COMANDOS BASICOS GIT

git init : creará un nuevo .gitsubdirectorio en su directorio de trabajo actual. Esto también creará una nueva rama principal.

git clone: Si un proyecto ya se configuró en un repositorio central, el comando clonar es la forma más común para que los usuarios obtengan un clon de desarrollo local. Como git init, la clonación es generalmente una operación de una sola vez. Una vez que un desarrollador ha obtenido una copia de trabajo, todas las operaciones de control de versiones se gestionan a través de su repositorio local.

git add : grega un cambio en el directorio de trabajo al área de ensayo. Le dice a Git que desea incluir actualizaciones de un archivo en particular en la próxima confirmación.

git commit: guarda los archivos en el repositorio local.

git commit - m "mensaje" : para agregar un mensaje descriptivo.

git status : ve el estado de los archivos, en que version del commit y queda algo para enviar.

git reset head : quita los archivos de staged o los devuelve a su estado anterior.

git rm : elimina el archivo de tu directorio de trabajo y no elimina su historial del sistema de versiones.

git rm --cached : mueve los archivos que indiquemos al estado untracked

git rm --force : elimina los archivos de git y del disco duro.

git branch "nombre_rama" : crea una nueva rama en el repositorio.

git show: muestra los cambios que han existido sobre un archivo.

git checkout "nombre_rama": se usa para migrar de una rama a otra.

git diff : ej: git diff commit1 commit2 / muestra la diferencia entre una version y otra.

git diff --stage : vemos los cambios entre versiones por etapas.

git log: obtenemos el ID de nuestros commits. Muestra el historial de confirmaciones del repo local.

git log -p : muestra el historial de confirmacion incluyendo todos los archivos y sus cambios.

git checkout -b nombre rama : permite crear una rama y cambiarnos a ella con un solo comando.

git checkout IDcommit : permite volve a cualquier version anterior de un archivo. Tambien permite moverse entre ramas.

git reset : tambien permite volver a versiones anteriores pero ademas se borran todos los cambios que hicimos despues de ese commit.

git reset --hard : reestablece el arbol de trabajo y el indice. Cualquier cambio en el arbol desde el commit se descartan (reestable tu commit, zona de staging y tu directorio de trabajo).

git reset --soft : no reinicia el archivo indice o el arbol de trabajo, sino que reinicia el HEAD para commit. Cambios todos los archivos a "cambios a ser commited" (HEAD: version del commit que estoy viendo.)

git branch -d nombre_rama : cuando se termine de trabajar con una rama y se haya fusionado, se puede eliminar usando este comando.

git add remote URL : agrega un repositorio remoto al repositorio local.

git pull origin "nombre_rama" : trae de la rama seleccionada del repositorio remoto los archivos.

git push origin "nombre_rama" : envia los archivos desde el repositorio local al repositorio remoto.

Configuración Básica

Configurar Nombre que salen en los commits

```
git config --global user.name "dasdo"
```

Configurar Email

```
git config --global user.email dasdo1@gmail.com
```

Marco de colores para los comando

```
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
```
GIT CLONE
Clonamos el repositorio de github o bitbucket

```
git clone <url>
```
Clonamos el repositorio de github o bitbucket ?????

```
git clone <url> git-demo
```
GIT ADD
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 especifica

```
git add docs/*.txt
```
Añadimos todos los archivos dentro de un directorios

```
git add docs/
```
GIT COMMIT
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 ultimo commit, este no se muestra como un nuevo commit en los logs. Se puede especificar un nuevo mensaje

```
git commit --amend -m "Texto que identifique por que se hizo el commit"
```
GIT PUSH
Subimos al repositorio

```
git push <origien> <branch>
```

Subimos un tag

```
git push --tags
```

GIT LOG
Muestra los logs de los commits

```
git log
```

Muestras los cambios en los commits

```
git log --oneline --stat
```

Muestra graficos de los commits

```
git log --oneline --graph
```

GIT DIFF
Muestra los cambios realizados a un archivo

```
git diff
git diff --staged
```

GIT HEAD
Saca un archivo del commit

```
git reset HEAD <archivo>
```

Devuelve el ultimo commit que se hizo y pone 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^^
```

Rollback merge/commit

```
git log
git reset --hard <commit_sha>
```

GIT REMOTE
Agregar repositorio remoto

```
git remote add origin <url>
```

Cambiar de remote

```
git remote set-url origin <url>
```

Remover repositorio

```
git remote rm <name/origin>
```

Muestra lista repositorios

```
git remote -v
```

Muestra los branches remotos

```
git remote show origin
```

Limpiar todos los branches eliminados

```
git remote prune origin
```

GIT BRANCH

Crea un branch

```
git branch <nameBranch>
```

Lista los branches

```
git branch
```

Comando -d elimina el branch y lo une al master

```
git branch -d <nameBranch>
```

Elimina sin preguntar

```
git branch -D <nameBranch>
```

GIT TAG

Muestra una lista de todos los tags

```
git tag
```

Crea un nuevo tags

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

GIT REBASE

Los rebase se usan cuando trabajamos con branches esto hace que los branches 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 pauso

```
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 un branch 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 branch

```
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>
```

FORK

Descargar remote de un fork

```
git remote add upstream <url>
```

Merge con master de un fork

```
git fetch upstream
git merge upstream/master
```