git init Para empezar un proyecto de git

git status -s Para ver el estado de los archivos del proyecto (?? Significa que no se está haciendo seguimiento del archivo, M significa que está modificado, A que está en el área de ensayo)

git add para subir un archivo al área de ensayo.

git add . para subir todos los archivos al área de ensayo.

git commit -m “comentario” para subir el archivo en el repositorio local.

git log --oneline permite conocer todas las modificaciones hechas.

git reset --hard [código del log] permite restaurar a una copia anterior del log.

git commit -am “comentario” con el -am se puede hacer el add y commit en un solo comando.

git commit --amend permite abrir el editor vm para modificar la descripción de un commit.

\*\*VM: al estar dentro del editor comenzamos escribiendo :i para editar los commits. Para salir del editor se coloca :wq

git tag {fecha}{versión} -m “comentario” el tag nos permite darle una versión a nuestro programa.

git push --tags se subirán los tags creadas.

crear rama

- git branch (nombre de la rama"No debe llevar ni paréntesis ni corchetes")

- git log --oneline o git branch (confirma visualmente la creación de la rama)

moverse de rama

- git checkout (nombre de la rama a la que se mueve “No debe llevar ni paréntesis ni corchetes")

- git branch (para confirmar la ubicación de la rama )

APARTIR DE AQUI, SE PUEDEN UTILIZAR LOS COMANDOS PARA AGREGAR VERSIONES

Moverse a master

- git checkout master

Unir ramas

- git merge (nombre de la rama a unir "No debe llevar ni paréntesis ni corchetes")

Eliminar ramas

-git branch -d nombreRama

CODEACADEMY

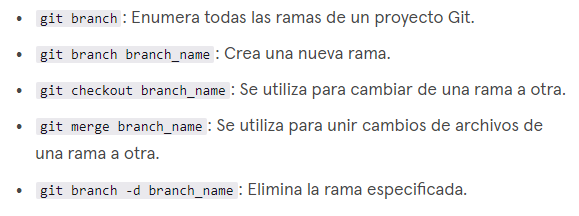
Git es el sistema de control de versiones estándar de la industria para desarrolladores web

Use los comandos de Git para ayudar a realizar un seguimiento de los cambios realizados en un proyecto:

* git init crea un nuevo repositorio de Git
* git status inspecciona el contenido del directorio de trabajo y el área de preparación
* git add agrega archivos del directorio de trabajo al área de preparación
* git diff muestra la diferencia entre el directorio de trabajo y el área de preparación
* git commit almacena permanentemente los cambios de archivos del área de preparación en el repositorio
* git log muestra una lista de todas las confirmaciones anteriores
* Mostrar el último registro de commits:
  + En Git, la confirmación en la que estás actualmente se conoce como confirmación HEAD. La salida del comando git show HEAD mostrará todo lo que el comando git log muestra para la confirmación HEAD, además de todos los cambios en los archivos que fueron confirmados.
* Reset de Git usando SHA:
  + En Git, el comando git reset commit\_SHA puede usarse para establecer HEAD a la confirmación commit\_SHA. El argumento commit\_SHA son los siete primeros dígitos del SHA de una confirmación anterior. En este ejemplo, el HEAD fue restablecido a la confirmación hecha el miércoles 6 de enero. Puedes usar git log para ver un registro de las confirmaciones anteriores y sus valores SHA.
* Puesta en escena de múltiples archivos:
  + En Git, el comando git add nombre\_de\_archivo\_1 nombre\_de\_archivo\_2 se utiliza para añadir múltiples archivos al área de preparación a la vez. Puedes utilizar git status para comprobar si has añadido correctamente tus archivos al área de preparación.
* Eliminar un archivo del área de preparación:
  + En Git, el comando git reset HEAD filename eliminará el archivo del área de staging. Ten en cuenta que este comando no descarta los cambios en los archivos del directorio de trabajo. Puedes usar este comando si has añadido un archivo al área de preparación, pero el archivo incluye ediciones incorrectas. Puedes usar el comando git status para asegurarte de que tu archivo fue eliminado correctamente del área de preparación.
* Retroceder al último commit:
  + En Git, el comando git checkout HEAD nombredearchivo retrocede todos los cambios que se han hecho en nombredearchivo desde la última confirmación. En otras palabras, este comando cambiará tu directorio de trabajo para que se vea exactamente igual que cuando hiciste la última confirmación. Puedes usar el comando git diff para ver si el retroceso fue exitoso. Si git diff no muestra nada, significa que tu directorio de trabajo coincide con tu última confirmación.
  + Git checkout – filename hace lo mismo

Ramificación de Git:

* Master Branch:
  + En Git, el proyecto principal se completa en la rama maestra. Al hacer tu primer commit en un nuevo repositorio git, se creará automáticamente una rama maestra. Crea nuevas ramas a partir de la rama maestra para desarrollar nuevas características para un proyecto. Estas ramas pueden fusionarse con la maestra más adelante para incorporar las nuevas características. Puedes usar git branch para comprobar en qué rama estás.
* Creación de una nueva sucursal:
  + En Git, el comando git branch branch-name se utiliza para crear una nueva rama llamada branch-name. Las ramas deben tener un nombre que describa el propósito de la rama. Ten en cuenta que los nombres de las ramas no pueden contener espacios en blanco: new-feature y new\_feature son nombres de rama válidos, pero new feature no lo es.
* Ver la rama actual:
  + En Git, el comando git branch mostrará todas las ramas. La rama actual mostrará \* antes de su nombre.
* Merge Conflicts:
  + En Git, se produce un conflicto de fusión cuando se modifica el mismo archivo en la rama actual y en la rama que se está fusionando. Aparecerá un error mostrando CONFLICTO (contenido): Conflicto de fusión en [nombre de archivo]. Git editará automáticamente el archivo con el conflicto para mostrar dónde está el conflicto. El texto de la rama actual estará entre <<<<<<< HEAD y =======. El texto de la rama que se está fusionando en la rama actual estará entre ======= y >>>>>>> branch-name Para resolver un conflicto de fusión, edite el archivo con el conflicto, decida qué partes de las ediciones de cada rama deben mantenerse, y luego añada y confirme el archivo.
* Deleting a Branch:
  + En Git, el comando git branch -d nombre\_rama se utiliza para eliminar la rama nombre\_rama. Es una buena práctica eliminar una rama después de que se haya fusionado con la rama maestra.
* Merging Branches:
  + En Git, el comando git merge nombre-de-rama añadirá los cambios de nombre-de-rama a la rama actual. Utiliza este comando cuando hayas terminado de construir una característica en una rama separada y quieras traer esos cambios a tu rama actual.



Git Teamwork:

* Flujo de trabajo de colaboración Git:
  + Un flujo de trabajo de colaboración Git común es: 1. Recoger y fusionar los cambios del remoto. 2. Crear una rama para trabajar en una nueva característica del proyecto. 3. Desarrollar la función en una rama y confirmar el trabajo. 4. Recoger y fusionar de nuevo desde el remoto (en caso de que se hagan nuevos commits). 5. Empujar la rama a la remota para su revisión. Los pasos 1 y 4 son una salvaguarda contra los conflictos de fusión, que ocurren cuando dos ramas contienen cambios en los archivos que no pueden ser fusionados con el comando git merge.
* Lista de los Git remotos:
  + En Git, el comando git-remote -v devuelve una lista de repositorios remotos a los que está vinculado el proyecto actual.
    - Git lista el nombre del repositorio remoto, así como sus ubicaciones.
    - Git nombra automáticamente este origen remoto, porque se refiere al repositorio remoto de origen. Sin embargo, es posible cambiar su nombre de forma segura.
    - El remoto se lista dos veces: una para (fetch) y otra para (push).
* Empujar los cambios de rama:
  + En Git, el comando git push origin branch-name empuja la rama, y todos los cambios comprometidos, a la remota. Esta rama puede ahora ser revisada por los colaboradores. En el ejemplo, la rama actual que contiene los cambios confirmados se llama bio-questions.
* Clonación de un repositorio remoto:
  + En Git, el comando git clone remote\_location clone\_name crea una copia local de un repositorio remoto. remote\_location indica a Git dónde encontrar el repositorio remoto y puede ser una ruta de archivo o una dirección web. clone\_name es el nombre del directorio donde se copiará el contenido del repositorio remoto. En el ejemplo, my-quizzes es un nuevo directorio creado como copia local del proyecto Git science-quizzes. Los cambios que se realicen en my-quizzes no afectarán a science-quizzes.
* Obtención de cambios de origen remoto (fetch):
  + En Git, el comando git fetch descarga objetos del repositorio remoto de origen. Los cambios, sin embargo, no se fusionan en la rama actual de nombre-rama. En su lugar, se almacenan en la rama origen/nombre de la rama, a la espera de ser fusionados. En el ejemplo proporcionado, utilizando el comando git branch -a para ver las ramas existentes, podemos ver que los datos obtenidos se han almacenado en una nueva rama origin/master.
* Git Remote
  + Un control remoto es un repositorio compartido de Git que permite que varios colaboradores trabajen en el mismo proyecto de Git desde diferentes ubicaciones. Los colaboradores trabajan en el proyecto de forma independiente y combinan los cambios cuando están listos para hacerlo.
* En Git, el git merge origin/branch-namecomando fusionará los cambios obtenidos, almacenados en origin/branch-namela branch-namerama actual. En el ejemplo, masteres el nombre de la rama que se fusiona.