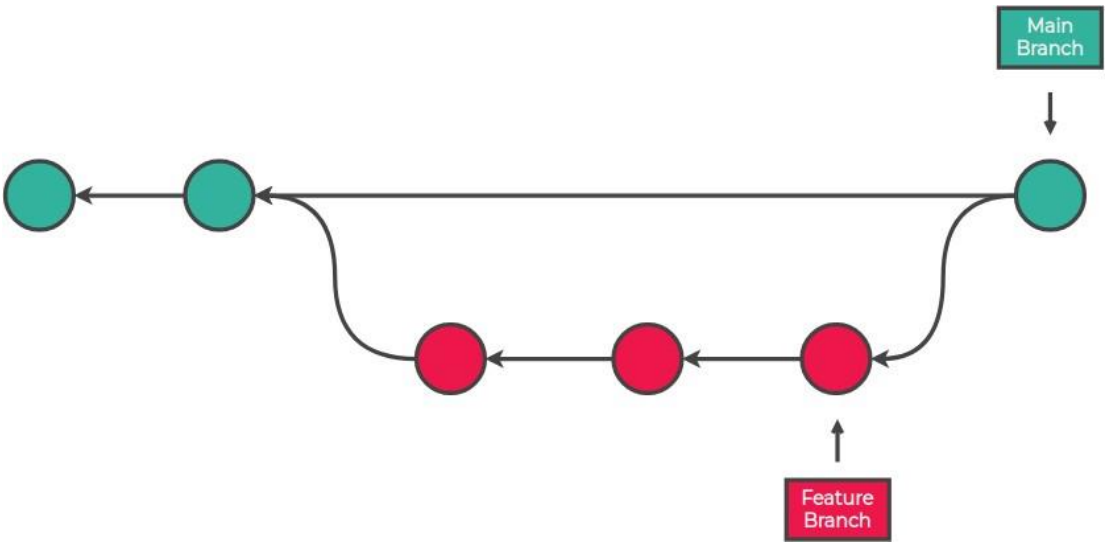
Ramas de Git

¿Qué es y cómo se utiliza una branch?

Una **rama o branch de Git** nos permitirá trabajar en versiones de nuestros archivos de manera mucho más ordenada. Hasta ahora vimos que el repositorio que creamos con Git o clonamos de Github tiene por defecto una rama raíz llamada `Main` — o `Master` para repositorios *legacy* — en la cual vamos guardando las versiones de nuestro trabajo cada vez que hacemos un commit.

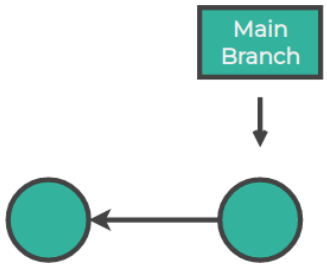
En los proyectos que realizaremos como desarrolladores, todos los cambios que hagamos en un repositorio deben ser socializados con un equipo. Ese equipo será el encargado de aprobar nuestras modificaciones. Entonces, cuando trabajemos en nuevas funcionalidades, debamos arreglar errores o simplemente queramos ordenar de una mejor manera nuestros archivos, deberíamos hacerlo sobre un espacio separado, de manera que nuestros cambios puedan ser integrados escalonadamente con el resto del trabajo del equipo. Para ello se utilizan las **ramas** o **branches.**

Una **branch** funcionará como un espacio de trabajo solo para nosotros, para que acumulemos nuestros cambios y hagamos todas las pruebas necesarias hasta que estemos seguros de socializar esos cambios con el resto del equipo.

