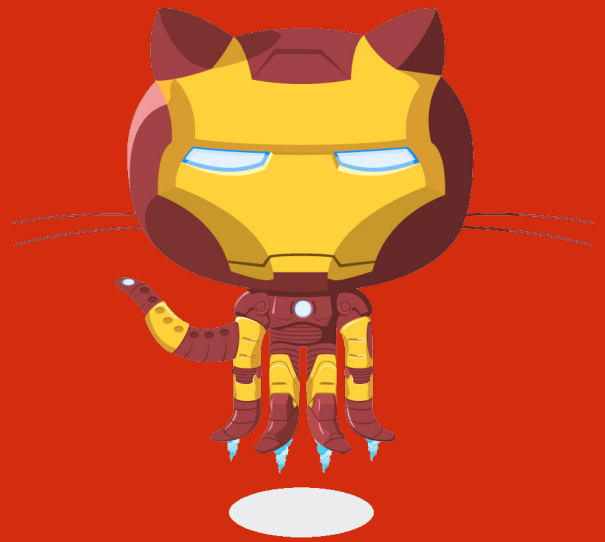
Trabajando en Equipo con Git y Github

Harys Vizcaino

@harysvizcaino



harysvizcaino@gmail.com

Git

Agenda

- Conocer mejor los commit
- Repositorios remotos
- Fetch
- Pull
- push
- branches
- Merge
- Conflictos
- Fork
- Pull request

Extras

- Git ignore
- Zenhub
- No me llevo mucho con la Consola

Git

$ git commit

El ‘commit’ nos ayuda a guardar los cambios que fueron hechos a un archivo o directorio en nuestro registro de git.

Importante: cuando hacemos un commit el commit nos pedira que introduzcamos un mensaje, es recomendable que este mensaje contenga un poco de información de ‘que exactamente tocamos en el codigo o en el directorio y el porque’.

git commit –m “MENSAJE”

$ Moviendonos entre commits

En cualquier momento podemos querer deshacer cambios, modificar un commit o volver a algún estado anterior de nuestro proyecto.

git commit

Git

$ git commit --amend

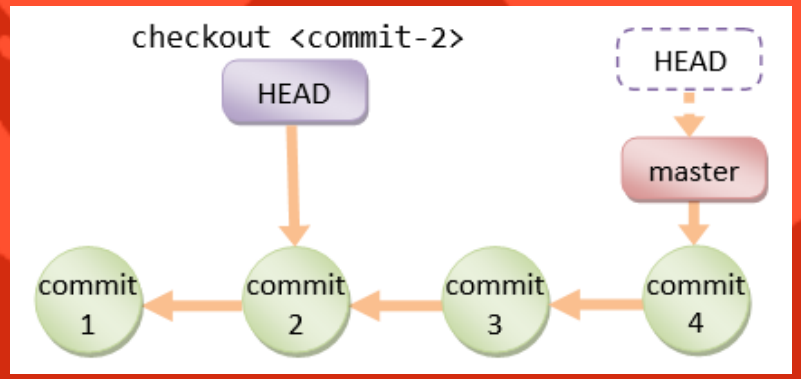
Este comando utiliza lo que haya en tu área de preparación para el commit. Si no has hecho ningún cambio desde la última confirmación (por ejemplo, si ejecutas este comando justo después de tu commit anterior), esta instantánea será exactamente igual, y lo único que cambiarás será el mensaje de confirmación.

git commit --amend

Git

$ git checkout ID

Con git checkout podemos movernos entrer commits y entre branches, haciendo un git checkout ID del commit podemos regresar a un commit anterior y ver como estaba nuestro código o nuestro directorio en ese entonces, también podemos modificar cosas, solo que para traer esos cambios al presente necesitaríamos crear un nuevo branch y luego hacer un merge de este branch

