alias <aliasName>=”comando que tendrá ese alias”

1. Disco Duro
2. Memoria Ram
3. Repositorio

git init (Iniciar repositorio)

git status(Saber estado)

git config -l (ver configuración de git)

git config - -global <nombre confi(user.name ejemplo)>

git log (Ver Cambios detalladamante) - -stat(muestra cambios en cada archivo)

git log - -all - -graph - -decorate - -oneline(Ver cambios mejor)

git show (Mostrar ultimo cambio)

git config - -global alias.<nameAlias> “código” (alias para codigo de git)

git reflog (tiene el historial completo de lo que hago)

git commit -m “Mensaje” (Subir a repositorio)

Git commit - -amend (resubir commit)

Si no agrego un mensaje: (Guardar y Salir de “Vim”)

* Para escribir el mensaje es Esc + i
* Para salir es Esc + Shift + z + z

(Guardar y Salir de “Nano”)

* Guardar Ctrl + x, preguntara si queremos guardar le damos y
* Para salir después de guardar en Enter

rm (Borra 1)

git rm - - cached (Borra 2,3)

git rm - -force (Borra 1,2,3)

git diff n°Commit1(Version Vieja/Original) + n°Commit2(Version Nueva)(Recomendable) (Compara Cambios)(También puedo ver cambios entre staging y cambios que no están en el staging)

git reset - -hard(Deshace todo, absolutamente todo. Toda la información de los commits y del área de staging se elimina del historial.)

git reset - -soft(Borra el historial y los registros de Git de commits anteriores, pero guarda los cambios en Staging para aplicar las últimas actualizaciones a un nuevo commit.)(Borra 3)

git reset - -mixed (Borra todo, exactamente todo. Toda la información de los commits y del área de staging se elimina del historial.)(Borra 2,3)

git reset - -HEAD (Contrario al add)

git checkout n°Commit(volver al commit indicado guardándolos en un Branch temporal(se borra si se sale))

git swich -c <Nombre nueva rama> (Guardar los cambios hechos del checkout(en la rama temporal) para poder hacerle merge a la rama principal)

## Ramas

git Branch <newName> (Crear Rama)

git Branch -D <rama> (Borrar Rama)

git Branch -m <oldName> <newName>

git branch -r (ver ramas remotas)

git Branch -a (ver todas las ramas)

git checkout <rama> (cambiar entre ramas)

**Merge**

git merge <rama de donde viene>(Unir Ramas), (si hay conflictos es buena la comunicación con el equipo)

git merge –abort (Cancelar Merge)

## Llaves Publicas y privadas

Llave publica: por medio de esta se envia algo al usuario que te dio la llave para que el mismo la decifre con su llave privada que solo funciona para el

Crear Llave

Ssh-keygen -t rsa -b 4096 -C “correo”

eval $(ssh-agent -s)(Verificar que hay agente corriendo)

ssh-add ~/.ssh/id\_rsa.pub (Añadir llave privada a la pc)

Traer repositorio/Repositorios de GitHub

Git remote add <nombre repositorio(origin normalmente)> <url github> (nuevo Origen remoto)

Git remote set-url <nombreRepo> <url nuevo> (Modificar origen remoto ya existente)

git clone <link de git hub>(Clonar repositorio ajeno)

HACER GIT PUSH SOLO CUANDO YA ESTE LISTO PARA ENVIAR PARA NO FORZARLO

git push <remoto> master(subir a repositorio remoto)

git pull <remoto> master(traer al repositorio mio y hacer merge al mismo tiempo)

## Git Tag

Git tag -a <versión ejemplo:v1 o nombre del tag> -m “mensaje” <n°Commit> (Agregar Tag)

Git show-ref - -tags (Visualizar Tags)

Git tag(visualizar tags)

Git push origin - -tags(Subir tags)

Git tag -d <nameTag>(Eliminar Tag de git bash)

Git push origin :refs/tags/<nameTag> (Borrar de git-hub)

Al clonar un repositorio ajeno me tienen que agregar a colaboradores con username preferiblemente

## Pull Requests (Lo hace el DevOps)

Estado Intermedio antes de enviar un merge, es una característica de github(sitio web) no de git

Creando una rama aparte, se hacen cambios, se envían al origin , github indica si se puede hacer un pull request(merge) a la rama main usando “compare and pull request”. No ejecuta el merge de por si

Se puede asignar ese pull a alguien mas, decirle que otro que lo revise, colocarle etiquetas, milenstons que son como objetivos

### Para colaborar a proyectos ajenos (Codigo abierto por ejemplo, nada mas publicos)

Fork al proyecto, git clone, hacer el push, luego hacer un newPullRequest y creamos un pullrequests. El dueño puede aprobar el merge y hacer merge el mismo. El fork se queda atrás y

Se puede actualizar desde github creando un pull requests desde el fork y hacer todo el proceso del merge y esas cosas

O se puede crear otro remoto(git remote add se puede llamar upstream) que se traiga los cambios del proyecto original

## Correccion de Errores

Solo se hace en el local, nunca en el repositorio

Creamos una rama exp

Git Branch exp ------ hacemos los cambios

Primero se hace en la rama que creamos ósea si estamos en “ej: exp”: independientemente si se modifico main

Git rebase main, Corregir los Conflictos usando git rebase - -continue y git add .

Y luego hacerlo en la rama main Git rebase main

## Git Stash(Sirve para probar cosas simples, entre otros)

Guardar Cambios que hice en un lugar temporal antes de hacer un add

Git stash(quitara los cambios y los guardara en el stash)

Git stash list(listar los stash)

Git stash pop (volver a aplicar los cambios)

Git stash drop()

Git stash Branch <nombreRama>(esto, después de hacer git stash, creara un branch con ese cambio) si no lo guardo en el branch que cree, dejara el cambio fuera de staging en la rama main

## Git Clean

Git clean - -dry-clean simulara lo que va a eliminar

Git clean -f borrara lo que simulo

## Git Cherry-pick

Sirve para traer commits viejos de otras ramas a la rama en donde lo necesito

Git Cherry-pick <n°Commit del cambio> esto hacerlo en la rama en donde quieres que se aplique el cambio

## Otros usos de Git

Git grep <palabra> (buscara la palabra en el proyecto)

-n línea exacta

-c cantidad de veces

Para errores de sintaxis usa “”

Git log -S “palabra” (buscara la palabra en el commit)

Git shortlog (diseñado pata ver commits que hace cada miembro del equipo)

-sn (personas que hacen los commits)

- - all (todos los commits)

- -no-merges(no incluyas los merges)

Git blame <archivo> (ver quien hizo cambios en cada línea)

-L <n°linea menor>,<n° línea mayor>

-c mas formato

Git <comando> - -help manual del comando

## GitHub Pages

Crear repositorio nuevo para una pagina(Publico) colocar <nombreUser>.github.io para que use la pagina base