Comandos Git
→ Ayuda
☐ git help
→ Comando específico
☐ git help add
☐ git help commit
☐ git help <cualquier_comando_git></cualquier_comando_git>
→ Establecer el usuario y el e-mail
☐ git configglobal user.name "nombre de usuario"
☐ git configglobal user.email email@email.com
→ Eliminar todos los registros que se refieren al usuario y al e-mail
☐ git configglobalunset user.name "nombre de usuario"
☐ git configglobalunset user.email email@email.com
→ Ver la configuración de Git
☐ git config —list
→ Crear un nuevo repositorio
☐ git init
→ Verificar el estado de los archivos/directorios
☐ git status (muestra el estado de los archivos en su repositorio)
→ Añadir un archivo
☐ git add nombre_archivo_directorio (archivo específico)
☐ git add . / git addall (todos los archivos)
→ Commitear un archivo/directorio
☐ git commit nombre_archivo -m "mensaje del commit"
→ Remover un archivo o directorio
☐ git rm archivo
☐ git rm -r directorio (remueve el directorio y los archivos que contiene)
→ Ver el historial de actividad
☐ git log (muestra el historial)
☐ git log <ruta archivo="" del=""> (muestra el historial de un archivo específico)</ruta>
☐ git logauthor=usuario (muestra el historial de un usuario en particular)
Deshacer operaciones
→ Deshaciendo el cambio local en su directorio de trabajo local
☐ git checkout archivo (solo debe usarse mientras el archivo no se haya añadido todavía al área de trabajo
temporal)
→ Deshaciendo el cambio local en el área de trabajo temporal (staged area)
☐ git reset HEAD archivo (debe usarse cuando el archivo ya ha sido añadido en el área temporal)
"Unstaged changes after reset:M archivo" (si se muestra el siguiente resultado, el comando reset no ha cambiado el
directorio de trabajo)
☐ git checkout nombre_archivo (permite realizar el cambio de directorio)
Repositorio Remoto
→ Ver los repositorios remotos (para saber a dónde se envían los cambios o de dónde los descargamos)
☐ git remote
☐ git remote -v
☐ git remote add origin git@github.com:minombre/archivo-git.git (enlaza el repositorio local con un repositorio
remoto)
🗖 git remote show origin (nermite ver la información de los renositorios remotos)

master.  git branch nuevaBranch_nombre (crea una nueva branch) git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre. git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout master (vuelve a la branch principal-master-) git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch -d nuevaBranch_nombre (apagando una branch) git branch (lista branches) git branch -v (lista branches con información de los últimos commits) git branchmerged (lista branches que ya se han unido (merged) con la master) git branchno-merged (listar branches que no se han unido (merged) con la master) git pull origin nombreeBranch (saca los archivos de una branch existente) git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)	☐ git remote rename origin nombre_nuevo (renombra un repositorio remoto)
git push (los otros pushs no necesitan otras informaciones)  → Actualizar el repositorio local según el repositorio remoto git pull (actualizar los archivos contra la branch actual)  → Clonar un repositorio remoto existente git clone git@github.com:minombre/archivo-git.git  Branches  I master es la branch principal de Git. El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master. git branch nuevaBranch_nombre (crea una nueva branch) git branch nuevaBranch_nombre (crea una nueva branch) git checkout nuevaBranch_nombre (crea una nueva branch) git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout -b nuevaBranch_nombre (rerea una nueva branch y apunta a ella) git theckout -b nuevaBranch_nombre (resu una nueva branch y apunta a ella) git there nuevaBranch_nombre (resulev la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch -d nuevaBranch_nombre (apagando una branch) git branch -v (lista branches con información de los últimos commits) git branch -v (lista branches con información de los últimos commits) git branch -ino-merged (listar branches que no se han unido (merged) con la master) git pullo origin nuevaBranch_nombre (erea una branch existente) git pullo origin nuevaBranch_nombre (crea una branch existente) git pullo origin nuevaBranch_nombre (apagando una branch remota con el mismo nombre) git merge -abort o git reset -merge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) git reset HEAD (cuando queremos volv	☐ git remote rm nombre_git (desvincula un repositorio remoto)
Actualizar el repositorio local según el repositorio remoto  git pull (actualizar los archivos contra la branch actual)  → Clonar un repositorio remoto existente  git clone git@github.com:minombre/archivo-git.git  Branches  El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.  git branch nuevaBranch_nombre (crea una nueva branch)  git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.  git checkout nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout -b nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch -d nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch -v (lista branches con información de los últimos commits)  git branch -v (lista branches con información de los últimos commits)  git branch -no-merged (listar branches que ya se han unido (merged) con la master)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD-2)  → Reescribiendo la historia  git commitamendm "Mi nuevo mensaje" (cambia los mensajes del commit)  Commandos de la terminal  crtH-lo clear: Limpiar la consola  mkdir nombre: Borrar archivo  la m-r nombre: Borrar ar	☐ git push -u origin master (el primer push en el repositorio debe contener su nombre y branch)
