

¿QUÉ ES GIT Y CÓMO FUNCIONA?

INTRODUCCION

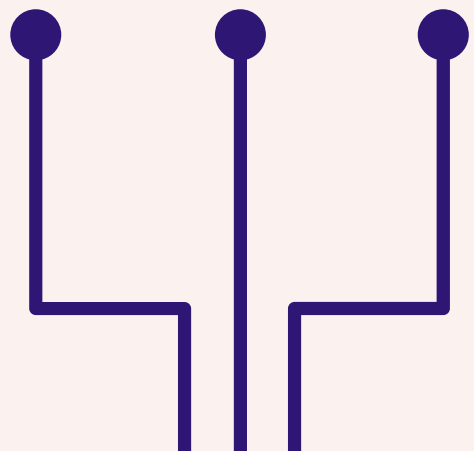
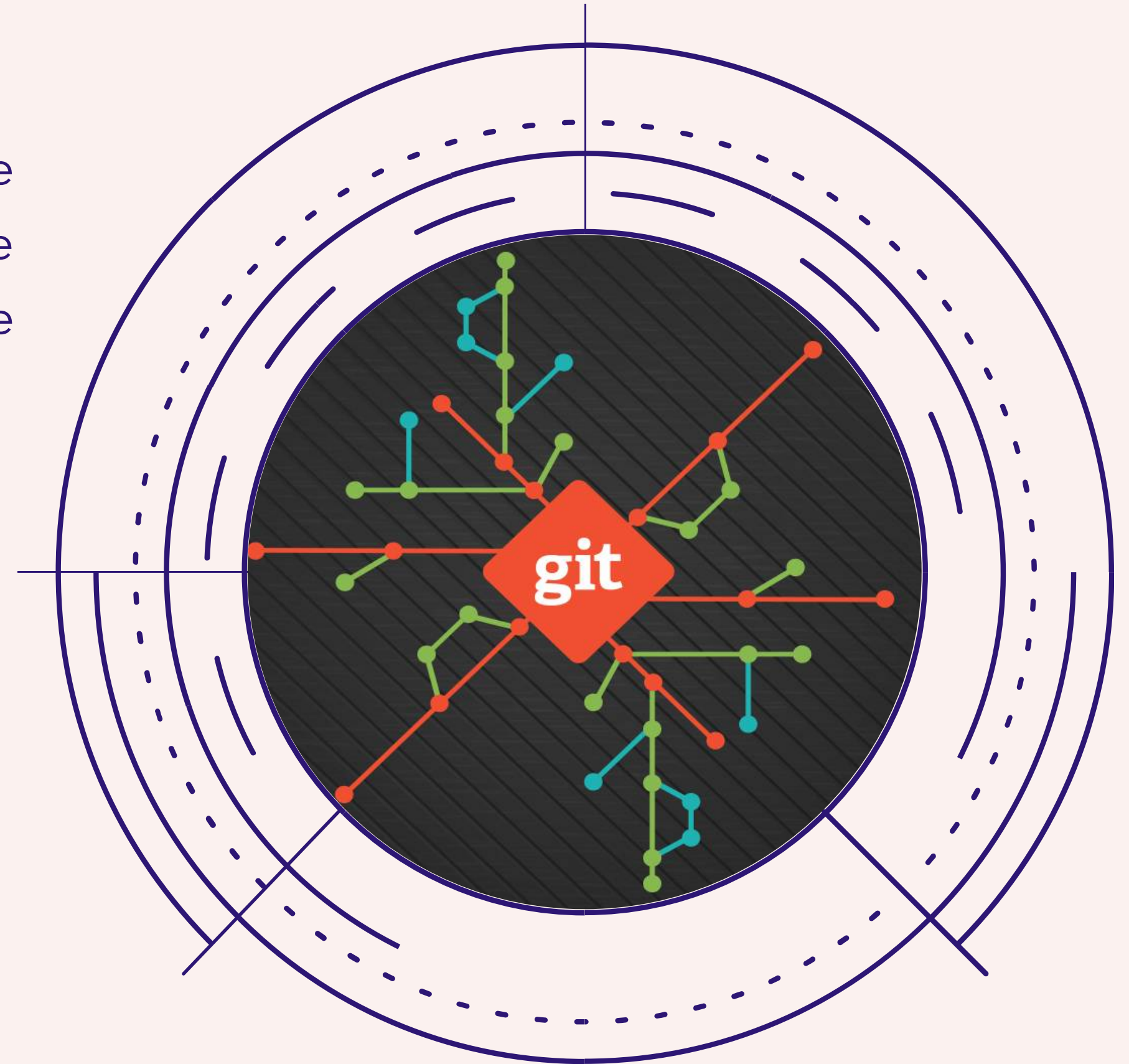
En el mundo del software, puedes llevar un historial, así como en un diario. No es algo que se crea en un día por arte de magia. El software tiene una vida, las aplicaciones tienen una vida y tienen versiones.

En esa vida le añadimos nuevas funciones, corregimos errores, agregamos nuevas versiones. Para registrar el día a día de un proyecto se usa un sistema de control de versiones (VCS), un software que permite trabajar de forma colaborativa.

Cada desarrollador que está involucrado en el proyecto puede agregar los trabajos, cambios y correcciones que aporta al proyecto. Este sistema de control de versiones tiene un historial de todo lo que se ha hecho en el proyecto y quiénes lo han hecho. Este sistema de control de versiones es GIT.

