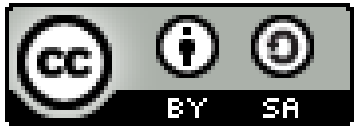
Conceptos relacionados con la Integración Continua

Asignatura: Integración Continua en el Desarrollo Ágil

Máster Universitario en Desarrollo Ágil de Software para la Web

José Ramón Hilera González



Conceptos

1. CI/CD
2. Entornos: desarrollo, pre-producción, producción
3. CI/CD pipeline
4. DevOps
5. Desarrollo ágil
6. Flujos de trabajo en ramas

1. CI/CD (1/3)

- CI/CD hace referencia a la combinación de Integración Continua (CI) con:
  - Entrega continua: Continous Delivery (CD)
  - Despliegue Continuo: Continous Deployment (CD)

1. CI/CD (2/3)

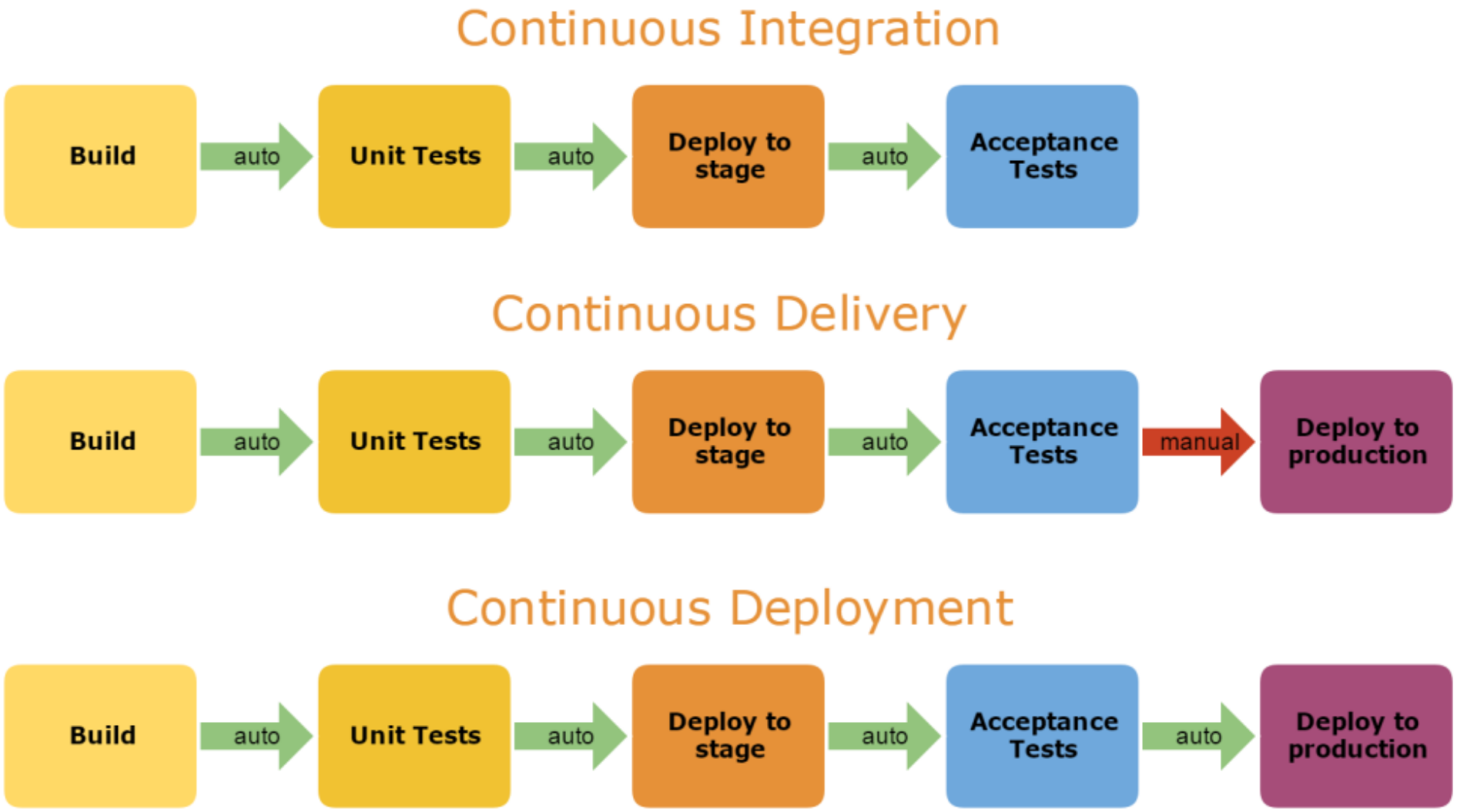


Imagen: www.linuxnix.com

1. CI/CD (3/3)

- **Entrega Continua (Continous Delivery-CD):** En un proceso de desarrollo, en el que se aplica integración continua (CI), se realiza entrega continua si el despliegue del producto final a producción se realiza de forma manual, con la intervención de una persona.
- **Despliegue Continuo (Continous Deployment-CD):** En un proceso de desarrollo, en el que se aplica integración continua (CI), se realiza despliegue continuo si el despliegue del producto final a producción se realiza de forma automática, sin la intervención de una persona.

