

git y github

git para que?

Git es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad.

Git se ha ideado para posibilitar la ramificación y el etiquetado como procesos de primera importancia (a diferencia de SVN) y las operaciones que afectan a las ramas y las etiquetas (como la fusión o la reversión) también se almacenan en el historial de cambios.

github para que?

GitHub es un sitio "social coding". Te permite subir repositorios de código para almacenarlo en el sistema de control de versiones Git.

git y github

control de versiones

una version de control (tambien conocida como control de versiones) es un sistema que facilita que los desarrolladores puedan organizar y llevar el registro de cualquier modificacion en el codigo de un proyecto de software.

La finalidad de administrar los cambios es recuperar versiones puntuales si lo deseamos en algun momento. Ademas de permitir al programador poder realizar sus labores de desarrollo de forma segura mediante dos procesos claves:



Una bifurcacion: mediante la cual el desarrollador duplica una parte del codigo fuente (conocido como repositorio). Con esta funcion, se puede alterar esa porcion de codigo sin que el resto del proyecto se vea afectado.



Una fusion: cuando el programador consigue que su parte del codigo se ejecute correctamente, puede fusionar su porcion al codigo fuente principal y volverlo oficial. Los cambios son registrados y revertidos, en caso de ser necesario.

git y github

comandos de git

git init: inicia un repositorio local en nuestro proyecto

git add: anade los cambios realizados en nuestro proyecto

git commit -m "comentario": Este sea quizas el comando mas utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizas despues de una tarea o asunto especifico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver mas tarde si es necesario.

Tambien necesitamos escribir un mensaje corto para explicar que hemos desarrollado o modificado en el codigo fuente.

Importante: Git commit guarda tus cambios unicamente en local.

git status: El comando de git status nos da toda la informacion necesaria sobre la rama actual.

Podemos encontrar informacion como:

- Si la rama actual esta actualizada
- Si hay algo para confirmar, enviar o recibir (pull).
- Si hay archivos en preparacion (staged), sin preparacion(unstaged) o que no estan recibiendo seguimiento (untracked)
- Si hay archivos creados, modificados o eliminados

git y github

comandos de git

Git clone: Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). En otras palabras, Git clone básicamente realiza una copia idéntica de la última versión de un proyecto en un repositorio y la guarda en tu ordenador.

cd nombreRepositorio: permite acceder al repositorio en la nube GitHub con todas sus ramas

Git branch: Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.

Creando una nueva rama:

```
git branch <nombre-de-la-rama>
```

Este comando creará una rama en local. Para enviar (push) la nueva rama al repositorio remoto, necesitarás usar el siguiente comando:

```
git push <nombre-remoto> <nombre-rama>
```

Visualización de ramas:

```
git branch  
git branch --list
```

Borrar una rama:

```
git branch -d <nombre-de-la-rama>
```

git y github

comandos de git

Git pull: El comando git pull se utiliza para recibir actualizaciones del repositorio remoto. Este comando es una combinación del git fetch y del git merge lo cual significa que cuando usemos el git pull recogeremos actualizaciones del repositorio remoto (git fetch) e inmediatamente aplicamos estos últimos cambios en local (git merge).

Git merge ramaDePrueba: agrega los cambios de una rama a la rama master o main

Git checkout: Este es tambien uno de los comandos mas utilizados en Git. Para trabajar en una rama, primero tienes que cambiarte a ella. Usaremos git checkout principalmente para cambiarte de una rama a otra. Tambien lo podemos usar para chequear archivos y commits.

`git checkout <nombre-de-la-rama>`

Hay algunos pasos que debes seguir para cambiarte exitosamente entre ramas:

- Los cambios en tu rama actual tienen que ser confirmados o almacenados en el guardado rapido (stash) antes de que cambies de rama.
- La rama a la que te quieras cambiar debe existir en local.

Hay tambien un comando de acceso directo que te permite crear y cambiarte a esa rama al mismo tiempo:

`git checkout -b <nombre-de-tu-rama>`

Este comando crea una nueva rama en local (-b viene de rama (branch)) y te cambia a la rama que acabas de crear.

git y github

comandos de git

Git merge ramaDePrueba: Cuando ya hayas completado el desarrollo de tu proyecto en tu rama y todo funcione correctamente, el ultimo paso es fusionar la rama con su rama padre (dev o master). Esto se hace con el comando `git merge`.

Git merge basicamente integra las caracteristicas de tu rama con todos los commits realizados a las ramas dev (o master). Es importante que recuerdes que tienes que estar en esa rama especifica que quieres fusionar con tu rama de caracteristicas.

Por ejemplo, cuando quieres fusionar tu rama de caracteristicas en la rama dev:

Primero, debes cambiarte a la rama dev:

```
git checkout dev
```

Antes de fusionar, debes actualizar tu rama dev local:

```
git fetch
```

Por ultimo, puedes fusionar tu rama de caracteristicas en la rama dev:

```
git merge <nombre-de-la-rama>
```

Pista: Asegurate de que tu rama dev tiene la ultima version antes de fusionar otras ramas, si no, te enfrentaras a conflictos u otros problemas no deseados.