git pull (actualizar los archivos contra la branch actual) git fetch (obtener los cambios, pero no aplicarlos a la branch actual) → Clonar un repositorio remoto existente git clone git@github.com:minombre/archivo-git.git  Branches  El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master. git branch nuevaBranch_nombre (crea una nueva branch) git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre. git checkout - bnuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout - bnuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout master (vuelve a la branch principal-master-) git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch duevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch va (lista branches con información de los últimos commits) git branch va (lista branches con información de los últimos commits) git branch va (lista branches con información de los últimos commits) git pull origin nombreeBranch (saca los archivos de una branch existente) git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) git merge -abort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Rescoribiendo la historia git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal crtH o clear: Limpiar la consola mkdir nombre: Borrar archivo la rur rombre: Borrar directorio y todos los archivos que cont	☐ git push (los otros pushs no necesitan otras informaciones)
git fetch (obtener los cambios, pero no aplicarlos a la branch actual)  → Clonar un repositorio remoto existente git clone git@github.com:minombre/archivo-git.git  Branches El master es la branch principal de Git. El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master. git branch nuevaBranch_nombre (crea una nueva branch) git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre. git checkout - b nuevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout unevaBranch_nombre (crea una nueva branch y apunta a ella) git checkout be nuevaBranch_nombre (crea una nueva branch y apunta a ella) git breckout master (vuelve a la branch principal-master-) git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch (lista branches) git branch (lista branches) git branch - nuevaBranch_nombre (apagando una branch) git branch - merged (lista branches que ya se han unido (merged) con la master) git pull origin nombreeBranch (saca los archivos de una branch existente) git pull origin nombreeBranch (saca los archivos de una branch existente) git puls origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) git merge -abort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia git commitamendm 'mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal lo ctil-lo clear: Limpiar la consola lo mkdir nombre_de_carpeta: Crear una carpeta lo ci. : Salir de la carpeta lo ci. : Salir de la carpeta lo ce de la carpeta	→ Actualizar el repositorio local según el repositorio remoto
→ Clonar un repositorio remoto existente  git clone git@github.com:minombre/archivo-git.git  Branches  El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.  git branch nuevaBranch_nombre (crea una nueva branch)  git checkout nuevaBranch_nombre (crea una nueva branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.  git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout master (vuelve a la branch principal-master-)  git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch -d nuevaBranch_nombre (apagando una branch)  git branch (lista branches)  git branch (lista branches con información de los últimos commits)  git branch -v (lista branches con información de los últimos commits)  git branch -no-merged (listar branches que ya se han unido (merged) con la master)  git pull origin nombreeBranch (saca los archivos de una branch existente)  git pull origin nombreeBranch (saca los archivos de una branch existente)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git merge -abort o git reset -merge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD^2)  → Reescribiendo la historia  git commitamendm "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd .: Salir de la carpeta  ls: Ver lo que hay dentro de la carpeta  lr m nombre: Borrar archivo	☐ git pull (actualizar los archivos contra la branch actual)
git clone git@github.com:minombre/archivo-git.git  Branches  El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.  git branch nuevaBranch_nombre (crea una nueva branch)  git checkout nuevaBranch_nombre (crea una nueva branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.  git checkout nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout master (vuelve a la branch principal-master-)  git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch - (lista branche que debe recibir los cambios.  git branch - (lista branches)  git branch (lista branches con información de los últimos commits)  git branch (lista branches con información de los últimos commits)  git pid branch no-merged (lista branches que ya se han unido (merged) con la master)  git pul origin nombreeBranch (saca los archivos de una branch existente)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD^2)  A Reescribiendo la historia  git commitamendm "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd .: Salir de la carpeta  ls: Ver lo que hay dentro de la carpeta  rm nombre: Borrar archivo  mm -r nombre: Borrar archivo	☐ git fetch (obtener los cambios, pero no aplicarlos a la branch actual)