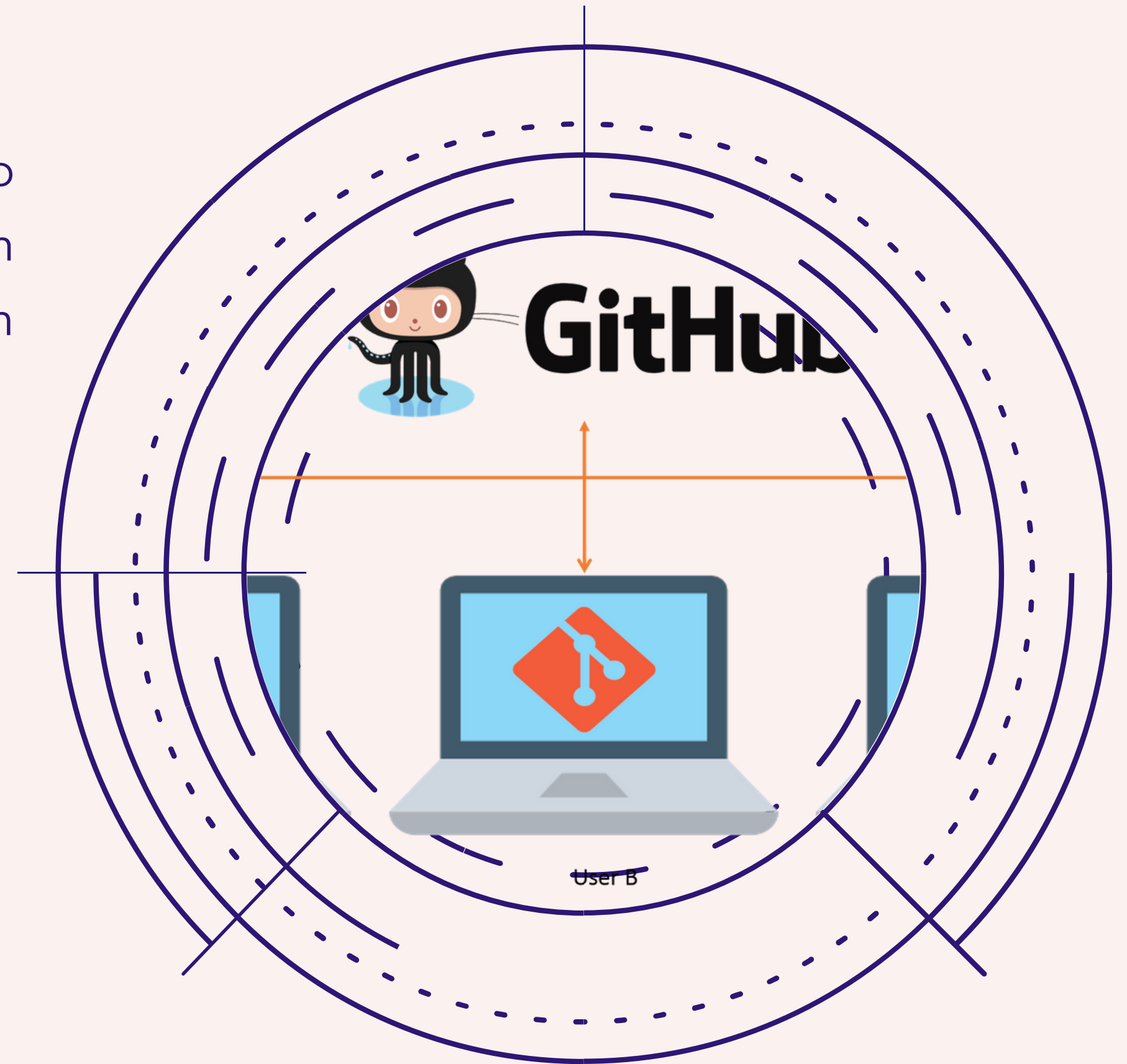
GIT

Es un sistema de control de versiones: un software que permite registrar el historial de cambios de un proyecto