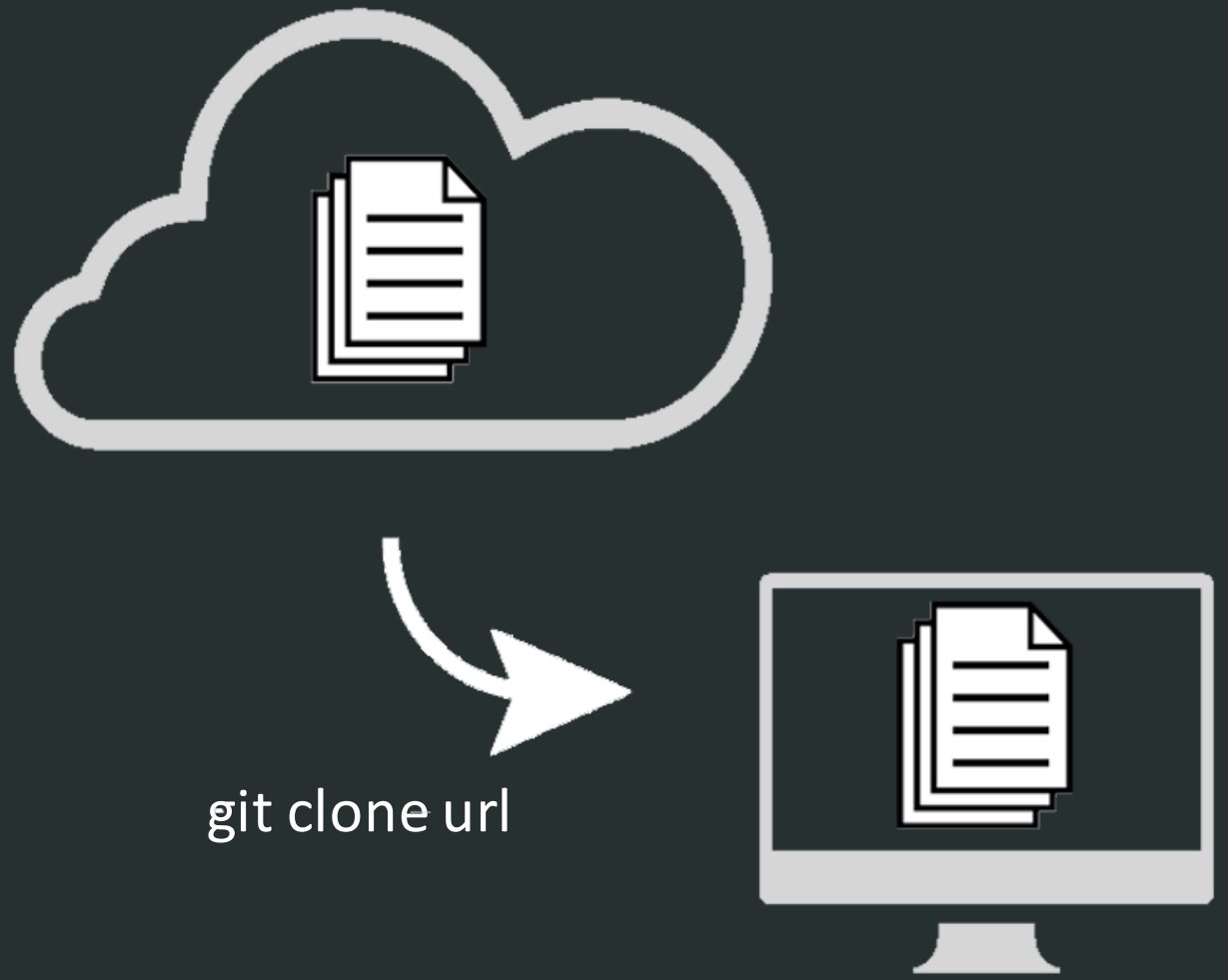


$ Trabajando con Repositorios remoto

Para poder colaborar en cualquier proyecto Git, necesitas saber cómo gestionar tus repositorios remotos. Los repositorios remotos son versiones de tu proyecto que se encuentran alojados en Internet o en algún punto de la red. Puedes tener varios, cada uno de los cuales puede ser de sólo lectura, o de lectura/escritura, según los permisos que tengas. Colaborar con otros implica gestionar estos repositorios remotos, y mandar (push) y recibir (pull) datos de ellos cuando necesites compartir cosas.

$ git clone [url]

Si deseas obtener una copia de un repositorio Git existente —por ejemplo, un proyecto en el que te gustaría contribuir— el comando que necesitas es git clone



$ git remote

Para ver qué repositorios remotos tienes configurados, puedes ejecutar el comando git remote. Mostrará una lista con los nombres de los remotos que hayas especificado el repositorio que hemos clonado debe aparecer como "origin"

$ Agregando repositorios remotos

Para agregar un repositorio remoto usamos el comando ‘ git remote ’ con la opción ‘ add ’ seguido del nombre con el que nombraremos el enlace del repositorio mas la url del repositorio

```
git remote add [nombre] [url]
```

$ git fetch & git pull

git fetch: Este comando recupera todos los datos del proyecto remoto que no tengamos aun en nuestro copia local del proyecto. Luego de ejecutar este comando deberíamos tener referencia a todas las ramas del repositorio.

git pull: para recuperar y unir automáticamente la rama remota con tu rama actual.

$ git push

Cuando tu proyecto se encuentra en un estado que quieres compartir, tienes que enviarlo a un repositorio remoto. El comando que te permite hacer esto es sencillo: git push Si quieres enviar tu rama maestra (master) a tu servidor origen (origin), ejecutarías esto para enviar tu trabajo al servidor:

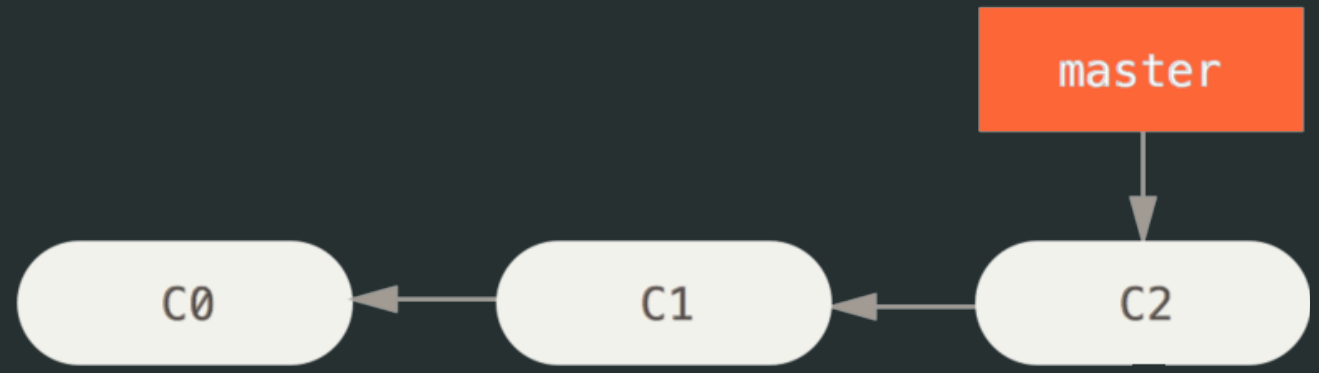
git push [nombre-remoto][nombre-rama].

$ Eliminando y renombrando repositorios remotos

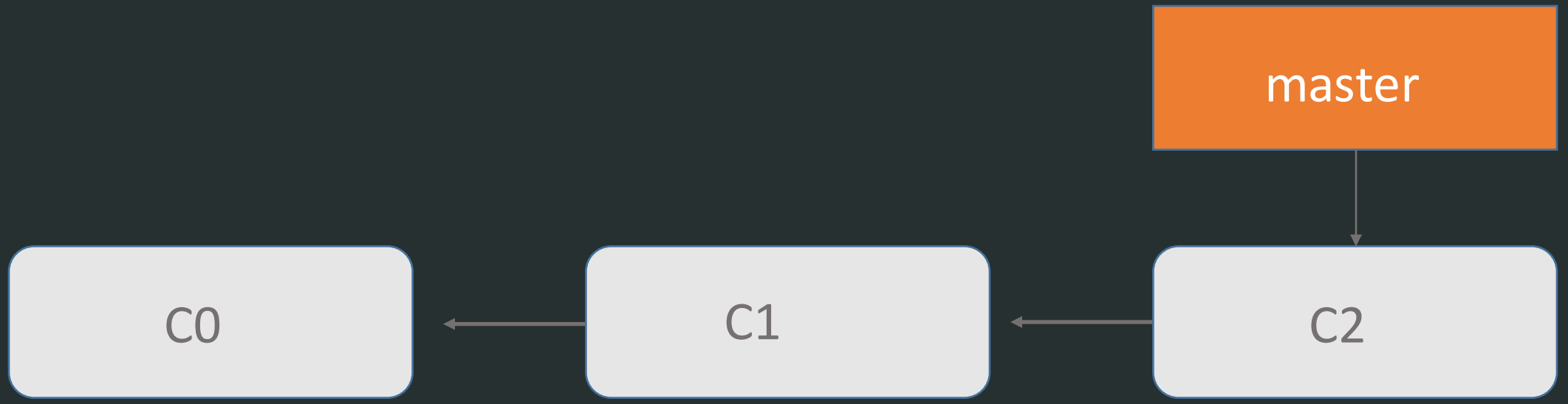
Si queremos renombrar una referencia a un repositorio remoto, podemos ejecutar el comando git remote rename en caso que queramos eliminar el repositorio podemos ejecutar el comando git rm

$ Branches [git branch]

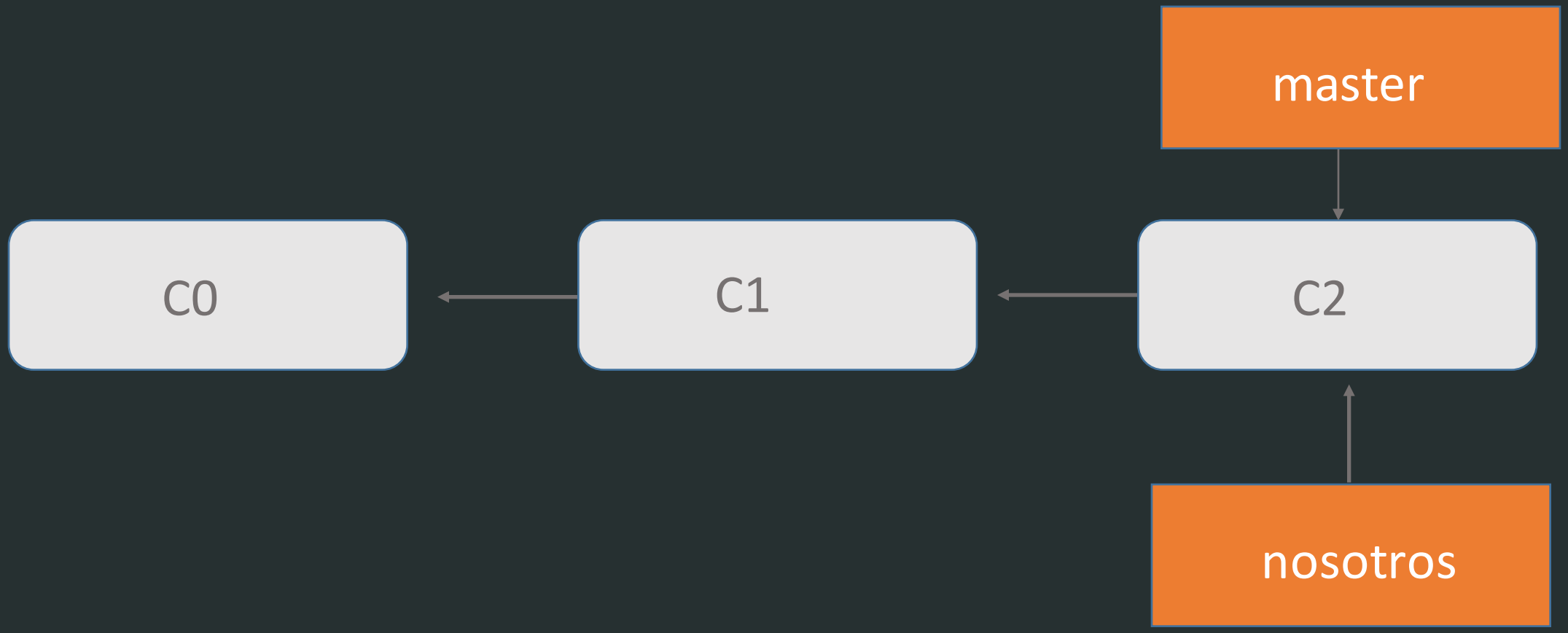
Las ‘ramas’ branches es esencialmente una línea de desarrollo independiente. Podemos tomar ventaja de ellas cuando se trabaja en nuevos ‘features’ o arreglando errores ‘bugs’.



