

Git y personas

Trabajar en equipo con git

Jesús López de la Cruz

Tech Lead - QaShops (Vente Privee group)

@jeslopclu

jesuslc.com

¿Quién utiliza Git?

¿De qué vamos a hablar?

- ¿Qué es git?
- A tener en cuenta
- Estrategias de branching
- Lecciones aprendidas
- Reflexiones

¿Qué es git?

¿Qué es git?

Es un software de **control de versiones** diseñado por Linus Torvalds, que te ayuda a llevar el registro de los cambios en el código fuente.

5 comandos básicos de git

- git checkout
- git commit
- git pull
- git merge
- git push

5 comandos básicos de git y las bases para trabajar en equipo

- git checkout
- git commit
- git pull
- git merge
- git push
- Comunicación
- Confianza
- Simplicidad

¿Cómo incorporamos cambios al repositorio?

Pull request

Solicitar al equipo
que revise el código
antes de incorporarlo

¿Cómo incorporamos cambios al repositorio?

Pull request

Solicitar al equipo que revise el código antes de incorporarlo

vs

Pair programming

Programación por parejas, dos personas trabajan en el mismo desarrollo

¿Cómo trabajamos con ramas?

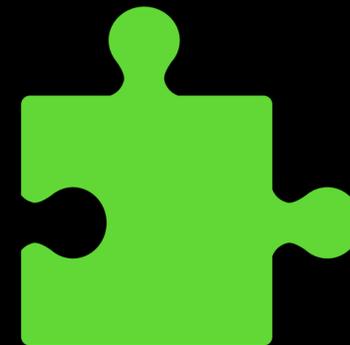
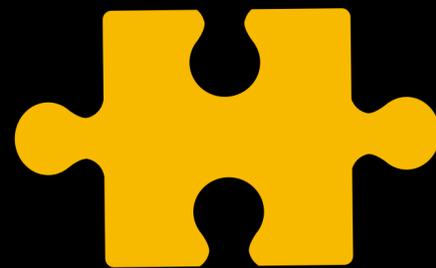
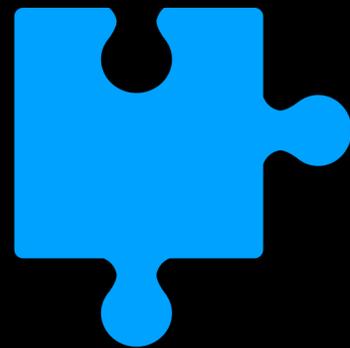
- ¿Solo una rama?
- ¿Varias ramas?
- ¿Cómo afrontar los bugs en producción?

OBJETIVO

Aportar valor al negocio a través del código minimizando el riesgo

Cómo elegir bien una estrategia de ramas

Puzzle



Cómo elegir bien una estrategia de ramas

Puzzle

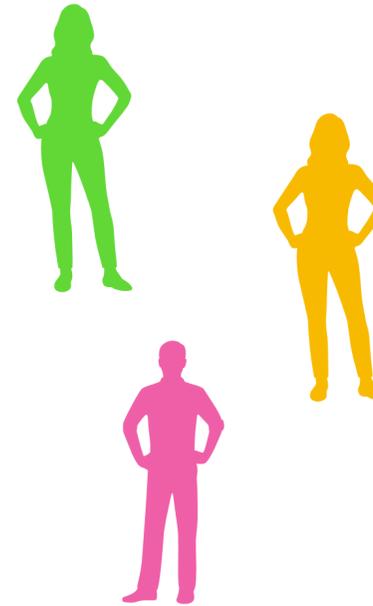


El equipo

 3 ó 10 personas

 ¿distribuido?

 ¿Madurez?



El tipo de proyecto

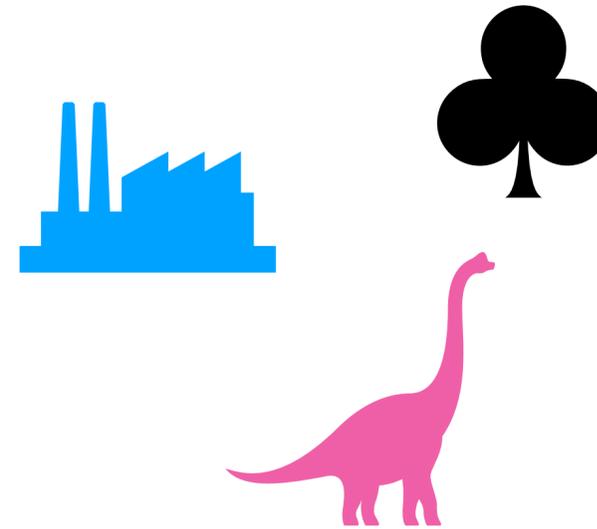
 ¿aplicación web?

 ¿android? ¿iOS?



La compañía

-  Historias de usuario
-  ¿Validación/QA?
-  Agilidad en los procesos



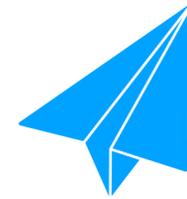
El desarrollo

 Tests

 Integración continua

 Entornos

 Automatización



¿Cómo lo hacemos?



