Git + GITHUB

Ill be back

# Instalaciones Necesarias

<https://gist.github.com/Klerith/90a612344dc2d4e4455ea810fdacbe69>

Instalar extensión Terminal, para que aparezca el icono de la Terminal en la barra

# Primeros comandos de GIT

|  |  |
| --- | --- |
| Comando | Descripción |
| git –version | Muestra la versión de GIT |
| git help | Muestra la ayuda de GIT |
| git help commit | Muestra ayuda sobre el comando |
| git config -- global user.name “usuario” | Configura de forma global el usuario |
| git config -- global user.email “correo” | Establece de forma global el correo |
| git config -- global core.editor “code -- wait” | Establece como editor predeterminado a visual studio code |
| git config -- global -e | Lista todas las configuraciones globales |

## Primer Repositorio

|  |  |
| --- | --- |
| Comando | Descripción |
| git init | Inicializa el repositorio |
| ls -a | Visualiza los archivos ocultos como .git |
| git status | Da información sobre los commits, la rama en la que se esta trabajando |
| git add . | Agrega los archivos al stage |
| Git reset nombre\_archivo | Saca un archivo del Stage |
| Git commit -m “Nombre del Commit” | Hace un commit |
| Git config core.autocrlf true | Estandarizar los saltos de línea en Windows |
| Git checkout -- . | Reconstruye el repositorio a como estaba en el último commit |

## Cambiar Nombre de la Rama Master a Main

|  |  |
| --- | --- |
| Comando | Descripción |
| git branch | Nos dice en que rama estamos |
| git branch -m master main | Cambia el nombre de la rama, en este caso la rama master a main |
| git config –global init.DefaultBranch main | Cambia el nombre de la rama de manera global |
| Git commit -am “Texto” | Agrega al stage y hace commit en un solo paso, pero solo funciona si ya le están dando seguimiento al archivo |
| Git log | Para ver los logs del repositorio |

## Diferentes Formas de añadir al Stage

**Se utilizó el archivo 02-bases**

|  |  |
| --- | --- |
| Comando | Descripción |
| Git add index.html main.html | Agrega archivos al stage por partes |
| Git reset index.html main.html | Saca el archivo del Stage |
| Git add \*.extension | Agrega todos los archivos al stage con la extensión indicada |
| Git add js/\*.js | Agrega archivos al stage que se encuentran dentro de una carpeta |
| Git no le da seguimiento a las carpetas vacías, para que le de seguimiento a las carpetas se debe crear un archivo llamado .gitkeep | |
| Git add css/ | Añade todos los archivos al stage que se encuentran dentro de la carpeta indicada |

## Alias para los Comandos

|  |  |
| --- | --- |
| Comando | Descripción |
| Git status -- short | Permite visualizar un resumen condensado de la información de los archivos |
| Git log --oneline | Muestra de forma resumida los datos de los commits |
| git log --oneline --decorate --all --graph |  |
| Creación de Alias | |
| Git config –global alias.s “status --short” | Crea un alias para los comandos especificados, donde el alias es el que se encuentra después de la palabra alias |
| git config --global alias.lg "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" | Un alias de log para ver de una forma más formal la información de los commits |

## Cambios en los Archivos

|  |  |
| --- | --- |
| Comando | Descripción |
| Git diff | Muestra los cambios en los archivos |
| Git diff –staged | Muestra los cambios en los archivos que están en el stage |

## Actualizar Mensajes de Commits y Revertir Commits

|  |  |
| --- | --- |
| Comando | Descripción |
| Git commit --amend -m “Nuevo nombre del commit” | Este comando permite cambiar el nombre del último commit. Este es el comando recomendado para cambiar los nombres |
| Git reset --soft HEAD^  Git reset --soft HEAD^2 | En este comando el HEAD apunta al último commit, el ^ indica el último commit antes del HEAD. Este comando nos mueve en los commits, si queremos hace cambios en el. El soft no borra los cambios |

