Git-Hub y Git

Para entender el funcionamiento de Git-Hub primero hay que comprender Git.

[](https://git-scm.com/)

Software de control de versiones (GIT):

El propósito por el cual fue creado este software gratis y abierto desarrollado por Linus Torvalds es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por muchas personas, que incide en numerosos detalles de rendimiento.

Características:

Desarrollo NO lineal, por lo tanto rapidez en la gestión de ramas y mezclado de diferentes versiones.

Gestión distribuida: Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.

Comentarios: Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio.

Re-almacenamiento periódico en paquetes (ficheros): Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el re-empaquetado(con base en diferencias) no ocurre cada cierto tiempo.

Modalidad conveniente de trabajo

Una buena práctica para hacer uso de Git ya sea en forma personal o en equipo es la de utilizar 4 ramas; Master, Development(Desarrollo), Features (características) y Hotfix (Revisión).

Rama Master: Es la rama principal. Contiene el repositorio que se encuentra publicado en producción, por lo que debe estar siempre estable.

Rama de Desarrollo: Es una rama que se desprende de la principal. Es la rama de integración, todas las nuevas funcionalidades se deben integrar en esta rama. Luego que se realice la integración y se corrijan los errores (en caso de existir), es decir que la rama se encuentra estable, se puede hacer un “merge” (fusión) de development sobre la rama master.

Rama de características: Cada nueva funcionalidad se debe realizar en una rama nueva, específica para esa funcionalidad. Estas se deben sacar de la rama de desarrollo. Una vez que la funcionalidad esté desarrollada, se hace una unión de la rama sobre la de desarrollo, donde se integrará con las demás funcionalidades.

Rama de revisión: Son errores de software que surgen en producción, por lo que se deben arreglar y publicar de forma urgente. Es por ello, que son ramas sacadas de la principal. Una vez corregido el error, se debe realizar una unificación de la rama sobre la principal. Al final, para que no quede desactualizada, se debe realizar la unificación de Master sobre Development.

Órdenes básicas

Aclaración: Git está pensado y diseñado para ser utilizado tan solo con la terminal, pero por una cuestión de facilidad nosotros utilizaremos Git-hub (puede ser usado desde la web) o Git-hub Desktop. Sin embargo, a continuación dejo los comandos más comunes en caso de que quieran utilizarlos:

Git clone + <url obtenida online en la página de git-hub:

Copia todo el repositorio a la carpeta/directorio en el que me encuentre parado.

Git push origin <nombre\_rama>:

Sube la rama “nombre\_rama” al servidor remoto

Git status:

Muestra el estado actual de la rama, como los cambios realizados sin comentarios

Git add <nombre\_archivo>:

Agrega a mi repositorio local ese archivo. No lo sube al servidor.

Git commit –am <”comentario”>:

Comenta el cambio realizado posterior a git add y confirma el cambio realizado.

Git push:

Este comando sube todos los cambios hechos en mi repositorio local y los sincroniza con el repositorio de git-hub que tenga.

Git pull:

Este comando se utiliza una vez que ya tengo clonado el repositorio en mi máquina, para bajar los cambios que se hayan realizado en la nube (servidor remoto).

Git-hub:

Es una plataforma de desarrollo colaborativo para alojar proyectos utilizando Git.

Forma de trabajo con git-hub Desktop o git-hub web:

1º) Creación del repositorio:

Agregar los “contribuidores” o colaboradores del proyecto para que todos puedan editarlo.

2º) Creación de rama/s:

Lo importante es siempre no trabajar sobre la rama principal a menos que estemos seguro del desarrollo o característica a implementar.

3º) Comentar los cambios:

Es importante que a pesar de que git-hub indique que sea una breve descripción, los comentarios sean lo más exactos posibles para una mejor y más rápida comprensión del nuevo código.

4º) Solicitar una revisión de los cambios:

Esta es, quizás, la característica más importante de Git-hub. “Pull request” va a mostrar las diferencias, cambios que existan entre dos ramas. Se muestran en rojo y verde.

Cuando se realiza esta solicitud Git-hub permite que el usuario indique con quién quiere discutir los cambios realizados, es decir, que colaborador quiere que revise esos cambios. En caso de no llegar a un acuerdo o que se decida que todavía hay que seguir modificando el código, es posible cerrar la solicitud.

5º) Fusionar la solicitud de revisión:

Este último paso se realiza cuando los cambios realizados han sido aceptados entre los programadores y se desea incorporar esos cambios a la rama en cuestión. Una vez que se ha unido la rama en cuestión a la principal por ejemplo, es una buena práctica eliminar la rama que ahora ha quedado obsoleta.