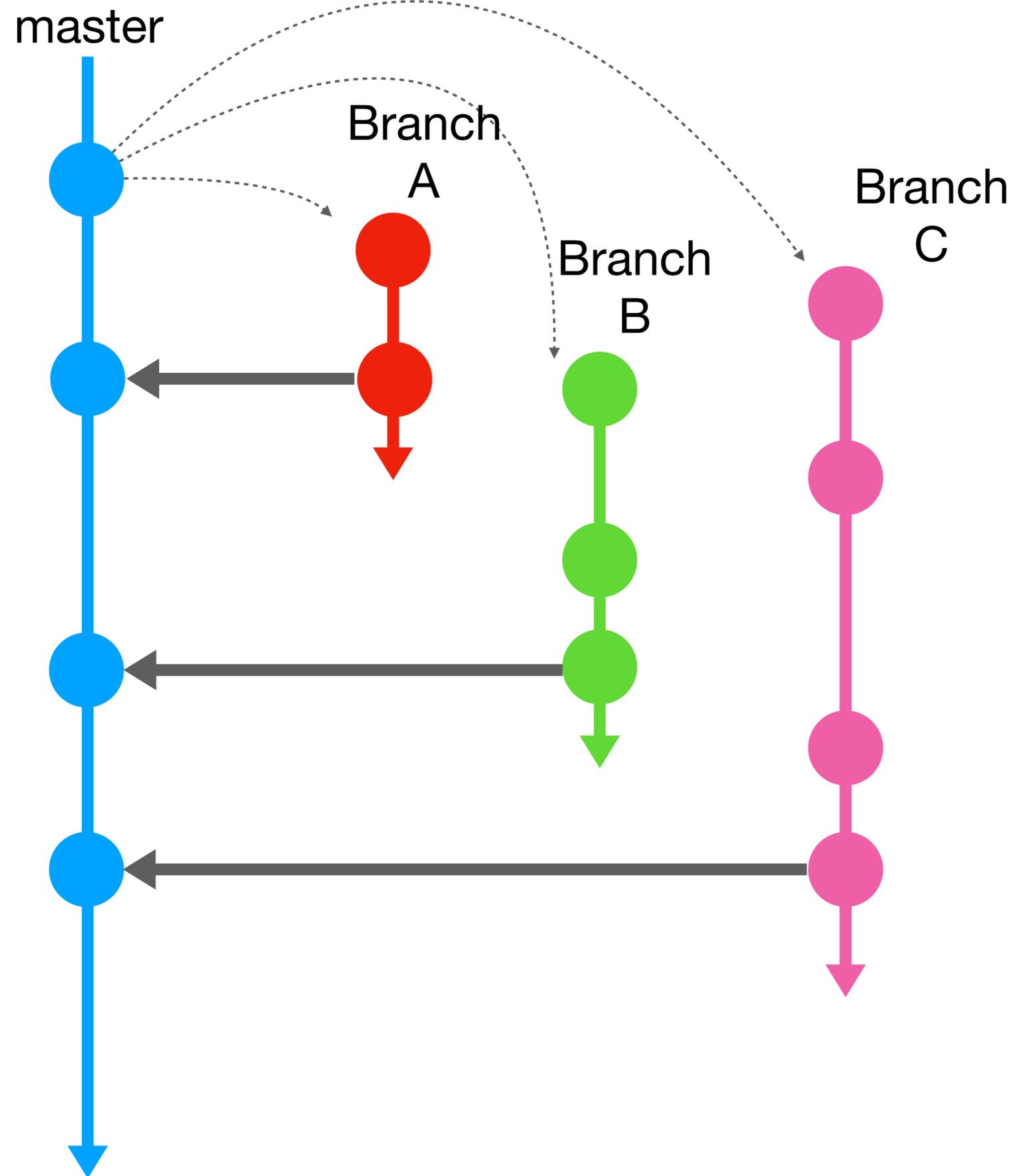
Estrategias de branching

Mainline branch

Simplicidad

Mainline branch

simplicidad



Mainline branch

simplicidad

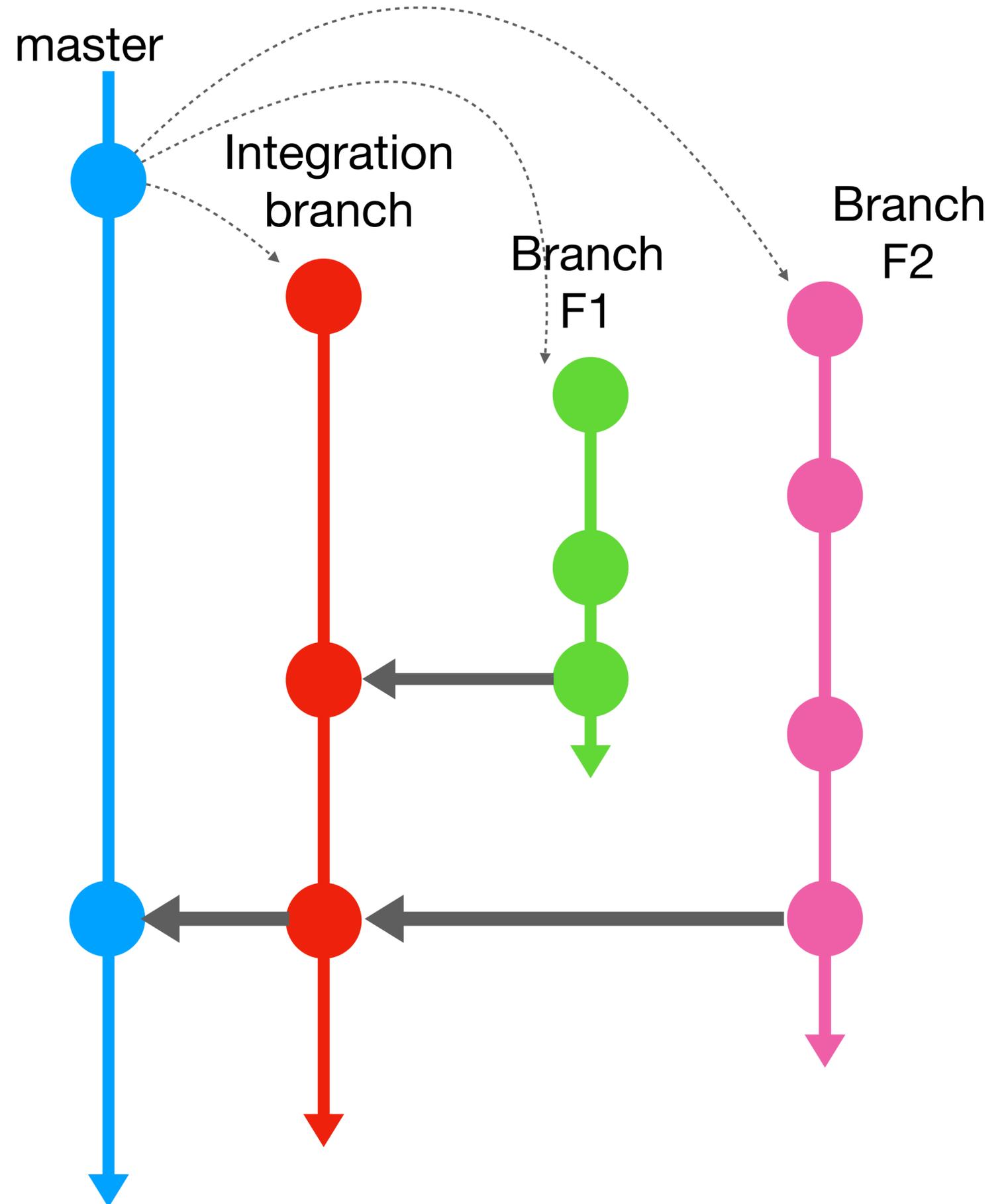
- ✓ Flujo de trabajo simple
- ✓ Listo para desplegar
- ✗ Cuidado con las ramas “largas”
- ✗ Si no hay tests es arriesgado

