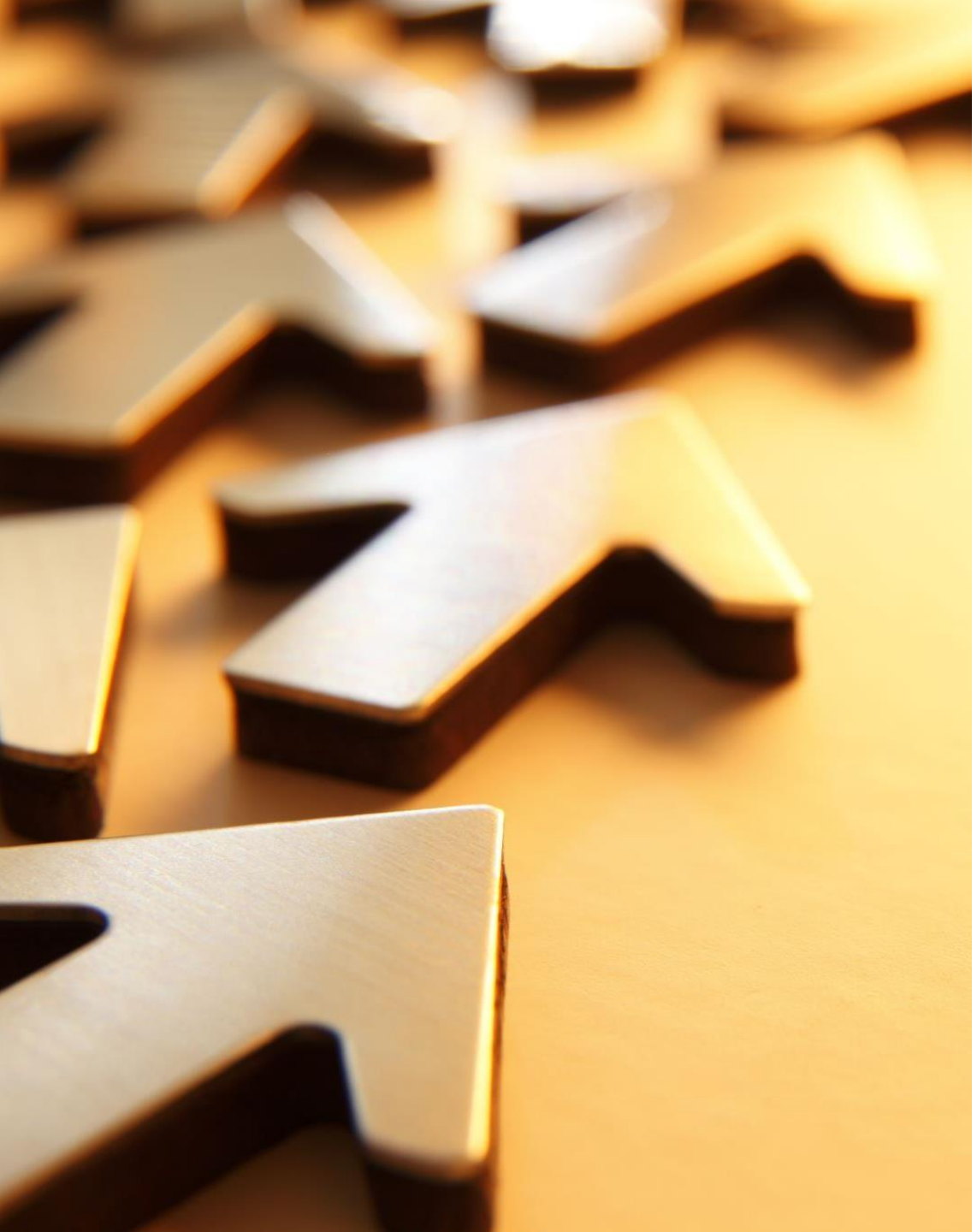
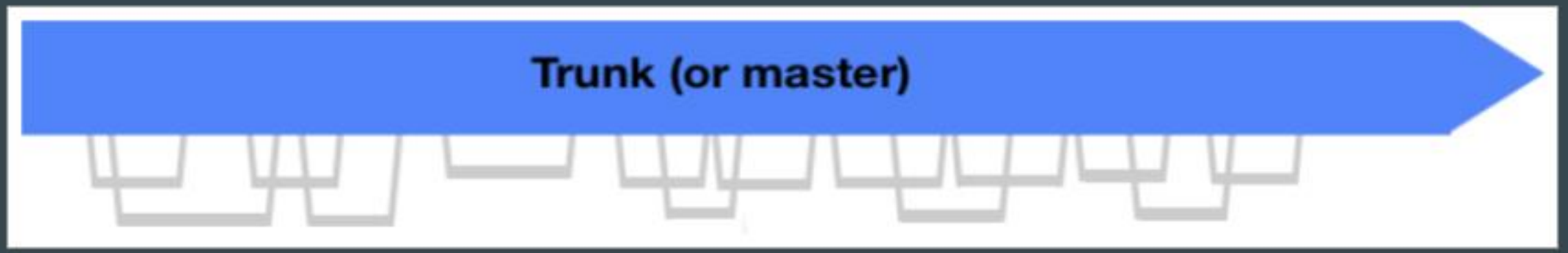
Trunk based development