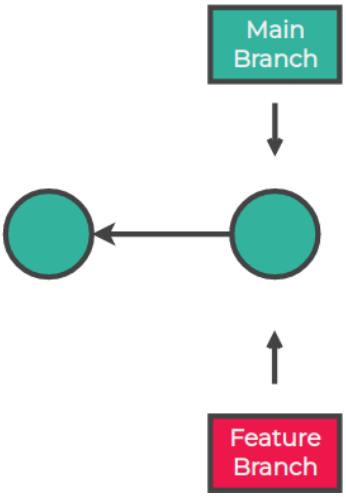

Main
Branch
Feature
Branch

Creación de ramas

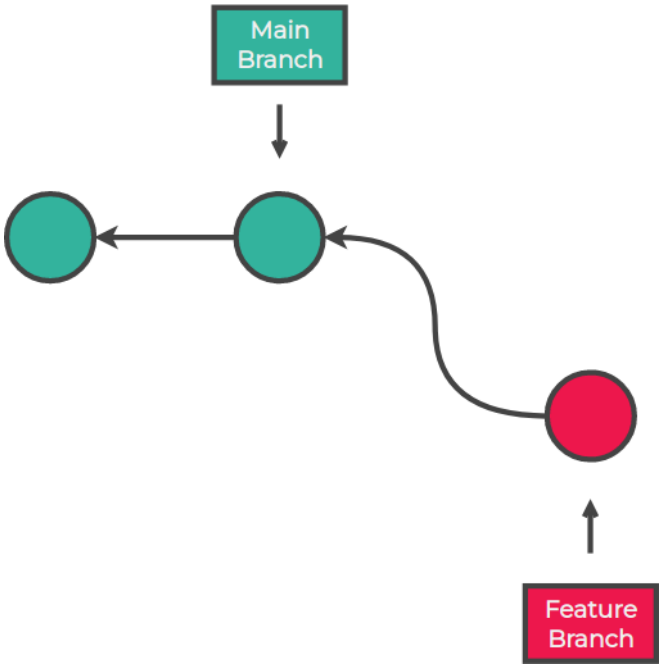
Es importante entender que una rama es solo una línea de desarrollo que tiene como base el repositorio, pero que no actúa directamente sobre él, o sea que no lo modifica. Si comenzamos con un repositorio que tiene este aspecto:



Al crear una rama, comenzamos desde el último commit de la rama principal:



Al generar commits en la nueva rama, los nuevos commits no afectan la rama principal, peroparten de ella:



Al utilizar ramas no solo es posible trabajar en un repositorio de forma paralela, sino que se **evita** que subamos **archivos no intencionados** o **código dudoso** a la **rama principal**.

Git branch

Para trabajar con ramas utilizamos el comando **git branch**. Git branch nos permite crear, enumerar, cambiar el nombre y eliminar ramas. No permite cambiar entre ramas o volver a unir un historial bifurcado.

- **`git branch`**

  Enumera todas las ramas de tu repositorio, es similar a git branch --list.

- **`git branch <nombre_rama>`**

  Crea una nueva rama llamada <branch>.

- **`git branch -d <nombre_rama>`**

  Elimina la rama llamada <branch>. Git evita que eliminemos la rama si tiene cambios queaún no se han fusionado con la rama Main.

- **`git branch -D <nombre_rama>`**

  Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.

Git checkout

Para moverse de una rama a otra, se ejecuta el comando

- **`git checkout <nombre_rama>`**

También, podremos crear la rama y movernos en un mismo comando:

- `git checkout -b <nombre_rama>`

Generalmente, Git solo permitirá que nos movamos a otra rama si no tenemos cambios. Si tenemos modificaciones, para cambiarnos de rama, debemos:

1. Eliminarlos (deshaciendo los cambios).
2. Confirmarlos (haciendo un git commit).

Guardar cambios y subirlos al repositorio remoto

Una vez que terminamos de realizar los cambios que queremos en nuestra branch, ejecutamoslos mismos comandos que vimos hasta ahora: git add, git commit, git status y git log. Pero cuando queramos subir esos cambios, la primera vez, deberemos configurar a qué rama remota queremos vincular esta rama local. Para eso utilizaremos el comando git push de la siguiente manera:

- `git push -u origin <nombre_rama>`

Subsecuentes push se podrán hacer con el comando visto:

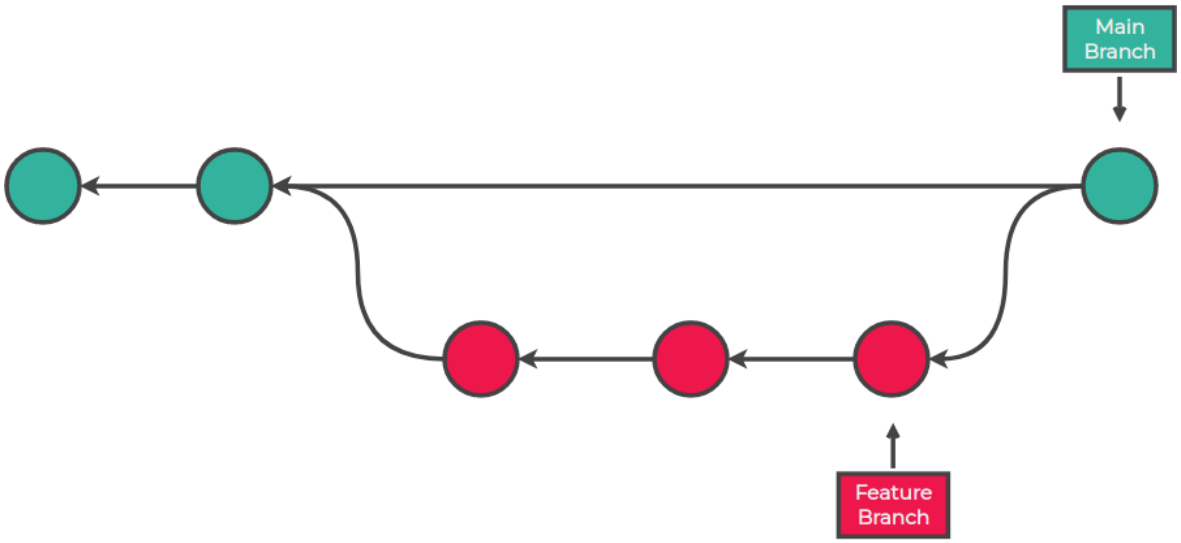
- `git push`

Así también, para traer los cambios de esa rama utilizamos el git pull:

- `git pull`

Git Merge

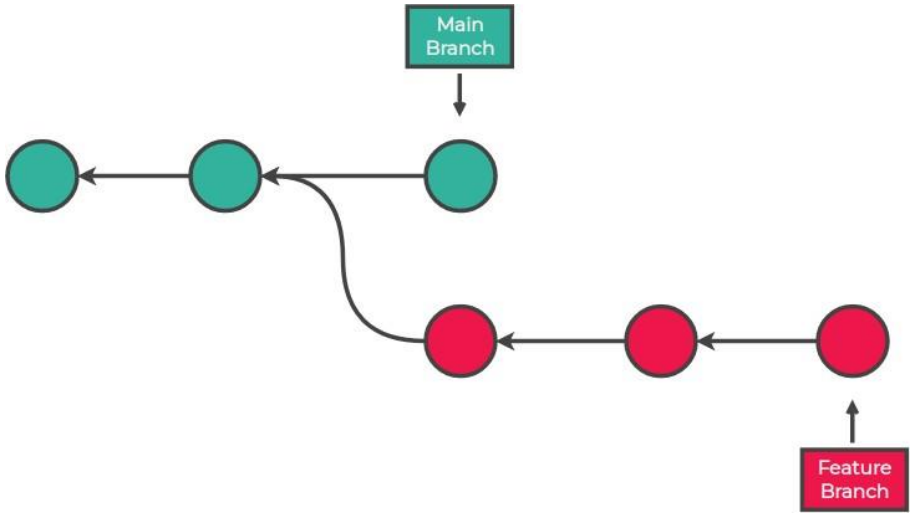
Hasta ahora vimos que podíamos crear ramas, pero el objetivo final es que las mismas eventualmente vuelvan a unificarse con `main` o incluso entre ellas. Por ejemplo:



Para esto existe el comando `git merge`.

Creando un merge

Supongamos que estamos en la rama `feature_branch` y nuestro repo se ve así:

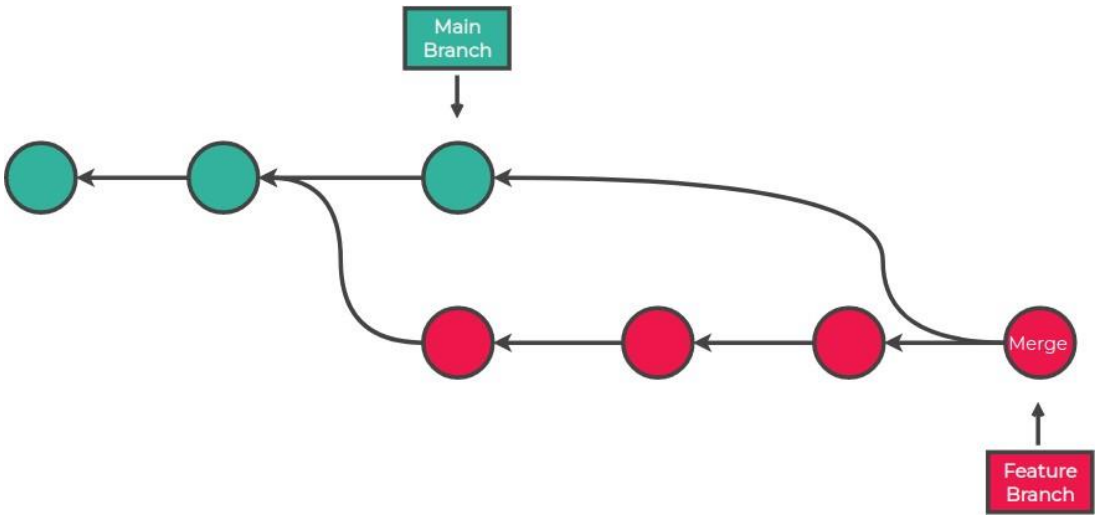


En este caso no podemos simplemente mergear **`feature_branch`** en **`main`** porque hay un commit en **`main`** que no tenemos en **feature_branch**. Esto puede ocasionar conflictos, los cuales no pueden estar en nuestra **rama principal**.

Entonces, antes de eso, debemos actualizar la rama en la que estamos. El flujo de trabajo recomendado es siempre hacer lo siguiente:

- **`git checkout feature_branch`** - Nos aseguramos de estar en la rama que queremos mergear
- **`git pull`** - Traemos todos los cambios del remoto.
- **`git merge origin/main`** - (Mergeamos: main → feature_branch)

Luego de esto, nuestro repo queda así:



DH Tips

Este proceso es **recomendable hacerlo dándole prioridad y de forma habitual** - por ejemplo, antes de empezar a trabajar cada día- , para traernos cambios de `main` y para que nuestra rama no se vaya desactualizando con respecto a los cambios que haya socializado el resto del equipo. Una cosa interesante de esto es que **nosotros** elegimos cuándo hacer parte de nuestra rama los cambios de los demás, sin que sea una sorpresa y un bloqueo para pushear.