Branch per feature

Master nunca está roto

Branch per feature

Master nunca está roto



Branch per feature

Master nunca está roto

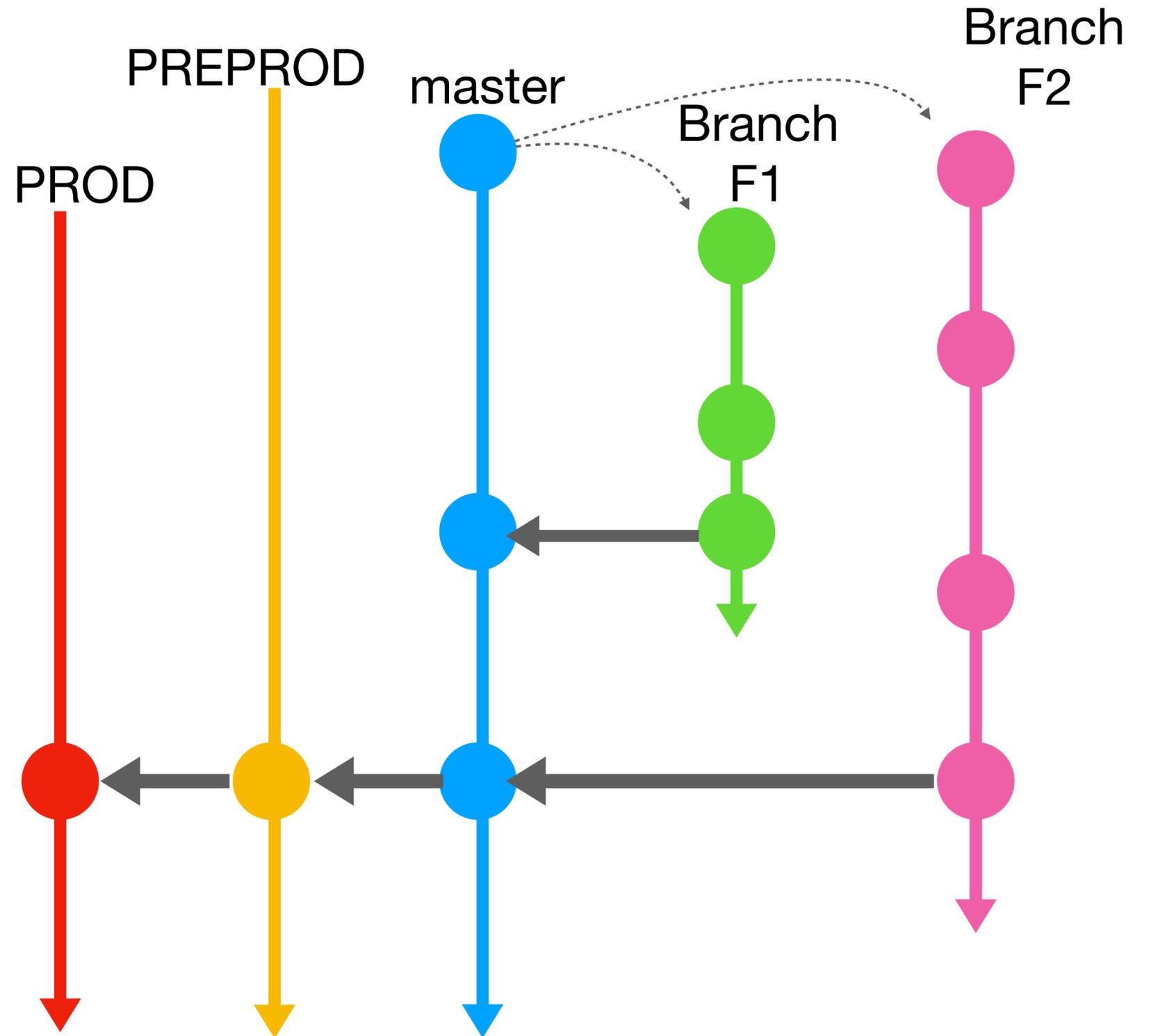
- ✓ Master nunca está “roto”
- ✓ Tenemos una rama para comparar
- ✗ Cuidado con las ramas “largas”
- ✗ Si hay un rollback se pierde todo el valor entregado