"Commit little and often"



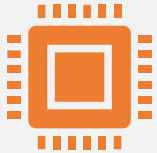
Índice



Introducción al TBD

- Branching model.
- Colaboración en la rama main o "trunk".
- Evolucionando desde los 80.
- Usado por empresas como Google o Facebook.

Principios básicos de TBD



Colaboración en una sola rama (trunk/main).

Resistencia a las ramas de desarrollo de larga duración.

Mantenimiento de la integridad del release.

Beneficios generales

AGILIDAD Y VELOCIDAD DE DESARROLLO.

RESPUESTA ANTE CAMBIOS.

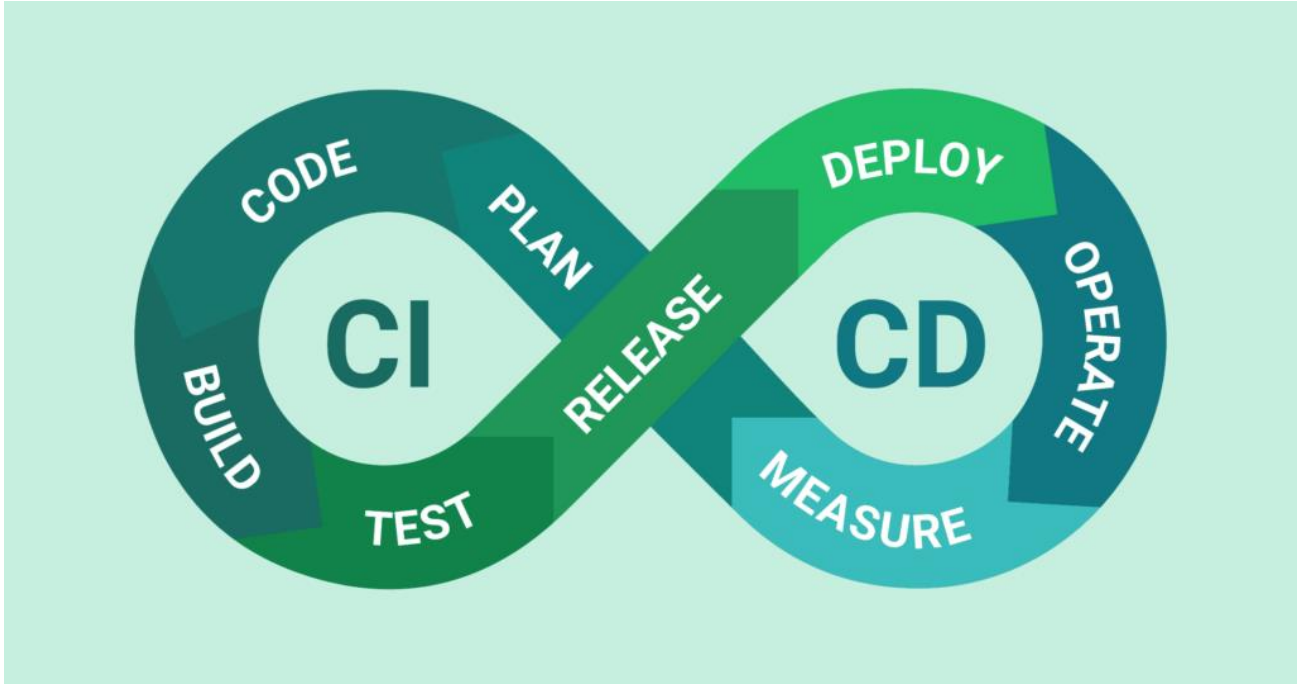
COLABORACIÓN EFECTIVA.

