**GIT AND GITHUB**

|  |  |
| --- | --- |
| **CONFIGURACIÓN BÁSICA** | |
| git config --global user.name "nombreUsuario" | Configurar Nombre que salen en los commits |
| git config --global user.email "correoUsuario@gmail.com” | Configurar Email |
| git config --global color.ui true | Marco de colores para los comando |
| **INICIANDO REPOSITORIO** | |
| git init | Iniciamos GIT en la carpeta donde esta el proyecto |
| git clone <url> | Clonamos el repositorio de github o bitbucket |
| git add . | Añadimos todos los archivos para el commit |
| git commit -m "Texto que identifique por que se hizo el commit" | Hacemos el primer commit |
| git push origin master | subimos al repositorio |
| **GIT CLONE** | |
| git clone <url> | Clonamos el repositorio de github o bitbucket |
| git clone <url> git-demo | Clonamos el repositorio de github o bitbucket ? |
| **GIT ADD** | |
| git add . | Añadimos todos los archivos para el commit |
| git add <archivo> | Añadimos el archivo para el commit |
| git add --all | Añadimos todos los archivos para el commit omitiendo los nuevos |
| git add \*.txt | Añadimos todos los archivos con la extensión especificada |
| git add docs/\*.txt | Añadimos todos los archivos dentro de un directorio y de una extensión especifica |
| git add docs/ | Añadimos todos los archivos dentro de un directorio |
| **GIT COMMIT** | |
| git commit -m "Texto que identifique por que se hizo el commit" | Cargar en el HEAD los cambios realizados |
| git commit -a -m "Texto que identifique por que se hizo el commit" | Agregar y Cargar en el HEAD los cambios realizados |
| git commit -a | De haber conflictos los muestra |
| git commit --amend -m "Texto que identifique por que se hizo el commit" | Agregar al ultimo commit, este no se muestra como un nuevo commit en los logs. Se puede especificar un nuevo mensaje |
| **GIT PUSH** | |
| git push <origien> <branch> | Subimos al repositorio |
| git push --tags | Subimos un tag |
| **GIT LOG** | |
| git log | Muestra los logs de los commits |
| git log --oneline --stat | Muestras los cambios en los commits |
| git log --oneline –graph | Muestra graficos de los commits |
| **GIT DIFF** | |
| git diff  git diff –staged | Muestra los cambios realizados a un archivo |
| **GIT HEAD** | |
| git reset HEAD <archivo> | Saca un archivo del commit |
| git reset --soft HEAD^ | Devuelve el ultimo commit que se hizo y pone los cambios en staging |
| git reset --hard HEAD^ | Devuelve el ultimo commit y todos los cambios |
| git reset --hard HEAD^^ | Devuelve los 2 ultimo commit y todos los cambios |
| git log  git reset --hard <commit\_sha> | Rollback merge/commit |
| **GIT REMOTE** | |
| git remote add origin <url> | Agregar repositorio remoto |
| git remote set-url origin <url> | Cambiar de remote |
| git remote rm <name/origin> | Remover repositorio |
| git remote -v | Muestra lista repositorios |
| git remote show origin | Muestra los branches remotos |
| git remote prune origin | Limpiar todos los branches eliminados |
| **GIT BRANCH** | |
| git branch <nameBranch> | Crea un branch |
| git branch | Lista los branches |
| git branch -d <nameBranch> | Comando -d elimina el branch y lo une al master |
| git branch -D <nameBranch> | Elimina sin preguntar |
| **GIT TAG** | |
| git tag | Muestra una lista de todos los tags |
| git tag -a <verison> - m "esta es la versión x" | Crea un nuevo tags |
| **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 | |
| git rebase | Une el branch actual con el mastar, esto no se puede ver como un merge |
| git rebase --continue | 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 --skip | Omite el conflicto y sigue su camino |
| git reabse --abort | Devuelve todo al principio del rebase |
| git rebase <nameBranch> | Para hacer un rebase a un branch en especifico |
| **GIT DIFF** | |
| El comando Diff se usa para vizualisar las diferencias entre objetos (ramas, commits, archivos, entre otros) | |
| git diff | Muestra los archivos que no se encuentran en el repositorio en la rama actual |
| git diff 1234abc 6789def #antiguo nuevo | Muestra las diferencias entre commits |
| git diff --staged | Muestra las diferencias de archivos de la "staged area" |
| git diff rama1 rama2 | Muestra las diferencias entre ramas |
| git diff miarchivo.txt | Muestra las diferencias en un archivo especifico |
| git diff documentation | Muestra las archivos con cambios en un directorio |
| **OTROS COMANDOS** | |
| git status | Lista un estado actual del repositorio con lista de archivos modificados o agregados |
| git checkout -- <file> | Quita del HEAD un archivo y le pone el estado de no trabajado |
| git checkout -b newlocalbranchname origin/branch-name | Crea un branch en base a uno online |
| git pull origin <nameBranch> | Busca los cambios nuevos y actualiza el repositorio |
| git pull origin <nameBranch> --allow-unrelated-histories | Busca los cambios y forza los nuevos y actualiza el repositorio |
| git checkout <nameBranch/tagname> | Cambiar de branch |
| git merge <nameBranch> | Une el branch actual con el especificado |
| git fetch | Verifica cambios en el repositorio online con el local |
| git rm <archivo> | Borrar un archivo del repositorio |
| **FORK** | |
| git remote add upstream <url> | Descargar remote de un fork |
| git fetch upstream  git merge upstream/master | Merge con master de un fork |
| git stash  git pull <remote name> <remote branch name> (or) switch branch  git stash apply --index | Please commit your changes or stash them before you merge. |

