GitFlow

Flujos de trabajo colaborativo con Git y comandos respectivos.

¿Qué es un GitFlow?

¿Qué es un GitFlow?

Es la manera en la que el equipo de desarrollo va a utilizar Git para poder trabajar de manera colaborativa.

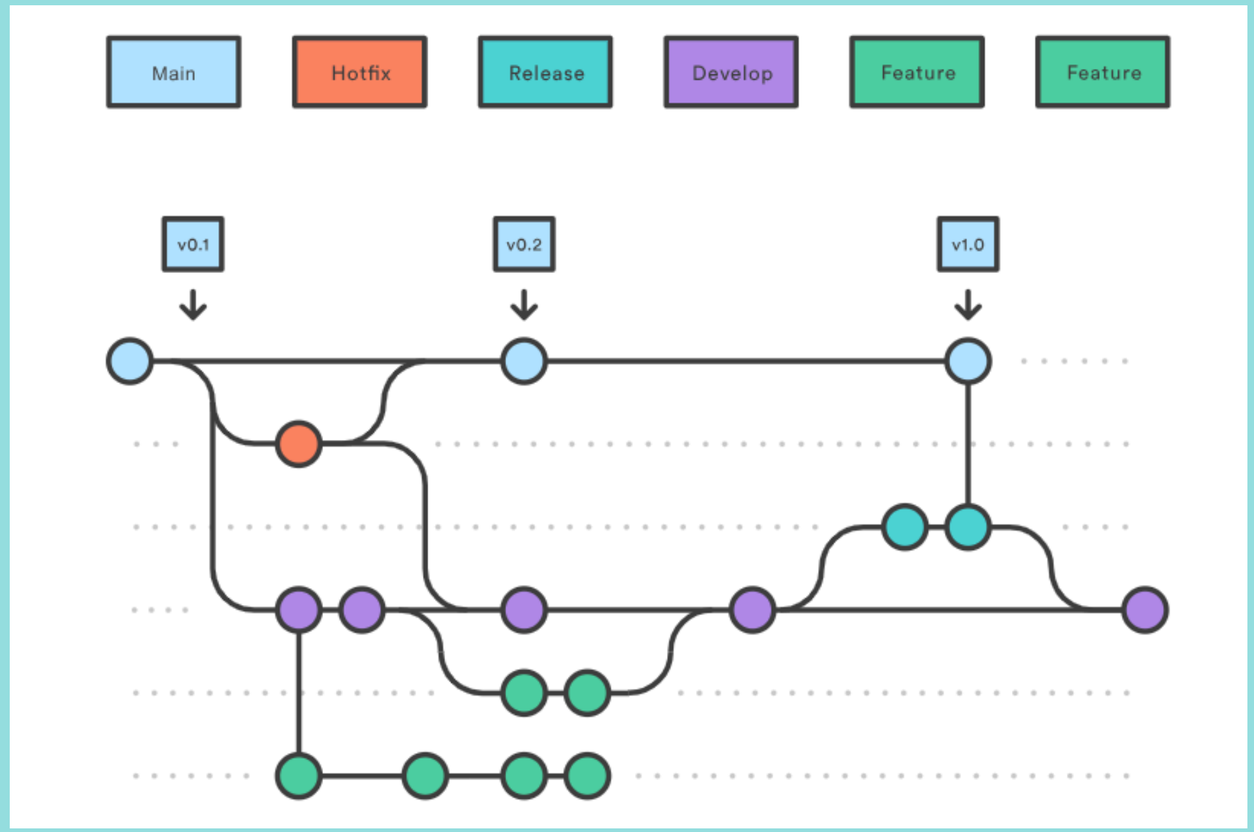
Siguiendo ciertas pautas que promueve el GitFlow

Git Flow

Es el flujo de trabajo más antiguo, utiliza las ramas:

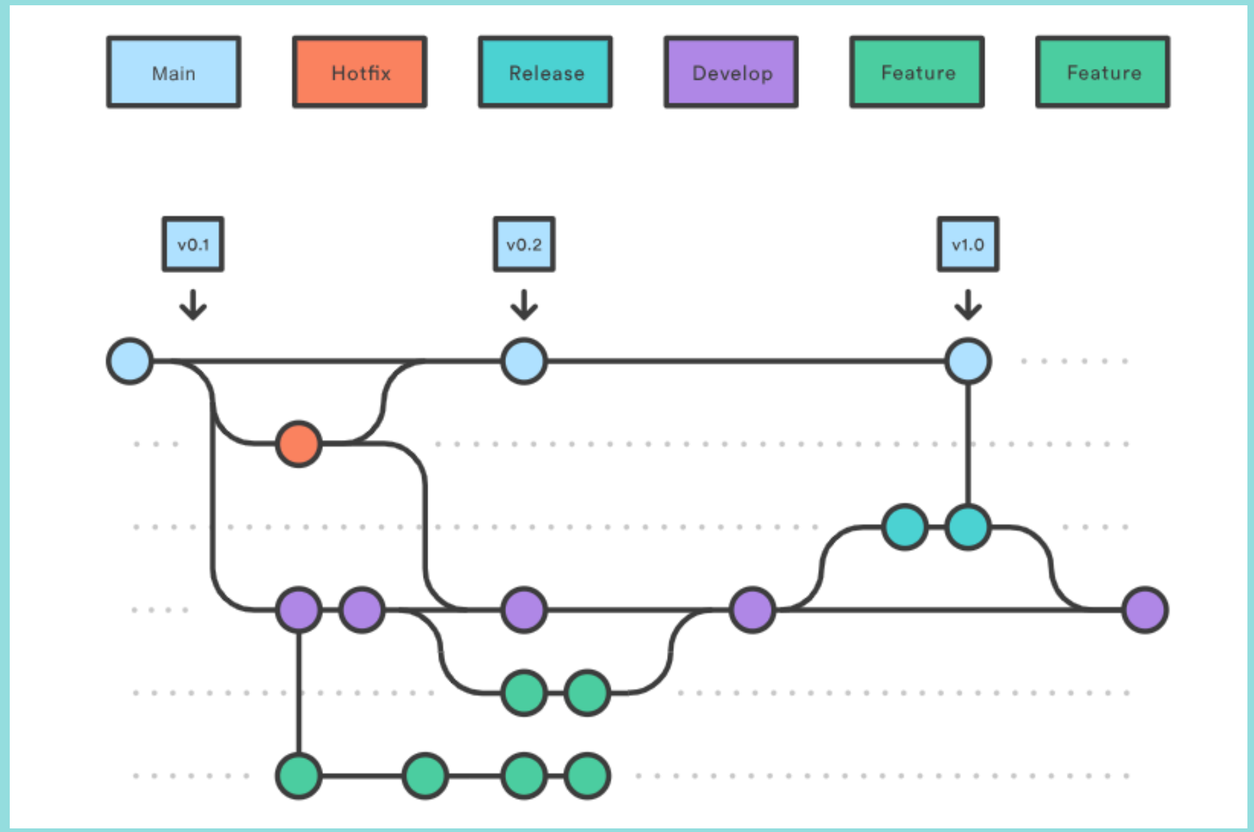
1. main (o master): Contener el código de producción.

2. develop: Código de pre-producción que todavía tienen que ser probadas y validadas