2. Entornos: desarrollo, pre-producción, producción (1/2)

- Habitualmente en un proyecto hay, al menos, tres entornos en los que se instala el producto software desarrollado
  - Desarrollo (Development), Pre-producción (Stage), Producción (Production)

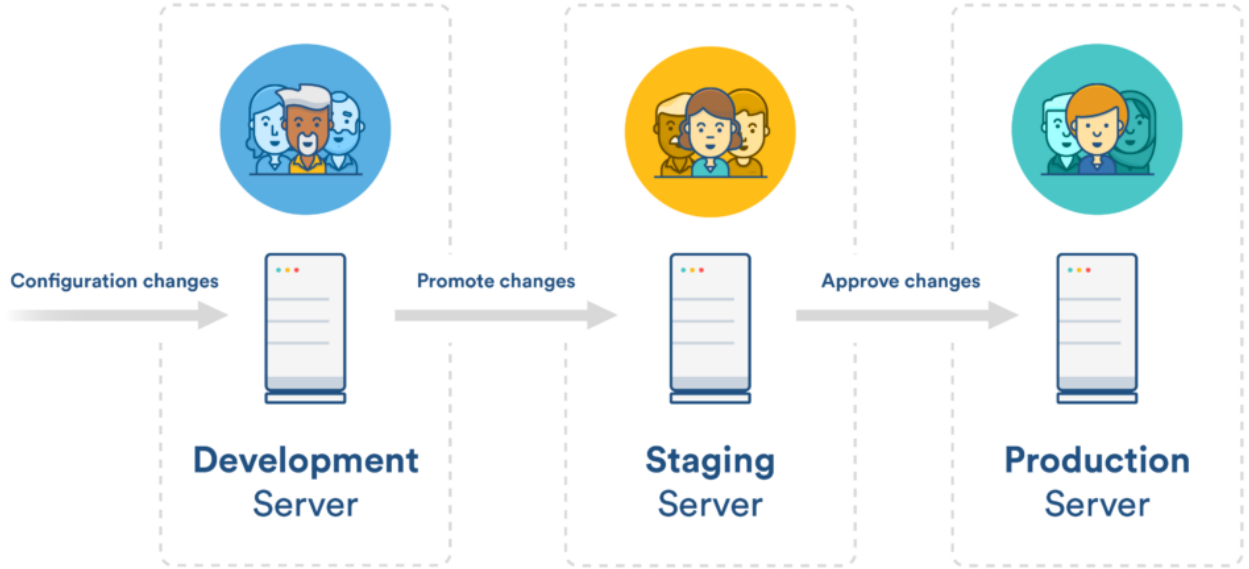


Imagen: V. Pabba

2. Entornos: desarrollo, pre-producción, producción (2/2)

- Desarrollo (Development): Suele haber un entorno local en la máquina de cada desarrollador para sus pruebas particulares antes de subir cambios al repositorio, y un entorno compartido gestionado por el servidor de integración continua
  - Ej. Aplicación web: Entorno basado en un contenedor compuesto por un servidor web en el que se instala la aplicación y un servidor base de datos en el que se instala una base de datos con datos de prueba
- Pre-producción (Stage): Réplica privada del entorno de producción, que sólo utilizan los desarrolladores, personal de aseguramiento de la calidad (QA) y, en su caso, un pequeño grupo de usuarios, para poder probar y asegurarnos de que esté todo correcto y que cuando se exponga en producción no haya ningún fallo
  - Ej. Aplicación web: Servidor web y servidor de base de datos con datos de prueba, en la nube.
- Producción (Production): Entorno de acceso público en el que queda instalado el producto software para su utilización por cualquier usuario
  - Ej. Aplicación web: Servidor web público y servidor de base de datos empresarial con datos reales, en la nube.

Referencias: Autentia, Milecia McG

3. CI/CD pipeline

- El pipeline de un proyecto de desarrollo es la secuencia de pasos automatizados establecidos para conseguir obtener el producto software aplicando integración, entrega y/o despliegue continuo.
- Un servidor de integración continua se encarga de ordenar la ejecución de los pasos del pipeline del proyecto cada vez que se sube un cambio (commit) al repositorio de código compartido.

