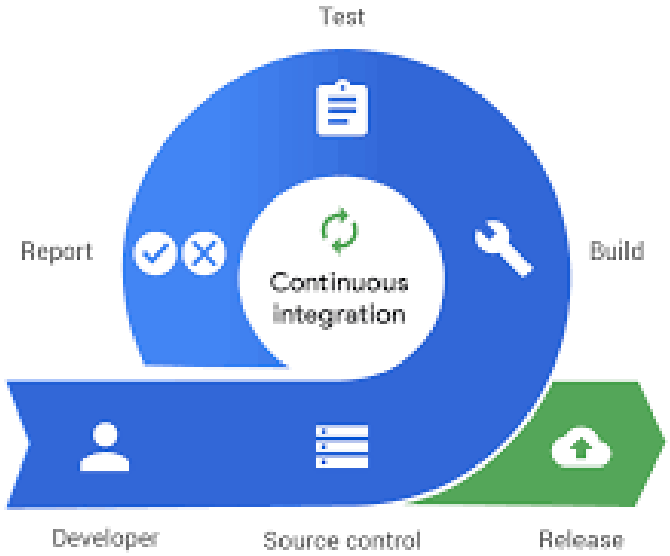
Continuous Integration & Continuous Delivery

Natalia Reyes Altamirano

Continuous Integration (CI)

- Es la práctica de automatizar la integración de los cambios de código de varios contribuidores en un único proyecto de software. Las herramientas automatizadas sirven para verificar que el nuevo código es correcto antes de la integración.
- Un sistema de control de versiones del código fuente es el punto clave del proceso de CI. El sistema de control de versiones también se complementa con otras comprobaciones como las pruebas automatizadas de calidad del código, las herramientas de revisión de estilo de sintaxis y mucho más.

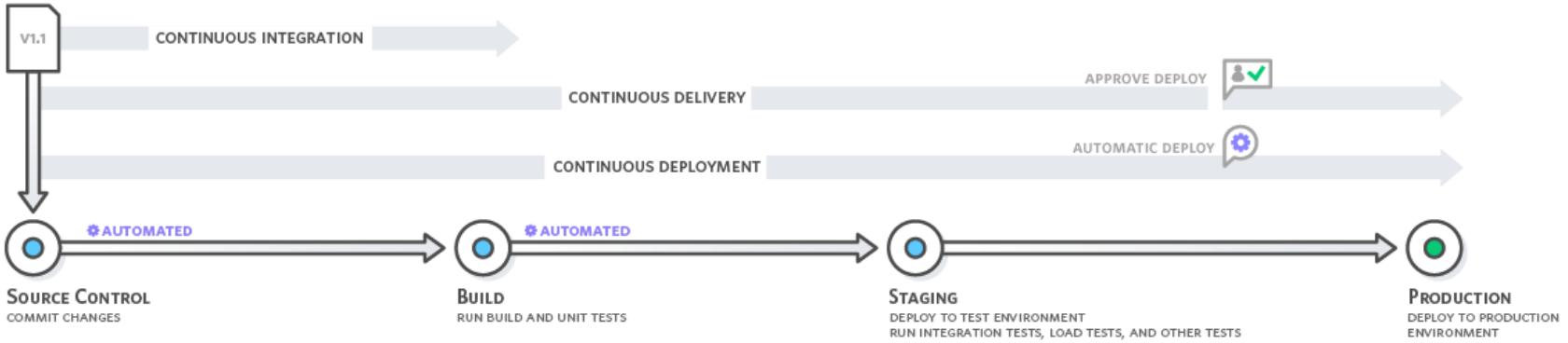


Beneficios de CI

- **Mejora la productividad de desarrollo:** Mejora la productividad del equipo al liberar a los desarrolladores de las tareas manuales y fomentar comportamientos que ayudan a reducir la cantidad de errores y bugs enviados a los clientes.
- **Permite encontrar y arreglar los errores con mayor rapidez**: Gracias a la realización de pruebas más frecuentes, el equipo puede descubrir y arreglar los errores antes de que se conviertan en problemas más graves.
- **Permite entregar las actualizaciones con mayor rapidez:** le permite a su equipo entregar actualizaciones a los clientes con mayor rapidez y frecuencia.

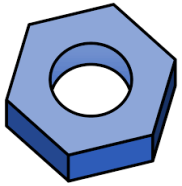
¿En qué consiste la Continuous Integration ?

- Los desarrolladores envían los cambios de forma periódica a un repositorio compartido con un sistema de control de versiones como Git.
- Antes de cada envío, se puede elegir ejecutar pruebas de unidad local en el código como medida de verificación adicional antes de la integración.
- Un servicio de integración continua crea y ejecuta automáticamente pruebas de unidad en los nuevos cambios realizados en el código para identificar inmediatamente cualquier error.
- Con la entrega continua, se crean, prueban y preparan automáticamente los cambios en el código y se entregan para la fase de producción.



