git cheat sheet

V1.0.0 @nicoarato

Inicialización

```
$ git help <command>  //detalla un comando
$ git init            //inicia el repo
$ git clone <url>     //clona un repo
$ git remote add origin <link>  //agrega un repo remoto
$ git remote -v       //muestra mi repo remoto
$ git config user.name <user_name> //asigna name al repositorio
$ git config user.mail <user_mail> //asigna mail al repositorio
$ git config --global --edit  //edita configuración global
$ git config --global -e      //edita configuración global
$ git config --global init.defaultBranch <default_name> //asigna un deafult_name al branch principal
$ git config --global alias.lg "git log --oneline --graph --decorate --all" //genera el alias lg
$ git log     //lista los commit
$ git log -p  //lista los commits con detalles
$ git log --merges    //lista sólo los merge commits
$ git log --no-merges  //lista commits sin merge commits
$ git log --reflog     //lista reflogs como si fueran commits
$ git reflog           //lista registro de referencias de todo lo que ha sucedido en orden cronológico
$ git log --pretty=oneline //lista log en una sola línea hash largo
$ git log --oneline        //lista log en una sola línea hash corto
$ git log --oneline --graph --decorate --all  //gráficos con hash corto
$ git log --oneline <branch> -10              //muestra los últimos 10 commits del branch
$ git log --oneline since="6am"               //muestra los commits desde las 6am
```

tip pro:

```
$ git config --global alias.lg "git log --graph --abbrev-commit --decorate
--format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset)
%C(white)%s%C(reset) %C(dim white)- %an%C(reset)%C(bold yellow)%d%C(reset)' --all"
```

Commits y viajes en el tiempo

```
$ git status          //muestra diferencia entre HEAD commit y trabajo actual
$ git status -s       //muestra formato corto
$ git status -u       //muestra archivos sin seguimiento
$ git add .           //agrega todos los archivos al stage
$ git add <file1> ... //agrega el archivo al stage
$ git add folder/     //agrega todos los archivos de un directorio al stage
$ git add folder/*.js  //agrega todos los archivos JS de un directorio al stage
$ git commit -m "Mensaje de commit"  //commit con mensaje
$ git commit --amend -m "Nuevo Mensaje de commit"  //modifica mensaje de último commit
$ git commit --amend       //permite agregar mensaje de último commit y brinda más info
$ git commit -am "Mensaje de commit" //git add + git commit -m sólo a archivos en seguimiento
$ git reset  -- <file>          //permite quitar el archivo del stage
$ git reset  --soft <hash_commit>  //permite volver al commit manteniendo los cambios actuales :)
$ git reset  --hard <hash_commit>  //permite volver al commit eliminando los cambios actuales :(
$ git reset  --soft HEAD^          //permite volver al commit anterior de donde apunta HEAD
$ git reset  --hard                //reconstruye los archivos al estado del último commit
$ git restore --staged <path>      //quita <path> del stage y lo coloca en unstaged
$ git push <remoto> <branch>       //actualiza el repositorio remoto <remoto> en la rama <branch>
```

git cheat sheet

V1.0.0 @nicoarato

Branch

```
$ git branch                    //lista todos las ramas del repo
$ git branch -m <name_old> <name_new>   //renombra una rama
$ git branch -M <name>          //modifica nombre de rama actual
$ git push <remote> --delete <branch>  //elimina rama remota
$ git branch -d <branch> -f     //borrado forzado
$ git checkout -b <branch_name>     //crea una rama a partir de la rama actual y nos mueve a ella
$ git checkout <branch>                 //cambia a rama <branch>
$ git branch -d <branch>   //borra una rama local
$ git checkout <commit_hash>   //nos mueve al commit de la rama actual
$ git checkout -- <file>          //vuelve el archivo a como estaba en el último commit
$ git checkout -- .    //vuelve todos los archivos a como estaban en el último commit
$ git checkout -       //cambia a la rama visitada previamente
```

Stash y diff

```
$ git stash             //guarda los cambios actuales en el stash stack
$ git stash pop       //recupera los cambios guardados en el último stash y los impacta en branch actual
$ git stash clear     //borra todos los stash sin que afecte el repo
$ git stash show stash@{1} //Muestra info corta de ese stash
$ git stash list --stat           //muestra info corta de todos los stash
$ git stash save "Descripción del Work In Progress"  //guarda el stash con una descripción
$ git stash apply stash@{2}   //aplica el stash2 al repo. NO lo borra de la stash stack
$ git stash drop stash@{2}    //lo borra de stash stack
$ git stash drop                    //por defecto borra el stash@{0}
$ git diff                  // muestra los cambios entre WIP y el último commit
$ git diff --staged         // muestra los cambios entre WIP y los archivos en el stage
$ git diff <commit1> <commit2> <file_name> //muestra los  cambios del archivo en 2 commits
$ git diff <commit1> <commit2>     //muestra los  cambios de los archivos en 2 commits
$ git diff --color-words                //muestra los  cambios resaltados
$ git diff <branch1> <branch2> <file_name>  //muestra los cambios del archivo entre 2 ramas
$ git diff <branch1> <branch2>       //muestra los cambios de los archivos entre 2 ramas
$ git diff --output=file.txt <branch1> <branch2> <file_name>  //genera un archivo con los cambios
```

Tags

```
$ git tag         //muestra todos los tags del repo
$ git tag <name>  //crea un tag a partir del último commit
$ git tag -d <name> //elimina el tag
$ git tag -a <name> -m "Versión X.Y.Z - LTS"  //genera un tag con una descripción
$ git tag -a <name> <hash_commit> -m "Versión X.Y.Z - LTS" //genera un tag desde un commit
$ git show <tagname>      //muestra la info relacionada al tag
$ git push --tags         //comprueba con el repo remoto y sube los tags
```

Fetch

```
$ git fetch          //comprueba si hay cambios en el repo remoto y los descarga. Archivos y commits.
$ git branch -r              // permite visualizar las actualizaciones de fetch
$ git checkout <branch>  // permite activar la nueva rama descargada con fetch
$ git fetch -v               // permite el modo verbose cuando se ejecuta el comando
$ git fetch --dry-run        // Muestre lo que se haría, sin hacer ningún cambio.
```

tip: `$ git fetch -v --dry-run`