MANUAL DE USUARIO/COMANDOS DE GITHUB

* **Configuración Básica**
* Configurar Nombre que salen en los commits

git config --global user.name "SANTI”

* Configurar correo electrónico

git config --global user.email “[SANTI1@gmail.coM](mailto:SANTI1@gmail.coM)”

* Marco de colores para los comandos

git config --global color.ui true

## Iniciando repositorio

* Iniciamos GIT en la carpeta donde esta el proyecto

git init

* Clonamos el repositorio de github o bitbucket

git clone <url>

* Añadimos todos los archivos para el commit

git add .

* Hacemos el primer commit

git commit -m "Texto que identifique por que se hizo el commit"

* subimos al repositorio

git push origin master

## CLONAR GIT

* Clonamos el repositorio de github o bitbucket

git clone <url>

* Clonamos el repositorio de github o bitbucket ?????

git clone <url> git-demo

## GIT AÑADIR

* Añadimos todos los archivos para el commit

git add .

* Añadimos el archivo para el commit

git add <archivo>

* Añadimos todos los archivos para el commit omitiendo los nuevos

git add --all

* Añadimos todos los archivos con la extensión especificada

git add \*.txt

* Añadimos todos los archivos dentro de un directorio y de una extensión específica

git add docs/\*.txt

* Añadimos todos los archivos dentro de un directorio

git add docs/

## COMPROMISO GIT

* Cargar en el HEAD los cambios realizados

git commit -m "Texto que identifique por que se hizo el commit"

* Agregar y Cargar en el HEAD los cambios realizados

git commit -a -m "Texto que identifique por que se hizo el commit"

* De haber conflictos los muestra

git commit -a

* Agregar al último compromiso, este no se muestra como un nuevo compromiso en los registros. Se puede especificar un nuevo mensaje

git commit --amend -m "Texto que identifique por que se hizo el commit"

## GIT EMPUJAR

* Subimos al repositorio

git push <origien> <branch>

* Subimos una etiqueta

git push –tags

## REGISTRO DE GIT

* Muestra los logs de los commits

git log

* Muestras los cambios en los commits

git log --oneline –stat

* Muestra grafica de los commits

git log --oneline –graph

## DIFERENCIA GIT

* Muestra los cambios realizados a un archivo

git diff

git diff –staged

## CABEZA GIT

* Saca un archivo del compromiso

git reset HEAD <archivo>

* Devuelve el ultimo commit que se hizo y puso los cambios en staging

git reset --soft HEAD^

* Devuelve el ultimo commit y todos los cambios

git reset --hard HEAD^

* Devuelve los 2 ultimo commit y todos los cambios

git reset --hard HEAD^^

* Fusión/compromiso de reversión

git log

git reset --hard <commit\_sha>

## GIT REMOTO

* Agregar repositorio remoto

git remote add origin <url>

* Cambiar de control remoto

git remote set-url origin <url>

* removedor repositorio

git remote rm <name/origin>

* removedor repositorio

git remote rm <name/origin>

* Muestra las sucursales remotas

git remote show origin

* Limpiar todos los branch eliminados

git remote prune origin

## SUCURSAL GIT

* Crea una rama

git branch <nameBranch>

* Lista las sucursales

git branch

* Comando -d elimina el branch y lo une al master

git branch -d <nameBranch>

* Eliminar sin preguntar

git branch -D <nameBranch>

## ETIQUETA GIT

* Muestra una lista de todos los tags

git tag

* Crea una nueva etiqueta

git tag -a <verison> - m "esta es la versión x"

## REBASE DE GIT

* Los rebase se usan cuando trabajamos con branch esto hace que los branch se pongan al día con el master sin afectar al mismo

Une el branch actual con el mastar, esto no se puede ver como un merge

git rebase

* Cuando se produce un conflicto no das las siguientes opciones:

cuando resolvemos los conflictos --continue continua la secuencia del rebase donde se pausa

git rebase --continue

* Omite el conflicto y sigue su camino

git rebase –skip

* Devuelve todo al principio del rebase

git reabse –abort

* Para hacer un rebase a un branch en especifico

git rebase <nameBranch>

## OTROS COMANDOS

* Lista un estado actual del repositorio con lista de archivos modificados o agregados

git status

* Quita del HEAD un archivo y le pone el estado de no trabajado

git checkout -- <file>

* Crea una sucursal en base a uno online

git checkout -b newlocalbranchname origin/branch-name

* Busca los cambios nuevos y actualiza el repositorio

git pull origin <nameBranch>

* Cambiar de sucursal

git checkout <nameBranch/tagname>

* Une el branch actual con el especificado

git merge <nameBranch>

* Verifica cambios en el repositorio online con el local

git fetch

* Borrar un archivo del repositorio

git rm <archivo>

## Tenedor

* Descargar remoto de un tenedor

git remote add upstream <url>

* Merge con master de un fork

git fetch upstream

git merge upstream/master