Environment based

Revisar antes de subir

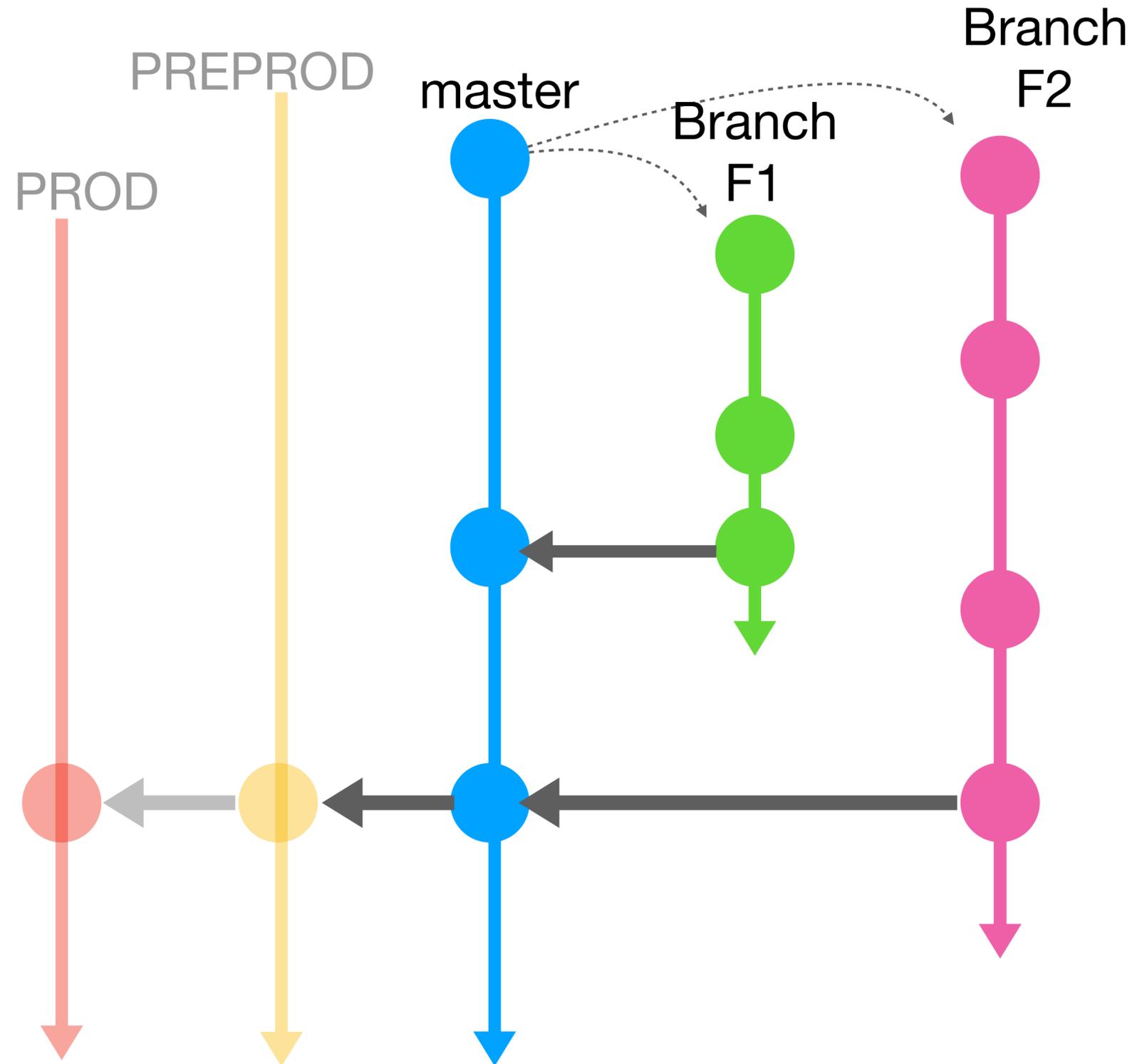
Environment based

Revisar antes de subir



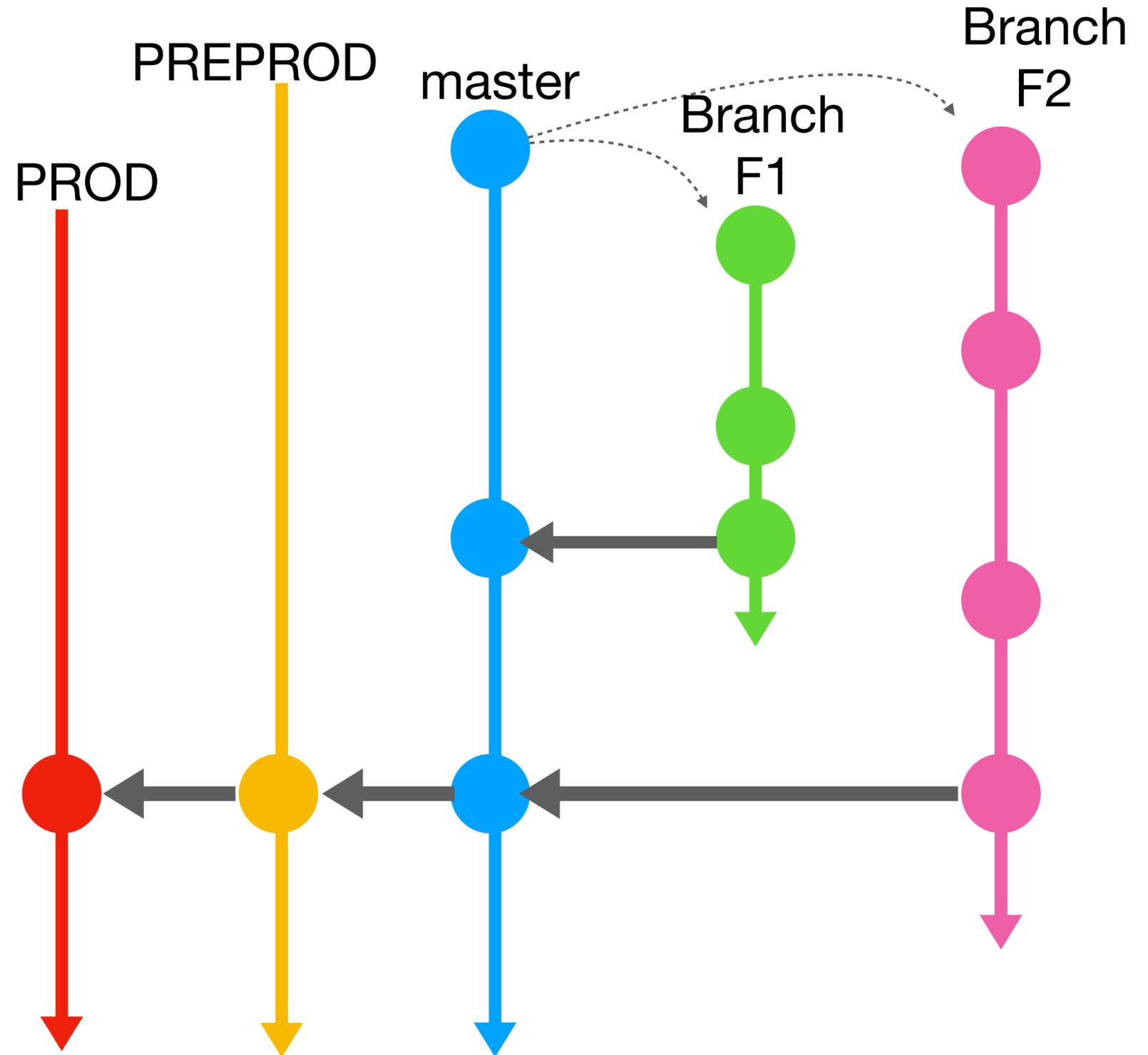
Environment based

Revisar antes de subir



Environment based

Revisar antes de subir



Environment based