**Git init**

**Git add .**

**Git status**

**Git log**

**Git commit -m “mensaje del commit”**

**Git push origin <branch>**

**Git pull origin <branch>**

**Git merge <branch>**

git checkout <nameBranch>

**LARAVEL**

|  |  |
| --- | --- |
| php artisan serve |  |
| php artisan make |  |
| composer install |  |
| cp .env.example .env |  |
| php artisan key:generate |  |
| php artisan migrate --seed |  |
| npm install |  |
| php artisan make:migration create\_users\_table | C |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| php artisan list | Para listar todos los comandos disponibles |
| php artisan serve | Levantar un servidor local |
| php artisan serve –port=8080 | Levantar un servidor local en otro puerto |
| php artisan make:controller UserController | Crear un controlador en app\Http\Controllers\: |
| php artisan make:controller SubFolder/UserController | Crear un controlador en app\Http\Controllers\SubFolder |
| php artisan make:controller UserController –r | Si deseas utilizar una ruta resource es probable que necesite este otro comando, el cual le creará los métodos necesarios para hacer un CRUD (Create, read, update, delete) |
| php artisan make:model User | Crear un modelo |
| php artisan make:migration CreateUserTable | Crear una migración |
| php artisan make:migration CreateUserTable -m | Para crear una migración y su respectivo modelo al mismo tiempo usa –m al final: |
| php artisan migrate | Ejecutar las migraciones |
| php artisan migrate:refresh | Si modificaste alguna migración y quieras volver a ejecutarla |
| php artisan migrate:fresh | Vuele a ejecutar las migraciones desde cero y vacia las tablas |
| php artisan migrate –seed | Ejecuta las migraciones y los seeders |
| php artisan migrate:refresh –seedg  php artisan migrate:fresh --seed |  |
| php artisan make:seeder UserSeeder | Crea una seeder |
| php artisan route:list | Lista todas las rutas disponibles |
| php artisan app:name nombre-namespace | Cambiar el namespace de la aplicación: defautl App |
|  |  |
| php artisan make:middleware NameExample | Crear un middleware |
|  |  |
| php artisan db:seed | Ejecuta los seeder en DatabaseSeeder |
| php artisan migrate:fresh --seed | eliminará todas las tablas y volverá a ejecutar todas sus migraciones. Y Ejecuta los seeder en DatabaseSeeder |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| cp .env.example .env | Crear el archive env. Apartir del archivo env.example |
| php artisan key:generate | Generar una llave para hashing para la app |