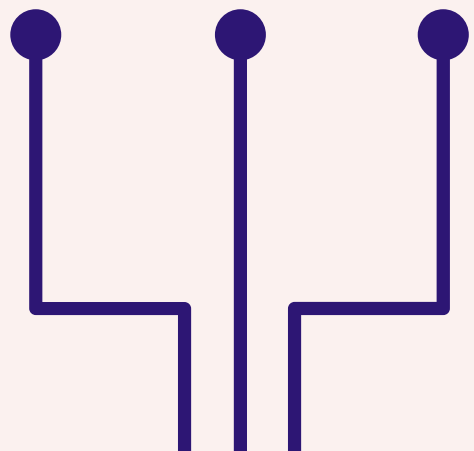
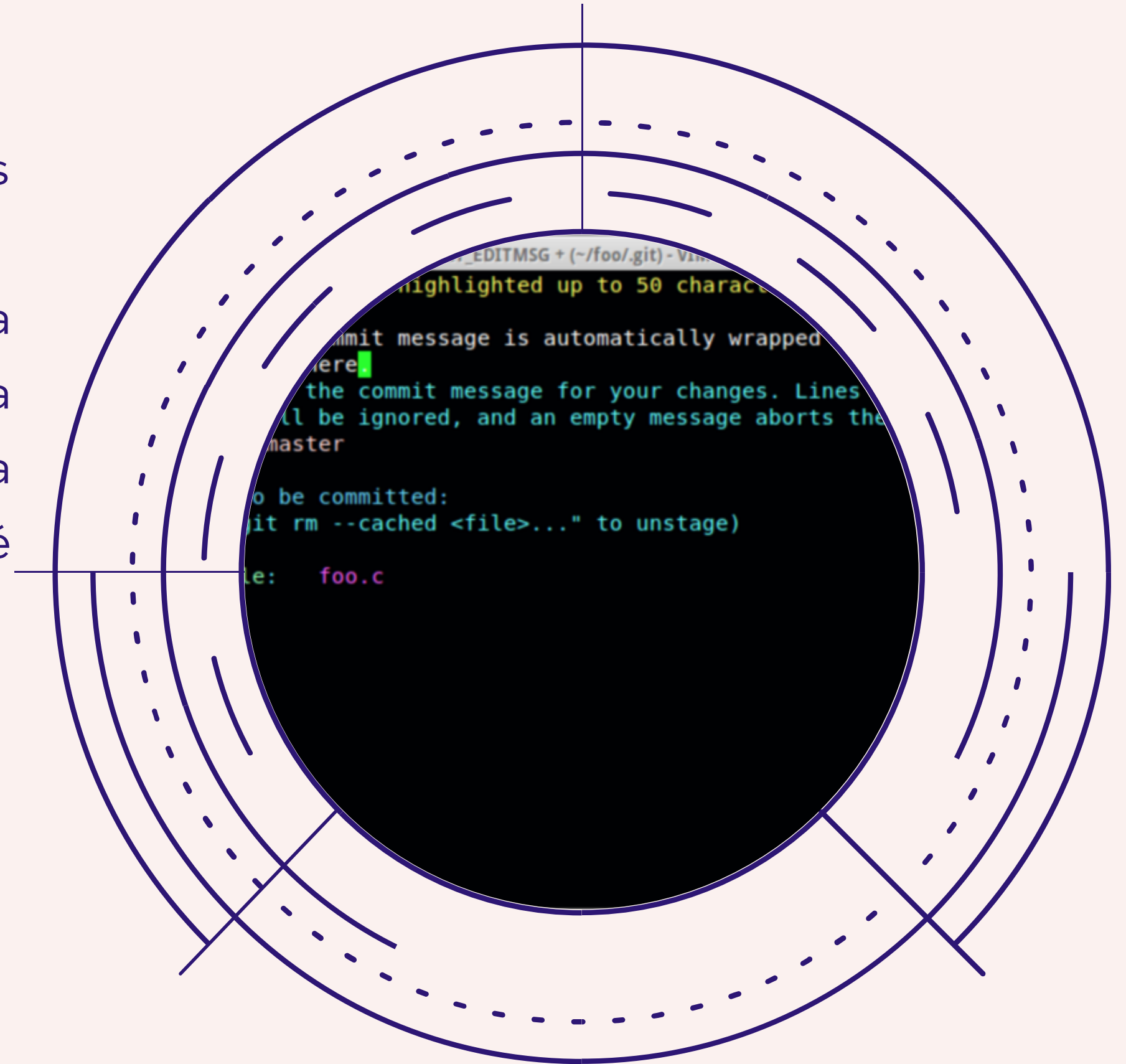
REPOSITORIO

Es todo proyecto que está siendo seguido por GIT, ya tiene un historial de GIT en el que se están registrando sus cambios.



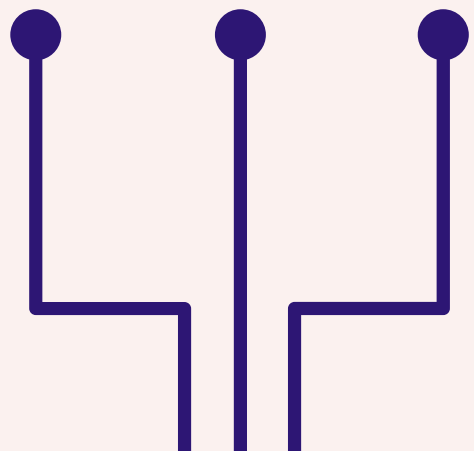
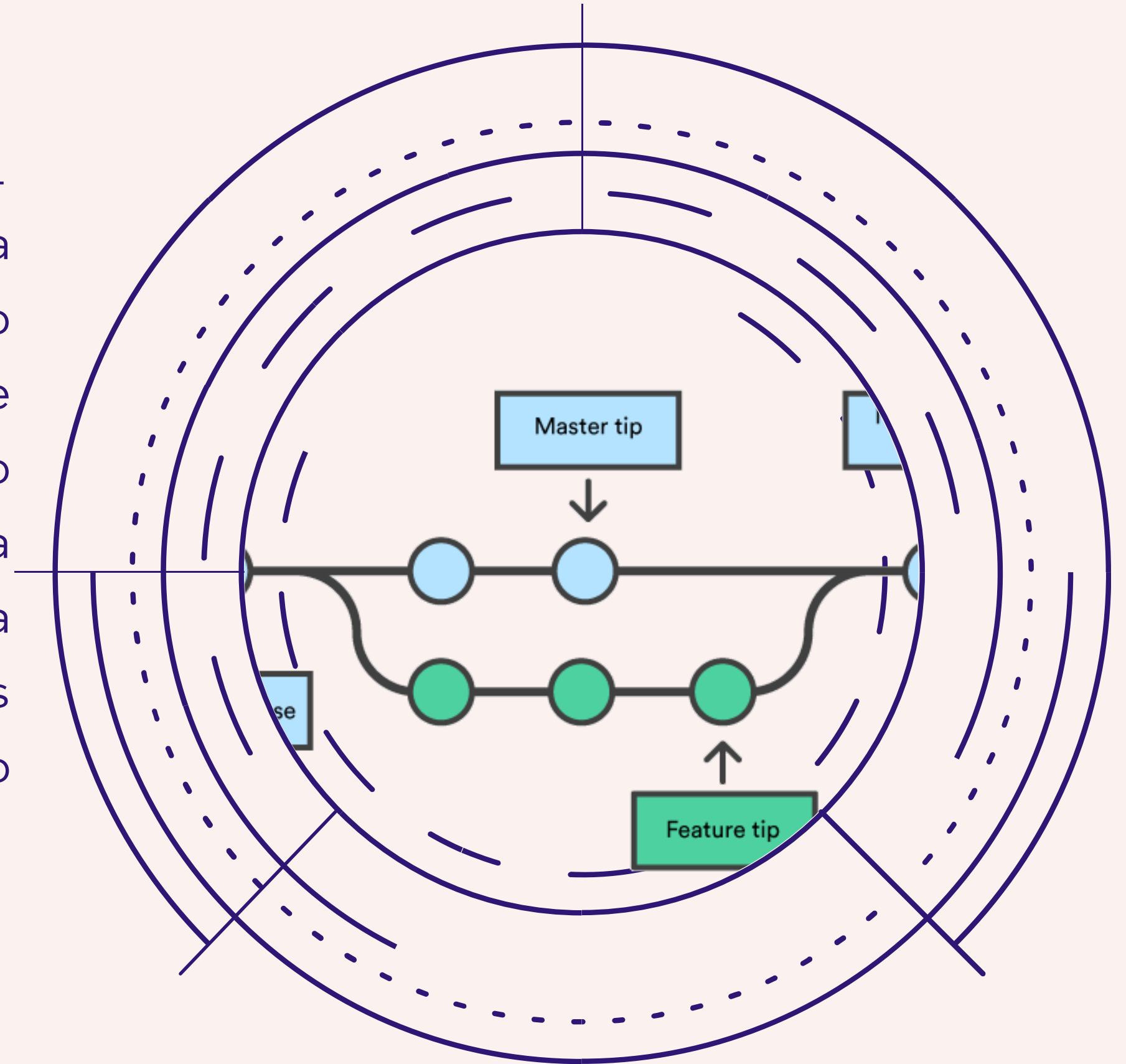
COMMIT