Revisar antes de subir

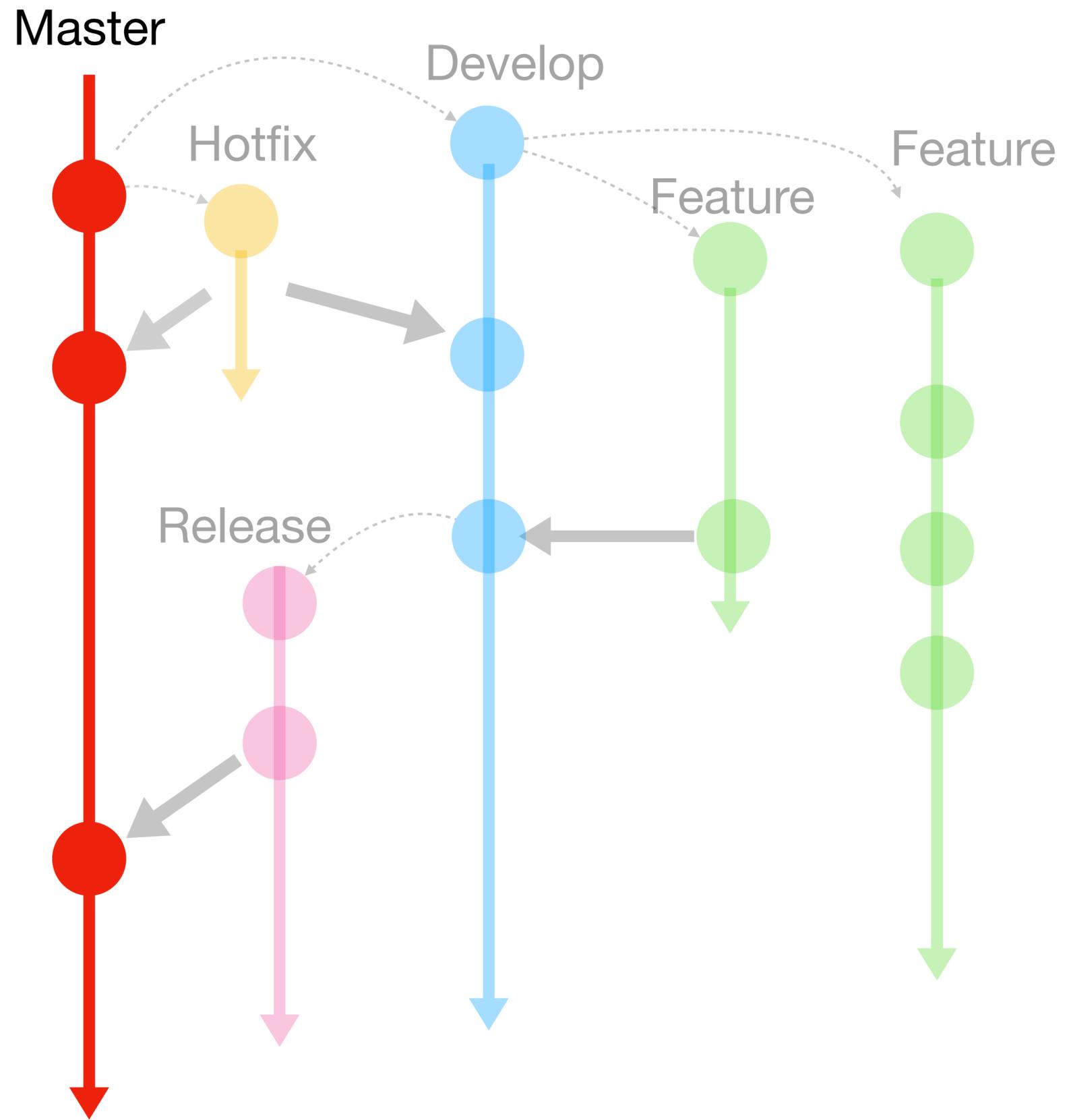
- ✓ Master nunca está “roto”
- ✓ Probar con distintas configuraciones
- ✗ Cuidado con las ramas “largas”
- ✗ Si hay un rollback se pierde todo el valor entregado
- ✗ Automatizar el proceso de despliegue