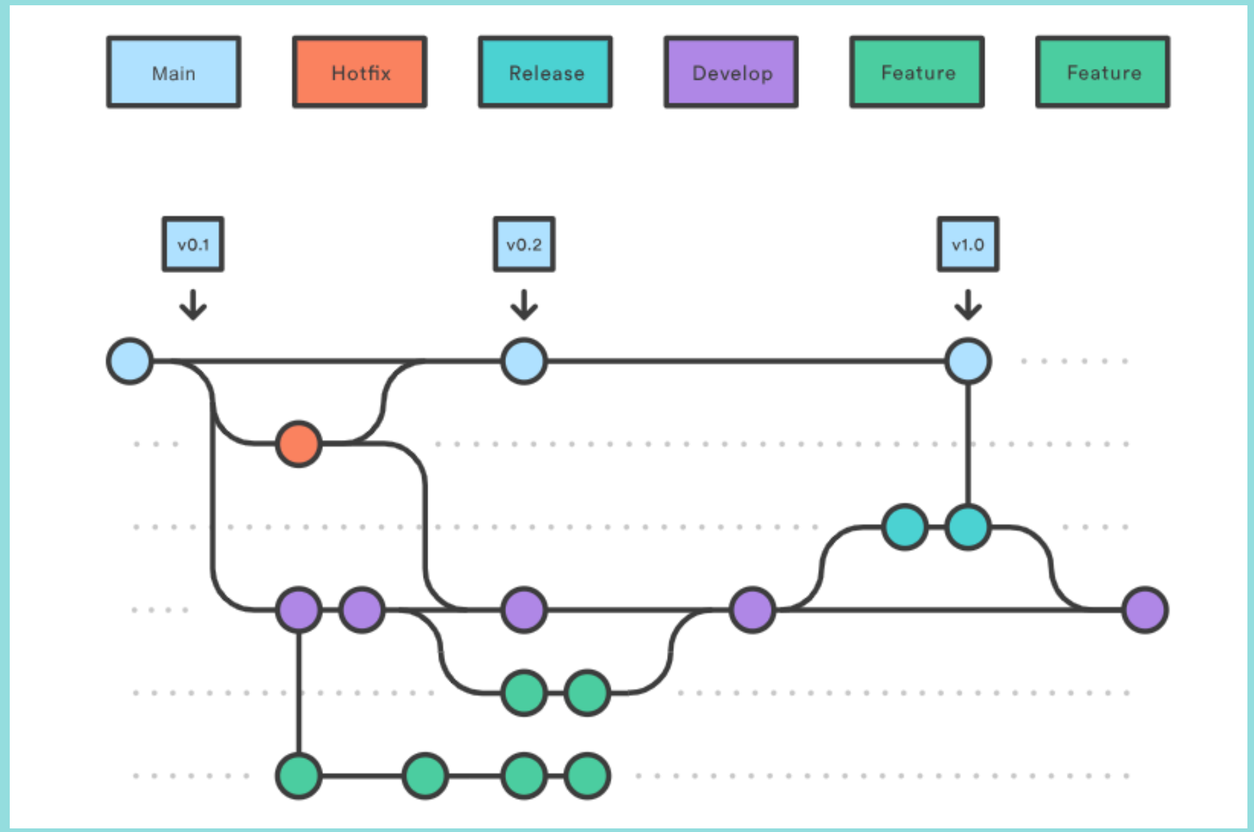
Git Flow

3. Feature: Características nuevas para el proyecto.
4. Release: Cambios de último momento
5. Hotfix: Parches o arreglar bugs pequeños



