**GIT**

¿Qué es?

Es un sistema de control de versiones, este sirve para se controle de una mejor manera todos los cambios y pruebas de un proyecto.

**Repositorio:** Así se le llama al proyecto.

**Repositorio distribuido:** Son las copias del proyecto para cada miembro del equipo.

**Commit:** Es la forma para guardar los cambios que se hicieron al repositorio.

**LA PALABRA HEAD APUNTA AL ULTIMO COMMIT QUE SE HIZO.**

**GIT IGNORE se agrega el archivo con la extension .gitignore , dentro de este archivo se ponen los los archivos o carpetas de los cuales no quieres que git tenga registro. Este archivo gitignore si se agrega a git y se hace su commit.**

**Rama:** Es una linea de tiempo de commits

**Merge (Uniones):**

**Fast Forward:** Cuando no hay ningun cambio en la rama principal, y Git los une como si nunca hubiera habido una separacion.

**Uniones Automaticas:** Git detecta que en la rama principal hubo algun cambio que la otra rama desconoce, y Git los puede unir sin ningun problema.

**Union Manual:** Git no puede unir los cambios que se hicieron, aquí toca hacerlo de manera manual para unir las ramas y a esto se le conoce como **MERGE COMMIT.**

**Tags (Etiquetas):** Es una referencia a un commit especifico.

**Stash:** Toma algunos archivos y los coloca en una area temporal.

Es como una caja fuerte donde puedes guardar el repositorio con los cambios que estabas trabajando, para despues volver a trabajar en ellos

**GITHUB:** Es una plataforma de desarrollo colaborativo de software para alojar proyectos.

**PUSH:** Es la manera para subir el proyecto a una plataforma , en este caso GITHUB.

COMANDOS:

git --version :Para ver la versión de git

git config --global user.name " ": Para asignar un nombre de quien hace los cambios.

git config --global user.email " ": Para asignar un correo de quien hace los cambios.

git config --global -e: Para revisar las personas que estan registradas en config y tambien ver los alias que hemos creado.

git unit: Crea un nuevo repositorio , importante estar en la ruta del proyecto.

git status: Te muestra el estado del repositorio y sus archivos.

git status -s: Muestra el estado del repositorio y sus archivos de manera mas breve.

git add . : Agrega todos los archivos al repositorio.

git add --all: Se agregan todos los archivos que se modificaron.

git add -u: Agrega la actualizacion de los archivos “Ejemplo si se borra o renombra un archivo”

git add pdfs/: Se agregan todos los archivos que esten dentro de una carpeta ejemplo carpeta “pdfs”.

git reset \*.xml : Excluye todos los archivos que fueron agregados con la extension xml.

git commit -m " " : Se crea un commit y se guarda con un nombre.

git checkout -- . : Restaura el repositorio al punto del commit.

git log: Muestra el historial de cada commit que fue creado, con informacion relevante.

git log –oneline: Muestra el historial de manera mas breve.

git log --oneline --decorate --all –graph: Muestra el historial de manera breve y mas decorada.

git diff: Muestra los cambios que se hicieron.

git diff --staged: Muestra los cambios que se hicieron si ya se enviaron al escenario.

git reset HEAD README.md: Remueve el archive que selecciones del stage.

git commit --amend -m "Actualizamos el README": Edita el mensaje del commit en caso de equivocaciones.

git reset --soft HEAD^: Revierte los cambios del commit, y lo deja tambien afuera del escenario.

git reset --mixed 9ac5a77: Remueve los commits que se hicieron pero aun conserva los cambios en los archivos.

git reset --hard 9ac5a77: Elimina por complete el commits y los archivos guardados en ese punto.

git reflog: Muestra todo el registro que se hizo, cada moviento, si se hizo commit , si se restauro, todos los registros.

git mv destruir-mundo.txt salvar-mundo.txt: Modifica el nombre de un archivo.

git rm salvar-mundo.txt: Remueve el archivo seleccionado.