git flow

El que todo el mundo conoce

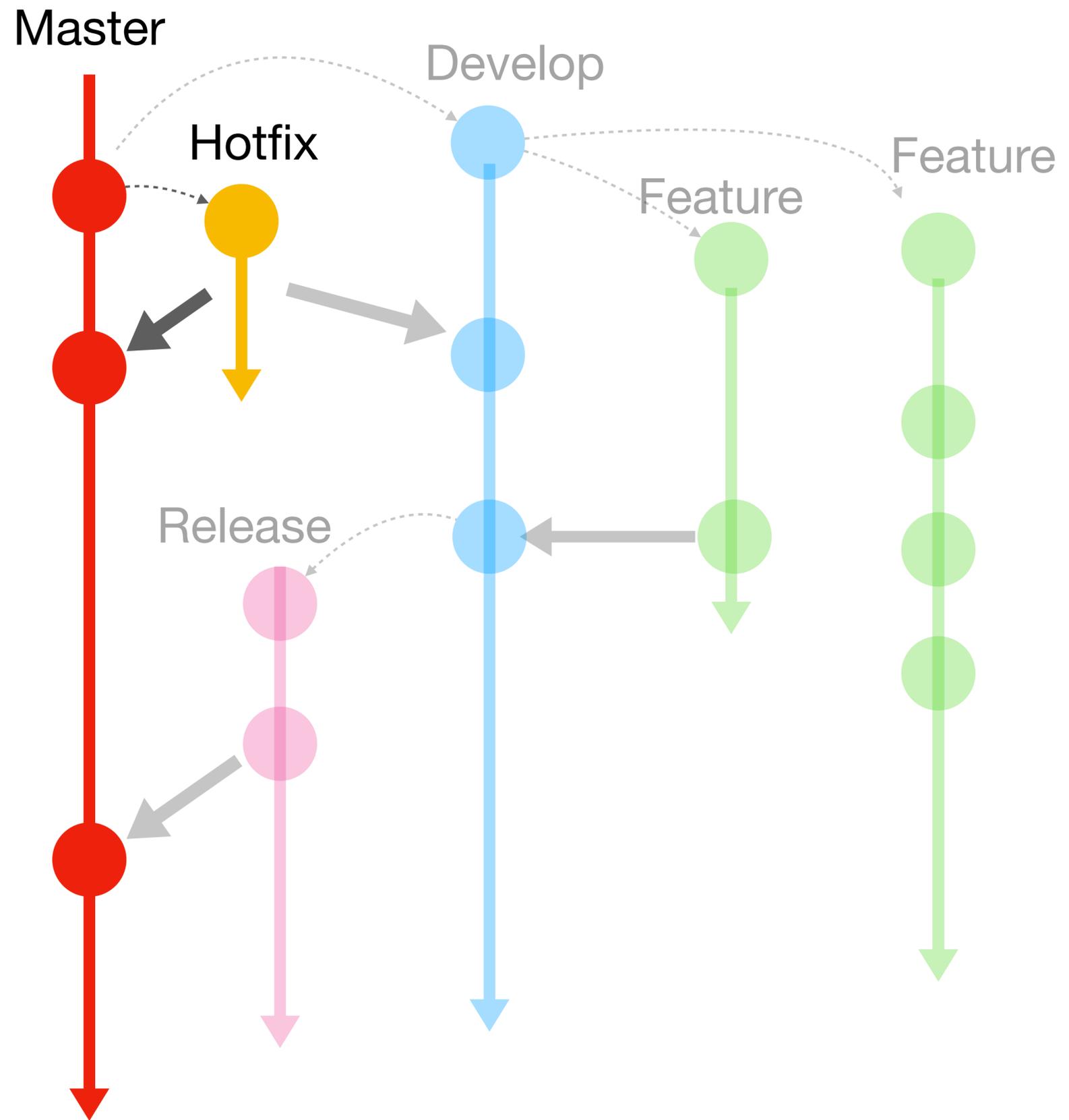
git flow

El que todo el mundo conoce



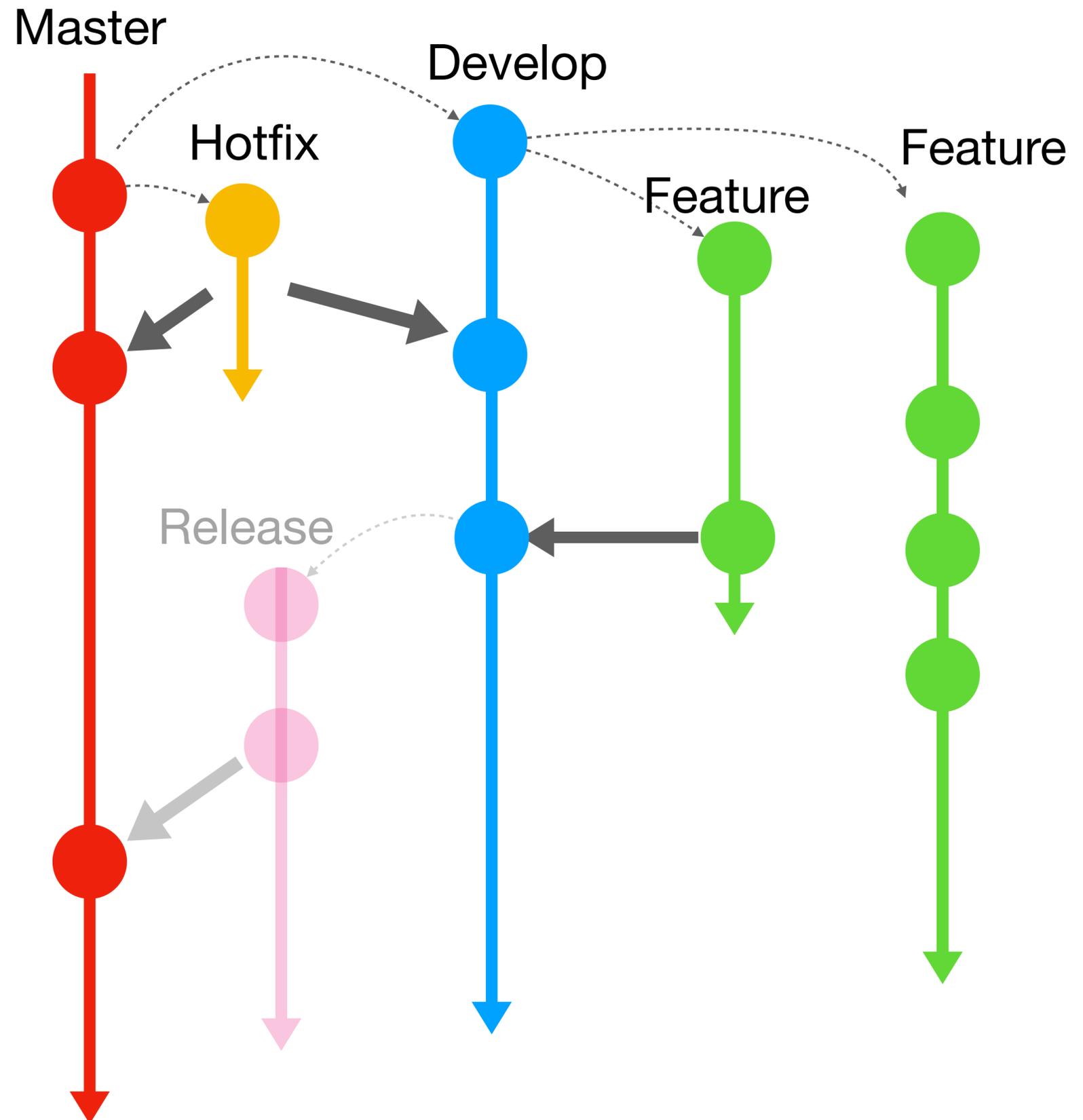
git flow

El que todo el mundo conoce



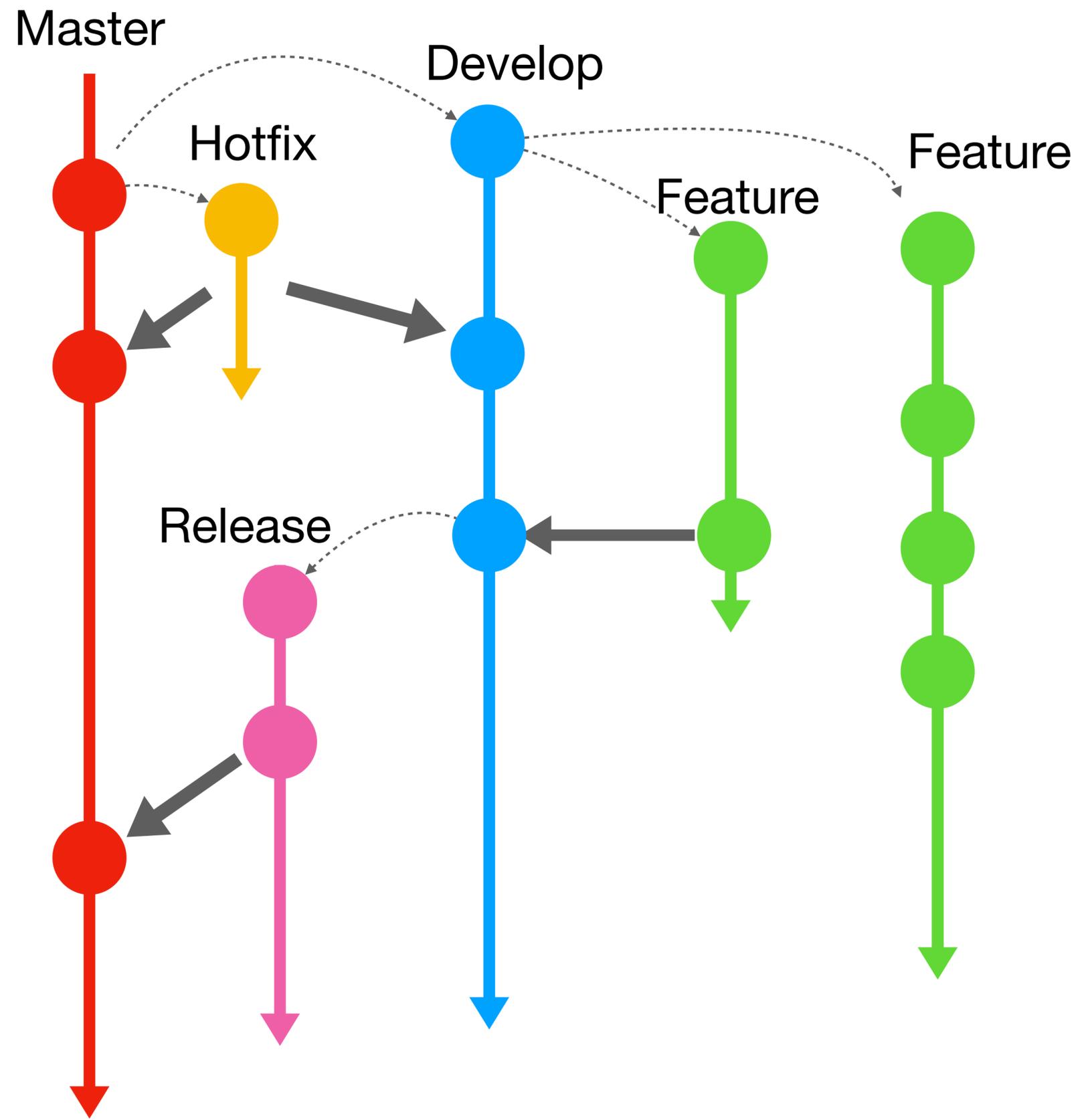
git flow

El que todo el mundo conoce



git flow

El que todo el mundo conoce



git flow

El que todo el mundo conoce

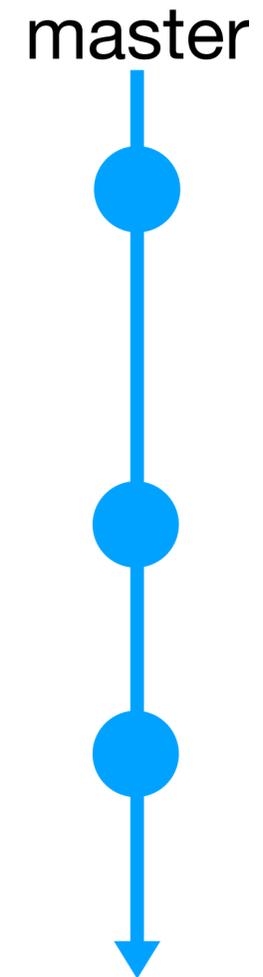
- ✓ Mucha documentación
- ✓ Ramas ordenadas
- ✗ Cuidado con las ramas “largas”
- ✗ Si hay un rollback se pierde todo el valor entregado
- ✗ Complejo para equipos pequeños

Trunk based

Las ramas son para los monos

Trunk based

Las ramas son
para los monos



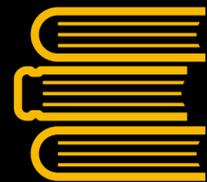
Trunk based

Las ramas son
para los monos

- ✓ Simple
- ✓ Estar al día de los cambios
- ✗ Equipos distribuidos
- ✗ Es necesario tener **confianza**: test, automatización y profesionalidad

Lecciones aprendidas

KISS - Keep It Simple, Stupid!



Lecciones aprendidas