Luego de esto, debemos comprobar que no se hayan incorporado errores del merge, es decir: resolver los posibles conflictos y probar el proyecto completo para corroborar que no se hayan agregado bugs.

Una vez que estamos seguros que la rama se puede socializar con el resto del equipo, podemos mergear esta rama con `main`

Para esta última tarea, tenemos los **pull requests** en Github.

Pull Requests

Para evitar cambiar el código de la rama principal “sin avisar”, y tener un proceso estructuradode review con nuestros pares, se crearon los pull requests.

Básicamente **un pull request es una petición para integrar nuestras propuestas o cambios de código a un proyecto que tienen que aprobar otros integrantes del equipo**.

Esta herramienta permite no solo llevar de forma más ordenada las tareas en la etapa del desarrollo, sino también crear propuestas o cambios que puedan ser integrados posteriormente a dicho proyecto.

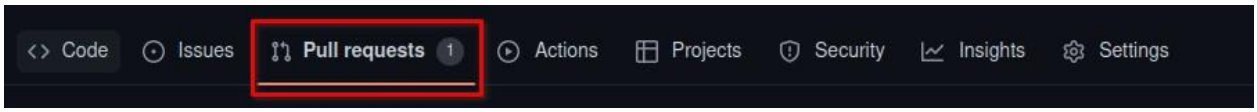
Creando un pull request

Para empezar con los pull requests, primero nuestra rama debe estar sincronizada en el remoto:

- `git push origin <nombre_rama>`

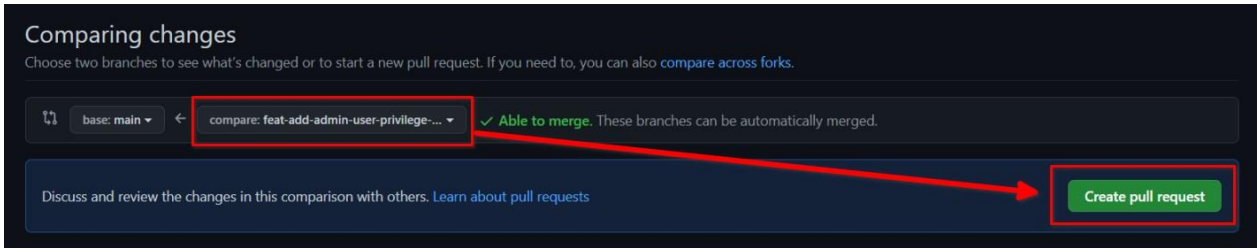
Supongamos que la rama se llama **“feat-add-admin-user-privilege-to-create-question”**

1. Vamos a la pestaña “**Pull requests**” y tocamos en “**New pull request**”

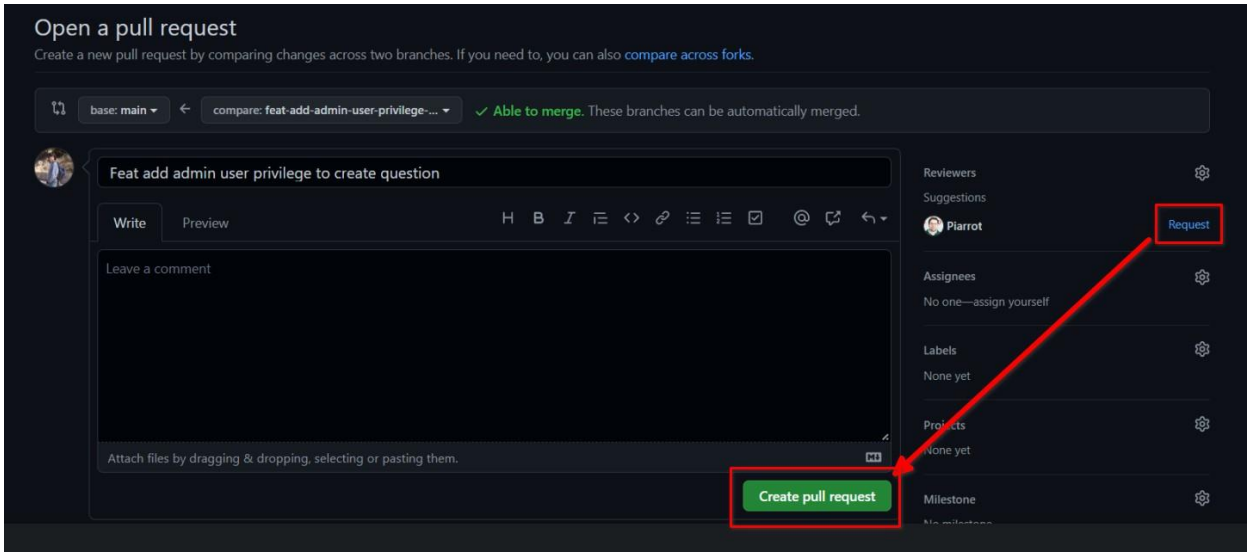


2. Seleccionamos la rama que queremos comparar y hacer merge con main. Nos aseguramos que en base esté la “rama objetivo” (**main**) y en “compare” esté la “rama fuente” (**feat-add-admin-user-privilege-to-create-question**)