REDUCIR LA COMPLEJIDAD

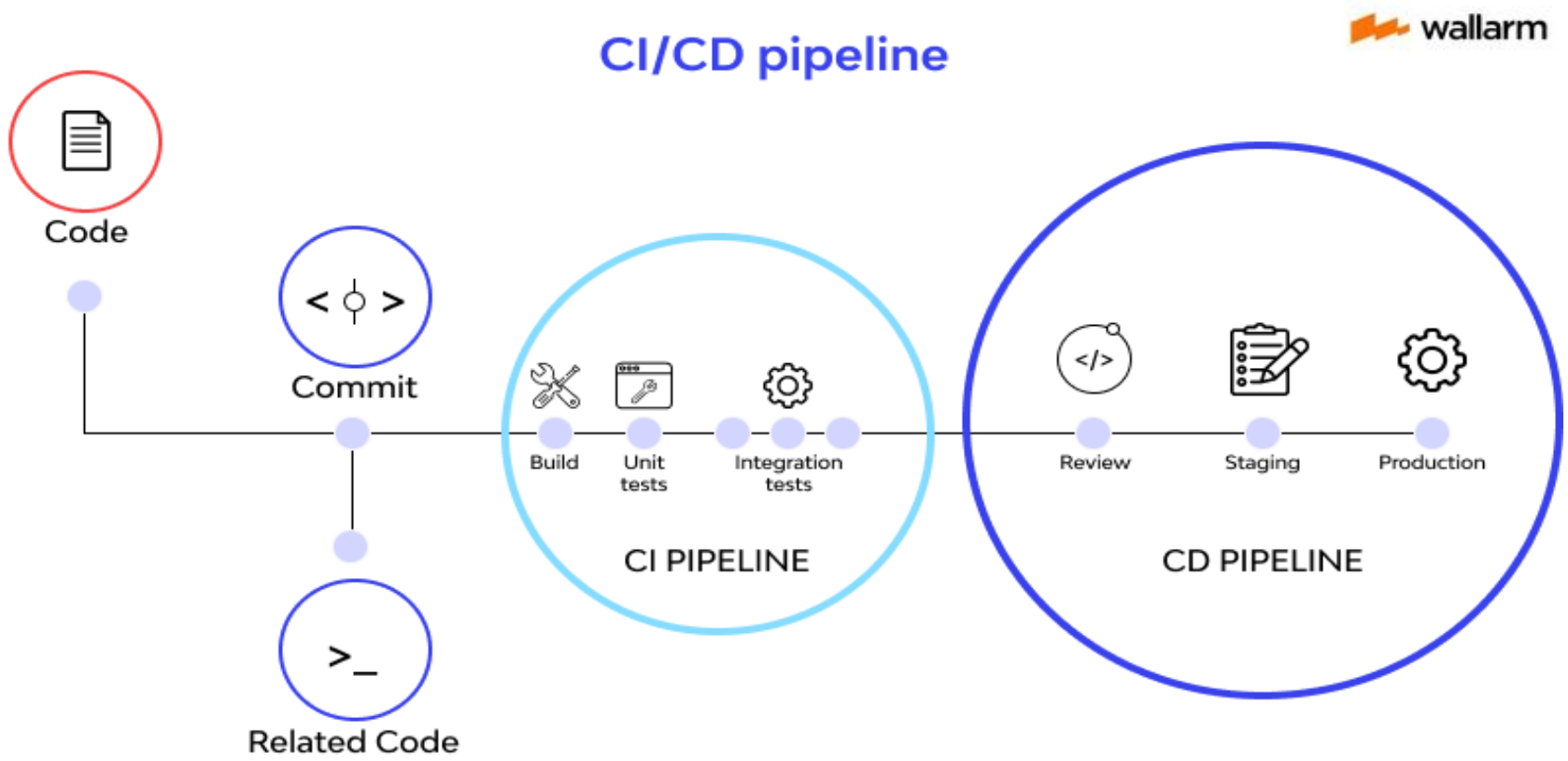
MEJORA CONTINUA

Integración continua y entrega continua

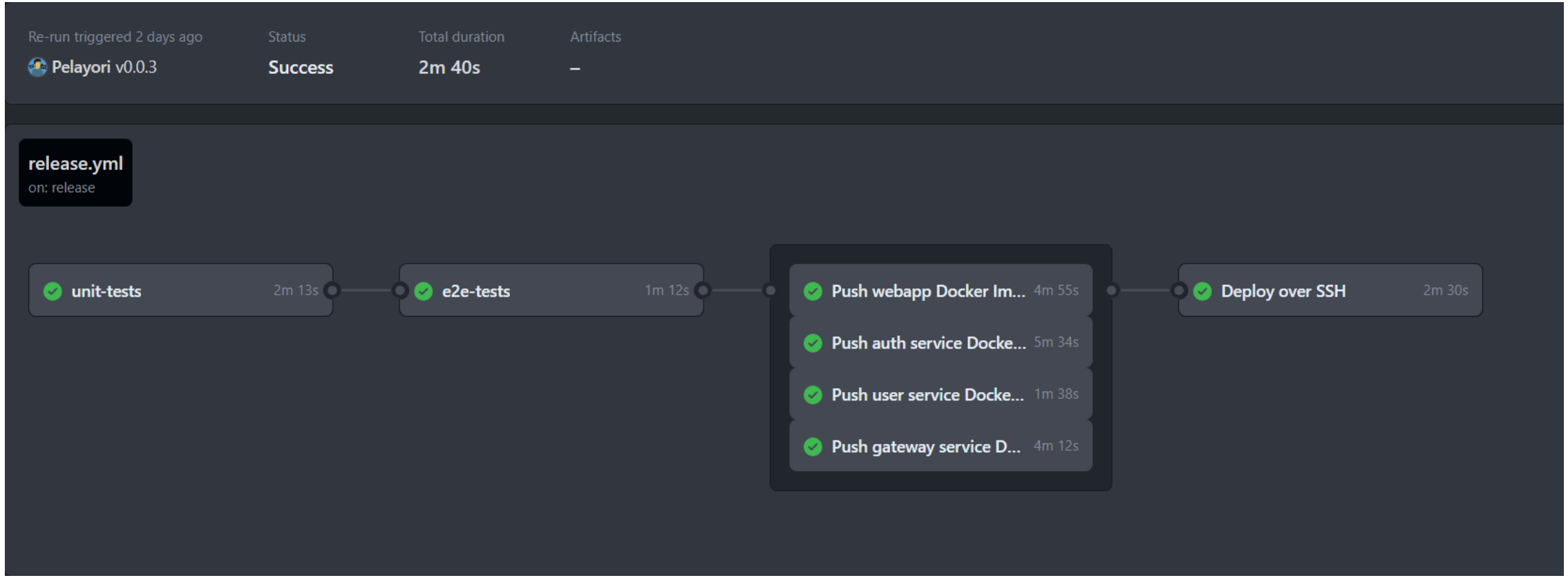
- Flujo de trabajo ágil
- Prueba constante de cambios en el código
- Código en el “tronco” siempre listo para release
- Commits frecuentes, (mínimo 1 en 24h)
- Merge hell



Integración continua y entrega continua



CI/CD en prácticas



Prácticas y claves técnicas

- Ramas de Características vs. Commit Directo al Trunk
  - Ramas de corta duración para desarrollo aislado, fusionadas rápidamente.
  - Commits directos favorecen integración continua, requieren disciplina y coordinación.
  - Flags de Características y Abstracción de Ramas

- Flags/Microservicios permiten activar/desactivar funcionalidades en ejecución, facilitan pruebas.
  - Abstracción de ramas reduce complejidad operativa, simplifica gestión del código.
  - Revisión Continua de Código y CI

- Revisiones de código identifican errores, mejoran calidad, promueven aprendizaje.
  - CI asegura calidad constante mediante pruebas automatizadas, mantiene código listo para despliegue.

