el repositorio de Github será actualizado con código nuevo que no tendremos en nuestro repositorio local.

Ya vimos que con el comando ***git push*** podemos enviar cambios locales hacia el repositorio de Github pero ¿Cómo podemos lograr el proceso inverso?

Para eso utilizaremos el comando ***git pull***, con él podremos descargar todo el historial nuevo de una rama en específico y además la información sobre nuevas ramas creadas en el repositorio.

Por ejemplo con el comando:

```
git pull origin master
```

descargará los cambios del historial de GIT del repositorio remoto y los fusionará con los cambios locales en tu máquina.

Si tienes varias ramas en tu repositorio remoto, debes especificar la rama que deseas descargar. Para hacerlo, reemplaza "master" en el comando anterior con el nombre de la rama que deseas descargar.

Es importante tener en cuenta que si has realizado cambios locales que entran en conflicto con los cambios del repositorio remoto, Git mostrará un mensaje de conflicto y te pedirá que los resuelvas antes de fusionar los cambios.

En resumen, descargar cambios en el historial de Git del repositorio remoto en tu máquina es un proceso simple que se puede realizar usando el comando **git pull origin <nombre de la rama>** en la terminal o línea de comandos.

Git Ignore

Git tiene una herramienta imprescindible casi en cualquier proyecto, el archivo ".gitignore", que sirve para decirle a Git qué archivos o directorios completos debe ignorar y no subir al repositorio de código.

Su implementación es muy sencilla, por lo que no hay motivo para no usarlo en cualquier proyecto y para cualquier nivel de conocimientos de Git que tenga el desarrollador. Únicamente se necesita crear un archivo especificando qué elementos se deben ignorar y, a partir de entonces, realizar el resto del proceso para trabajo con Git de manera habitual.

En el .gitignore se especificarán todas las rutas y archivos que no se requieren y con ello, el proceso de control de versiones simplemente ignorará estos archivos.

Por qué usar gitignore

Piensa que no todos los archivos y carpetas son necesarios de gestionar a partir del sistema de control de versiones. Hay código que no necesitas enviar a Git, ya sea porque sea privado para un desarrollador en concreto y no lo necesiten (o lo deban) conocer el resto de las personas. Pueden ser también archivos binarios con datos que no necesitas mantener en el control de versiones, como diagramas, instaladores de software, etc.

El ejemplo más claro que se puede dar surge cuando se trabaja con sistemas de gestión de dependencias, como npm, Bower, Composer. Al instalar las dependencias se descargan muchos archivos con documentos, tests, demos, etc. Todo eso no es necesario que se mantenga en el sistema de gestión de versiones, porque no forma parte del código de nuestro proyecto en concreto, sino que es código de terceros. Si Git ignora todos esos archivos, el peso total del proyecto será mucho menor y eso redundará en un mejor mantenimiento y distribución del código.

Otro claro ejemplo de uso de gitignore son los archivos que crean los sistemas operativos automáticamente, archivos que muchas veces están ocultos y no los vemos, pero que existen. Si no evitas que Git los procese, estarán en tu proyecto como cualquier otro archivo de código y generalmente es algo que no quieres que ocurra.

Implementar el gitignore

Como hemos dicho, si algo caracteriza a gitignore es que es muy fácil de usar. Simplemente tienes que crear un archivo que se llama ".gitignore" en la carpeta raíz de tu proyecto. Como puedes observar, es un archivo oculto, ya que comienza por un punto ".".

Dentro del archivo .gitignore colocarás texto plano, con todas las carpetas que quieres que Git simplemente ignore, así como los archivos.

La notación es muy simple. Por ejemplo, si indicamos la línea node_modules/

Estamos evitando que se procese en el control de versiones todo el contenido de la carpeta "node_modules".

Si colocamos la siguiente línea:

*.DS_Store

Estaremos evitando que el sistema de control de versiones procese todos los archivos acabados de .DS_Store, que son ficheros de esos que crea el sistema operativo del Mac (OS X) automáticamente.

Hay muchos tipos de patrones aplicables a la hora de especificar grupos de ficheros, con comodines diversos, que puedes usar para poder indicar, de manera muy específica, lo que quieres que Git no procese al realizar el control de versiones.

Eliminar archivos del repositorio si te has olvidado el .gitignore

Nos ha pasado a todos más de una vez que se nos olvida generar el correspondiente .gitignore después de haber hecho un commit. Observarás que, por mucho que estés diciendo que ahora sí quieres ignorar ciertas carpetas o rutas, éstas continúan en tu repositorio. Básicamente, ésto es así porque estaban allí antes de informar que se debían ignorar. En los siguientes

commits serán ignoradas todas las modificaciones de las carpetas en cuestión, pero lo que había antes perdurará en el repositorio.

Por ejemplo, imagina que estás en un proyecto NodeJS. Olvidas de hacer el gitignore de la carpeta "node_modules". Entonces haces un commit y metes un montón de dependencias a tu repositorio git, que no debían estar. Si ves ahora en un sistema de interfaz gráfica tu repositorio (o subiéndolo a Github) podrás observar que los archivos de "node_modules" están ahí.

¿Qué hacer entonces?

Básicamente lo solucionas con un comando de Git llamado "rm" que básicamente funciona igual que el comando del mismo nombre que usas para borrar archivos en una consola del estilo de Mac o Linux.

git rm -r --cached node_modules

Luego tendrás que hacer un commit para que esos cambios se apliquen al sistema de control de versiones.

git commit -m 'Eliminada carpeta node_modules del repo'

A partir de ahora esa carpeta no se verá en tu repositorio y gracias a tu .gitignore tampoco se tendrá en cuenta en las siguientes operaciones que realices mediante Git.

Descargar vs Clonar

Al inicio de uso de un sitio como GitHub, si no tenemos ni idea de usar Git, también podemos obtener el código de un repositorio descargando un simple Zip.

Sin embargo, descargar un repositorio así, aunque muy sencillo no te permite algunas de las utilidades interesantes de clonarlo, como:

- No crea un repositorio Git en local con los cambios que el repositorio remoto ha tenido a lo largo del tiempo. Es decir, te descargas el código, pero nada más.
- No podrás luego enviar cambios al repositorio remoto, una vez los hayas realizado en local.

En resumen, no podrás usar en general las ventajas de Git en el código descargado. Así que es mejor clonar, ya que aprender a realizar este paso es también muy sencillo.

Clonar el repositorio Git

Para vincular tu repositorio local con el remoto, debes clonar el repositorio remoto en tu máquina. Para hacerlo, sigue estos pasos:

1. Copia la URL del repositorio remoto de GitHub.

2. Abre la terminal o línea de comandos en tu máquina y navega hasta la carpeta donde deseas clonar el repositorio.

Escribe el siguiente comando en la terminal:

```
git clone <URL del repositorio remoto>
```

Por ejemplo:

```
git clone https://github.com/tu-usuario/tu-repositorio.git
```

Esto creará una copia del repositorio remoto en tu máquina.