## Viajes en el Tiempo, resets y reflog

Se utilizo proyecto Material-Heoes

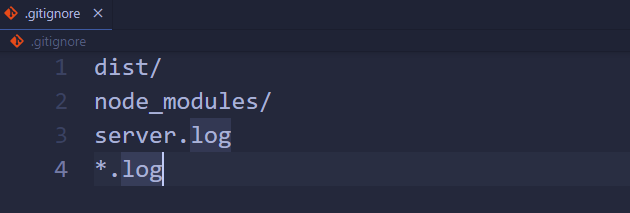
|  |  |
| --- | --- |
| Comando | Descripción |
| git reset --soft 8b35dff | Comando que ayuda a regresar a un punto en el tiempo de los commits, utilizando el Hash |
| Git reset --soft | Genera un reset de HEAD hacia otro commit. No modifica el índice de archivos ni el contenido local. Esto significa que los archivos agregados o eliminados y los cambios producidos durante estos commits reseteados se mantendrán en el contenido local. No modifica los cambios, no los saca del stage |
| Git reset –mixed | Modifca el HEAD y el índice de archivos pero no el contenido local. Cualquier archivo insertado o eliminado dentro de los commits reseteados no se agregará en el índice de commits futuros. Saca los archivos del Stage (Predeterminado) |
| Git reset --hard | Modifca el HEAD, el índice de archivos y el contenido local. El estado del proyecto es el equivalente al que se encontraba en el commit al que fue reseteado. |
| Git reflog | Es un log de las referencias de todo lo que ha sucedido en orden cronológico |

## Cambiar el nombre y eliminar archivos mediante GIT

|  |  |
| --- | --- |
| Comando | Descripción |
| Git mv nombrearchivoanterior nombrearchivonuevo | Mueve los datos de un archivo a otro y lo renombra |
| Git rm nombre\_del\_archivo | Elimina un archivo, pero lo deja en el stage |
| Git reset –hard | Regresa el archivo eliminado |

## Ignorar Archivos que no deseamos

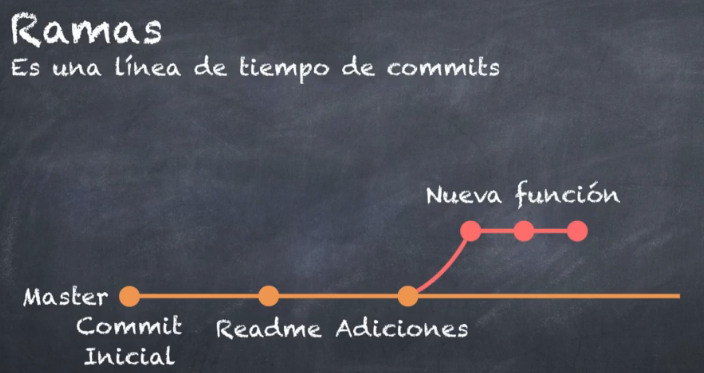
Para que los archivos sean ignorados por git se tiene que crear un archivo .gitignore, usualmente debe estar en la raíz del proyecto



# Ramas Uniones y tags

### Ramas

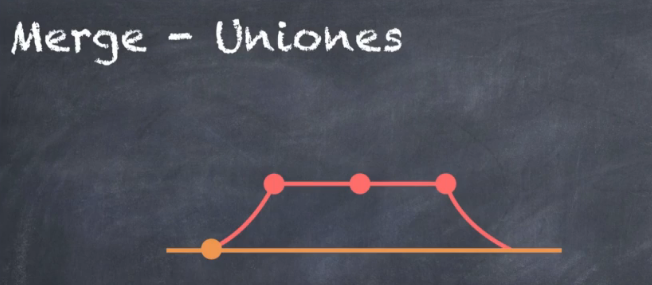
Una rama es una línea del tiempo de commits



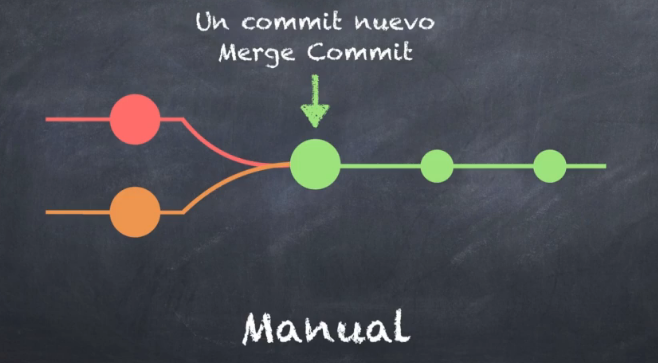
### Merge – Uniones

Cuando intentamos unir una rama con otra git intentara hacerlo por nosotros, escenarios de los merge:

1. Fast-forward – se dispara cuando git detecta que no hay ningún cambio en la rama principal y los cambios pueden ser reintegrados de forma transparente a la rama principal

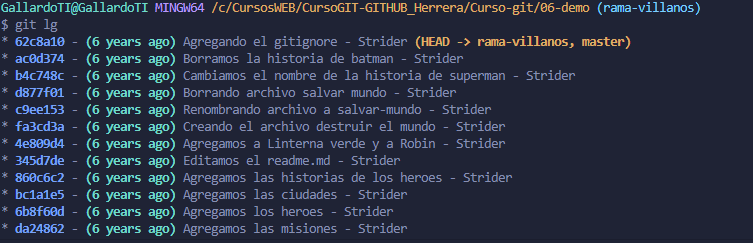


1. Uniones automáticas – Git detecta que en la rama principal hubo algún cambio que la rama secundaria desconoce, sino hay conflictos git une la rama a la principal de forma transparente
2. Uniones Manuales. Git no puede resolver de forma automática

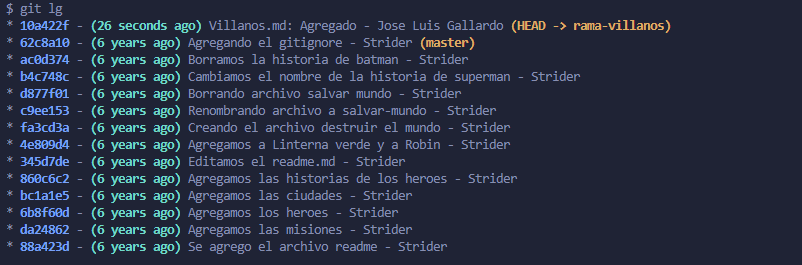


|  |  |
| --- | --- |
| Comando | Descripción |
| Git branch nombre\_rama | Crea una rama |
| Git branch | Visualiza las ramas y en la que estoy actualmente |
| Git checkout ‘Nombre de la rama ‘ | Se mueve a una rama |
| Git merge “Nombre de la rama” | Para realizar el merge se debe estar en la rama que se desea hacer el merge |

Al realizar un git log se muestra que la rama nueva y la rama main se encuentran apuntando al mismo lugar



Al realizar un commit nuevo y a ejecutar el git log se muestra lo siguiente:



## Merge - Fast-forward

|  |  |
| --- | --- |
| Comando | Descripción |
| Git branch nombre\_rama | Crea una rama |
| Git branch | Visualiza las ramas y en la que estoy actualmente |
| Git checkout ‘Nombre de la rama ‘ | Se mueve a una rama |
| Git merge nombre de la rama | Se debe posicionar en la rama que se quiere hacer merge |
| Git branch -d nombre de la rama | Elimina una rama |
| Git branch -d nombre de la rama -f | Borra de manera forzada, aunque no se hayan hecho cambios |
| Git checkout -b nombre de la rama | Este comando crea la rama y cambia a ella en un solo paso |