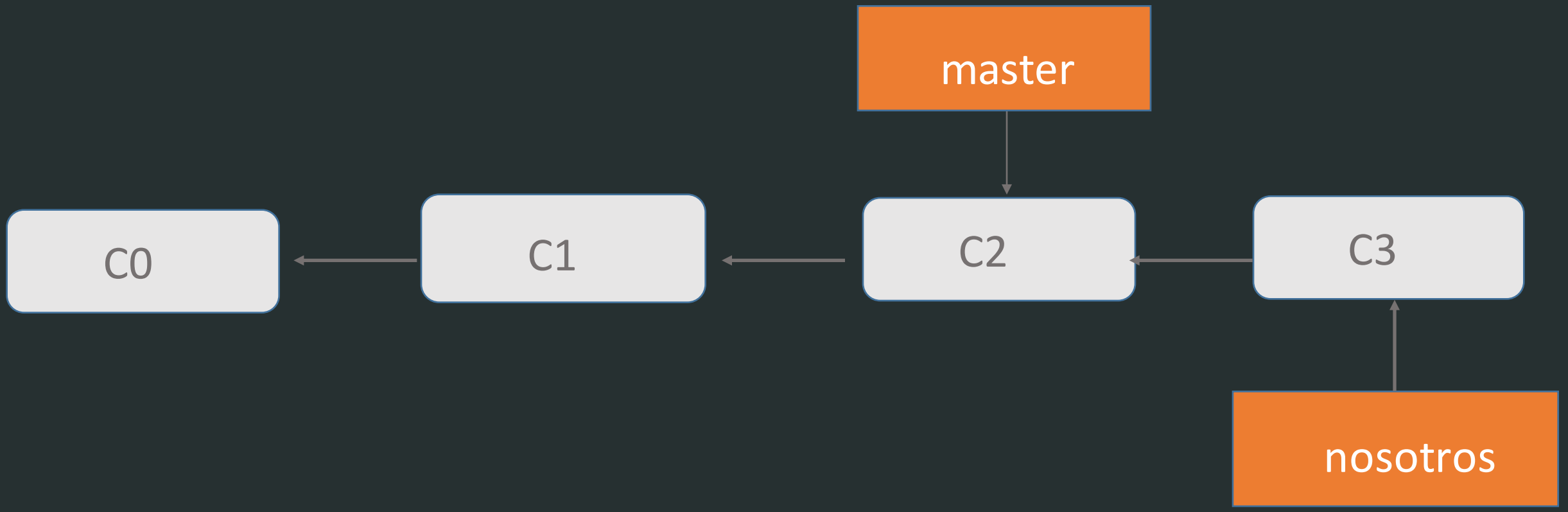
$ git & github



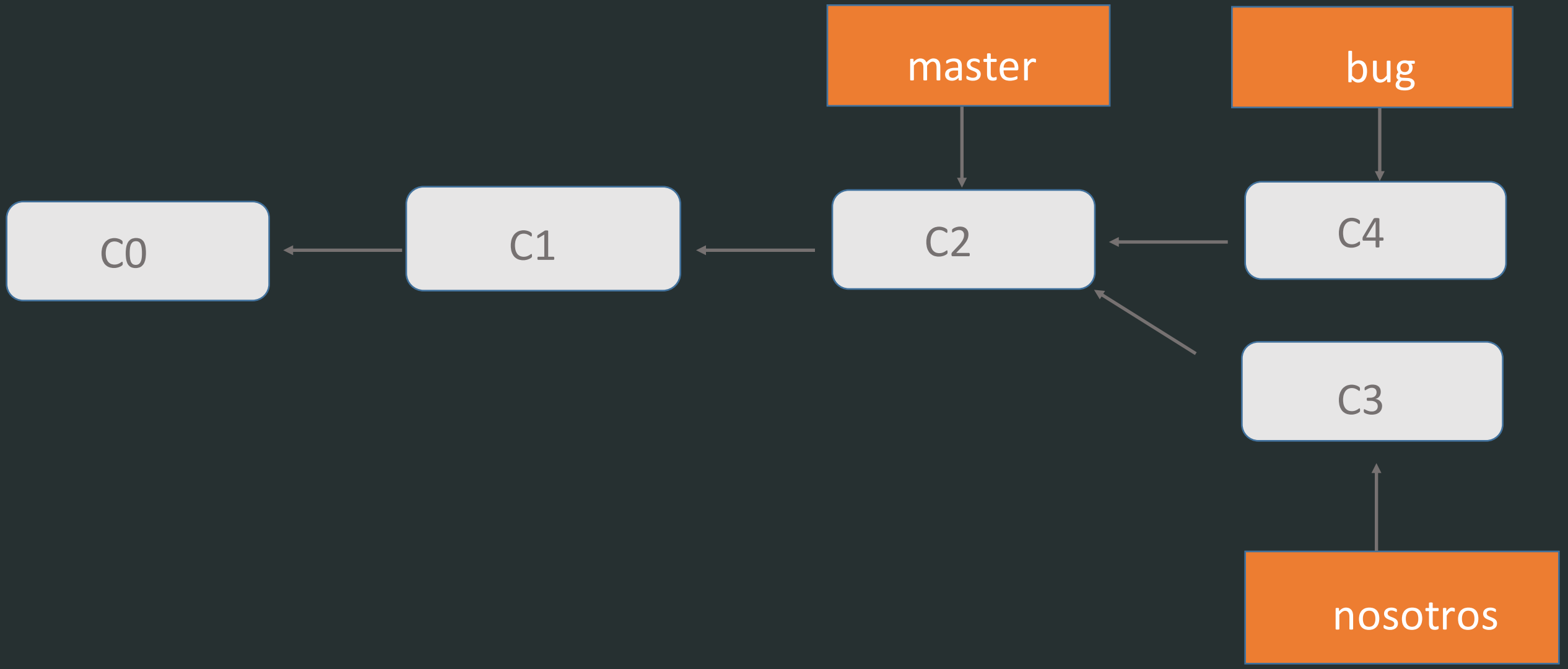
$ git & github

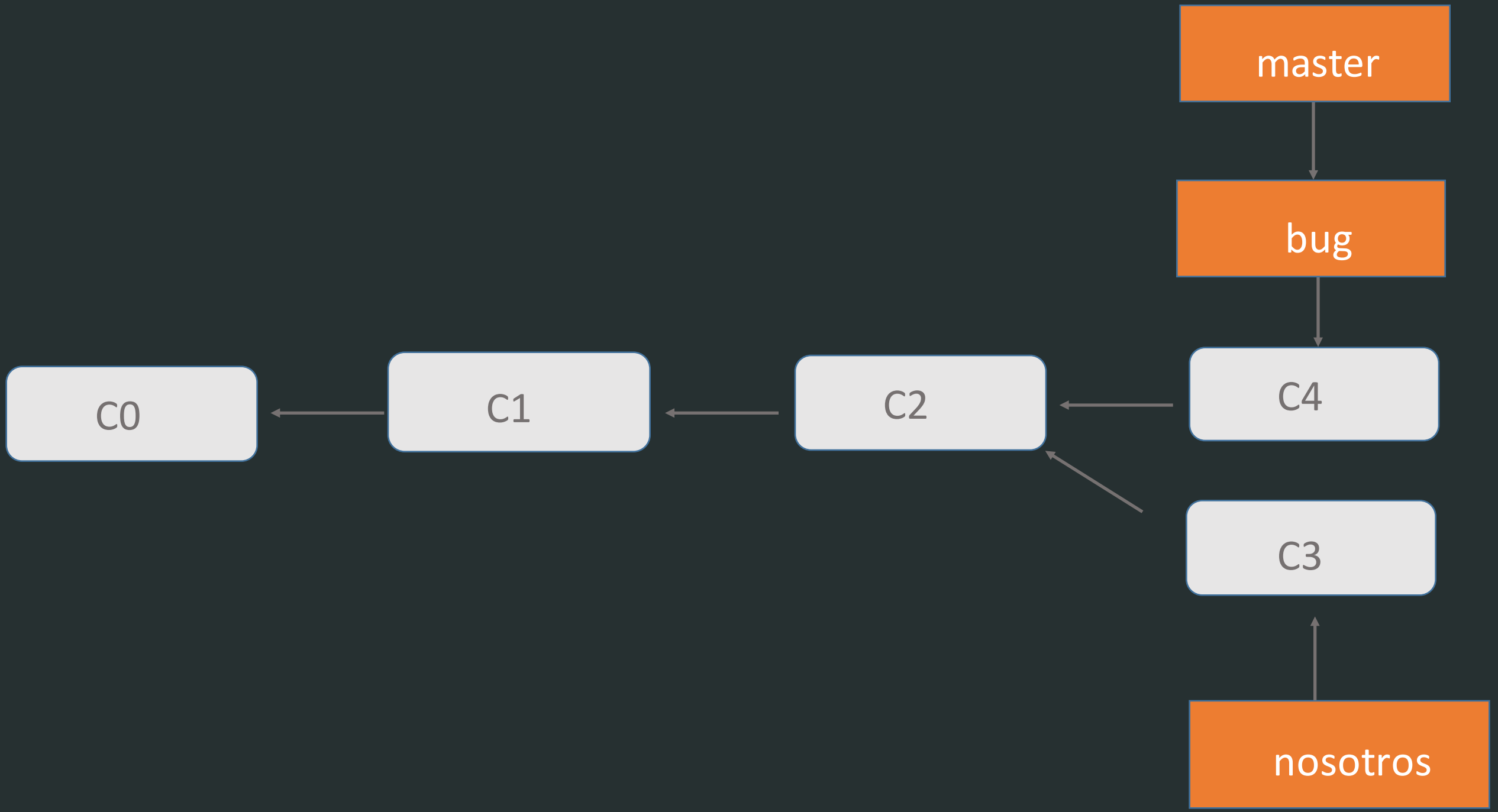


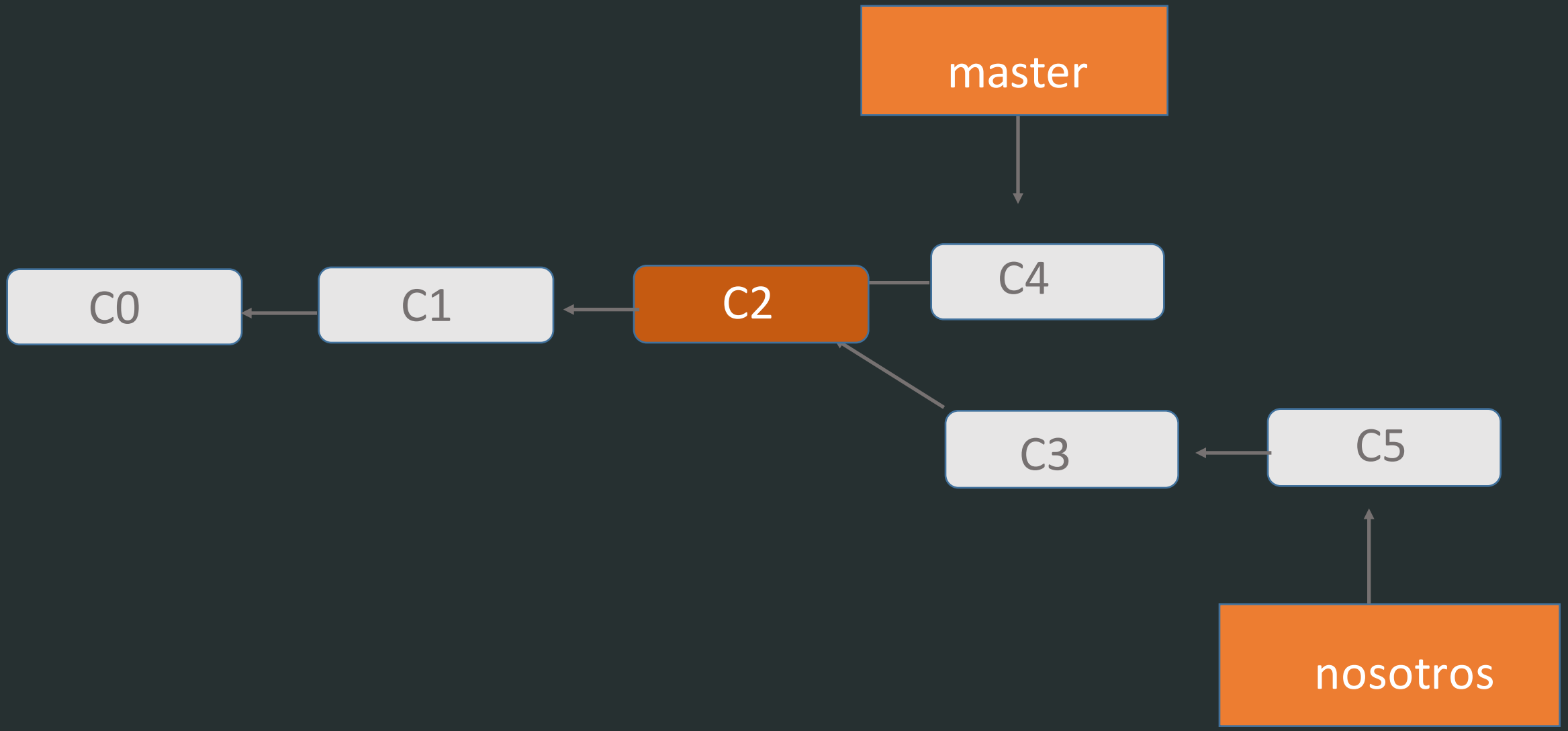
$ git & github

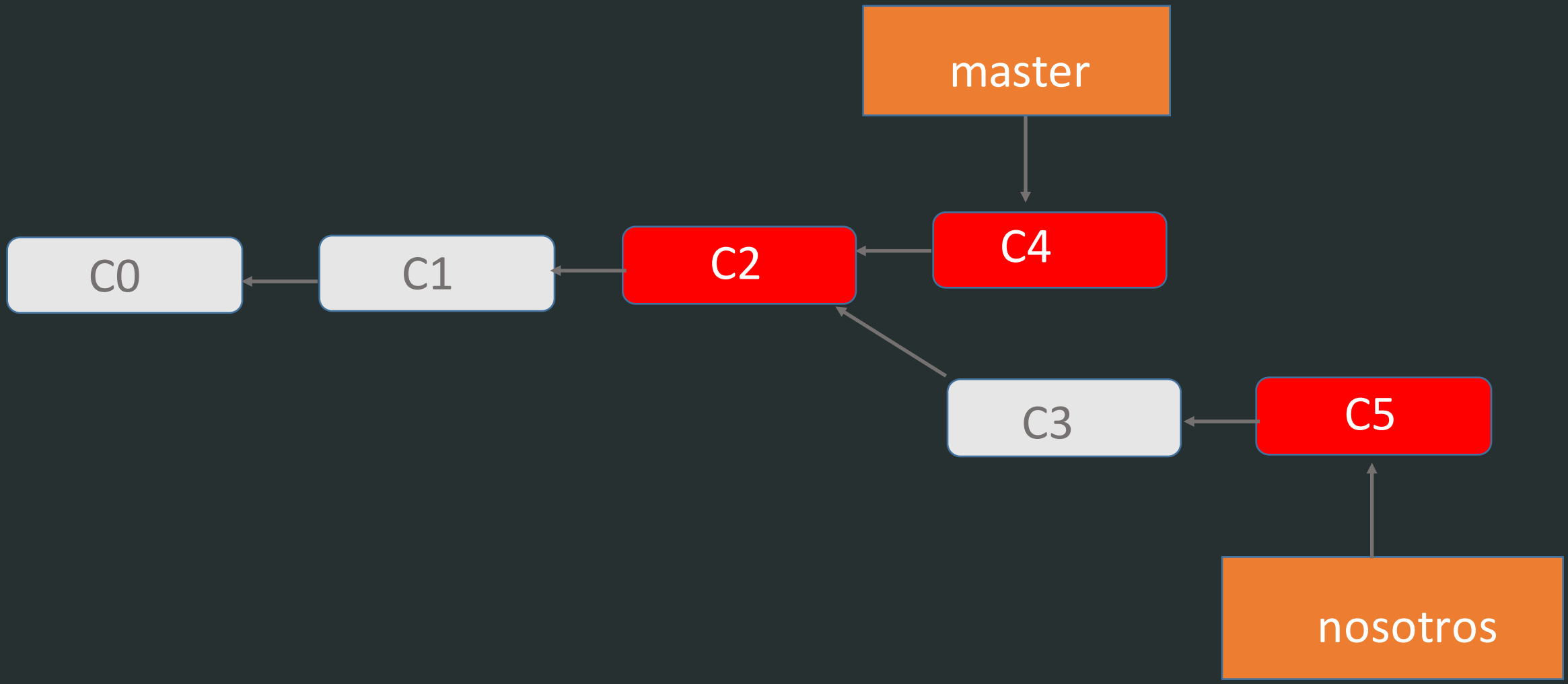


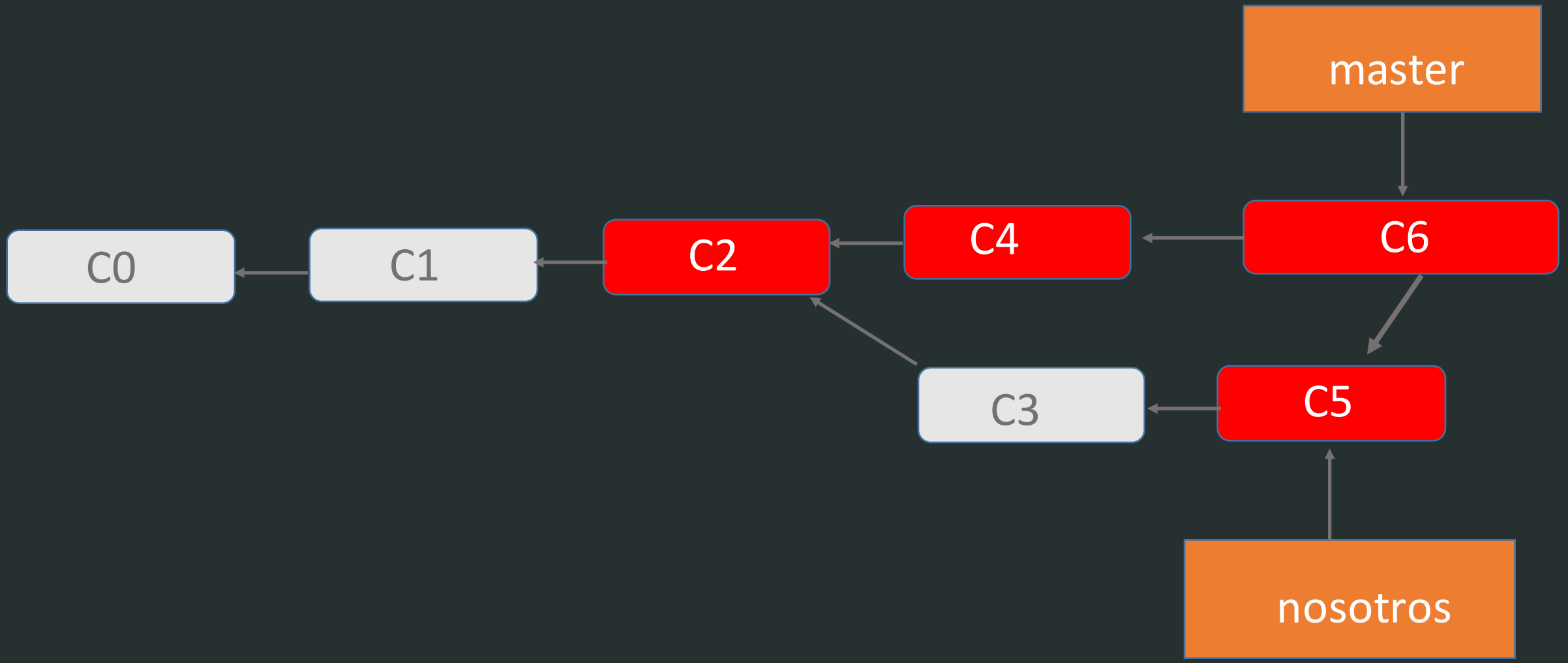
$ git & github




master
bug
C4
C2
C1
C0
C3
nosotros


master
C0
C1
C2
C4
C3
C5
nosotros


master
C0
C1
C2
C4
C3
C5
nosotros


master
C0
C1
C2
C4
C6
C3
C5