Es cada uno de los cambios registrados en el historial de GIT. Cada uno de los desarrolladores manda los commits de los cambios que ha hecho. No es automático, cada desarrollador tiene que decir qué hizo y por qué.



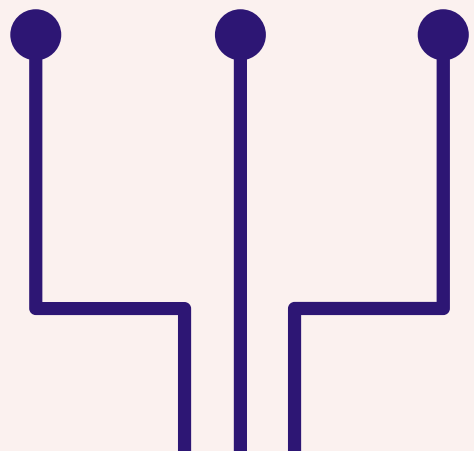
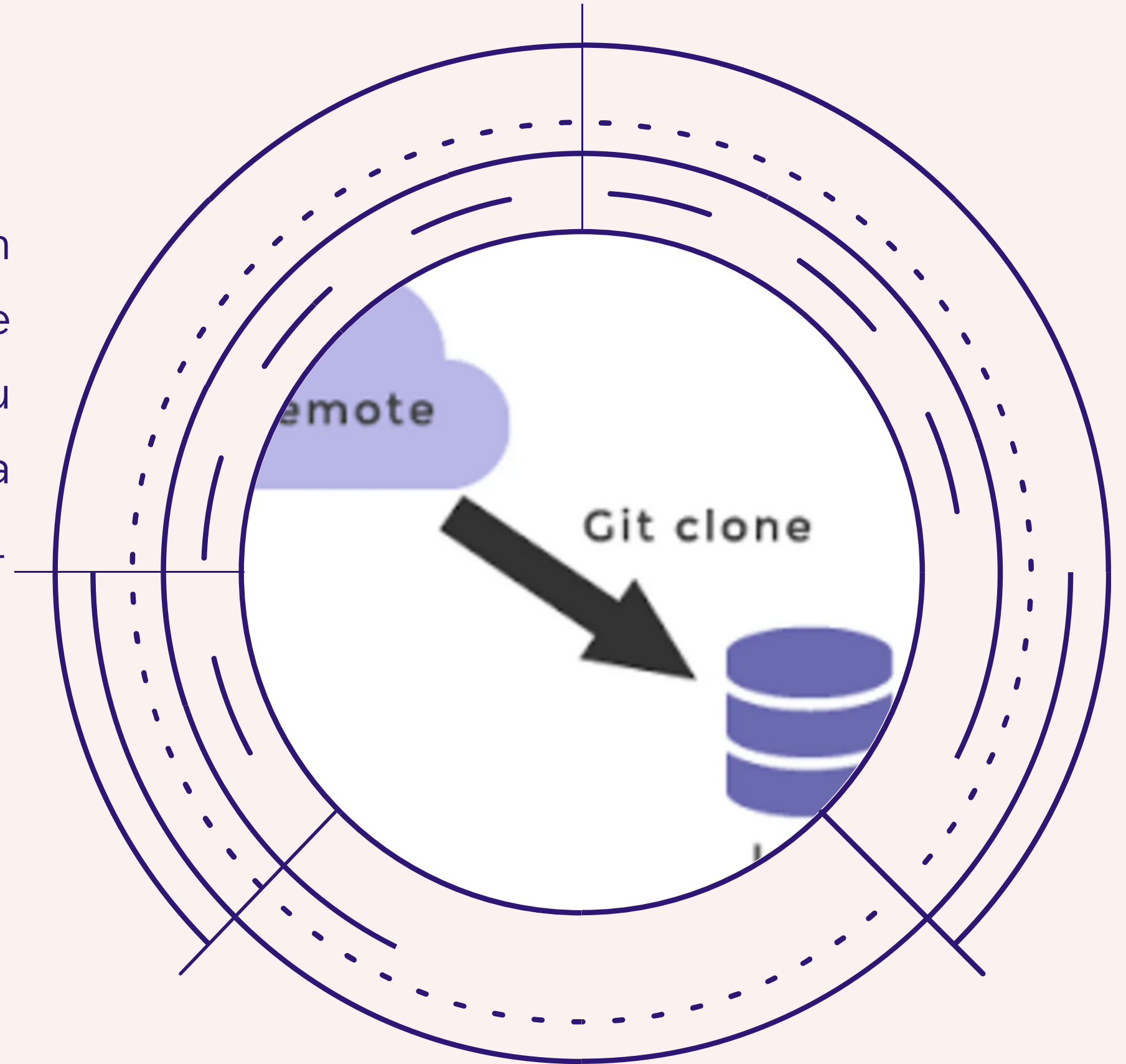
RAMAS