Herramientas de Continuous Integration de código abierto

- **Jenkins:** Permite a los desarrolladores compilar, integrar y probar código automáticamente en el momento en que lo confirman en el repositorio de origen, lo que facilita a los desarrolladores detectar los errores a tiempo y desplegar el software más rápido. El plugin de Docker está disponible en Jenkins.
- **Buildbot:** Puede automatizar todos los aspectos del ciclo de desarrollo de software. Como sistema de planificación de trabajos, pone en cola y ejecuta trabajos, e informa de los resultados.
- **Go:** Permite usar conductos, que facilitan el modelado de flujos de trabajo de compilación complejos.
- **Travis CI:** Ofrece soluciones alojadas fiables.
- **GitLab CI:** Es un servicio de alojado gratuito que proporciona una gestión detallada del repositorio git con funciones como control de acceso, seguimiento de problemas o revisiones de código, entre otras.



Formas de medir la CI

Factor a probar	Qué medir
Las confirmaciones de código activan una compilación del software.	El porcentaje de confirmaciones de código que dan como resultado una compilación de software sin intervención manual
Las confirmaciones de código activan una serie de pruebas automatizadas.	El porcentaje de confirmaciones de código que dan como resultado un conjunto de pruebas automatizadas que se ejecutan sin intervención manual
Las compilaciones y pruebas automatizadas se ejecutan de forma correcta todos los días.	El porcentaje de compilaciones y de pruebas automatizadas que se ejecutan de forma correcta todos los días
Las pruebas actuales están disponibles para las pruebas de exploración de los verificadores.	La disponibilidad de las compilaciones para los verificadores, o al revés, es decir, la no disponibilidad de las compilaciones para los verificadores
Los desarrolladores reciben comentarios de las pruebas de aceptación y rendimiento todos los días.	La disponibilidad de comentarios para los desarrolladores a partir de las pruebas de aceptación y rendimiento, es decir, el porcentaje de pruebas que proveen comentarios y que están disponibles para los desarrolladores en un día
Las compilaciones rotas se corrigen de inmediato.	El tiempo que transcurre entre la falla de la compilación y su corrección, ya sea con un registro que soluciona el problema o con una reversión del cambio que generó el problema

Continuous Delivery (CD)

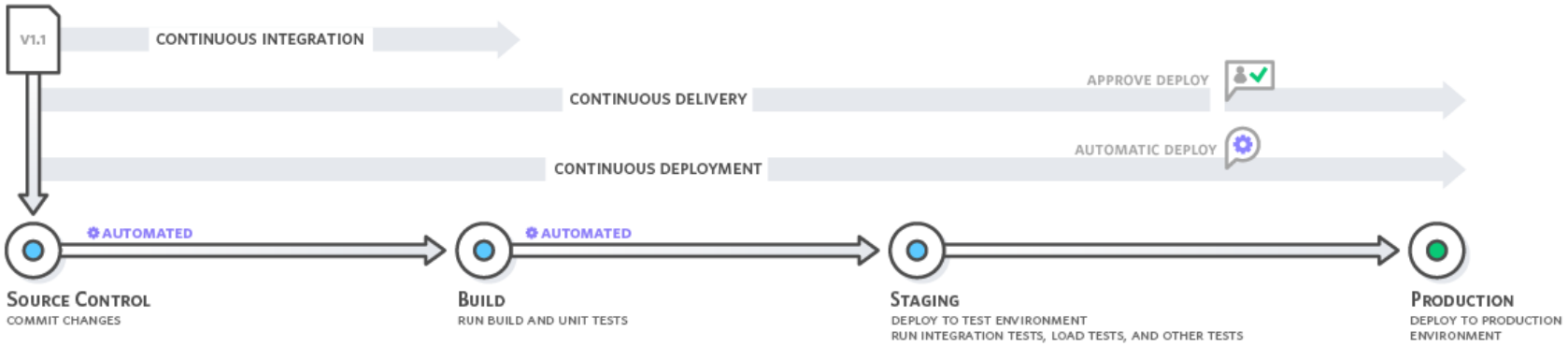
- Es una práctica de desarrollo de software mediante la cual se preparan automáticamente los cambios en el código y se entregan a la fase de producción. La entrega continua amplia la integración continua al implementar todos los cambios en el código en un entorno de pruebas o de producción después de la fase de compilación. Cuando la entrega continua se implementa de manera adecuada, los desarrolladores dispondrán siempre de un artefacto listo para su implementación que se ha sometido a un proceso de pruebas estandarizado.