Branches  El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.    git branch nuevaBranch_nombre (crea una nueva branch)   git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.   git checkout - b nuevaBranch_nombre (crea una nueva branch y apunta a ella)   git checkout master (vuelve a la branch principal-master-)   git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.   git branch - d nuevaBranch_nombre (apagando una branch)   git branch (lista branches)   git branch - v( lista branches con información de los últimos commits)   git branch - v( lista branches que ya se han unido (merged) con la master)   git pull origin nombreeBranch (saca los archivos de una branch existente)   git pull origin nombreeBranch (saca los archivos de una branch existente)   git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)   git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)   git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  -> Reescribiendo la historia   git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal   crtl+1 o clear: Limpiar la consola   mkdir nombre_de_carpeta: Crear una carpeta   cd .: Salir de la carpeta   lis: Ver lo que hay dentro de la carpeta   lis: Ver lo que hay dentro de la carpeta   lis: Ver lo que hay dentro de la carpeta	→ Clonar un repositorio remoto existente
El master es la branch principal de Git.  El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.  git branch nuevaBranch_nombre (crea una nueva branch)  git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.  git checkout - b nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout master (vuelve a la branch principal-master-)  git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch - d nuevaBranch_nombre (apagando una branch)  git branch (lista branches)  git branch - v (lista branches con información de los últimos commits)  git branch - v-nerged (listar branches que va se han unido (merged) con la master)  git pull origin nombreeBranch (saca los archivos de una branch existente)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  Resescribiendo la historia  git commitamendm "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd.: Salir de la carpeta  ls: Ver lo que hay dentro de la carpeta  ls: Ver lo que hay dentro de la carpeta  rm nombre: Borrar archivo  mnombre: Borrar archivo	☐ git clone git@github.com:minombre/archivo-git.git
El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.    git branch nuevaBranch_nombre (crea una nueva branch)   git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.   git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)   git checkout master (vuelve a la branch principal-master-)   git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.   git branch -d nuevaBranch_nombre (apagando una branch)   git branch (lista branches)   git branch -v (lista branches con información de los últimos commits)   git branch -v (lista branches con información de los últimos commits)   git branch -merged (listar branches que va se han unido (merged) con la master)   git pull origin nombreeBranch (saca los archivos de una branch existente)   git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)   git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)   git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)   Reescribiendo la historia   git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal   crtH o clear: Limpiar la consola   mkdir nombre_de_carpeta: Crear una carpeta   cd: Entrar en la carpeta   cd: Entrar en la carpeta   cd: Salir de la carpeta   lis: Ver lo que hay dentro de la carpeta   rm nombre: Borrar directorio y todos los archivos que contiene	Branches
El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la master.    git branch nuevaBranch_nombre (crea una nueva branch)   git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.   git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)   git checkout master (vuelve a la branch principal-master-)   git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.   git branch -d nuevaBranch_nombre (apagando una branch)   git branch (lista branches)   git branch -v (lista branches con información de los últimos commits)   git branch -v (lista branches con información de los últimos commits)   git branch -merged (listar branches que va se han unido (merged) con la master)   git pull origin nombreeBranch (saca los archivos de una branch existente)   git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)   git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)   git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)   Reescribiendo la historia   git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal   crtH o clear: Limpiar la consola   mkdir nombre_de_carpeta: Crear una carpeta   cd: Entrar en la carpeta   cd: Entrar en la carpeta   cd: Salir de la carpeta   lis: Ver lo que hay dentro de la carpeta   rm nombre: Borrar directorio y todos los archivos que contiene	El master es la branch principal de Git.