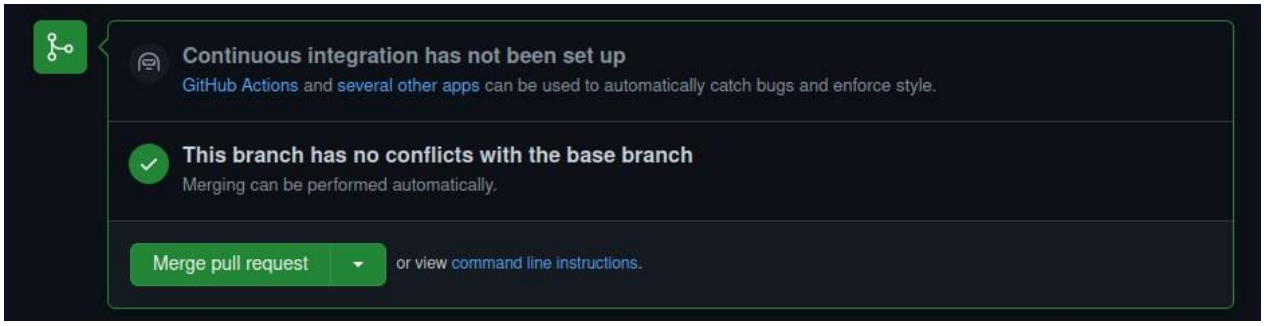
3. Apretamos el botón “**Create pull request**”.



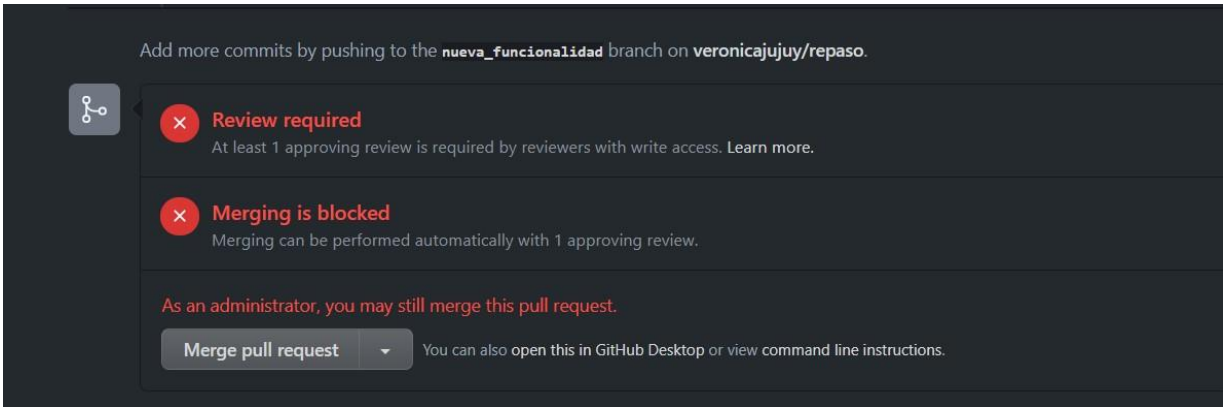
4. Agregamos comentarios para detallar nuestro aporte al proyecto y asignamos quién será la persona que harán el review de la nueva funcionalidad, y apretamos en “**Create pull request**”



5. Los reviewers serán los encargados de dar feedback, cerrar o aprobar el pull request. También, si no tenemos reviewers por algún motivo, podemos mergear “a mano” apretando el botón “Merge pull request” directamente nosotros.

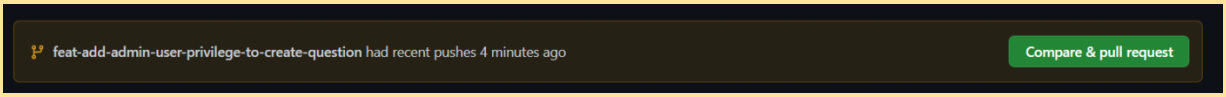


Cabe destacar que, según la configuración del repositorio, mergear a main podría necesitar si o si que haya un reviewer y el pull request sea aceptado por el mismo:



DH Tips:

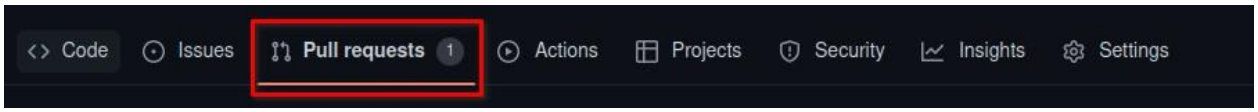
Si nuestro push es reciente, en el repositorio remoto (página de github) aparecerá la nueva incorporación:

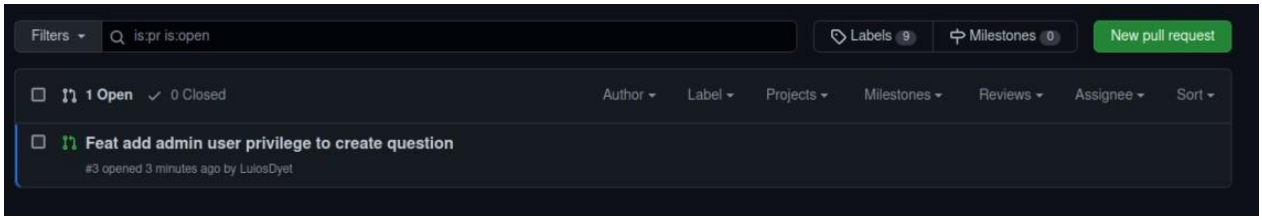


Hacemos click en el botón **“compare & pull request”** y podremos crear el pull request en menos pasos.

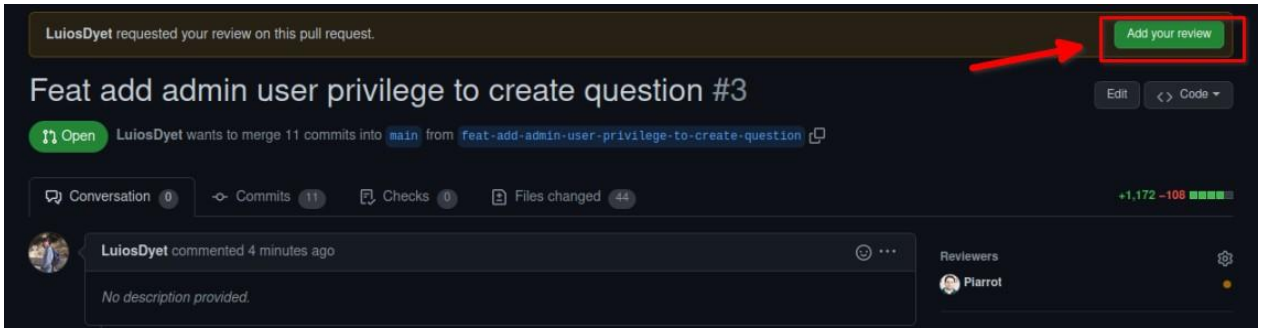
Aprobando un pull request

En nuestro equipo realizaron un pull request, y nos piden que lo revisemos. Para ver los pullrequest activos vamos a la pestaña pull request y nos aparecerán todos los pull requests realizados por nuestro equipo:

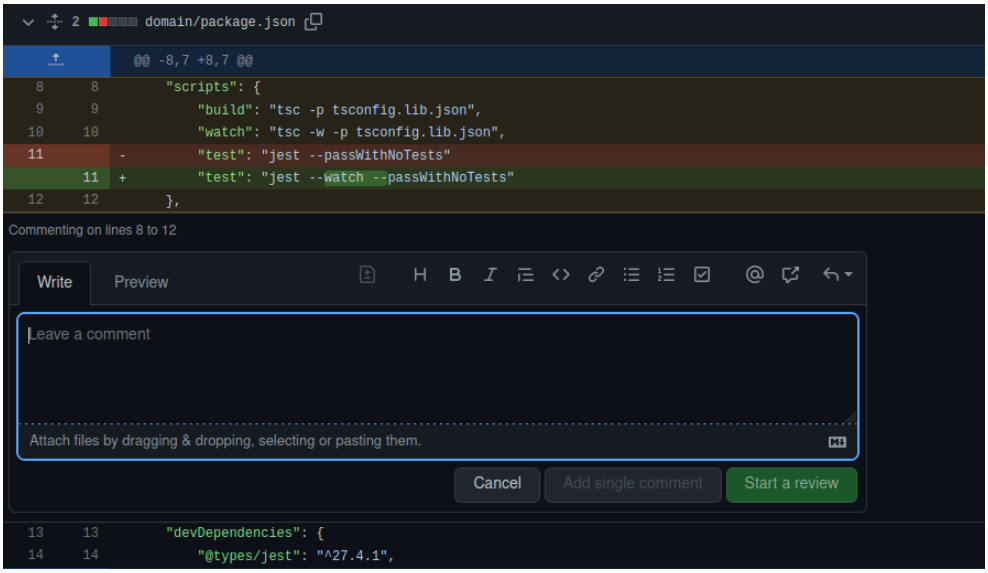




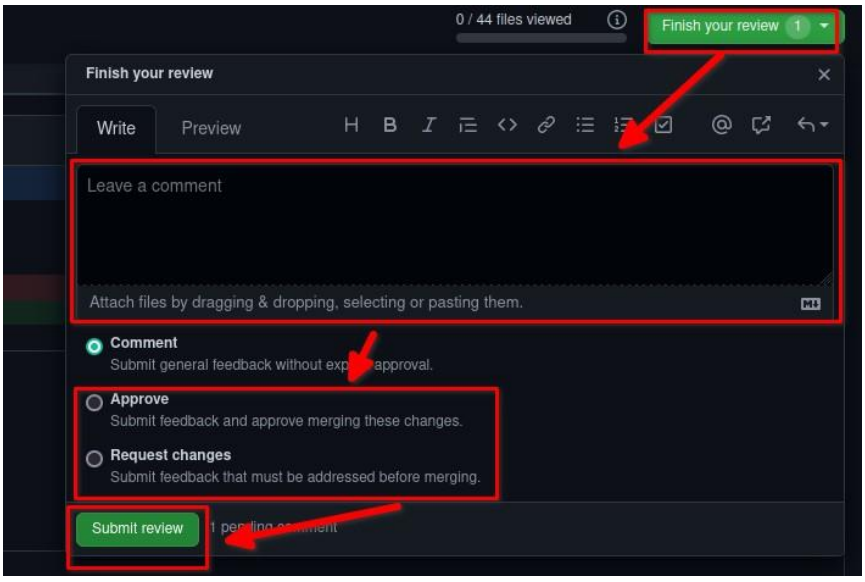
1. Una vez elegido el pull request, tocamos el botón “add your review”.



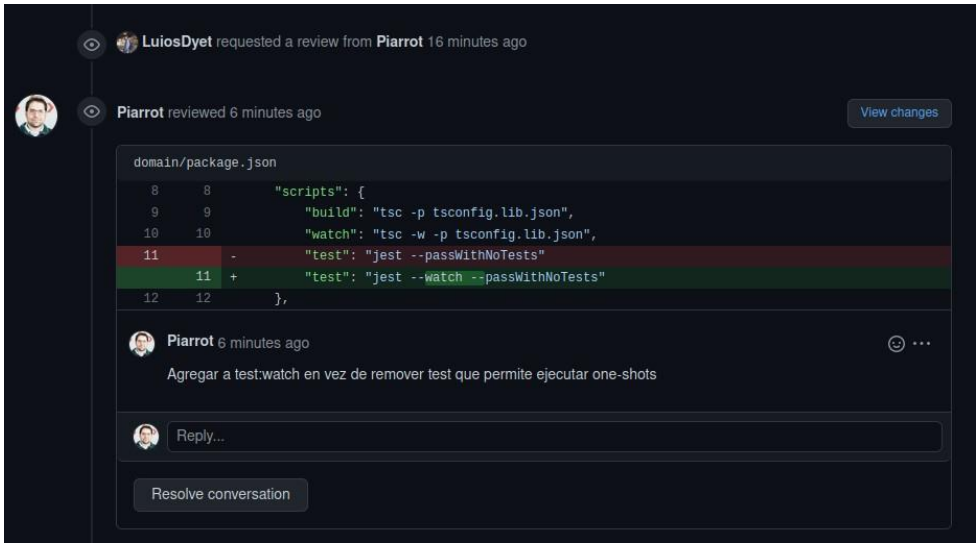
2. En esta pantalla podremos dejar comentarios en las distintas líneas de código pasando el mouse por arriba y tocando en el botón azul “+”. Para seleccionar más de una línea debemos arrastrar el mouse entre las líneas antes de soltar el botón del mouse.



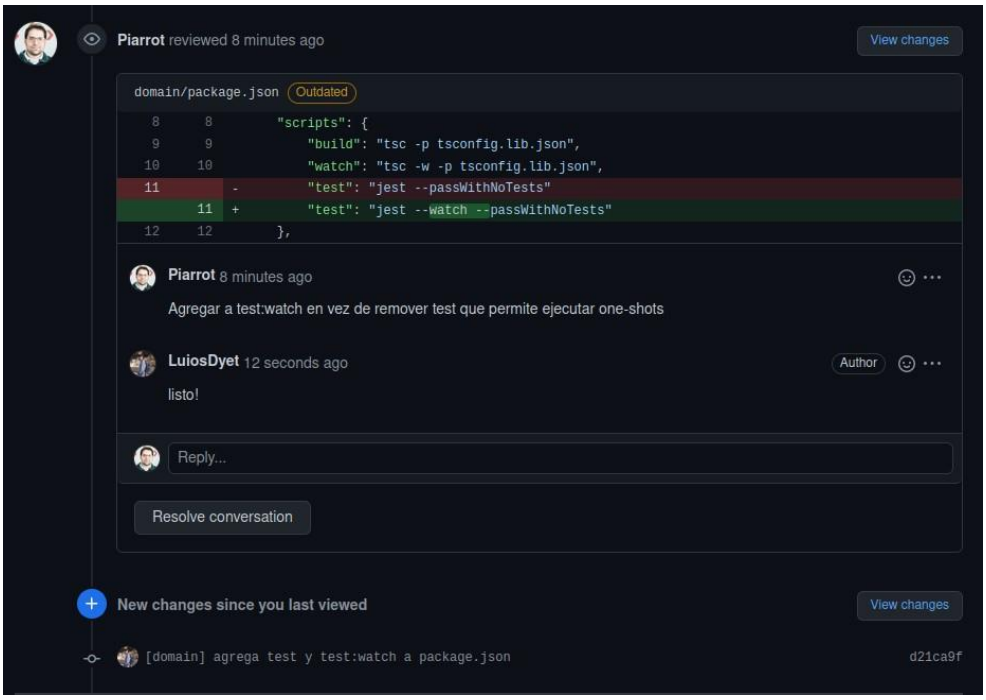
3. Una vez estamos satisfechos, tocamos el botón arriba a la derecha “Review Changes” y dejamos un comentario sobre la review, marcando si la aprobamos o no.