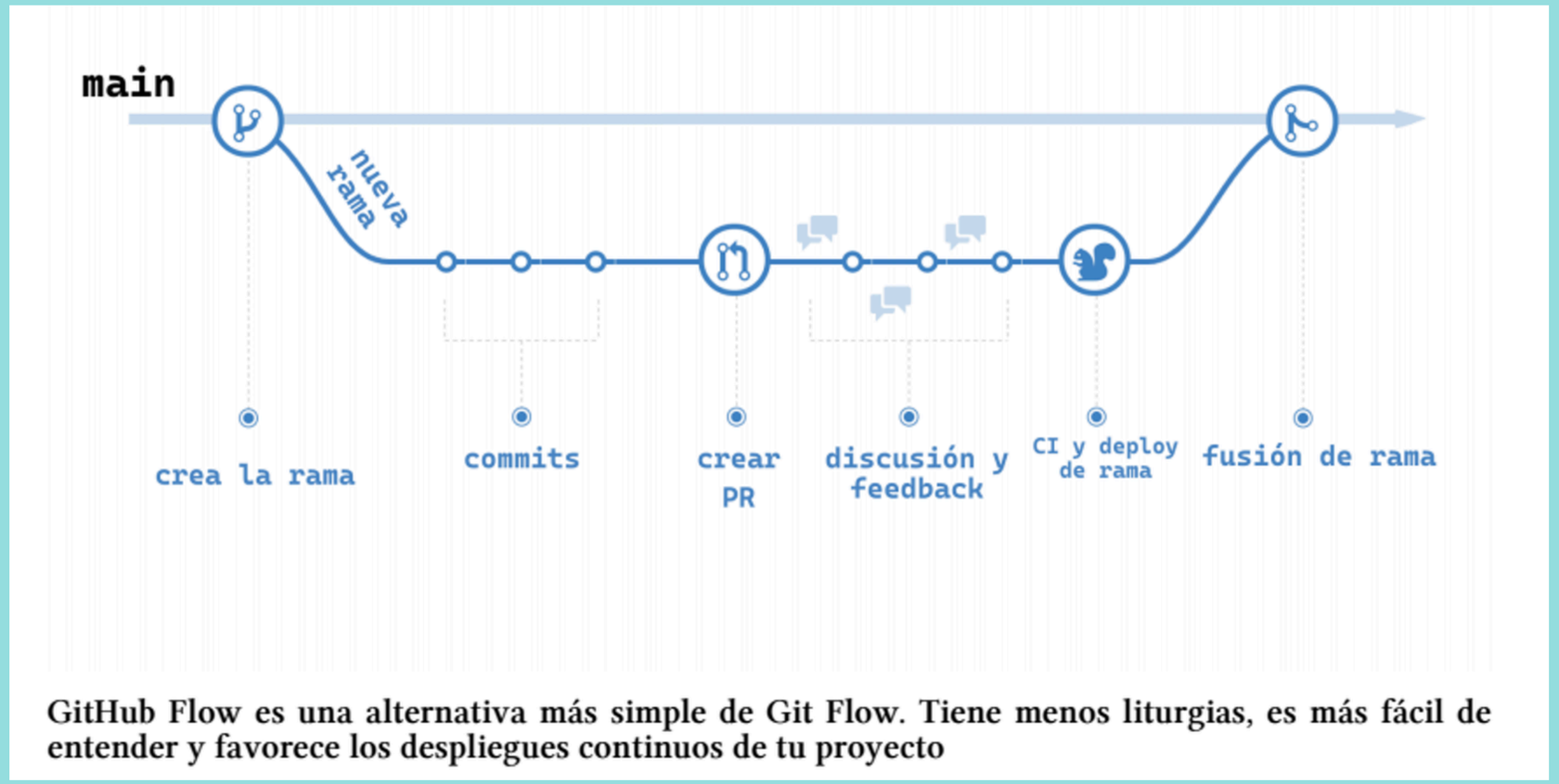
Git Flow

https://github.com/OpherV/gitflow4idea/branches/all



GitHub Flow

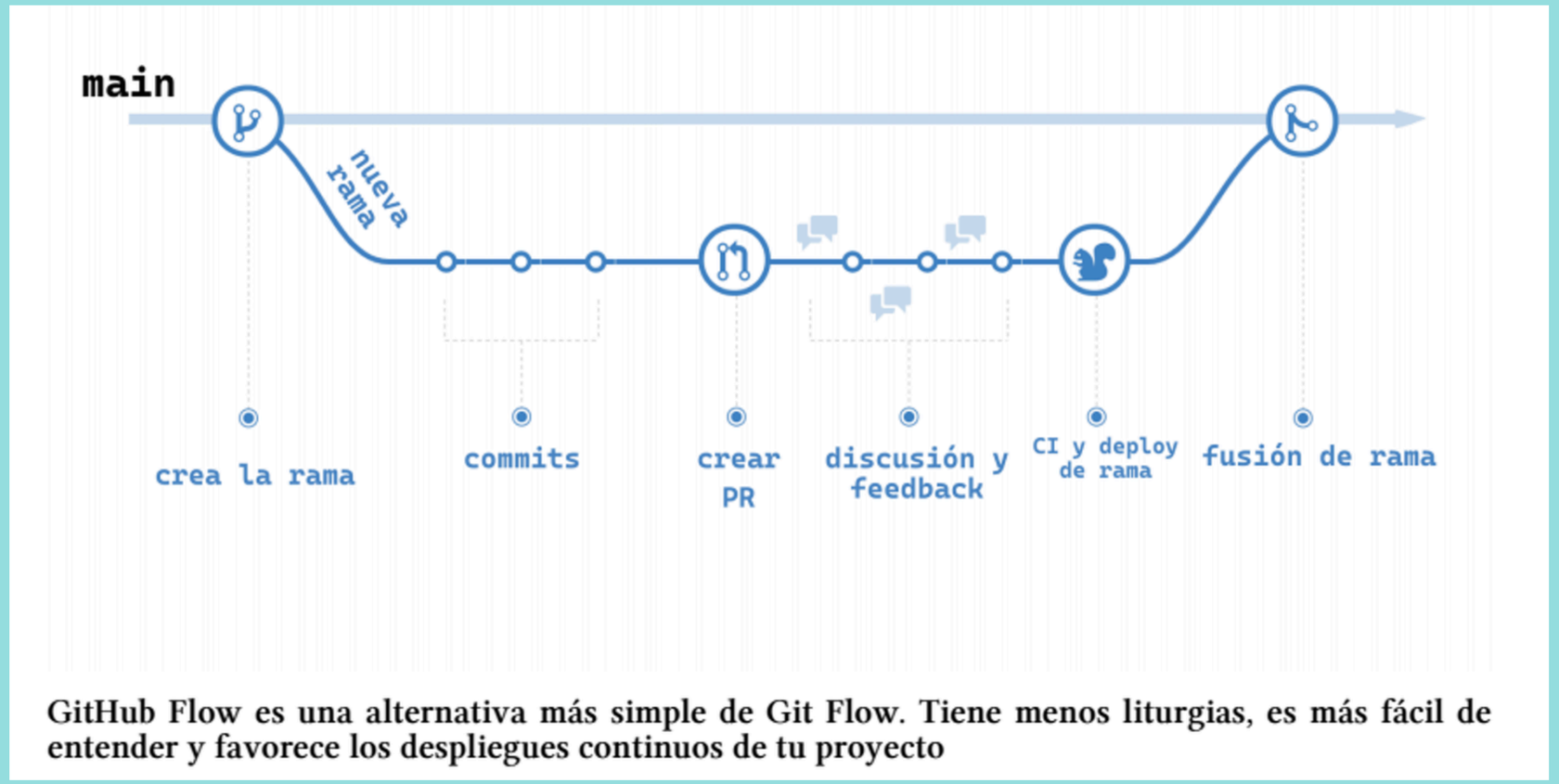
Rama **main** y cualquier otra rama que quiera ser integrada por medio de una Pull Request



GitHub Flow es una alternativa más simple de Git Flow. Tiene menos liturgias, es más fácil de entender y favorece los despliegues continuos de tu proyecto

