EVER HINOJOSA AGUIRRE

GIT BASICS

- Git es un Sistema de Control de Versiones(VCS) el cual permite mantenerse al tanto y manejar los archivos en un proyecto.

Cuando pienso en un grupo de desarrolladores trabajando en un proyecto, siempre me imagino un grupo de trabajadores trabajando en una casa. Git en este caso permitiría guardar la casa en los diferentes estados de su proceso.

GIT BASICS

Un Repositorio de Git o ¨Repo¨ es un workspace que se mantiene al tanto y maneja los archivos de un folder. Cuando se quiere iniciar un nuevo repositorio se puede generar uno desde cero desde el mismo perfil de GitHub de una persona o se puede iniciar el Repositorio de un proyecto que no tenga uno con el comando:

git init

Este no debe usarse dentro de un repositorio ya creado, para revisar si un proyecto ya tiene un repositorio de git se puede usar el comando:

git status

Este comando proporciona información sobre el estatus el repositorio actual y sus contenidos. (ver que archivos no hemos guardado).

GIT BASICS

En caso de que ya tengamos nuestro repositorio y ya tengamos algo de progreso en este podemos ¨Hacer commit¨ de nuestro progreso y guardar lo que tengamos hasta el momento pero antes de guardar este progreso tenemos que decidir que queremos guardar. En este caso usamos este comando:

git add . para poner todos los archivos con cambios en el área de guardado

git add ejemplo1 ejemplo2 En lugar de un ¨.¨ que representa todo podemos poner los nombres de los archivos específicos que queremos agregar.

El comando git add se usa para agregar o ¨stage¨ los cambios a una zona especial ¨Stage area¨ esta área para git son los cambios que se se subirán en nuestro próximo commit.

GIT BASICS

Una vez que se agregaron al ¨stage area¨ todos los cambios que se quieren agregar al repositorio, la manera de empaquetar los cambios en la ¨stage area¨ es con el comando previamente mencionado:

git commit

El commit requerirá que se agregue un mensaje y si la mejor manera de agregar este mensaje sin problemas es con:

git commit –m ¨mensaje¨

El mensaje, agregado con la bandera **–m**, siendo algo importante o relevante al commit en caso de que se quiera hacer commit a los cambios digamos de la casa que estamos construyendo y se pintaron las paredes el mensaje seria algo como ¨Se pintaron las paredes de verde¨.

GIT BASICS

En caso de que faltara algún documento en el commit o se causara un typo el mensaje del commit en lugar de generar otro commit se puede usar la opción –amend de la próxima manera:

git commit –m ¨mensaje¨

git add archivoolvidado

git commit --amend

Algunas buenas practicas cuando se haga commit es que se debe hacer temprano y amenudo, agrupar cambios similares / no hacerle commit a muchas cosas y escribir mensajes de commit que sean concisos y den una buena idea del punto del commit.

GIT BASICS

Al momento de subir archivos al repositorio la mejor manera de evitar que git tome archivos que no tienen importancia o NO se deben subir al repositorio como por ejemplo credenciales, logs, Dependencias o paquetes, configuraciones de desarrollo, etc. Se puede usar el archivo .**gitignore** en la raíz del repositorio para ignorar los archivos que no se quieran subir al repositorio.

Para ignorar ciertos archivos se ponen de la siguiente manera lso archivos dentro del .gitignore

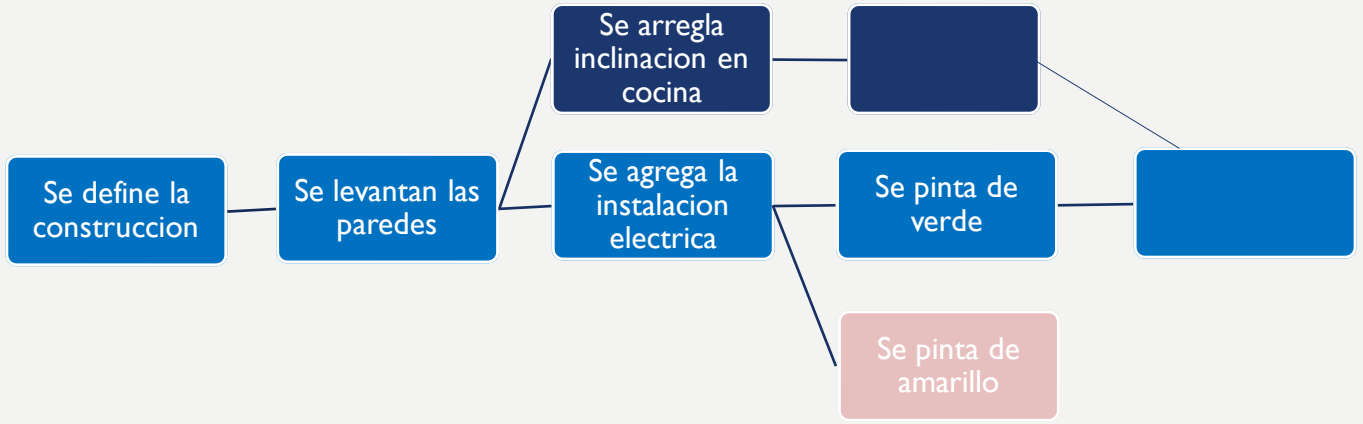
.archivo ignorara cualquier archivo llamado .archivo

Nombredelfolder/ ignorara todo el directorio señalado

***.log** ignorara todos los archivos con la extensión .log

BRANCHES

Branches o ramas son una parte importante de Git, nos permiten probar nuevas ideas en un proyecto, o trabajar varias ideas al mismo tiempo. Si trabajamos en una rama esta no afectara los cambios de otras ramas a menos que los unamos (merge changes).