## Merge - Uniones automáticas

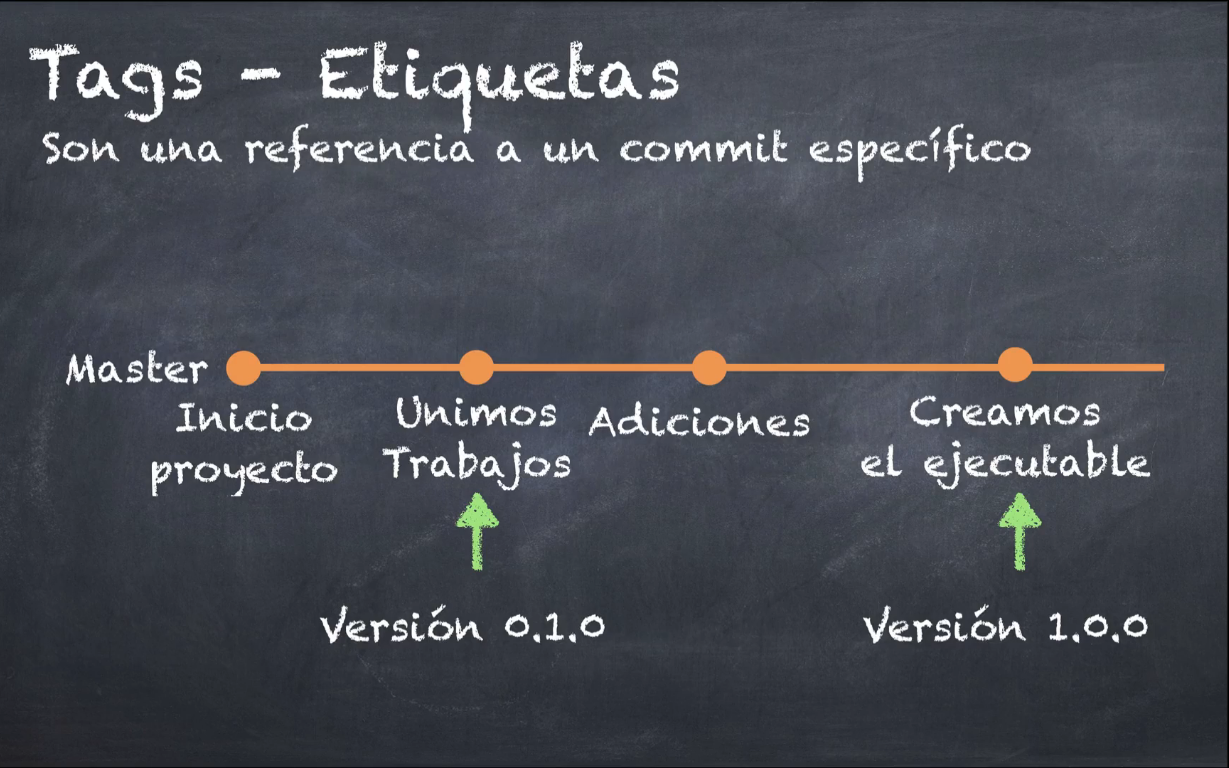
|  |  |
| --- | --- |
| Comando | Descripción |
| Git checkout -b nombre de la rama | Este comando crea la rama y cambia a ella en un solo paso |

## Merge - Uniones con conflictos

|  |  |
| --- | --- |
| Comando | Descripción |
| Git checkout -b nombre de la rama | Este comando crea la rama y cambia a ella en un solo paso |

## Tags – Etiquetas

Es un mensaje o etiqueta que se le pone a un commit



|  |  |
| --- | --- |
| Comando | Descripción |
| git tag nombre-del-tag | Crea una etiqueta o nombre para el commit |
| git tag -d nombre-del-tag | Elimina el Tag |
| git tag -a v1.0.0 -m "Version 1.0.0 lista" | Crea un tag versionado (Versión anotada) |
| git tag -a v0.1.0 7eb793b -m "Versión Alpha de nuestra app" | Le pone un tag por valor hash a cualquier commit |
| git show v0.1.0 | Para visualizar la información de los tags |

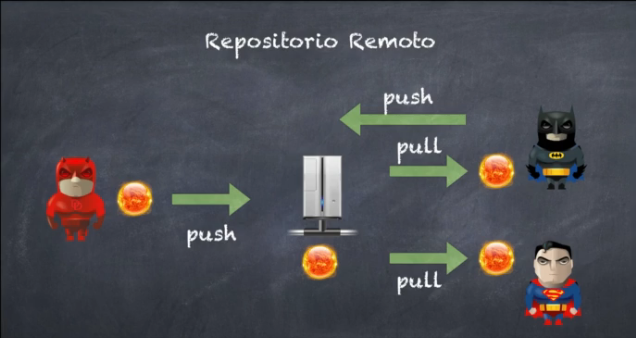
## Git Stash

|  |  |
| --- | --- |
| Comando | Descripción |
| git stash | Almacena en el stash |
| git stash list | Visualiza lo que esta en el stash |
| git stash pop | Toma el último stash y regresa a como estaba el trabajo |
| git stash clear | Borra todos los stash |
| git stash apply stash@{2} | Regresa un stash en particular(utilizando el código del stash) |
| git stash drop stash@{0} | Borra un stash en particular (utilizando el código del stash) |
| git stash drop | Borra el último stash que se ingreso |
| git stash show stash@{1} | Da información general del stash (utilizando el código del stash) |
| git stash save "nombre del stash" | Se hace un stash con nombre |
| git stash list –stat | Muestra información más detallada |