git checkout nuevaBranch_nombre (cambia a una branch existente) - En este caso, el principal puntero HEAD está apuntando a la branch llamada nuevaBranch_nombre.  git checkout - b nuevaBranch_nombre (crea una nueva branch y apunta a ella)  git checkout master (vuelve a la branch principal-master-)  git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch - d nuevaBranch_nombre (apagando una branch)  git branch - v(lista branches)  git branch - v(lista branches con información de los últimos commits)  git branch - v(lista branches que ya se han unido (merged) con la master)  git branch - no-merged (listar branches que ya se han unido (merged) con la master)  git pull origin nombreeBranch (saca los archivos de una branch existente)  git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git merge -abort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  Rescribiendo la historia  git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd: Entrar en la carpeta  cd: Estrar en la carpeta  ls: Ver lo que hay dentro de la carpeta  mnombre: Borrar directorio y todos los archivos que contiene	El HEAD es un puntero especial que indica cuál es la branch actual. Por defecto, HEAD apunta a la branch principal, la
apuntando a la branch llamada nuevaBranch_nombre.  □ git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)  □ git checkout master (vuelve a la branch principal-master-)  □ git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  □ git branch -d nuevaBranch_nombre (apagando una branch)  □ git branch (lista branches)  □ git branch -v (lista branches)  □ git branchwerged (lista branches que ya se han unido (merged) con la master)  □ git branchmerged (listar branches que no se han unido (merged) con la master)  □ git pull origin nombreeBranch (saca los archivos de una branch existente)  □ git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  □ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  □ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  → Reescribiendo la historia  □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  □ crtl+l o clear: Limpiar la consola  □ mkdir nombre_de_carpeta: Crear una carpeta  □ cd .: Salir de la carpeta  □ ls: Ver lo que hay dentro de la carpeta  □ ls: Ver lo que hay dentro de la carpeta	☐ git branch nuevaBranch_nombre (crea una nueva branch)
git checkout master (vuelve a la branch principal-master-) git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios. git branch -d nuevaBranch_nombre (apagando una branch) git branch (lista branches) git branch -v (lista branches con información de los últimos commits) git branchwerged (lista branches que ya se han unido (merged) con la master) git branchno-merged (listar branches que no se han unido (merged) con la master) git pull origin nombreeBranch (saca los archivos de una branch existente) git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal crti+l o clear: Limpiar la consola mkdir nombre_de_carpeta: Crear una carpeta cd: Entrar en la carpeta cd: Entrar en la carpeta ls: Ver lo que hay dentro de la carpeta rm nombre: Borrar archivo	
git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge), debe estar en la branch que debe recibir los cambios.  git branch -d nuevaBranch_nombre (apagando una branch)  git branch (lista branches)  git branch -v (lista branches con información de los últimos commits)  git branchv (lista branches que ya se han unido (merged) con la master)  git branchno-merged (listar branches que no se han unido (merged) con la master)  git pull origin nombreeBranch (saca los archivos de una branch existente)  git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)  git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)  git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  Reescribiendo la historia  git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd: Entrar en la carpeta  ls: Ver lo que hay dentro de la carpeta  rm nombre: Borrar archivo  rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git checkout -b nuevaBranch_nombre (crea una nueva branch y apunta a ella)
debe estar en la branch que debe recibir los cambios.  □ git branch -d nuevaBranch_nombre (apagando una branch) □ git branch (lista branches) □ git branch -v (lista branches con información de los últimos commits) □ git branch -v (lista branches que ya se han unido (merged) con la master) □ git branchno-merged (listar branches que no se han unido (merged) con la master) □ git pull origin nombreeBranch (saca los archivos de una branch existente) □ git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) □ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) □ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal □ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd: Salir de la carpeta □ cd: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git checkout master (vuelve a la branch principal-master-)