En este caso la rama azul es nuestra rama principal (**master o main**) es en la que siempre se esta trabajando. Cuando estamos trabajando el termino **HEAD** se refiere al ¨lugar¨ en el que estamos trabajando en el repositorio.

BRANCHES

Para ver en que rama estamos trabajando podemos usar el comando:

git branch

con este comando podemos ver nuestras ramas existentes, la rama defecto es main aunque esto se puede cambiar. Para crear una nueva rama basada en donde estemos ósea HEAD podemos usar:

git branch <branch-name>

Este comando crea una rama, pero no cambia la rama donde estemos trabajando (HEAD se queda dónde está).

BRANCHES

Para cambiar la rama en la que estamos trabajando podemos usar:

git switch <branch-name> Cambia la rama a la que indiquemos

git checkout <branch-name> hace muchas más cosas, pero también se puede usar sin problemas para cambiar la rama

git switch –c <branch-name> el –c se puede usar para indicar que se quiere crear una nueva rama y cambiar a esta en un solo comando.

git switch –c refactor la bandera –c nuevamente se usa para crear una nueva branch y en este caso refactorar hacia esta.

En todo caso las ramas nuevas que se generen estas se crean basadas en HEAD.

BRANCHES

- En cualquier momento podemos usar la bandera –v para revisar mas información sobre una rama:

  git branch -v

MERGING

Cuando se quiere combinar(merge) dos ramas se debe recordar 2 conceptos:

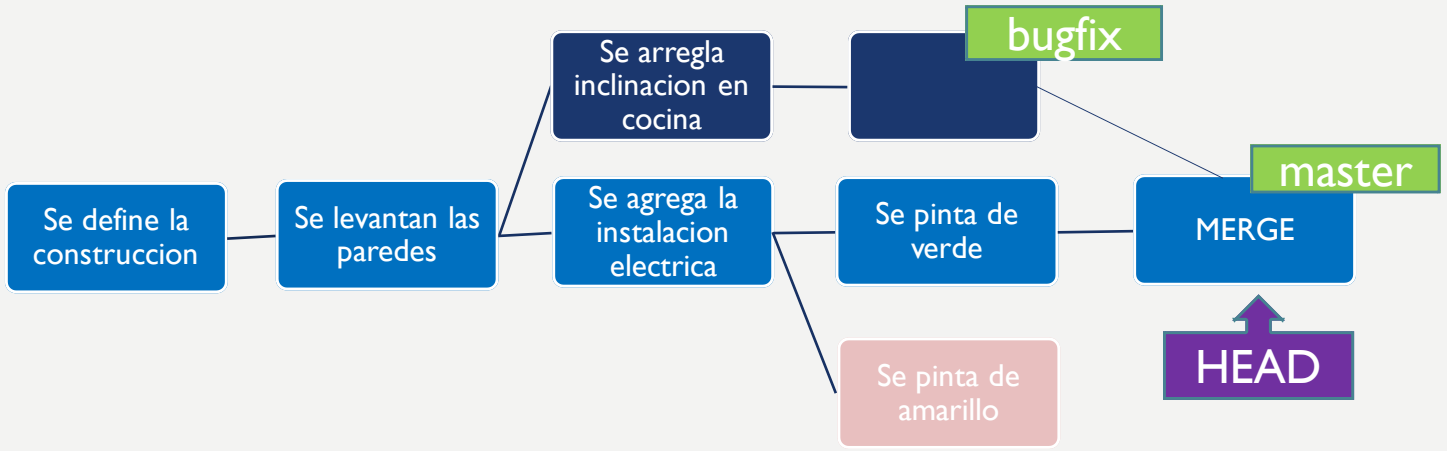
- Se hace merge a ramas No a commits.
- El merge siempre se dirige a la branch a la que HEAD este apuntando.

MERGING

Para hacer un merge se deben seguir ciertos pasos. Primero se tiene que cambiar con un switch o checkout hacia la rama a la que se quiere hacer el merge (la rama que recibirá los cambios) después usamos el comando git merge para combinar los cambios de una rama en especifico hacia la rama en la que estemos (HEAD).

git switch master

git merge bugfix



MERGING

En algunos casos Git no podría combinar las ramas automáticamente y esto resultara en un ***Merge Conflict*** que se necesitara resolver manualmente.

Para resolver dichos conflictos se deben seguir estos pasos:

1. Abrir el archivo conque tenga conflicto
2. Editar el archivo para remover dichos conflictos y decidir que contenido mantener.
3. Remover los marcadores de conflicto en el documento.
4. Agregar los cambios hechos y hacer un commit.

MERGING

Para borrar una branch se puede usar el comando donde la bandera –d borra:

git branch -d branchname

CONCLUSIONES

- En conclusión aunque ya tenia algo de conocimiento básico de GIT, si aprendí varias cosas sobre este que no sabia pues realmente lo que sabia de git era información que aprendí yo mismo, cualquier herramienta de versionamiento es necesaria para cualquier desarrollador si es que se busca trabajar en un equipo porque esta habilita esto y el saber las funciones de la herramienta facilita el trabajo en equipo lo que es la base de cualquier proyecto grande.

p.s. Grande Linus Torvalds