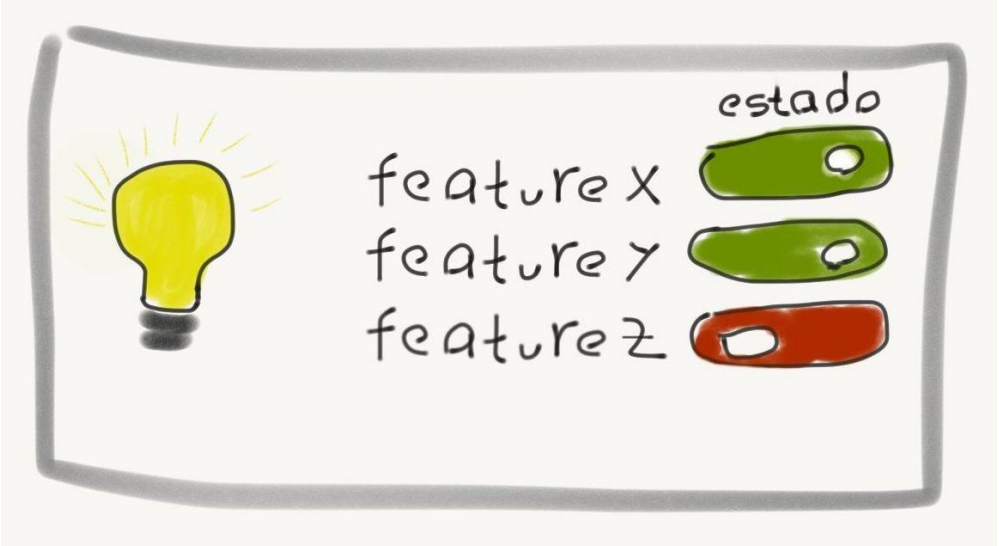
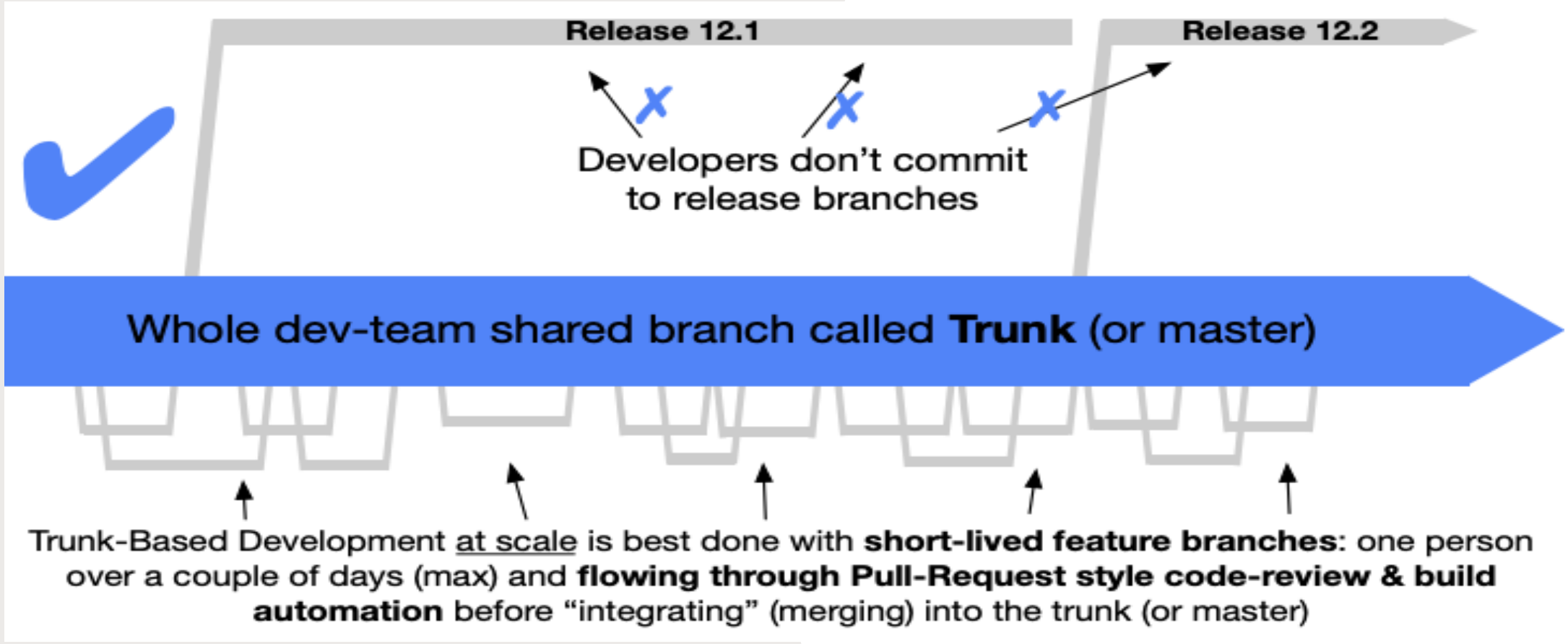
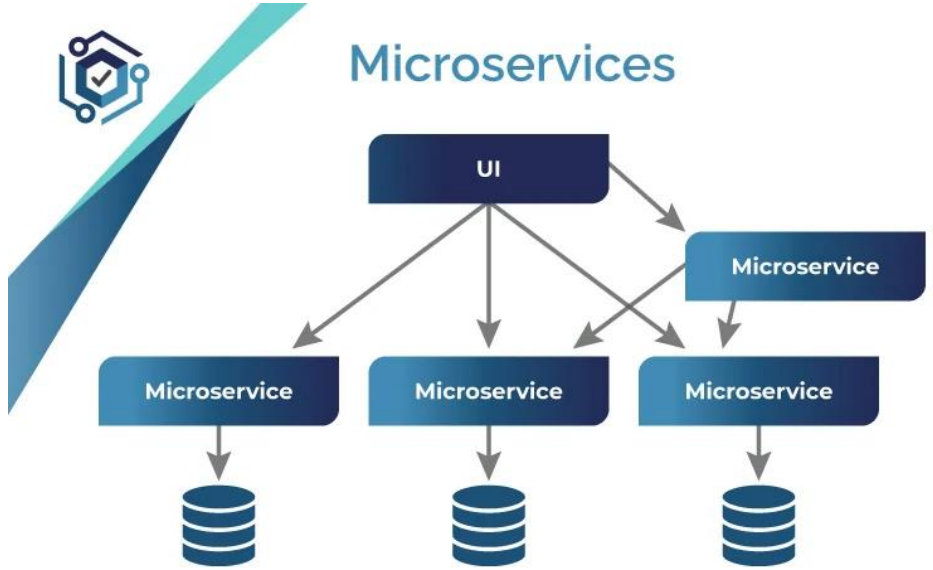