Son nuevos caminos que toma el proyecto. La rama principal se llama máster y es donde está el proyecto que sale a producción. Cada vez que se saca una nueva característica o que se quiere corregir algo se saca una rama, de tal manera que se pueda trabajar en un ambiente aislado. Es una copia exacta del proyecto, pero está separada.



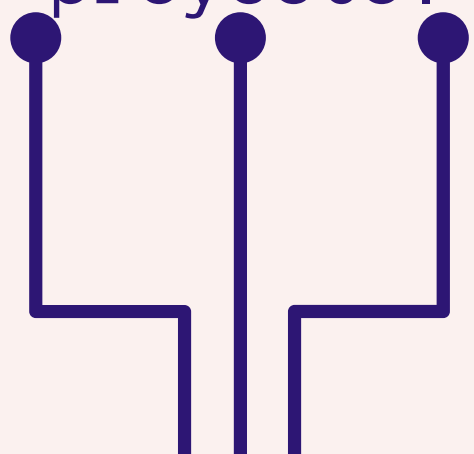
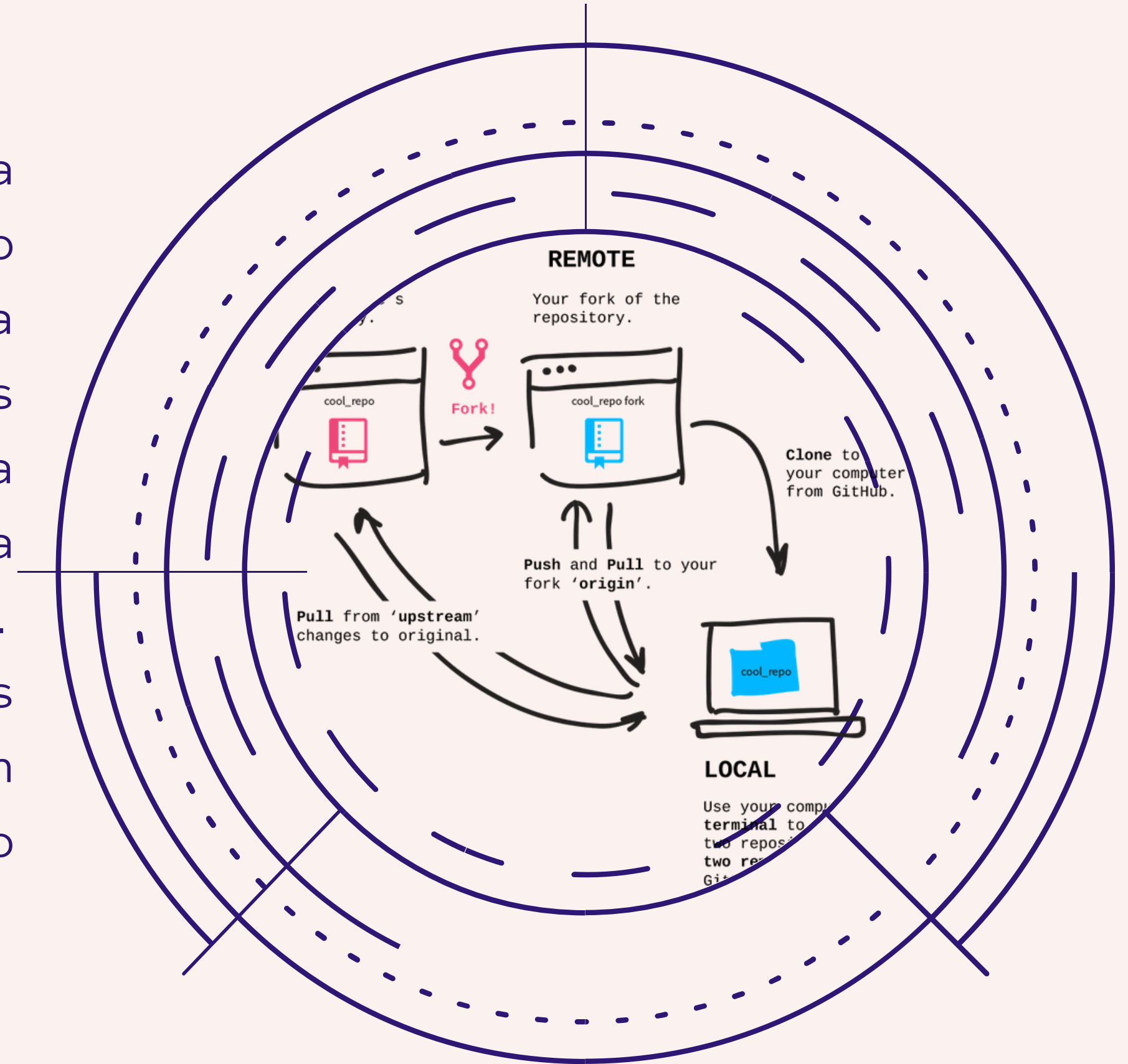
CLON