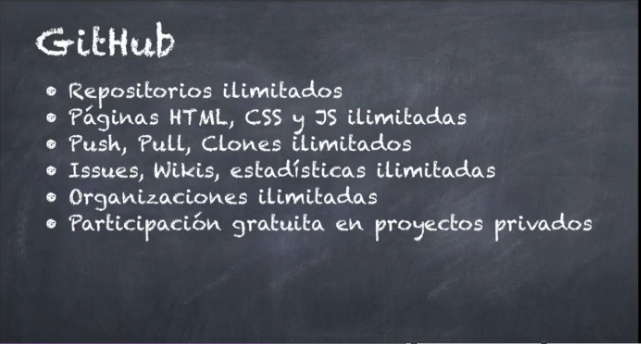
## Git Rebase / Rebase Squash / Rebase Reword/Rebase Edit

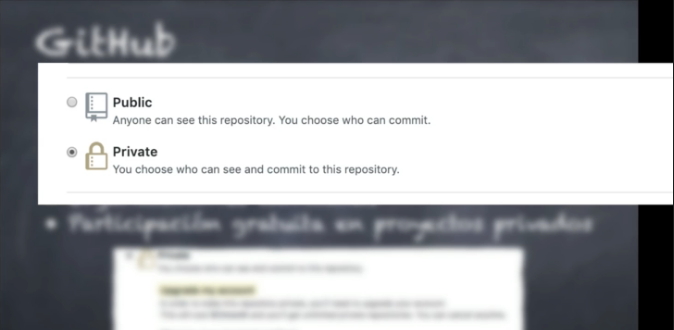
|  |  |
| --- | --- |
| Comando | Descripción |
| git rebase master |  |
| git rebase -i HEAD~4 | Rebase interactivo el HEAD~4 es mostrar los últimos 4 commits y para unirlos se utiliza la palabra squash en el archivo que abre, para cambiar los nombres se pone el comando reword |
| git checkout – nombre del archivo | Regresa solamente el archivo a como estaba en el ultimo commit |

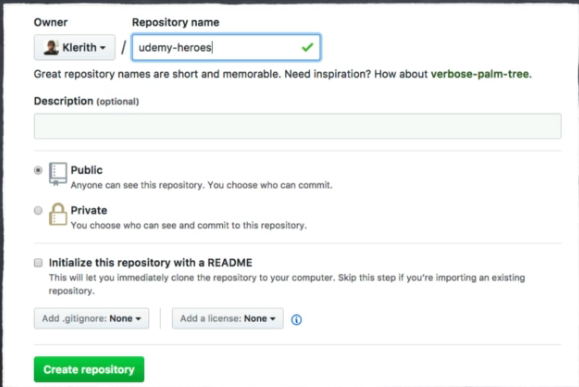
# Github

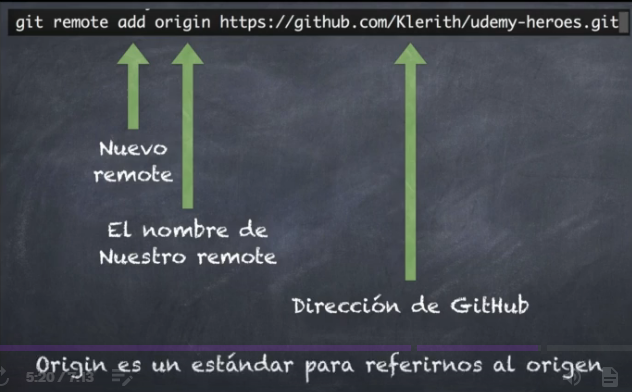












|  |  |
| --- | --- |
| Comando | Descripción |
| git remote add origin https://github.com/gallardohjl2/liga-justicia2.git  git branch -M main  git push -u origin main | Perimite poner el origen de nuestro repositorio remoto |
| Git tag | Permite ver todos los tags que se encuentran en el repositorio local |
| git push –tags | Agrega los tags a github |
| git remote -v | Permite visualizar el path donde se encuentra el repositorio |
| git pull | Obtener los datos del repositorio que esta establecido por defecto |
| git clone <https://github.com/gallardohjl2/liga-justicia2.git> | Realiza un clone del repositorio, esa url se obtiene en git hub en la parte de code y debe ser https |
| git pull –rebase |  |
| git config –global pull.rebase true | Configura el pull de forma global |
| Git rebase --continue | Acepta los cambios realizados por el rebase |
| git config --global pull.rebase true | Configura la forma en que hará el merge el pull, en este caso en forma rebase |