 **Flujo simple**

Lecciones aprendidas

Flujo simple

No florituras

Ramas pequeñas

Feature toggles /
branch by abstraction

Lecciones aprendidas

 Flujo simple

 **Comunicación, personas y confianza**

Lecciones aprendidas

 Flujo simple

 **Comunicación, personas y confianza**

Diagrama de flujo

Historias de
usuario

Profesionalidad

Lecciones aprendidas

 Flujo simple

 Comunicación, personas y confianza

 **Testing y automatización**

Lecciones aprendidas

 Flujo simple

 Comunicación, personas y confianza

 **Testing y automatización**

Tests

Scripts

Hooks

Lecciones aprendidas

 Flujo simple

 Comunicación, personas y confianza

 Testing y automatización

 **Deploy / Rollback**

Reflexiones

Para pensar



Reflexiones

- ¿Hay confianza en el equipo?: tests, proponer cambios, historias de usuario,...

Reflexiones

- ¿Hay confianza en el equipo?: tests, proponer cambios, desplegar,...
- ¿Hay code reviews? ¿pair programming? ¿Las code reviews se encolan? ¿code review a alguien sentado a 2 sillas de distancia?

Reflexiones

- ¿Hay confianza en el equipo?: tests, proponer cambios, desplegar,...
- ¿Hay code reviews? ¿pair programming? ¿Las code reviews se encolan? ¿code review a alguien sentado a 2 sillas de distancia?
- ¿Hay competiciones por no ser el último en hacer merge y así tener pocos conflictos?

Referencias

- <http://www.javacodegeeks.com/2015/11/git-branching>
- <https://gist.github.com/jbenet/ee6c9ac48068889b0912>
- <http://dymitruk.com/blog/2012/02/05/branch-per-feature/>
- <http://www.nomachetejuggling.com/2017/04/09/a-different-branching-strategy>
- <https://nvie.com/posts/a-successful-git-branching-model/>
- <https://trunkbaseddevelopment.com/>
- <https://about.gitlab.com/2016/07/27/the-11-rules-of-gitlab-flow/>
- <https://martinfowler.com/bliki/BranchByAbstraction.html>

Git y personas: trabajar en equipo con git

Gracias

Jesús López de la Cruz

jesuslc.com

Twitter: @jeslopclu

Email: jeslopclu@gmail.com