**¿Qué es Git-Hub?**

En un nivel más alto, GitHub es un sitio web y un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código. Para entender exactamente qué es GitHub, primero usted necesita conocer los dos principios que lo conectan:

* **Control de versión**
* **Git**

**¿Qué Es una Versión de Control?**

Una Versión de Control ayuda a los desarrolladores llevar un registro y administrar cualquier cambio en el código del proyecto de software. A medida que crece este proyecto, la versión de control se vuelve esencial.

Por esto, la versión de control permite al desarrollador trabajar de forma segura a través de una **bifurcación** y una **fusión**.

Con la bifurcación, un desarrollador duplica parte del código fuente (llamado repositorio). Este desarrollador, luego puede, de forma segura, hacer cambios a esa parte del código, sin afectar al resto del proyecto.

Luego, una vez que el desarrollador logre que su parte del código funcione de forma apropiada, esta persona podría fusionar este código al código fuente principal para hacerlo oficial.

Todos estos cambios luego son registrados y pueden ser revertidos si es necesario.

**¿Qué Es Git?**

Específicamente, Git es un sistema de control de versión distribuida, lo que quiere decir que la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones.

**HEAD:** Estado actual de nuestro código, es decir, donde nos colocó Git

**Working tree:** Lugar donde los archivos realmente están siendo almacenados

**index:** Lugar donde Git almacena lo que será commiteado, es decir, la ubicación entre el working tree y el repositorio de Git en sí.

**git commit -m**

Sirve para guardas los cambios realizados en el código.

**git log -p**

Nos muestra todos los cambios realizados a detalles. (:Q) para salir.

**git log --oneline**

Nos muestra todos los cambios realizados en una sola línea-

* Que un commit es la forma de guardar un estado o versión de nuestro código;
* Cómo agregar archivos para commitear con git add;
* Cómo commitear archivos usando el comando git commit;
* Cómo verificar el historial de commits, a través de git log y algunas de sus opciones:
* **git log --oneline**
* **git log -p**
* **git log --pretty="**parámetros de formato"
* Cómo hacer que Git no monitoree archivos, a través de .**gitignore**
* Que no debemos realizar commits, es decir, guardar un estado, de nuestra aplicación que no esté funcionando.

En esta aula aprendimos:

* Lo que son los repositorios remotos
* Cómo crear un repositorio de Git sin una copia de los archivos (con --bare) para usar como servidor;
* Cómo agregar links a repositorios remotos, con el comando git remote add;
* Cómo descargar un repositorio por primera vez clonándolo con el comando git clone;
* Cómo enviar nuestros cambios a un repositorio remoto, con git push;
* Cómo actualizar nuestro repositorio con los datos en el repositorio remoto, usando git pull;
* Qué es **GitHub** y para qué sirve;
* Cómo crear un repositorio en **GitHub**;
* Cómo agregar un repositorio de **GitHub** como repositorio remoto.