GitHub Flow

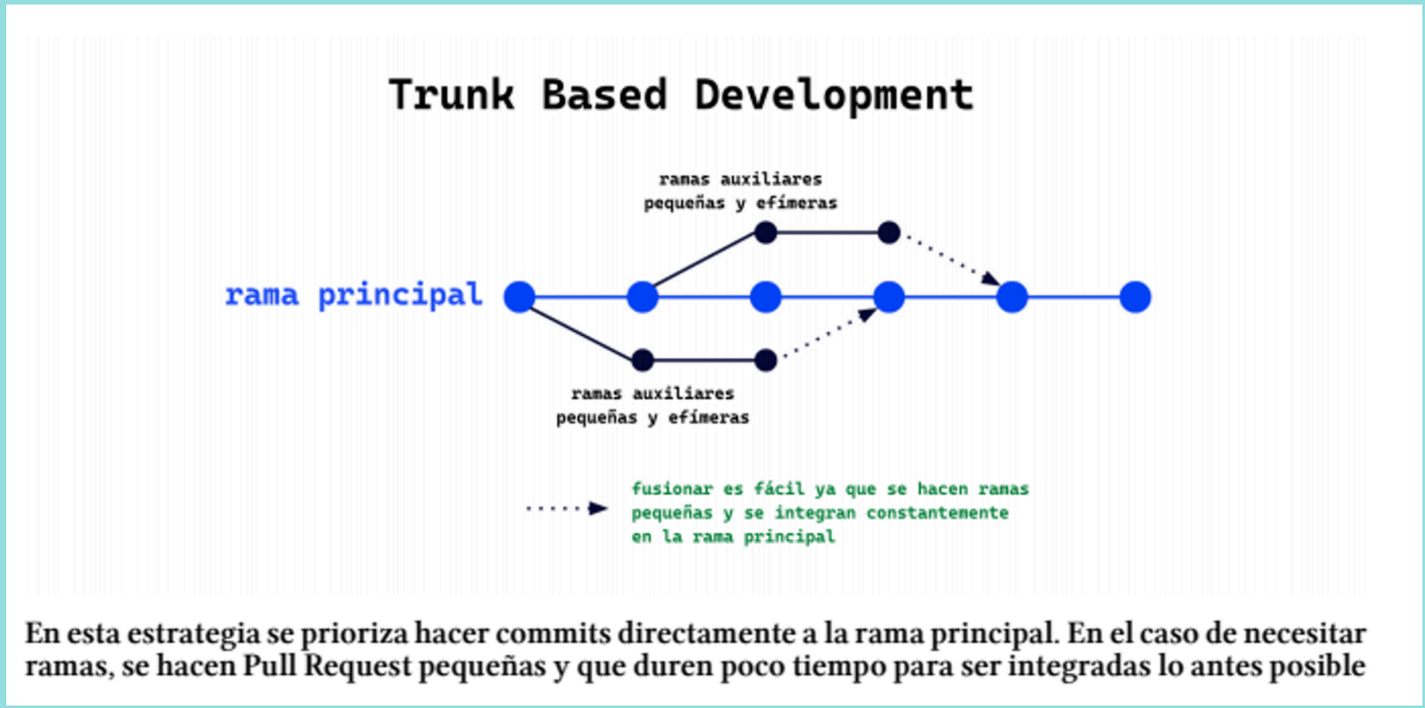
https://github.com/atmos/heaven



GitHub Flow es una alternativa más simple de Git Flow. Tiene menos liturgias, es más fácil de entender y favorece los despliegues continuos de tu proyecto