Es una copia exacta del repositorio. Cuando un programador se integra a un equipo de trabajo lo primero que debe hacer es clonar el repositorio en su equipo local. De esa manera cada miembro del equipo tiene un clon del repositorio en su equipo local



FORK

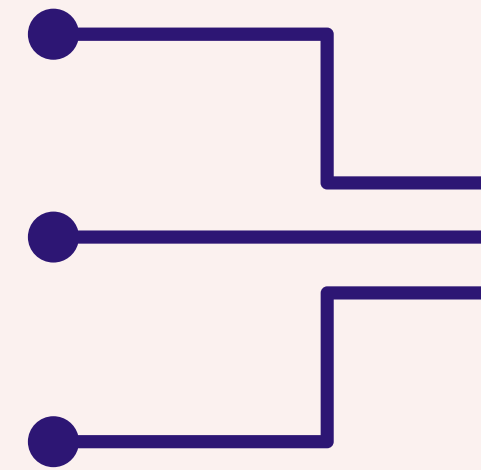
A diferencia de un clon y de una rama, un fork es un proyecto totalmente diferente que se crea a partir de otro. Por ejemplo, las distribuciones de Linux se crean a partir del mismo kernel y luego cada una se va por diferentes caminos. Cada vez que vemos que un proyecto es un fork de otro es porque se basó en lo que se había trabajado en el otro proyecto.



¿POR QUÉ GIT ES EL SISTEMA DE CONTROL DE VERSIONES MÁS USADO EN EL MUNDO?

ES DISTRIBUIDO

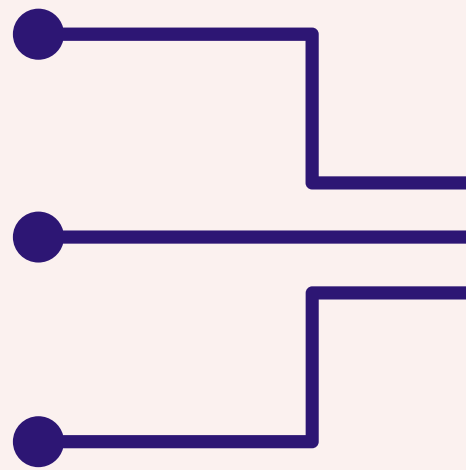
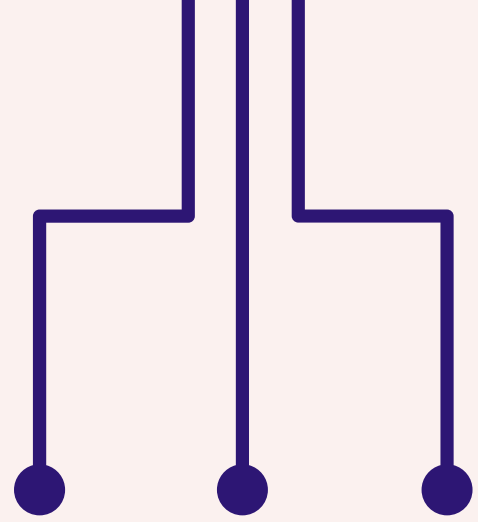
Cada uno de los desarrolladores tiene una copia idéntica del repositorio en su equipo local. No necesita conexión a un servidor local o a internet, puede trabajar en cualquier momento. Todos los involucrados tienen un backup del proyecto. Si algo llegara a ocurrir, cada uno tiene una copia del proyecto y lo mantiene a salvo.



¿POR QUÉ GIT ES EL SISTEMA DE CONTROL DE VERSIONES MÁS USADO EN EL MUNDO?

TIENE RAMAS Y FUSIONES

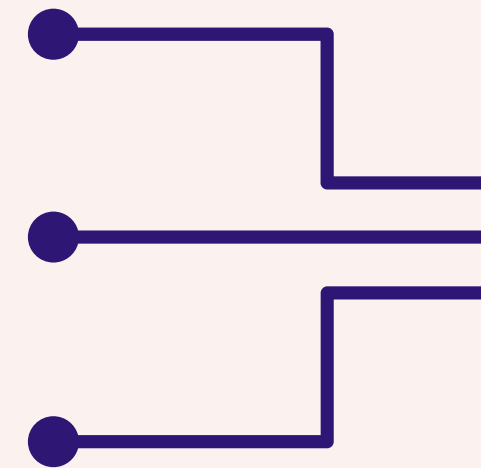
Las ramas son nuevos caminos, bifurcaciones, ramificaciones que toma el proyecto para no comprometer a la rama principal. De este modo, cada desarrollador puede crear diferentes ramas para hacer pruebas, corregir errores, agregar nuevas características. Luego estas ramas deben integrarse con la rama principal. Salen, toman su camino y luego se integran, esa es una fusión.



¿POR QUÉ GIT ES EL SISTEMA DE CONTROL DE VERSIONES MÁS USADO EN EL MUNDO?

INTEGRIDAD DE DATOS

GIT agrega una suma de comprobación a cada uno de sus archivos y de los commits, de tal manera que hay seguridad total de que cada uno de los desarrolladores tenga los mismos datos que los demás. Si hay datos errados, el proyecto podría estar en problemas.



FLUJO DE TRABAJO

Básico

el desarrollador va a trabajar en un proyecto, así que crea un repositorio.

Hay dos maneras de crearlo:

- 1.git init: Si es un repositorio nuevo, desde cero.
- 2.git clone: Si es un repositorio que ya existe.