nosotros

$ git conflict

El algunas ocasiones, los procesos de fusión ‘merge’ no suelen ser tan fluidos como en los casos anteriores, si hay modificaciones dispares en distintas ramas git no será capaz de fusionarlas directamente. Si alguien modifico la misma porsion de código que modificamos en nuestra rama cuanto intentemos hacer la fusión se creara un conflicto de fusion.

$ git fork & contribuir a proyectos de terceros

Hacer un fork de un repositorio nos permite experimentar libremente con cambios sin afectar un proyecto original.

El uso mas común de fork es para proponer cambios a proyectos de alguien mas o para usar el proyecto de alguien mas como punto de inicio de nuestro propio proyecto.

$ Mantener el proyecto actualizado

Para mantener el proyecto actualizado agregamos el proyecto original ‘proyecto desde el realizamos el fork’ como un proyecto remoto y hacemos fetch y luego el merge.

1 – agregamos el proyecto original como remoto git remote add upstream [url]

2 - hacemos el fetch del proyecto para descargar todos los cambios relizadosen en el proyecto original git fetch nombredadeo [upstream]

3 – nos movemos a master (en caso que no estemos en master) y hacemos merge del proyecto online en nuestro proyecto local. git merge upstream/master

$ Pull request

Básicamente un **pull request** es una petición para integrar nuestras propuestas o cambios de código a un proyecto.