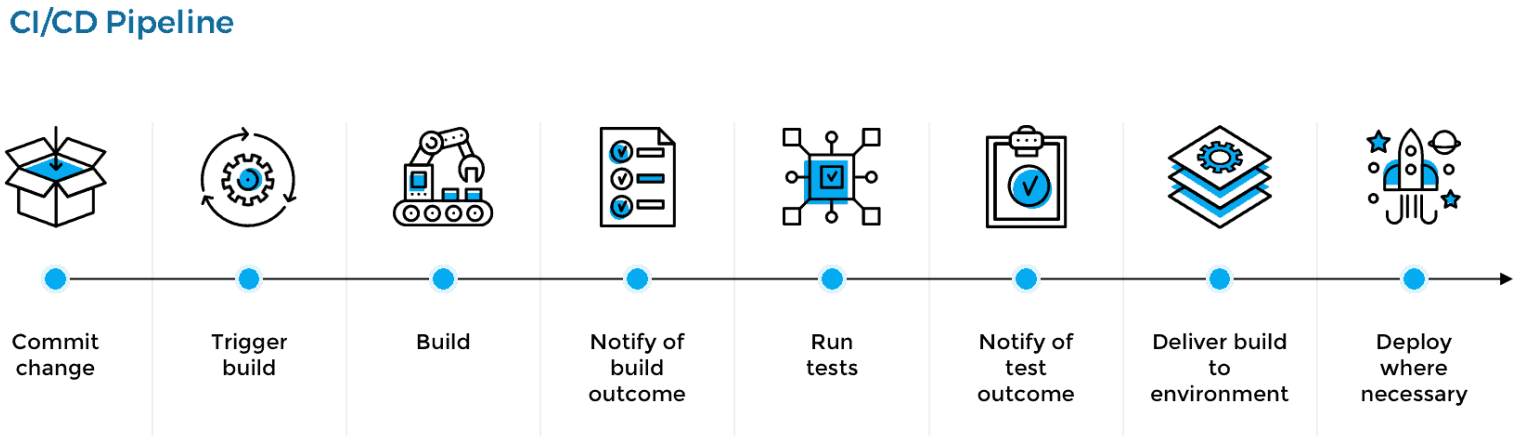


Imagen: TJ Simmons

3. CI/CD pipeline
Ejecución

- Cuando se ejecuta un pipeline, se van ejecutando en secuencia los pasos que lo componen.
  - Por ejemplo, la construcción o compilación (build)
- Si alguno de los pasos falla, se aborta la ejecución del pipeline y se avisa a los miembros del proyecto
  - Por ejemplo, por email.

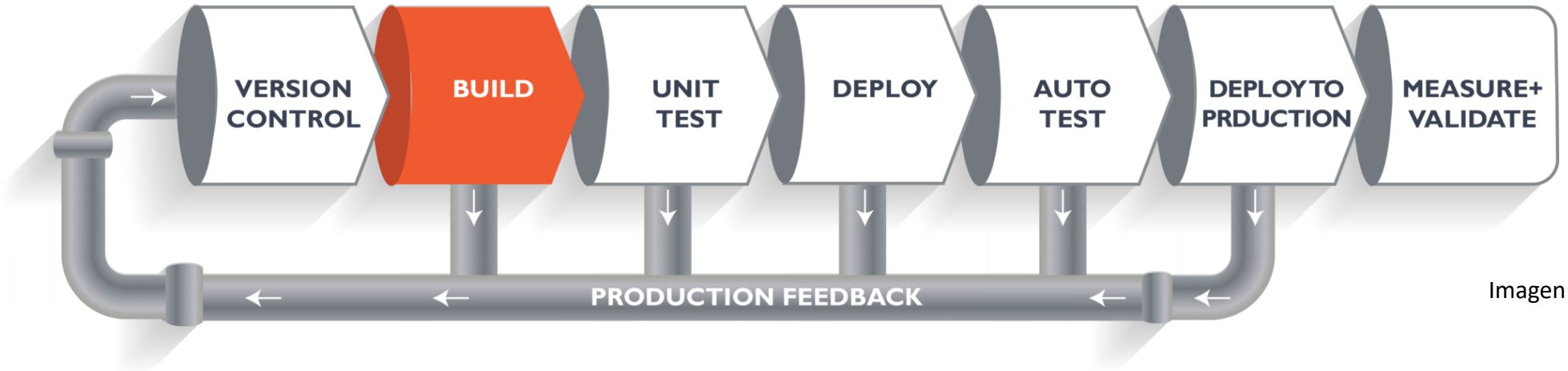


Imagen: Samarpit Tuli

4. DevOps

- DevOps = Dev (Desarrollo) + Ops (Operaciones).
- Marco de trabajo en el que tanto el departamento de desarrollo como el de operaciones (IT) colaboran estrechamente para producir y mantener software de calidad que cambia con frecuencia, automatizando al máximo las tareas relacionadas con la planificación, codificación, construcción, prueba, liberación, despliegue, operación y monitorización.
- No se trata de que los desarrolladores asuman las funciones de los administradores de sistemas ni viceversa, se trata de colaborar estrechamente entre ellos para acortar el tiempo de puesta en producción de nuevas funcionalidades del software.
- DevOps se basa en la aplicación de prácticas de CI/CD y en el uso de herramientas que automaticen los procesos.