**COMANDO PARA RAMAS**

git branch rama-villanos : Crea una nueva rama y le asigna el nombre que deseas.

git branch: Sirve para visualizar la lista de ramas creadas.

git checkout nombre de la rama: Sirve para posicionarte en la rama que quieres.

git merge rama-villanos: Estando en la rama master se utiliza para unir la rama master con la rama que seleccionas.

git branch -d rama-villanos: Se utiliza para eliminar la rama que seleccionaste una vez que se ha unido a la rama master.

git checkout -b rama-villano: Sirve para crea una rama y al mismo tiempo moverte a esa rama.

**COMANDO PARA TAGS**

git tag superRelease: Sive para crear y asignar una etiqueta.

git tag: Para ver todas las etiquetas que hemos creado.

git tag -d superRelease: Para eliminar una tag.

git tag -a v1.0.0 -m "Version 1.0.0": Para crear un tag , poner el nombre y un mensaje o dellates sobre esa etiqueta.

git tag -a v0.1.0 345d7de -m "Version Alfa": Para crear un tag para un commit en especifico.

git show v0.1.0: Sirve para mostrar los detalles de la etiqueta que selecciones.

**COMANDOS PARA STASH**

git stash: Sirve para guardar el repositorio en un stash.

git stash list: Sirve para mostrar todos los stash que tenemos guardos.

git stash pop: Extrae el ultimo elemento stash y lo eliminara y aplicara los cambios pendientes a master.

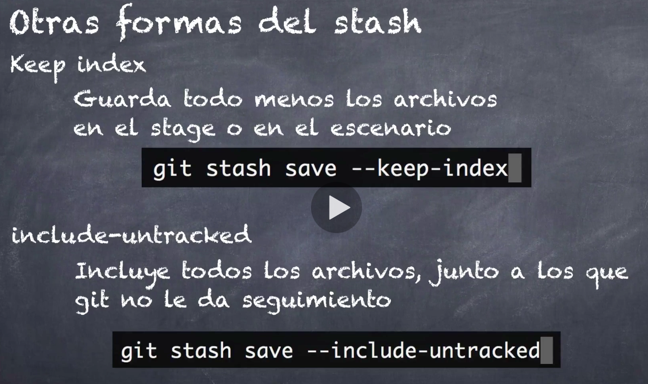
git stash drop: Se utiliza para eliminar solamente el ultimo stash.

git stash apply: Restaura el ultimo registro en el stash.

git show stash: Mostrara los cambios que se hicieron en cada stash.

git stash save “COMENTARIO”: Le agrega informacion a un stash.

git stash clear: Borra todos los stash que hay.



COMANDOS PARA REBASE

git rebase master: Sirve para poner los cambios que se hicieron en master debajo de una rama en proceso, esto facilita hacer una union (merge).

git rebase -i HEAD~4: Toma la ultima cantidad de commits indicados y sirve para utilizar otras funciones que nos ofrece un rebase interactivo como hacer **squash**.

El **squash** sirve para unir 2 commits y hacerlos uno mismo.

El **reword** sirve para modificar el nombre de un commit.

El **edit**: Utilizado para separar commits.

git reset HEAD^: Saca los commits del stage sin borrar los cambios que se hicieron en un archivo.

git rebase --continue: Finaliza el proceso del edit para aplicar todo los cambios hechos en este proceso.

**COMANDO UTILIZADOS PARA GITHUB**

git remote add origin https://github.com/Javier311098/Udemy-heroes.git: Se utiliza para subir el repositorio a una plataforma.

git push -u origin master: Se utiliza para mandar los archivos que tenemos en determinada rama, ejemplo la rama master.

git push: Sube los cambios que se hacen al repositorio.

git remote: Muestra los repositorios remotos que tenemos.

git remote -v:

git push --tags: Sube a Github los tags que ya estan creados en nuestro repositorio.

git pull: Sirve para descargar los archivos y modificaciones del proyecto.

git clone https://github.com/Javier311098/Udemy-heroes.git cualquier nombre: Hace una copia en tu computadora del repositorio que quieres clonar.

git fetch: Actualizar todos los cambios que se hicieron en el repositorio remoto de manera manual.

CREAR ALIAS PARA LOS COMANDOS LARGOS

git config --global alias.lg "log --oneline --decorate --all --graph": lg es el alias de todo el comando que esta en comillas.