Trunk Based Development

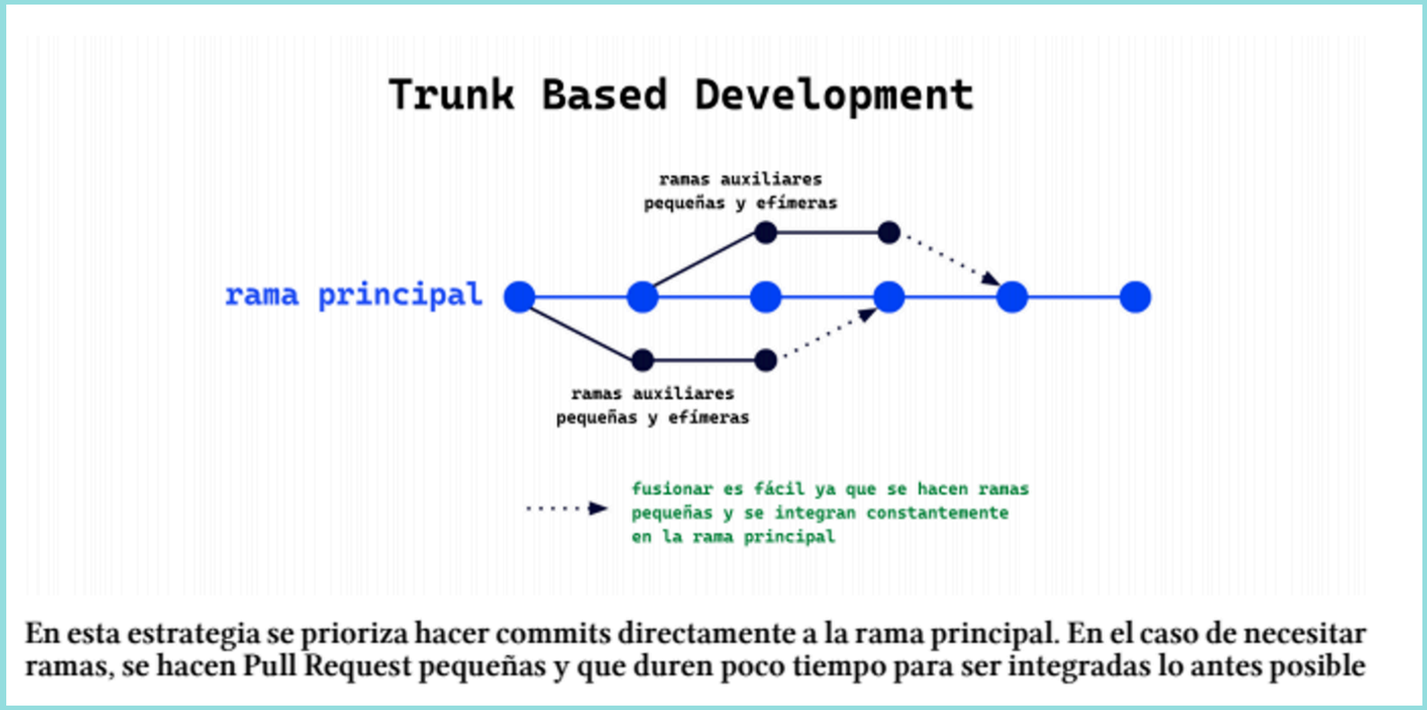
Solo la **rama main** y ramas auxiliares efímeras que quiera ser integrada por medio de una Pull Request.
Es útil si contamos con un buen sistema CI/CD



En esta estrategia se prioriza hacer commits directamente a la rama principal. En el caso de necesitar ramas, se hacen Pull Request pequeñas y que duren poco tiempo para ser integradas lo antes posible