Ilustración de TBD



Microservicios y TBD

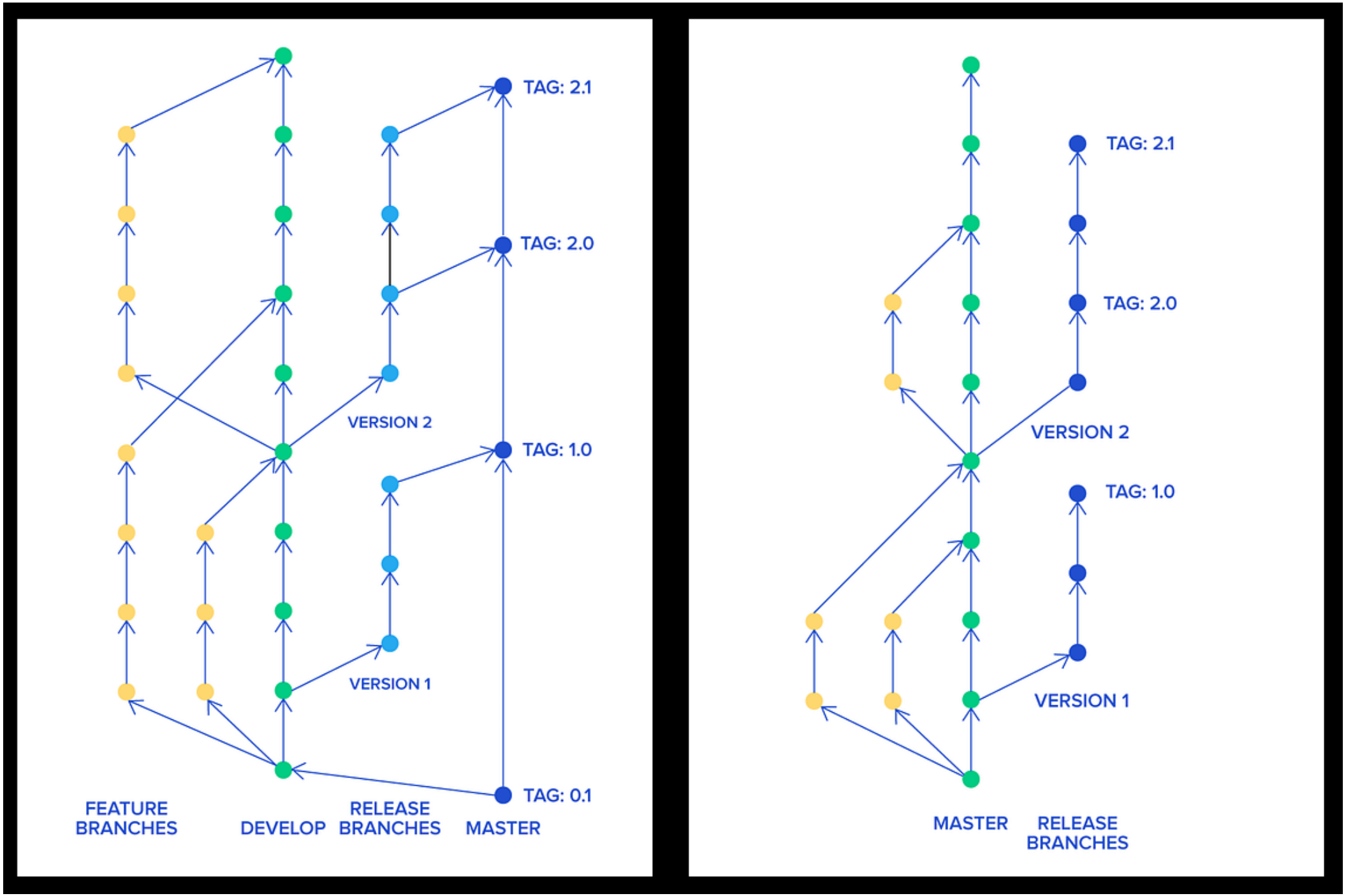
- Formados por diferentes APIs que se comunican entre si
- Integración en el workflow de TBD fácilmente
  - Ramas de corta duración
  - Fácil de mezclar con commits frecuentes