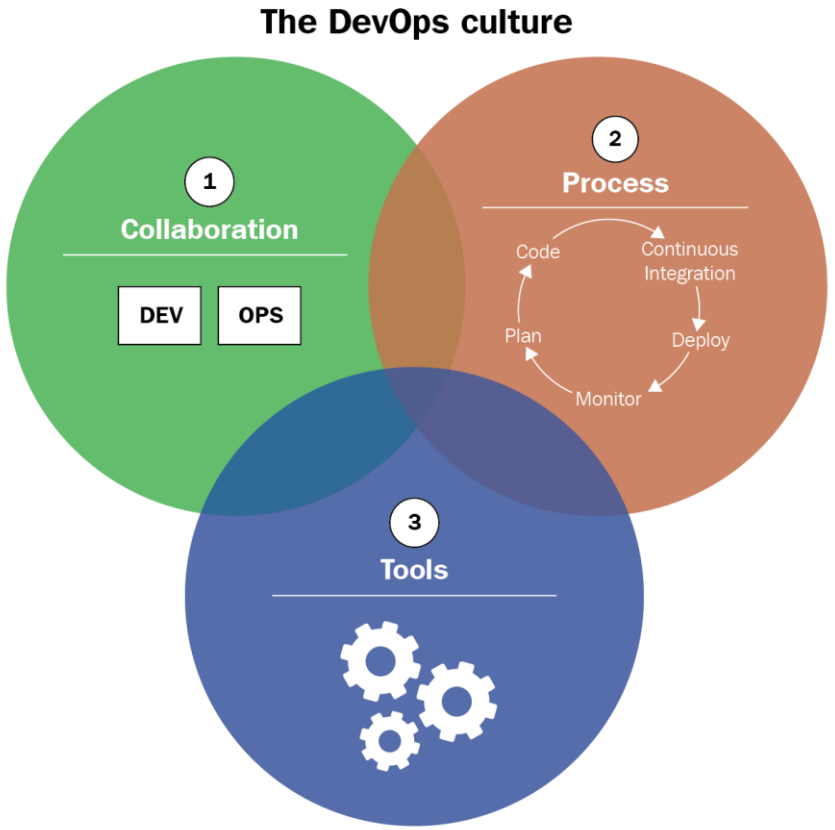
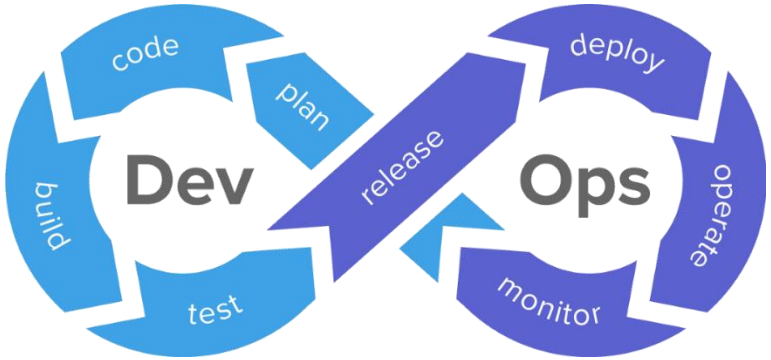


Imagen: Mikael Krief

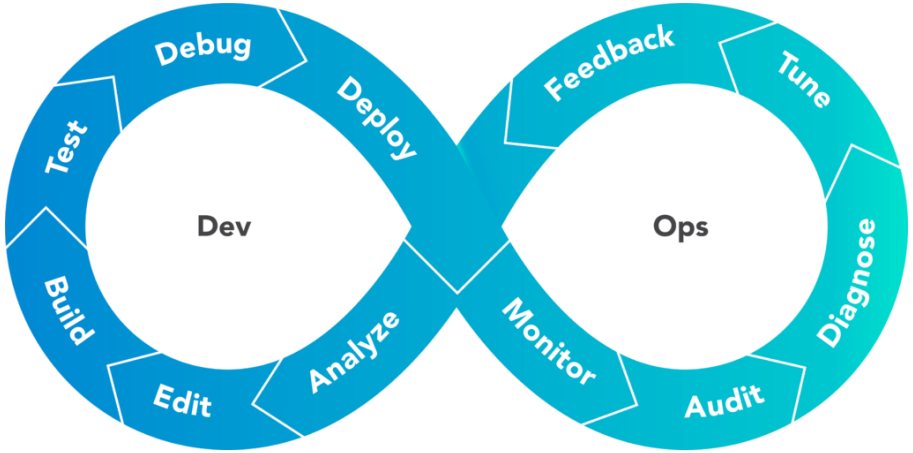
4. DevOps
Ciclo de vida, ejemplo con 8 fases

- **Plan (Planficación):** Esta fase incluye todo lo que sucede antes de que los desarrolladores empiecen a escribir código: requisitos, historias de usuario, etc.
- **Code (Codificación):** Cada desarrollador trabaja en su entorno personal elaborando una parte del código fuente que le corresponde.
- **Build (Construcción):** Cuando un desarrollador ha terminado su parte, la envía al repositorio compartido y el servidor de CI/CD ordena la compilación de todo el proyecto, y la ejecución de las correspondientes pruebas (unitarias, integración).
- **Test (Prueba):** Se completan las pruebas que no hayan hecho en la fase anterior, normalmente son pruebas funcionales (end-to-end) que requieren una instalación de la aplicación en un entorno para pruebas.
- **Release (Liberación):** Esta fase puede no ser necesaria si se despliega directamente a producción. Si existe se despliega la aplicación en un entorno de pre-producción para seguir haciendo pruebas.
- **Deploy (Despliegue):** Se despliega la aplicación en el entorno de producción.
- **Operate (Operación):** La aplicación queda operativa para que los usuarios la utilicen.
- **Monitor (Monitorización):** Se recogen datos sobre el funcionamiento de la aplicación y las opiniones de los usuarios, que se tendrán en cuenta en siguientes iteraciones