Trunk Based Development

https://github.com/santiagxf/trunkbased-mlops

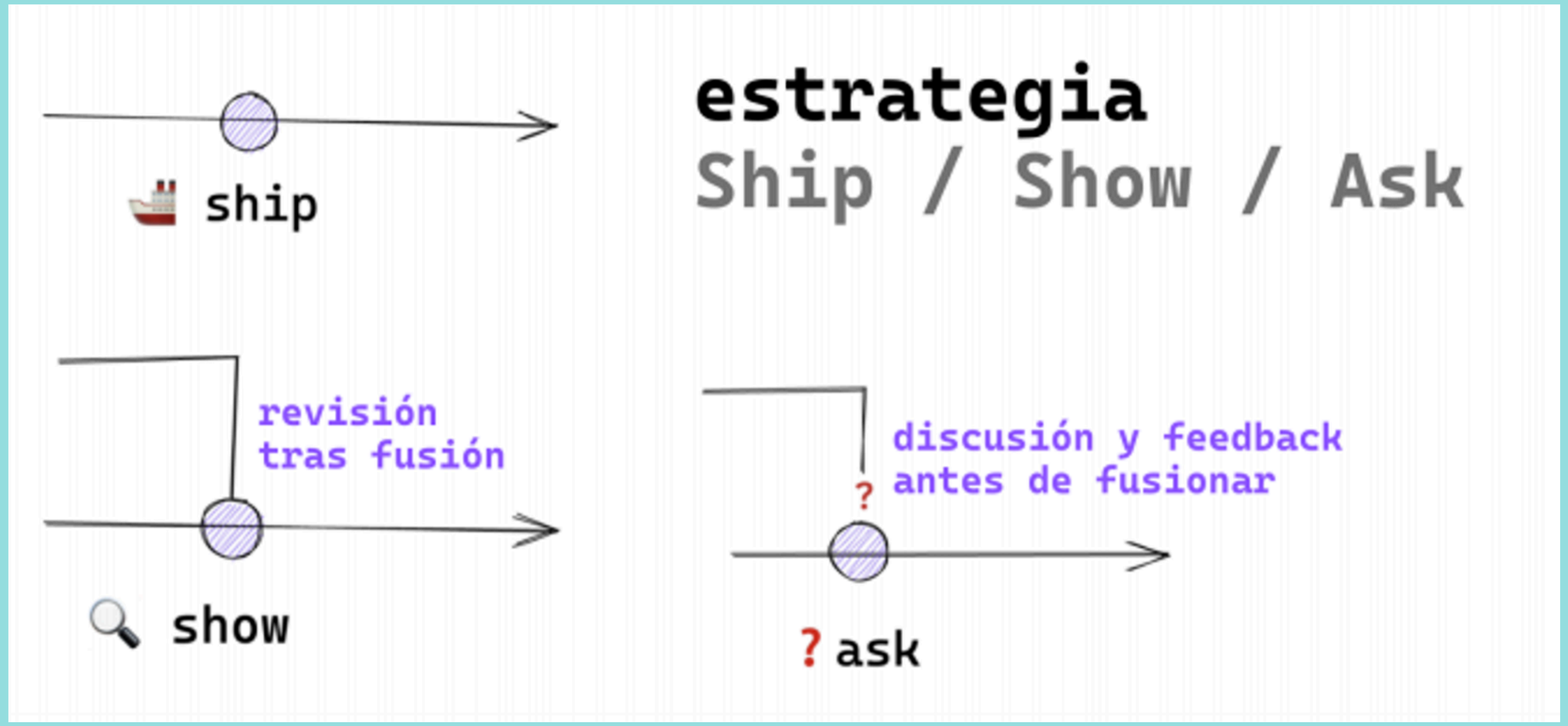


En esta estrategia se prioriza hacer commits directamente a la rama principal. En el caso de necesitar ramas, se hacen Pull Request pequeñas y que duren poco tiempo para ser integradas lo antes posible

Ship / Show / Ask

1. **Ship:** Se fusiona en la rama principal sin revisión.
2. **Show:** Abre una petición de cambios para que sean revisados por CI pero se fusiona inmediatamente.
3. **Ask:** Abre una PR para discutir los cambios antes de fusionarlos.

Ship / Show / Ask



Las reglas de Ship / Show / Ask

1. Tenemos un buen sistema de CI/CD.
2. Confiamos en el equipo y existen buenas prácticas de desarrollo. Pair programming, mob programming, seniority… y, sobretodo, existe responsabilidad. La persona se responsabiliza de decidir la categoría de su cambio. ***Un gran poder, poder hacer merge de tus propias Pull Request, conlleva una gran responsabilidad (no romper producción).***
3. Las revisiones de código no son requerimientos
4. Las ramas son lo más pequeñas posibles, tienen un tiempo de vida corto y siempre salen directamente desde la rama principal.
5. El equipo ha sabido lidiar con el ego individual, las personas confían en el resto del equipo y las pruebas automáticas pasan.

Ejemplos Ship/Show/Ask

Show	https://github.com/CodeMasterChefs/Backend-Ruta_del_Programador/pull/3
Ask	https://github.com/CodeMasterChefs/Backend-Ruta_del_Programador/pull/34/commits/1392ca4147fb4d461681f065b90958f09eccb759