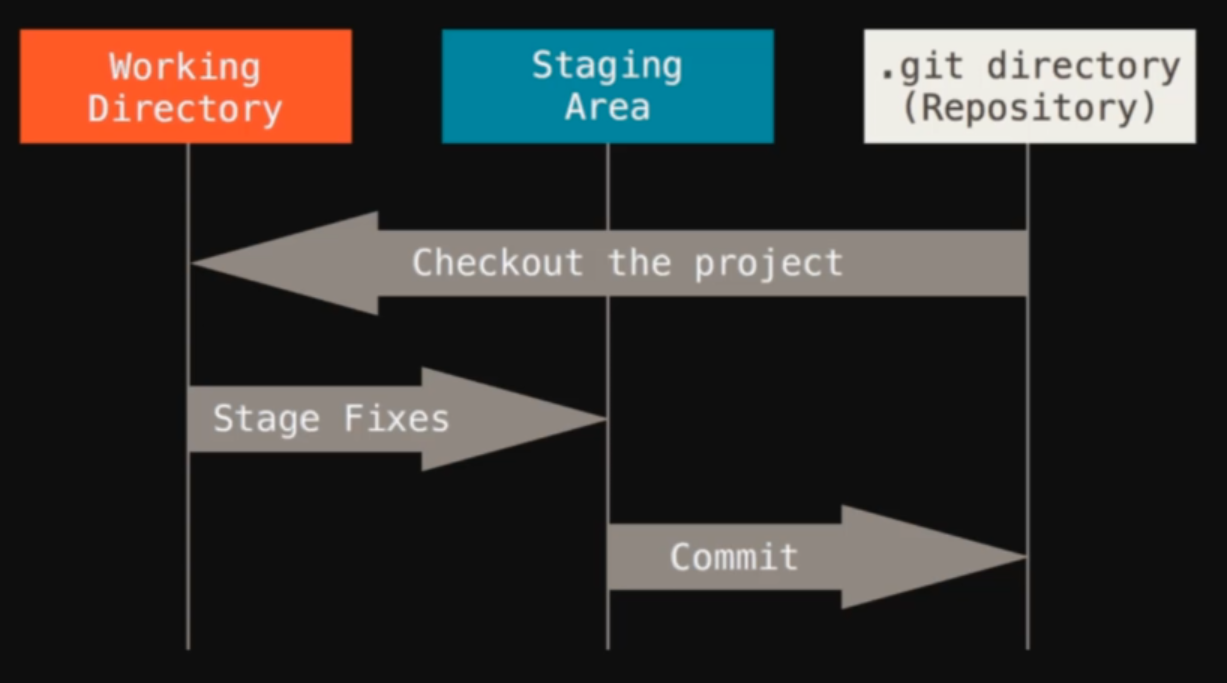
**Configurar nombre de usuario y mail que se verá cuando hagamos un commit**

git config -global user.name "NombreUsuario"

git config -global user.email "mail"

**Listar configuraciones**

git config –list



En esta elección hablaremos de distintos estados importantes que tiene github para administrar las versiones. Específicamente tiene tres estados principales en los que se pueden encontrar tus archivos: **modificado, preparado y confirmado**.

**Modificado** es el área de **working directory** que es el directorio de trabajo.

**Preparado** es el área de preparación o Staging Area

**Confirmado** o **Commit** es el área del repositorio local que básicamente es el directorio de git.

Para pasar un archivo del área de modificado al área de preparado debemos usar el comando **“git add”.** Luego para pasar el archivo desde el Preparado a confirmado debemos usar un comando que se llama **“git commit”.**

**Inicializar repositorio**

**git init**

git status

pasar archivos a staging área

git add .

Hacer un commit

git commit –m “mi primer commit”

sacar archivo del staging area

git reset HEAD “nombre archivo”

Encontrar diferencias del código actual con respecto al ultimo commit

git diff

volver un archivo a la versión anterior el ultimo commit guardado

git chekout ” nombre de archivo”

Comandos luego de crear repositorio

Clonar repo

git clone <https://link-con-nombre-del-repositorio>

crear rama

git branch <nombre-de-la-rama>

Ver ramas

Git branch

Git branch –list

**Borrar una rama:**

git branch -d <nombre-de-la-rama>

cambiarte a una rama

git checkout <nombre-de-la-rama>

* Los cambios en tu rama actual tienen que ser confirmados o almacenados en el guardado rápido (stash) antes de que cambies de rama.
* La rama a la que te quieras cambiar debe existir en local.
* **Hay también un comando de acceso directo que te permite crear y cambiarte a esa rama al mismo tiempo:**
* git checkout -b <nombre-de-tu-rama>

**Git status**

El comando de git status nos da toda la información necesaria sobre la rama actual.

git status

Podemos encontrar información como:

* Si la rama actual está actualizada
* Si hay algo para confirmar, enviar o recibir (pull).
* Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)
* Si hay archivos creados, modificados o eliminados

**Añadir un único archivo:**

* git add <archivo>

**Añadir todo de una vez:**

* git add –A

Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

* git commit -m "mensaje de confirmación"

7. Git push

Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.

git push <nombre-remoto> <nombre-de-tu-rama>

**De todas formas, si tu rama ha sido creada recientemente, puede que tengas que cargar y subir tu rama con el siguiente comando:**

**git push --set-upstream <nombre-remoto> <nombre-de-tu-rama>**

**or**

git push -u origin <nombre-de-tu-rama>

**Importante: Git push solamente carga los cambios que han sido confirmados.**

**8. Git pull**

El comando **git pull**se utiliza para recibir actualizaciones del repositorio remoto. Este comando es una combinación del **git fetch** y del **git merge** lo cual significa que cundo usemos el git pull recogeremos actualizaciones del repositorio remoto (git fetch) e inmediatamente aplicamos estos últimos cambios en local (git merge).