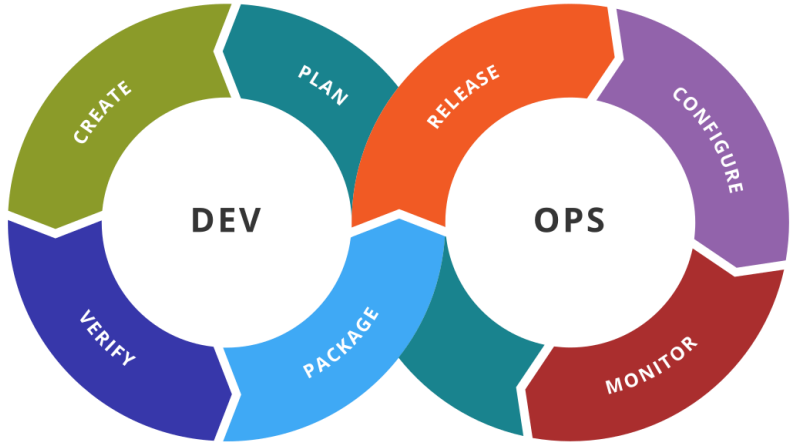


Fuente: JakobTheDev

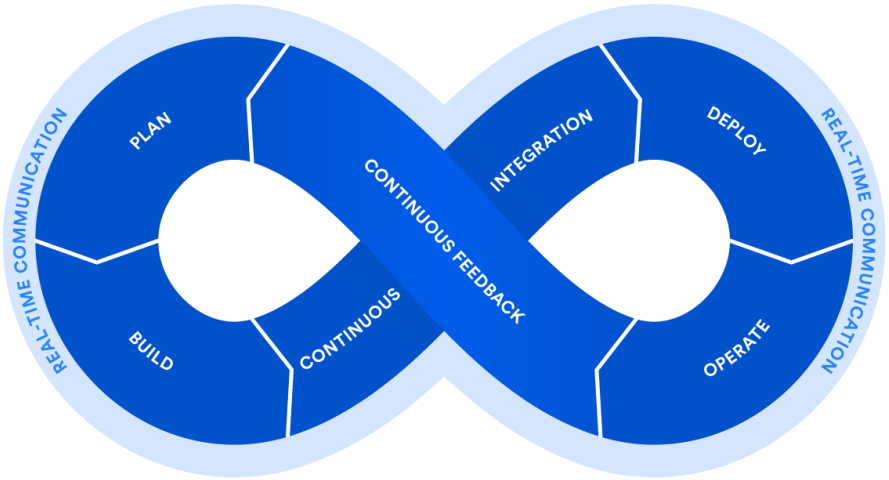
4. DevOps
Ciclo de vida, otros ejemplos



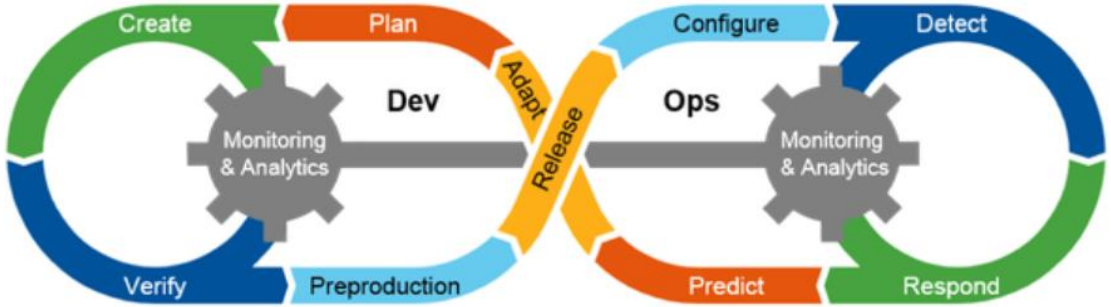
Fuente: compuware



Fuente: innovify



Fuente: atlassian



Fuente: quora

4. DevOps Herramientas

- DevOps se basa en el uso de un conjunto o combinación de herramientas (toolchain) que permitan la máxima automatización de cada tarea

