INTRODUCCION AL USO DE GIT / GITHUB

Git es un sistema de control de versiones, es decir, un sistema que nos permite llevar un historial automático de todas las actualizaciones que vayamos realizando en un directorio dado.

Para ello debemos inicializar un "repositorio" dentro de ese directorio, y a partir de allí git llevará un control de todos los movimientos de archivos que se realicen en la carpeta. Actualmente podemos administrarlo tanto desde línea de comandos como de forma visual, sea en Linux, Mac o Windows. Como alternativa, podemos utilizar también algún IDE de programación popular (como Visual Studio Code, Sublime Text, Atom, etc), y gestionar Git mediante un plugin dentro del mismo editor.

PASO 1, instalación de Git:

Lo descargamos gratuitamente desde https://git-scm.com/downloads. Una vez instalado, como decíamos, podremos manejarlo tanto de forma visual como desde consola, para crear y controlar repositorios. En nuestro caso, por comodidad, utilizaremos Visual Studio Code.

PASO 2, instalación de Visual Studio Code:

VSC es un IDE (Entorno Integrado de Desarrollo), es decir, un editor para programar cómodamente en distintos tipos de lenguaje. Es actualmente muy popular y muy completo, a través del uso de plugins para incorporar distintas capacidades según sea necesario. VSC incluye de forma predeterminada una serie de herramientas para trabajar con Git.

PASO 3, creación de una cuenta Github:

Github es esencialmente una plataforma online para administrar proyectos git, desde https://github.com/ nos crearemos gratuitamente una cuenta que nos permitirá comenzar a utilizar el sistema.

La comodidad de trabajar con una cuenta online, lógicamente, será que podremos colocar contenidos allí y mantenerlos sincronizados con nuestros repositorios locales. Por supuesto no es imprescindible, podemos tranquilamente trabajar solo local, pero

la alternativa de sincronizar con un repositorio remoto es muy práctica para disponer de una copia remota siempre actualizada.

PASO 4, creación de un repositorio en Github:

Una vez creada nuestra cuenta en Github, solo tendremos que ingresar con las credenciales (email y clave) que hayamos elegido, y crear un repositorio. Para ello nos dirigiremos al tabulador Repositorios (Repositories), y pulsaremos en el botón Nuevo (New). Con solo dar un nombre e indicar si queremos que nuestro repo sea público o privado, el sistema inicializará todo lo necesario.

La dirección en la que encontraremos el repo, siempre comenzará con la identificación de nuestra cuenta, por lo tanto, si nuestro usuario Github fuese "usuariogh", y el nombre del nuevo repositorio "repo01", la url del repo sería https://github.com/usuariogh/repo01. Obviamente, más allá de esto tendremos siempre listados los repositorios activos dentro de nuestra cuenta, con enlaces para acceder a ellos.

PASO 5, clonación de un repositorio:

Como mencionamos, mediante la propia herramienta git, podremos gestionar un repositorio en nuestra máquina y mantenernos trabajando de forma local, pero aprovecharemos la comodidad de Github y Visual Studio Code para crear un repositorio remoto, clonarlo de forma local y sincronizarlo cada vez que lo deseemos, de esa forma tendremos un backup en Github de todas las actualizaciones que vayamos realizando, manejándonos de manera cómoda desde el entorno visual de VSC.

Para clonar un repo, abriremos VSC y en la columna de opciones de la izquierda, elegiremos el ícono de Source Control (Control de versiones). Allí veremos un botón Clonar Repo (Clone Repository), solamente tendremos que pegar la URL del repositorio que nos interese (recordar la mención de url en el paso anterior), y el sistema automáticamente creará un repo local, copiando los archivos que se encuentren en el repo de Github. Para ello nos pedirá el nombre de una carpeta local donde deseemos colocarlos, y de no estar actualmente logueados en Github, nos solicitará nuestros datos de usuario para poder ingresar, luego procederá a la clonación.

PASO 6, actualizaciones:

La comodidad de esta clonación, es que podremos mantener relacionados nuestro repo local y el repo remoto en Github. Si intentan abrir en VSC cualquier archivo del repo local, lo modifican y lo guardan, verán que el ícono de Control de versiones (Source Control), detecta el cambio. Si ingresan allí verán que se ha creado un registro de qué archivo fue modificado y qué alteraciones se han hecho.

En cualquier momento, pulsando el tilde de la barra superior, podemos realizar lo que se denomina un "Commit". Un commit no es más que una confirmación, que indica que esos cambios son válidos y queremos mantenerlos (lógicamente entre las opciones dispondremos también la posibilidad de deshacer cambios). Siempre que realicemos un commit, el sistema nos pedirá que ingresemos un pequeño texto, indicando qué cambios se han hecho, será una breve referencia que aparecerá en los repositorios.

Una vez hecho el commit de todos los archivos que deseemos modificar, podremos optar por varias alternativas. En el menu superior, cerca del tilde, observarán 3 puntos (…) que despliegan un menu de opciones, de ellas nos concentraremos en 3:

- Pull: el comando pull de git, recupera contenidos desde un repositorio remoto (el de Github en este caso) hacia el local en el que estamos trabajando (el de nuestra máquina). Para este ejemplo que estamos siguiendo, NO debemos utilizarlo, ya que de hacerlo perderíamos las modificaciones que acabamos de confirmar con Commit localmente.

- Push: el comando push por su parte, opera en sentido contrario, envía contenidos desde el repositorio en el que estamos trabajando (el local de nuestra computadora en este caso) hacia el remoto (el de Github). Para este ejemplo, si usáramos push luego de realizar el commit, SI funcionaría, y lograríamos actualizar el contenido en Github con los cambios que acabamos de efectuar localmente.

- Sync: el comando sync, trabaja de forma bidireccional entre un repo local y uno remoto, sincronizando cualquier cambio hecho en cualquiera de los 2 repos hacia el otro. Este comando siempre tardará algo más de tiempo en ejecutarse respecto a pull o push, pero nos garantizará que cualquier cambio hecho sea de un lado o del otro, se reflejará, es decir, ambas copias (la local y la remota) tendrán en todo momento los mismos contenidos.

Si bien git es una herramienta que cuenta con muchas más alternativas, con estos pasos básicos estaremos ya en condiciones de crear un repo, tanto local como remoto, y mantener ambos en sincronía, pudiendo de esa forma trabajar cómodamente en un equipo local y actualizar cambios hacia el repo remoto, donde otros pueden públicamente acceder y visualizar o clonar esos contenidos en sus propios repos.