Si llegas a trabajar en un proyecto que ya existe no vas a usar git init, sino que clonarás el repositorio en tu equipo local con git clone. Una vez hecho esto, ahora debes comenzar a mandar tus cambios al repositorio a través de commits. Cada vez que el programador hace un cambio debe enviarlos manualmente. Estos commits deben representar una funcionalidad específica. Por ejemplo: un cambio en la interfaz o la corrección de un problema.

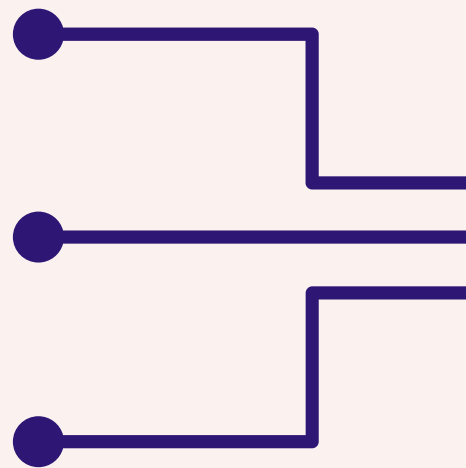
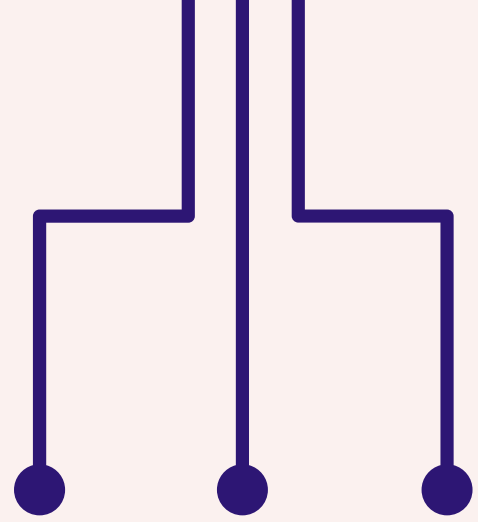
FLUJO DE TRABAJO

Los commits no se mandan directamente al repositorio. Hay una etapa intermedia llamada Stagin Area. Cuando un programador está corrigiendo un error en el código puede tocar 4 o 5 archivos, así que va tocando uno a uno y los va mandando al Stagin Area con el comando **git add**. No se mandan directamente al repositorio hasta no estar seguros de que las modificaciones se hicieron en todos los archivos necesarios. Una vez ahí, se mandan los cambios al repositorio con el comando **git commit** y se agrega un mensaje especificando el error que se está corrigiendo.

FLUJO DE TRABAJO

Trabajo en equipo

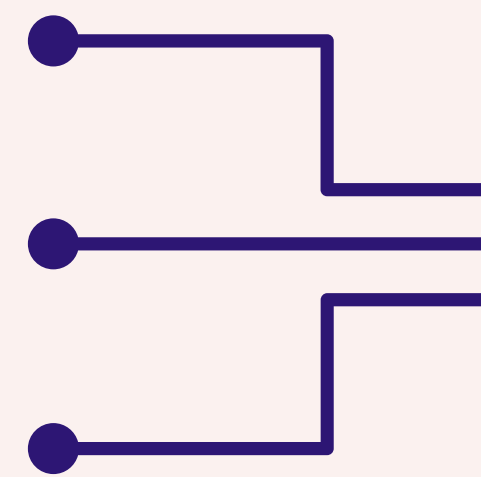
Cuando se trabaja en equipo, lo más común es tener dos ramas: una llamada Master, donde está el proyecto principal y una llamada Dev, que es donde se debe trabajar. Desde la rama Dev cada uno de los desarrolladores va a sacar nuevas ramas para comenzar a trabajar en lo que le corresponda a cada uno. Cuando el trabajo se complete, las ramas deben volver a integrarse al proyecto, hay que verificar que todo esté en orden y que nada se rompa. Esa integración se llama merge. En ese proceso puede haber conflictos y errores, por lo que es recomendable que el líder de proyecto supervise que no haya conflictos y devuelva los trabajos para que los corrijan.

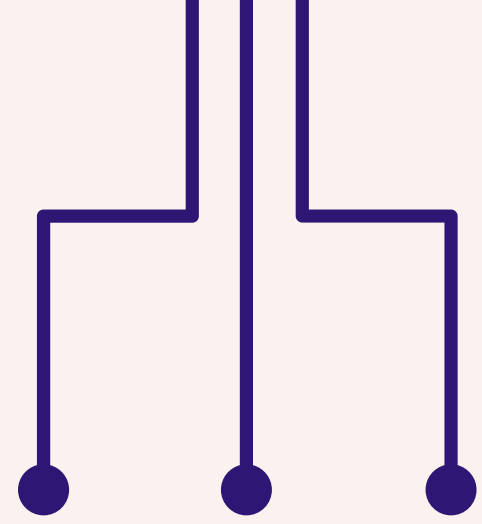




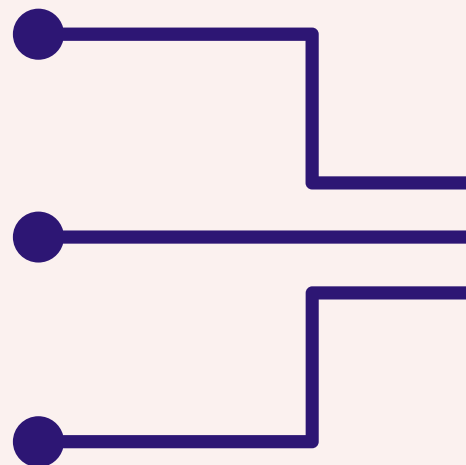
FLUJO DE TRABAJO

En flujos más avanzados existen sistemas de integración continua en donde se revisa si hay conflictos a través de tests automatizados. De haber conflictos, se les notifica a los programadores para que los corrijan y luego puedan enviar el merge. Una vez que se resuelven los conflictos, se hace merge y se integra el trabajo en Dev, también se integra Dev a Master. Master pasa a estar actualizado con todos los cambios que han hecho los programadores. Finalmente, se envía a producción, si todo va bien, funcionará.