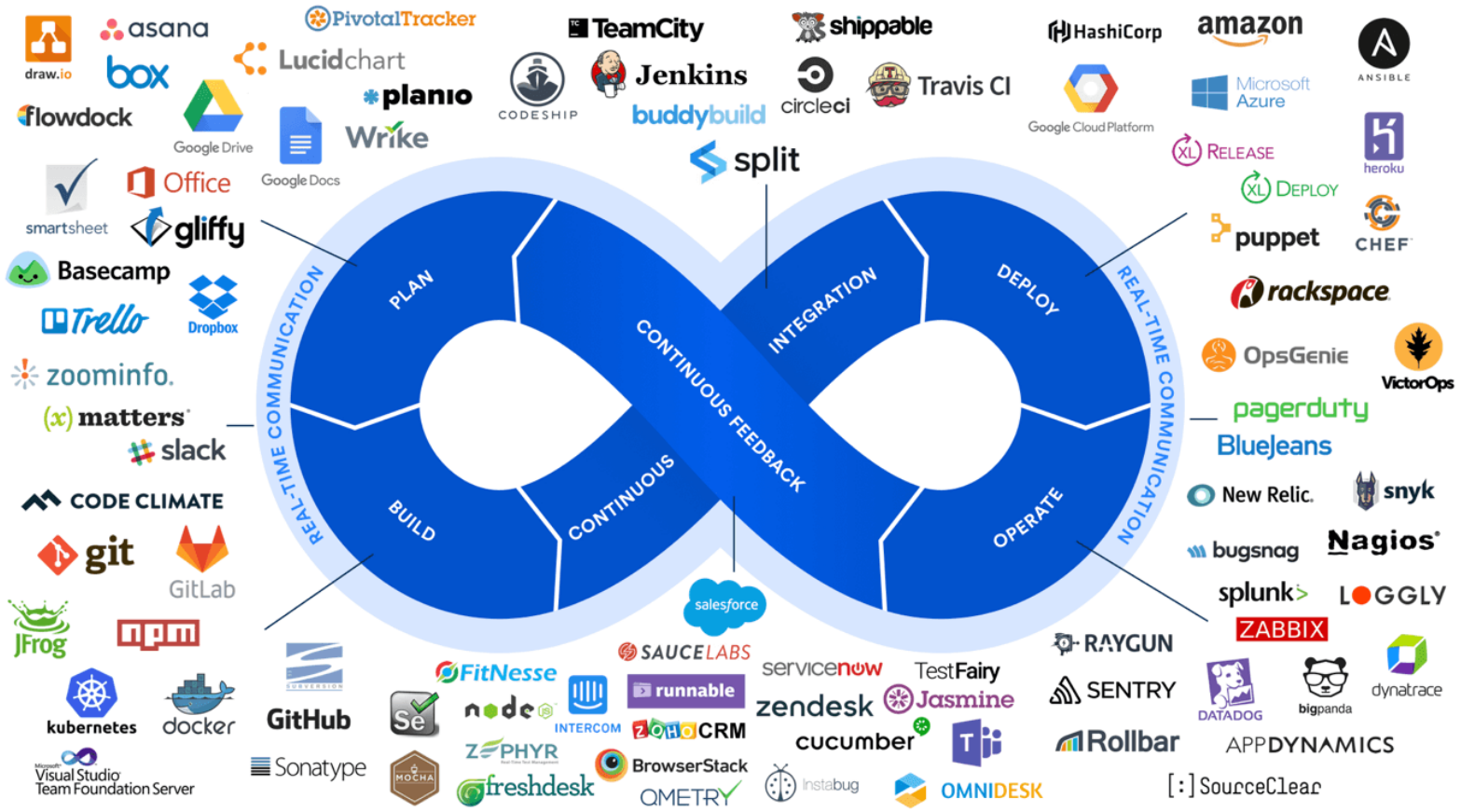


Imagen: osolabs

5. Desarrollo ágil

- CI/CD ayuda a cumplir los principios del manifiesto ágil, especialmente:
  - Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
  - Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
  - El software funcionando es la medida principal de progreso.
- DevOps está soportado por metodologías ágiles

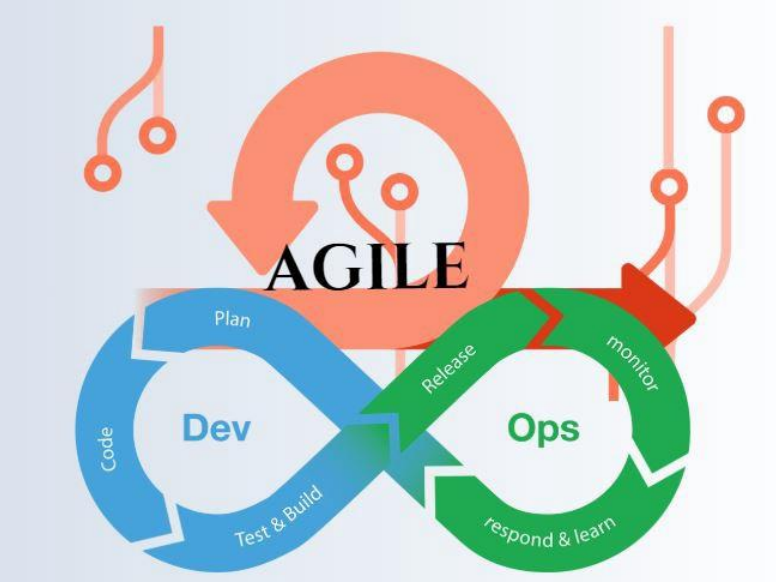


Imagen: Nanduri Balajee

6. Flujos de trabajo en ramas

- En un repositorio de código fuente utilizado para la integración continua en un proyecto, las ramas se organizan aplicando un tipo de flujo de trabajo en ramas, que se basa en una política o estrategia de ramas
- Los flujos más conocidos para repositorios Git son
  - GitFlow
  - GitLab Flow
  - GitHub Flow
  - Release Flow
  - Trunk-Based
  - Master-only flow

Referencia: CodelyTV

6. Flujos de trabajo en ramas Git-Flow

- Está basado en dos grandes ramas con infinito tiempo de vida:
  - rama master
  - rama develop
- varias ramas de apoyo
  - unas orientadas al desarrollo de nuevas funcionalidades (ramas feature-*),
  - otras al arreglo de errores (ramas hotfix-*)
  - y otras orientadas a la preparación de nuevas versiones de producción (ramas release-*).

