Que es github:

Es un servicio de hospedaje de repositorios basado en una web, es lo mismo que tener tus scripts en una nube, en un perfil en la red.

Es importante no poner en github información confidencial de la empresa o informaciones sensibles ya que es un blanco fácil para ataques cibernéticos, en su lugar, es mejor usar un servido privado.

Ya tengo una cuenta en github, aquí se pueden crear repositorios en la nube de manera fácil usando el wizzard.

Se puede copiar o clonar este repositorio descargándolo a nuestra computadora usando el siguiente comando **git clone** [**https://github.com/Mike-Ehl/health-check.git**](https://github.com/Mike-Ehl/health-check.git), donde ese url, es la ubicación, que es lo que descargaremos, para esto necesitaremos autenticar nuestro nombre de usuario y contraseña o autenticarnos desde el browser.

Github es super visual y hacer super fácil entender lo que ha pasado con los archivos.

Podemos enviar todas las informaciones de la historia a github con solamente accionar el comando **git push**. Otros parámetros que le podemos pasar a este comando son los siguientes **git push -u origin refactor**, siendo -**u**, un alias para crear una rama al vuelo, **origin** el nombre del repositorio, y **refactor** el nombre de la rama ala que queremos subir el cambio.

Cuando queremos descargar los cambios que se han hecho por otras personas, o simplemente cambios que hicimos en github a nuestro repositorio local, podemos usar **git pull** si ya nos hemos autentificado antes.

Cuando usamos **git clone** git hace un repositorio local con nombre **origin,** si usamos el comando **git remote -v** podemos ver las url que están indexadas a este repositorio. Normalmente hay dos urls, una para hacer push y la otra para hacer fetch, normalmente, la url debe ser la misma.

Normalmente todos los repositorios remotos tienen un nombre asignado, eso nos deja llevar cuenta de distintos repositorios.

Si queremos más información sobre el repositorio, podemos usar **git remote show origin** siendo origin el nombre del repositorio. Esto nos deja ver tanto las urls de subida u bajada como así también, las ramas que existen tanto local como remotamente.

Si estamos trabajando en un repositorio remoto, obviamente las ramas que obtendremso, serán ramas remotas o remote Branches, para acceder a las remotes Branches solamente tenemos que usar el mismo comando que usamos para acceder a las ramas del repositorio que estamos trabajando como haríamos normalmente, solo que con la extensión r:  **git branch -r**. Estas ramas son de solo lectura, no las podemos modificar, para poder modificarlas deberemos ir a través del flujo de trabajo normal que mencionamos antes para los repositorios locales.

Para esto necesitamos primero que todo introducir a nuestra rama local, algún cambio que haya habido a nivel remoto así no haya que hacer merge entre nuestra rama y algún contenido de la rama principal del repositorio remoto, luego hacemos merge entre la rama de trabajo que estamos llevando y la rama principal del repositorio remoto, y entonces, es que podemos hacer push, y enviar los cambios a la master branch remota.

Para conocer el estado de las ramas remotas simplemente debemos hacer uso de **git status** esto nos devolver un mensaje diciéndonos si estamos up to date o no con la rama principal remota.

Si corremos **git fetch** este comando lo que hace es descargarnos la información de todas las ramas remotas que hay en el repositorio remoto, y nos dice en que posición estamos con respecto a las demás ramas, esto no realiza ningún cambio como pull o push, simplemente es para reflejar los cambios hechos por otros. Para ver nuestra posición podemos usar **git log** y esto nos mostrara en que posición estamos con respecto a las demás ramas.

También podemos usar **git remote update** para poder ver esto con todos los elementos ya que fetch solamente lo hace con uno en específico.

Luego de esto podemos correr merge, y actualizar nuestra rama en conjunto con la rama remota.

Si queremos simplemente actualizar nuestra rama, sin hacer el fetch simplemente usamos **git pull**

Cuando estamos usando **git remote show origin** siendo origin el nombre el repositorio, y vemos que ahora existe una nueva rama que no tenemos en nuestro repositorio local, y la queremos incluir, es tan fácil como hacer **git checkout new-feature**, siendo new feature la nueva rama, y así se creará una copia de esta localmente, y el work tree, también se actualizará.

Rebase:

Una vez que nuestra rama ha sido probada pude ser fusionada en la rama principal, o podemos hacer rebase, o sea, podemos cambiar el commit base que usamos para esta rama. Cada rama tiene un commit base del que parte, lo que hacemos con rebase, es hacer que el commit base para esta rama, sea el ultimo commit, ósea, convertimos esto en una sola rama, una historia linear.

EJ:

push push | \ | 3 2.1 3—2.1 | / | 2 2 | | 1 1

En este ejemplo vemos que la rama principal se separa en dos, sigue a tres y luego, el cuarto commit es la unión de las ramas máster y la de la versión 2.1, y entonces esta versión 4 es la que podemos enviar al repositorio remoto y haciendo push.

En el segundo ejemplo, vemos que hemos cambiado el commit base de la rama, ya la rama no sale del commit 2 como una versión 2.1, sino que lo que hacemos es simplemente convertirla en un commit linear con la master, lo que no nos dará ningún error de conflicto obviamente siempre y cuando no se hayan hecho modificaciones en las mismas líneas que la master, y hará un fast-forward, luego de esto, podemos hacer pull,