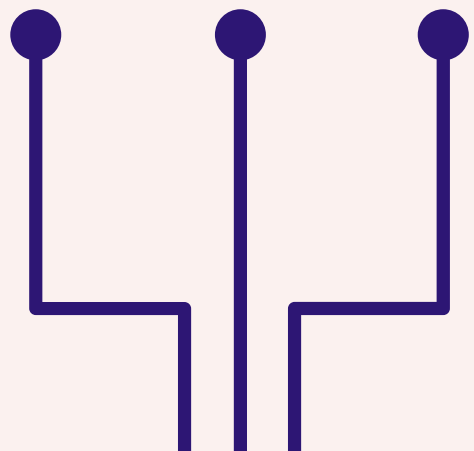
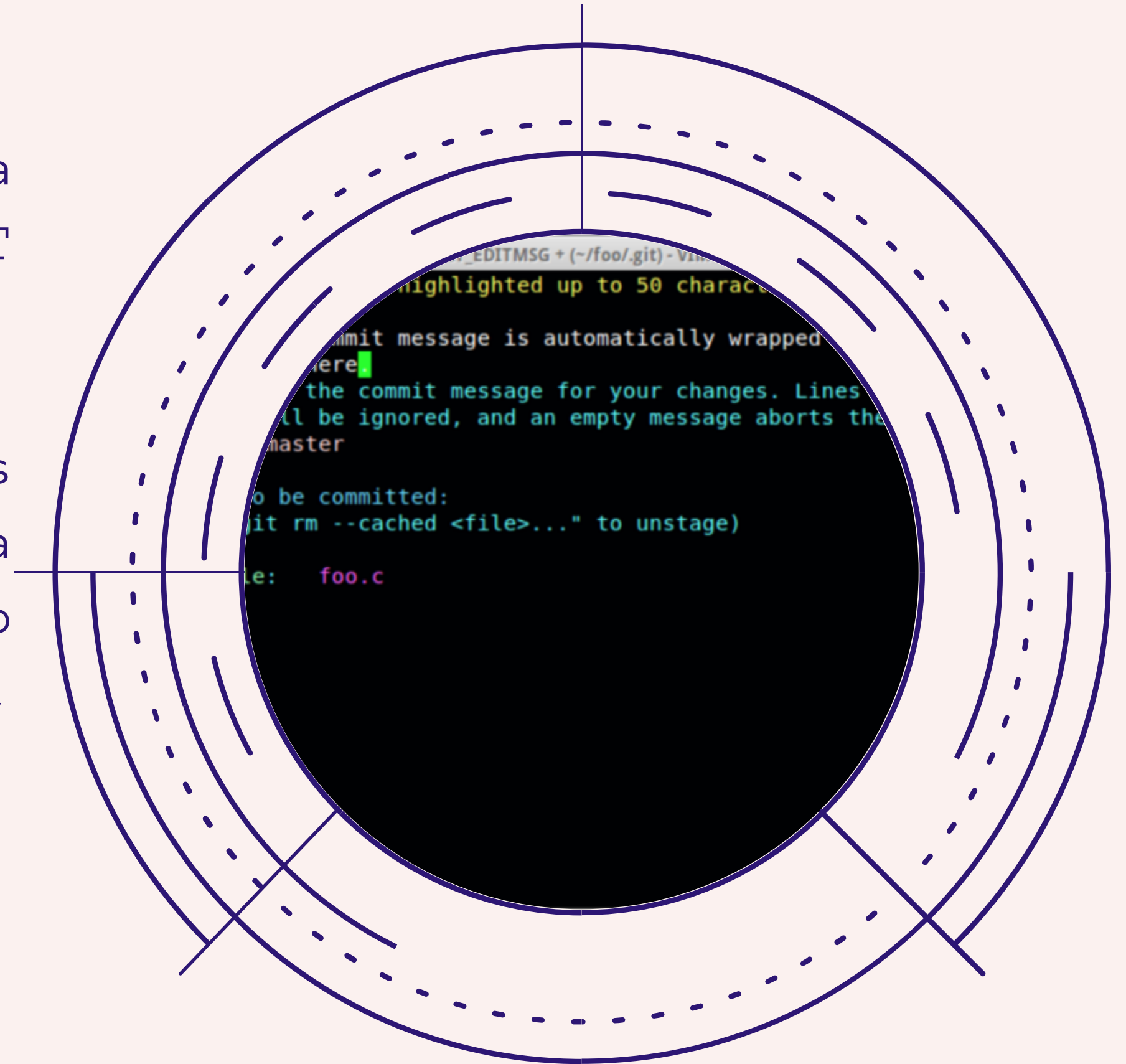
HERRAMIENTAS GIT



LÍNEA DE COMANDOS

Es la herramienta más importante para trabajar con GIT. No se aprende GIT con una interfaz gráfica.

Debes aprender sí o sí haciendo los comandos. Una vez entiendes la terminal, puedes acelerar tu trabajo con clientes gráficos como Gitkraken, SourceTree y GitHub Desktop.



REPOSITORIOS EN LA NUBE

Existen empresas que dan toda una plataforma para usar Git en la nube, estos repositorios son ramas, también llamadas ramas remotas. Por ejemplo, está la rama local **Master** y la rama remota suele llamarse **Origin**. Esta rama puede apuntar a **GitHub**, **Bitbucket** y **GitLab**. Cada empresa escoge la que mejor se adapte a su flujo de trabajo.