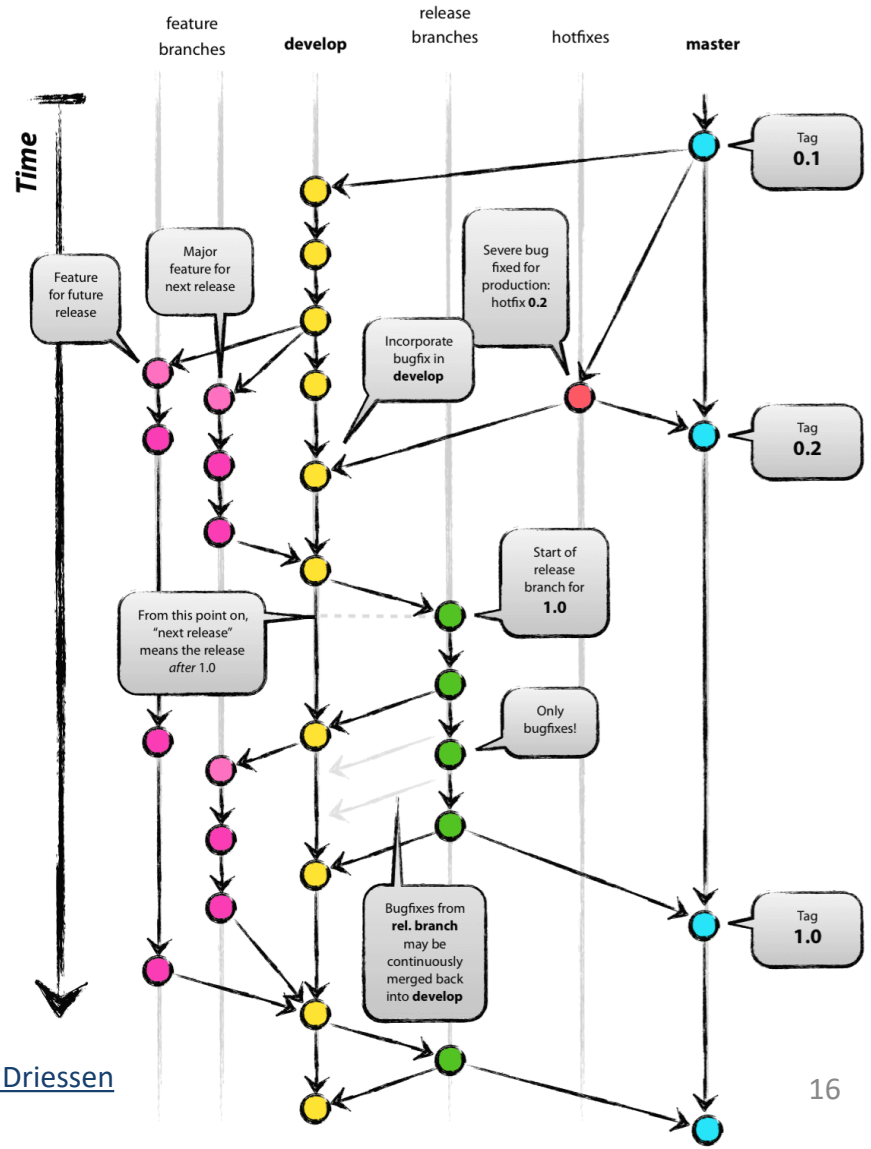


Imagen: V. Driessen

6. Flujos de trabajo en ramas
GitLab Flow

- Está basado en tres ramas con infinito tiempo de vida
  - rama master
  - rama pre-production
  - rama production
- Y varias ramas orientadas al desarrollo de nuevas funcionalidades o reparar errores
  - ramas feature o hotfix