Cuando estamos trabajando como parte de un equipo, debemos ser un poco mas cuidadosos, es alli donde entra en juego el uso de los pull request.

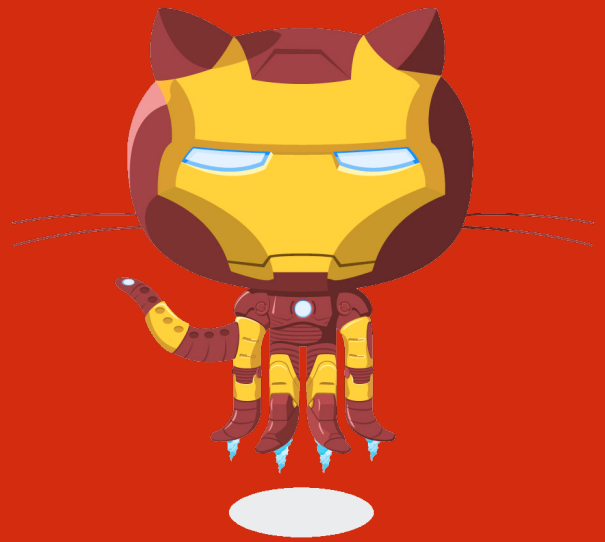
$ Git Ignore

Un archivo gitignore especifica los archivos que queremos que git ignore de forma intencional los archivos ya conocidos por git no se ven afectados.

F-!-N

Harys Vizcaino

@harysvizcaino



harysvizcaino@gmail.com

Git