TDB en la industria

- Google pionero
- Utilizado en grandes empresas
- Flujo de trabajo en expansión

TBD	vs	GitFlow
Enfoque en una sola rama		Enfoque en ramificaciones
Rama principal		Una rama por funcionalidad
Integraciones continuas		Integraciones al final de cada fase
Entrega rápida		Entrega lenta
Estructura simple		Estructura compleja


TAG: 2.1
TAG: 2.0
VERSION 2
TAG: 1.0
VERSION 1
TAG: 0.1
FEATURE BRANCHES
DEVELOP
RELEASE BRANCHES
MASTER
TAG: 2.1
TAG: 2.0
VERSION 2
TAG: 1.0
VERSION 1
MASTER
RELEASE BRANCHES

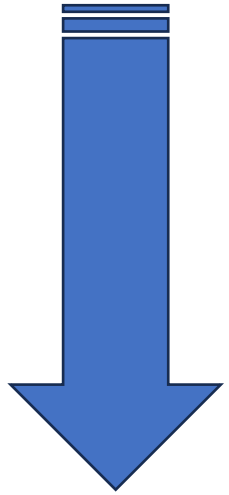
¿Cuál escoger?

- Composición del equipo
  - ¿Experimentado o principiantes?
- Tipo de producto
  - ¿GreenField o brownfield?
- Proceso de creación
  - ¿Dirigido por el equipo o gobernado?

De GitFlow a TBD

- Gradual
- Adaptarse a la cadencia anterior
- Cumplir con los compromisos funcionales
- Acuerdo de todas las partes

GitFlow



TBD

Desafíos

- Pruebas automatizadas
- Disposición del equipo
- Superposición de código
- Lanzamiento

Bibliografia

- https://www.atlassian.com/continuous-delivery/continuous-integration
- ¿Cuál debo escoger, TDB o GitFlow?
- https://www.codurance.com/es/publications/que-es-trunk-based-development
- https://openwebinars.net/blog/trunk-based-development-vs-git-flow-cual-elegir/#qu%C3%A9-es-trunk-based-development
- https://trunkbaseddevelopment.com/
- https://www.se-radio.net/2023/05/se-radio-564-paul-hammant-on-trunk-based-development/
- https://es.linkedin.com/pulse/branching-model-en-desarrollo-de-software-definición-y-ojeda-montoya
- TDB vs GitFlow

Trunk based development

"Commit little and often"