<ul> <li>git branch -d nuevaBranch_nombre (apagando una branch)</li> <li>git branch (lista branches)</li> <li>git branch -v (lista branches con información de los últimos commits)</li> <li>git branchv (lista branches que ya se han unido (merged) con la master)</li> <li>git branchno-merged (listar branches que no se han unido (merged) con la master)</li> <li>git pull origin nombreeBranch (saca los archivos de una branch existente)</li> <li>git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)</li> <li>git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)</li> <li>git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)</li> <li>→ Reescribiendo la historia</li> <li>git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)</li> <li>Comandos de la terminal</li> <li>crtl+l o clear: Limpiar la consola</li> <li>mkdir nombre_de_carpeta: Crear una carpeta</li> <li>cd: Entrar en la carpeta</li> <li>cd:: Salir de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>rm nombre: Borrar archivo</li> <li>rm nombre: Borrar directorio y todos los archivos que contiene</li> </ul>	☐ git merge nuevaBranch_nombre (resuelve la unión (merge) entre las branches) - Para realizar la unión (merge),
<ul> <li>git branch (lista branches)</li> <li>git branch -v (lista branches con información de los últimos commits)</li> <li>git branch -v (lista branches que ya se han unido (merged) con la master)</li> <li>git branchno-merged (listar branches que no se han unido (merged) con la master)</li> <li>git pull origin nombreeBranch (saca los archivos de una branch existente)</li> <li>git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)</li> <li>git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)</li> <li>git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)</li> <li>→ Reescribiendo la historia</li> <li>git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)</li> <li>Comandos de la terminal</li> <li>crtl+l o clear: Limpiar la consola</li> <li>mkdir nombre_de_carpeta: Crear una carpeta</li> <li>cd: Entrar en la carpeta</li> <li>cd: Entrar en la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>rm nombre: Borrar archivo</li> <li>rm -r nombre: Borrar directorio y todos los archivos que contiene</li> </ul>	debe estar en la branch que debe recibir los cambios.
<ul> <li>git branch -v (lista branches con información de los últimos commits)</li> <li>git branchmerged (lista branches que ya se han unido (merged) con la master)</li> <li>git branchno-merged (listar branches que no se han unido (merged) con la master)</li> <li>git pull origin nombreeBranch (saca los archivos de una branch existente)</li> <li>git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)</li> <li>git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)</li> <li>git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)</li> <li>→ Reescribiendo la historia</li> <li>git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)</li> <li>Comandos de la terminal</li> <li>crtl+l o clear: Limpiar la consola</li> <li>mkdir nombre_de_carpeta: Crear una carpeta</li> <li>cd :: Salir de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>rm nombre: Borrar archivo</li> <li>rm -r nombre: Borrar directorio y todos los archivos que contiene</li> </ul>	☐ git branch -d nuevaBranch_nombre (apagando una branch)
git branchmerged (lista branches que ya se han unido (merged) con la master) git branchno-merged (listar branches que no se han unido (merged) con la master) git pull origin nombreeBranch (saca los archivos de una branch existente) git pull origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal crtl+l o clear: Limpiar la consola mkdir nombre_de_carpeta: Crear una carpeta cd: Entrar en la carpeta cd: Entrar en la carpeta ls: Ver lo que hay dentro de la carpeta rm nombre: Borrar archivo rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git branch (lista branches)
<ul> <li>git branchno-merged (listar branches que no se han unido (merged) con la master)</li> <li>git pull origin nombreeBranch (saca los archivos de una branch existente)</li> <li>git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)</li> <li>git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)</li> <li>git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)</li> <li>→ Reescribiendo la historia</li> <li>git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)</li> <li>Comandos de la terminal</li> <li>crtl+l o clear: Limpiar la consola</li> <li>mkdir nombre_de_carpeta: Crear una carpeta</li> <li>cd: Entrar en la carpeta</li> <li>dc .:: Salir de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>rm nombre: Borrar archivo</li> <li>rm -r nombre: Borrar directorio y todos los archivos que contiene</li> </ul>	☐ git branch -v (lista branches con información de los últimos commits)
