COMANDOS GIT

1. 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
```

2. 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
```

3. GIT CLONE

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

```
git clone <url>
```

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

```
git clone <url> git-demo
```

4. 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/
```

5. 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 último 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"
```

6. GIT PUSH

- **Subimos al repositorio**

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

- **Subimos un tag**

```
git push --tags
```

7. 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
```

8. GIT DIFF

- **Muestra los cambios realizados a un archivo**

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

9. 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>
```

10. 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
```

11. 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>
```

12. 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"
```

13. 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>
```

14. 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>
```

15. Fork

- **Descargar remote de un fork**

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

- **Merge con master de un fork**

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