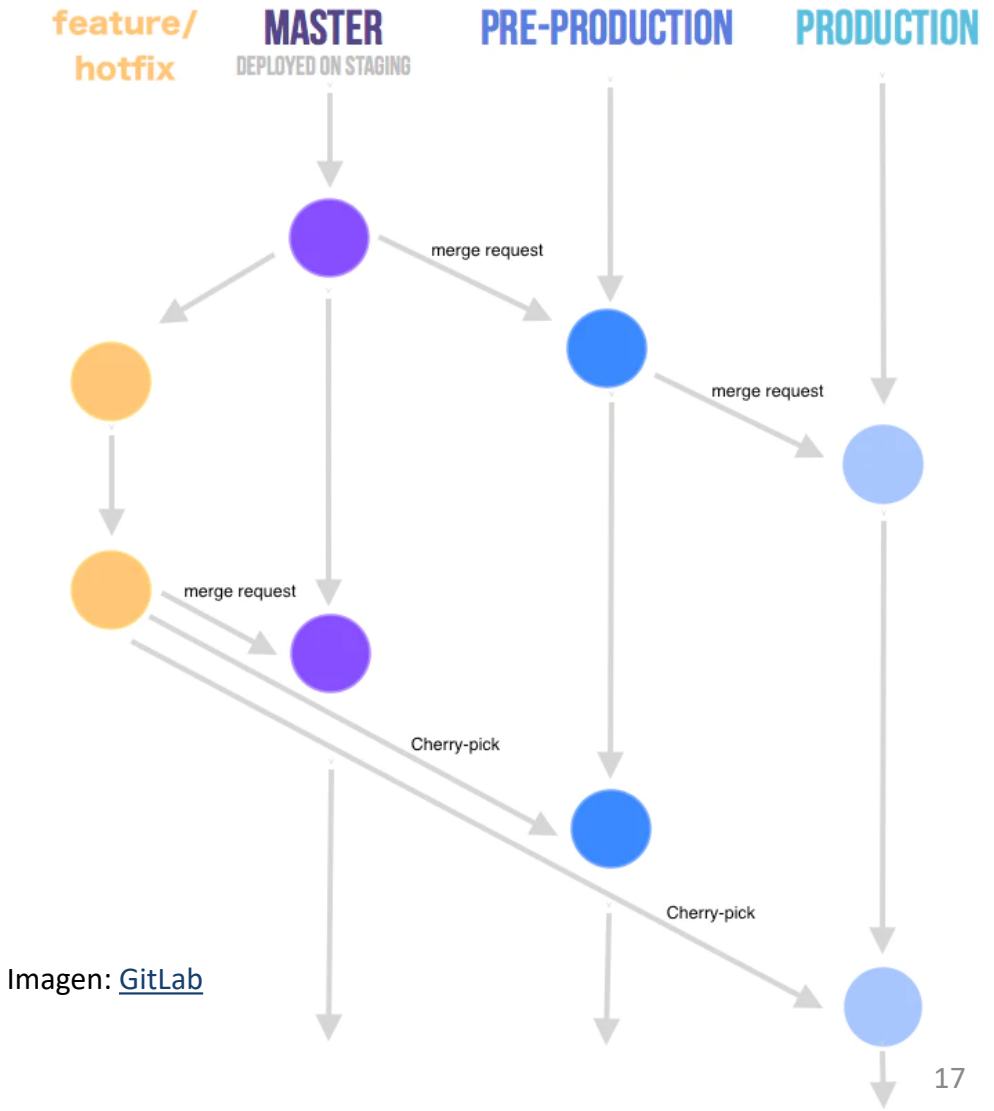


Imagen: GitLab

6. Flujos de trabajo en ramas
GitHub Flow

- Está basado en una rama con infinito tiempo de vida
  - rama master
- Y varias ramas orientadas al desarrollo de nuevas funcionalidades
  - ramas feature-*

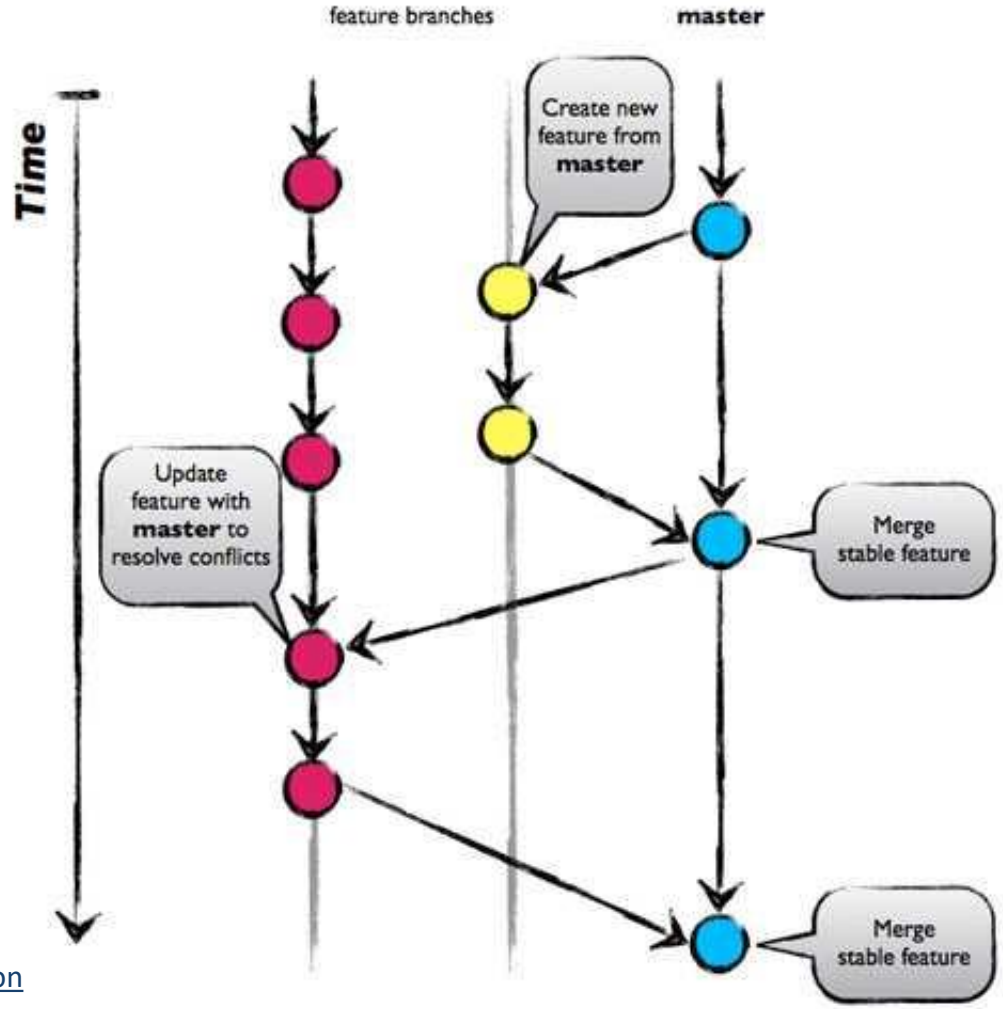


Imagen: nicoespeon

6. Flujos de trabajo en ramas
GitHub Flow (Principios)

- Todo lo que está en la rama master está listo para ser puesto en producción
- Para trabajar en algo nuevo, hay que crear una nueva rama a partir de la rama master con un nombre descriptivo. El trabajo se irá integrando sobre esa rama en local y regularmente también a esa rama en el repositorio compartido
- Cuando creamos que la rama está lista para integrarla en la rama master, se debe abrir una pull request (solicitud de integración de cambios).
- Alguien debe revisar y visar los cambios para fusionarlos con la rama master
- Los cambios integrados se pueden poner en producción.