□ git pull origin nombreeBranch (saca los archivos de una branch existente) □ git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) □ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) □ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal □ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd :: Salir de la carpeta □ cd .:: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git branchmerged (lista branches que ya se han unido (merged) con la master)
□ git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre) □ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) □ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal □ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd: Entrar en la carpeta □ cd .: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git branchno-merged (listar branches que no se han unido (merged) con la master)
□ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla) □ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2) → Reescribiendo la historia □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal □ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd: Entrar en la carpeta □ cd .: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git pull origin nombreeBranch (saca los archivos de una branch existente)
□ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)  → Reescribiendo la historia □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal □ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd: Entrar en la carpeta □ cd .: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git push origin nuevaBranch_nombre (crea una branch remota con el mismo nombre)
poner el número de commits después de HEAD. Ejemplo: HEAD~2)  → Reescribiendo la historia  □ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  □ crtl+l o clear: Limpiar la consola  □ mkdir nombre_de_carpeta: Crear una carpeta  □ cd: Entrar en la carpeta  □ cd: Salir de la carpeta  □ ls: Ver lo que hay dentro de la carpeta  □ rm nombre: Borrar archivo  □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git mergeabort o git resetmerge (cuando tenemos problemas con la unión (merge) y queremos deshacerla)
git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)  Comandos de la terminal  crtl+l o clear: Limpiar la consola  mkdir nombre_de_carpeta: Crear una carpeta  cd: Entrar en la carpeta  cd: Salir de la carpeta  ls: Ver lo que hay dentro de la carpeta  rm nombre: Borrar archivo  rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git reset HEAD (cuando queremos volver a un commit anterior, si queremos volver a más de un commit, debemos poner el número de commits después de HEAD. Ejemplo: HEAD~2)
Comandos de la terminal  crtl+l o clear: Limpiar la consola mkdir nombre_de_carpeta: Crear una carpeta cd: Entrar en la carpeta cd: Salir de la carpeta ls: Ver lo que hay dentro de la carpeta rm nombre: Borrar archivo rm -r nombre: Borrar directorio y todos los archivos que contiene	→ Reescribiendo la historia
□ crtl+l o clear: Limpiar la consola □ mkdir nombre_de_carpeta: Crear una carpeta □ cd: Entrar en la carpeta □ cd: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ git commitamend -m "Mi nuevo mensaje" (cambia los mensajes del commit)
<ul> <li>mkdir nombre_de_carpeta: Crear una carpeta</li> <li>cd: Entrar en la carpeta</li> <li>cd: Salir de la carpeta</li> <li>ls: Ver lo que hay dentro de la carpeta</li> <li>rm nombre: Borrar archivo</li> <li>rm -r nombre: Borrar directorio y todos los archivos que contiene</li> </ul>	Comandos de la terminal
□ cd: Entrar en la carpeta □ cd: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ crtl+l o clear: Limpiar la consola
□ cd: Salir de la carpeta □ ls: Ver lo que hay dentro de la carpeta □ rm nombre: Borrar archivo □ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ mkdir nombre_de_carpeta: Crear una carpeta
☐ Is: Ver lo que hay dentro de la carpeta ☐ rm nombre: Borrar archivo ☐ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ cd: Entrar en la carpeta
☐ rm nombre: Borrar archivo ☐ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ cd: Salir de la carpeta
☐ rm -r nombre: Borrar directorio y todos los archivos que contiene	☐ Is: Ver lo que hay dentro de la carpeta
	☐ rm nombre: Borrar archivo
☐ rm -rf nombre: Borrar directorio y todos los archivos que contiene en forma forzada	☐ rm -r nombre: Borrar directorio y todos los archivos que contiene
	☐ rm -rf nombre: Borrar directorio y todos los archivos que contiene en forma forzada