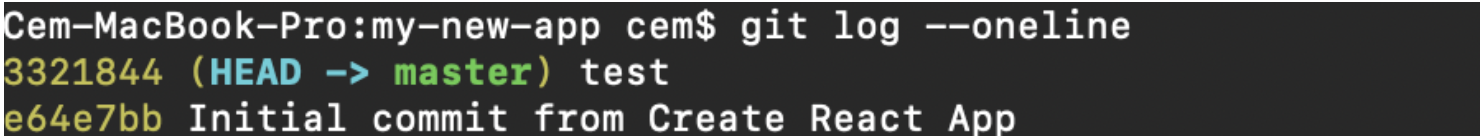
git pull <nombre-remoto>

**Esta operación puede generar conflictos que tengamos que resolver manualmente.**

**9. Git revert**

A veces, necesitaremos deshacer los cambios que hemos hecho. Hay varias maneras para deshacer nuestros cambios en local y/o en remoto (dependiendo de lo que necesitemos), pero necesitaremos utilizar cuidadosamente estos comandos para evitar borrados no deseados.

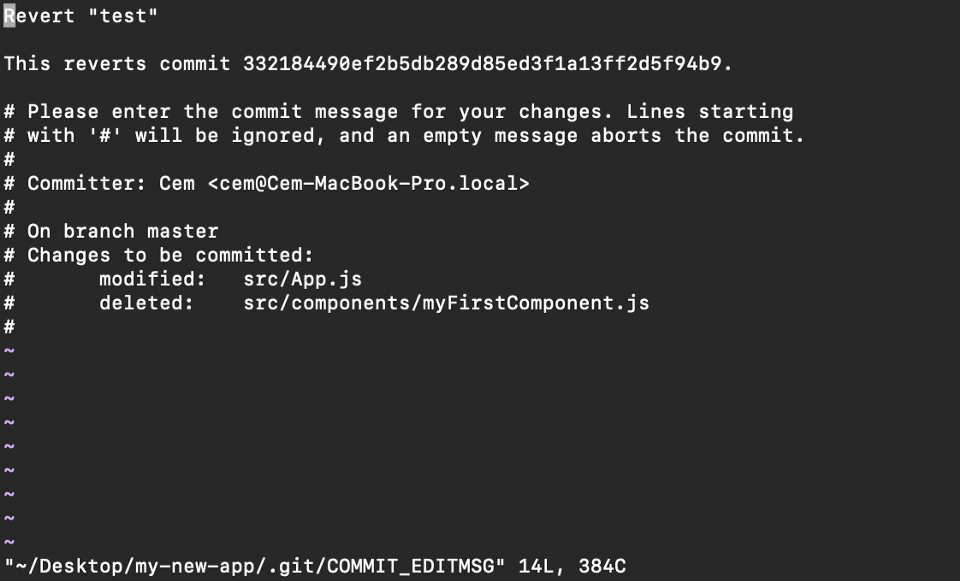
Una manera segura para deshacer nuestras commits es utilizar **git revert**. Para ver nuestro historial de commits, primero necesitamos utilizar el  **git log -- oneline:**

histórico de git en mi rama master

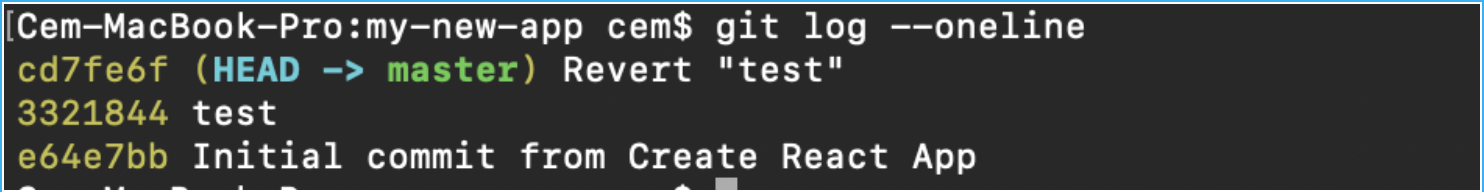
Entonces, solo necesitamos especificar el código de comprobación que encontrarás junto al commit que queremos deshacer:

git revert 3321844

Después de esto, verás una pantalla como la de abajo -tan solo presiona **shift + q** para salir:



El comando git revert deshará el commit que le hemos indicado, pero creará un nuevo commit deshaciendo la anterior:

commit generado con el git revert

La ventaja de utilizar **git revert** es que no afecta al commit histórico. Esto significa que puedes seguir viendo todos los commits en tu histórico, incluso los revertidos.

Otra medida de seguridad es que todo sucede en local a no ser que los enviemos al repositorio remoto. Por esto es que git revert es más seguro de usar y es la manera preferida para deshacer los commits.

**10. Git merge**

Cuando ya hayas completado el desarrollo de tu proyecto en tu rama y todo funcione correctamente, el último paso es fusionar la rama con su rama padre (dev o master). Esto se hace con el comando git merge.

Git merge básicamente integra las características de tu rama con todos los commits realizados a las ramas dev (o master).  Es importante que recuerdes que tienes que estar en esa rama específica que quieres fusionar  con tu rama de características.

Por ejemplo, cuando quieres fusionar tu rama de características en la rama dev:

**Primero, debes cambiarte a la rama dev:**

git checkout dev

**Antes de fusionar, debes actualizar tu rama dev local:**

git fetch

**Por último, puedes fusionar tu rama de características en la rama dev:**

git merge <nombre-de-la-rama>

**Pista: Asegúrate de que tu rama dev tiene la última versión antes de fusionar otras ramas, si no, te enfrentarás a conflictos u otros problemas no deseados.**