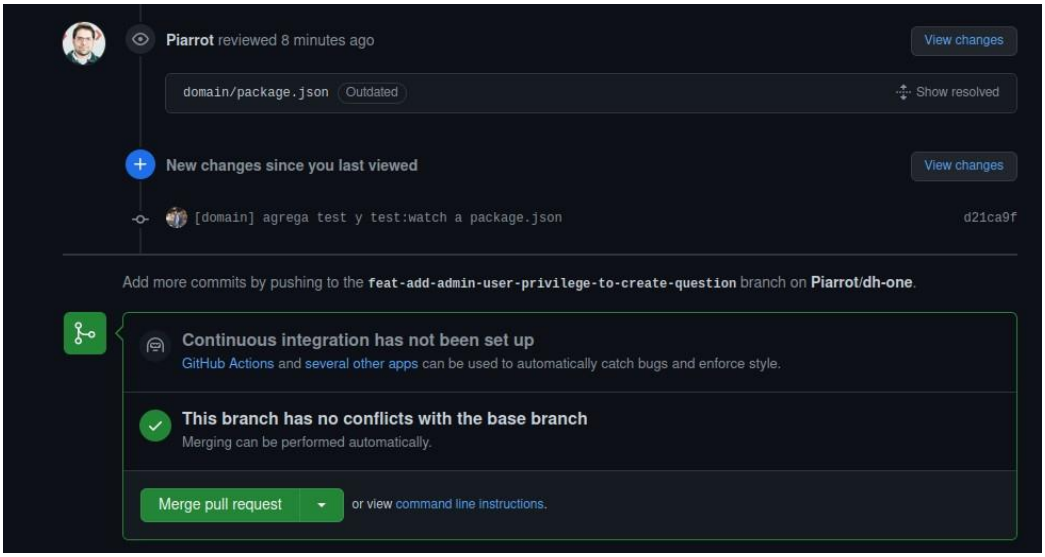
4. Si hay cambios para hacer, la conversación sobre los cambios requeridos se verá en el pull request.



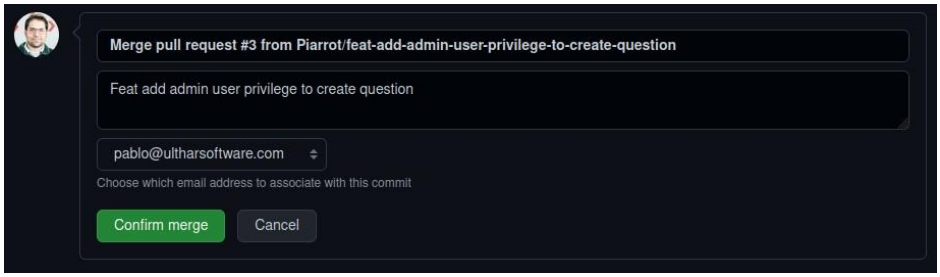
5. Cualquier nuevo commit (+push) a la misma rama, se verá en el pull-request y se podrá notificar a los reviewers que se hicieron los cambios requeridos.

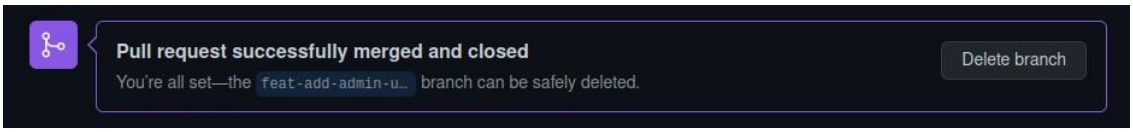


6. Una vez que el código fue revisado y aprobado, ya se podrá mergear la rama.



7. Tocamos en “Merge pull request” y confirmamos el merge:





8. Luego, la rama puede ser borrada de manera segura, como dice la imagen, apretando el botón “delete branch”, para no dejar nuestro repositorio con ramas que ya no van a ser utilizadas.

¡Hasta la próxima!