Ventajas y Desventajas de CD

Ventajas	Desventajas
Es mucho más fácil encontrar y eliminar los errores presentes en el software durante la etapa de desarrollo.	Factor de costo: es necesario contar con un servidor potente y fiable de integración de datos para llevar a cabo las pruebas automatizadas y conseguir una liberación correcta y segura del producto.
En el pasado, el desarrollo de software requería muchísimo esfuerzo. Gracias al continuous delivery, los desarrolladores pueden concentrarse exclusivamente en el desarrollo.	Las pruebas automatizadas tienen que funcionar a la perfección y no presentar errores de código. En caso contrario, pueden ocasionar graves daños al ser ejecutadas.
Gracias al pipeline de continuous delivery es mucho más sencillo para los desarrolladores llevar a cabo la resolución de problemas.	Requiere que haya una buena coordinación dentro del equipo porque los cambios introducidos en el código deben compilarse de forma eficiente y con una determinada frecuencia.
Otros procesos de prueba (como las pruebas alfa y beta) conllevan más costes asociados.	Requiere una buena y continua comunicación con los clientes y sus sistemas de destino.
Se pueden dedicar más recursos a la etapa conceptual que a la técnica durante el control de calidad para mejorar el software.	El cliente espera que haya mejoras y actualizaciones continuamente. Es difícil poder “pausar” el proyecto de desarrollo.
Normalmente el desarrollo de software se consigue de forma más rápida porque el proceso de liberación automatizado reduce la carga de trabajo de los desarrolladores y también la cantidad de descansos que deben tomar.	A la hora de implementar innovaciones, mejoras o modificaciones al producto, sigue siendo necesario hacerlo manualmente. Si se quiere automatizar este proceso, es necesario recurrir al continuous deployment.
Gracias a que las publicaciones de software ocurren más rápido y más a menudo, se recibe más feedback lo que se traduce en más mejoras.	El cliente tiene que estar dispuesto a utilizar el software cuando todavía se encuentra en una fase de desarrollo. Además, tiene que poner de su parte y devolvernos su feedback.
Los desarrolladores tienen que soportar menos presión a la hora de realizar cambios en el código fuente porque los errores se encuentran mucho más rápido. Esto sirve como motivación e inspiración para el trabajo.	

¿En qué consiste la Continuous Delivery ?

- Con la entrega continua, todos los cambios en el código se crean, se prueban y se envían a un entorno de almacenamiento o pruebas de no producción. Pueden efectuarse varias pruebas al mismo tiempo antes de la implementación en producción. La diferencia entre la entrega continua y la implementación continua es la diferencia de aprobación manual para actualizar la producción. Con la implementación continua, la producción tiene lugar de manera automática, sin aprobación explícita.
- La entrega continua automatiza todo el proceso de publicación de software. Cada revisión efectuada activa un proceso automatizado que crea, prueba y almacena la actualización. La decisión definitiva de implementarla en un entorno de producción en vivo la toma el desarrollador.
- .



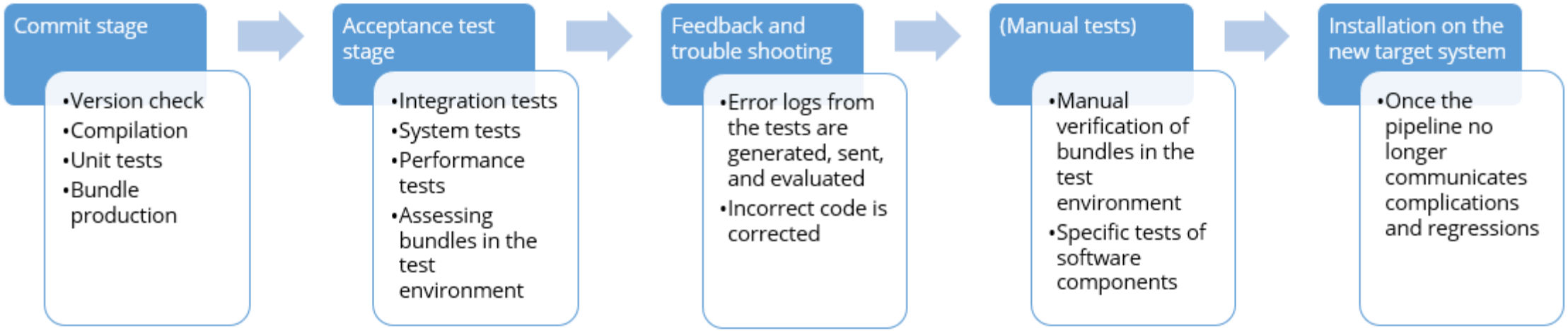
Herramientas de Continuous Delivery de código abierto

- **Jenkins:** facilita la integración continua de los componentes del software. Jenkins está escrito en Java, se ejecuta en un contenedor EJB y dispone de varias herramientas de build (Apache Ant, Maven/Gradle, CVS, Subversion, Git, etc.) y, de los procesos de pruebas automáticas tan importantes en el caso del continuous delivery (Junit, Emma).
- **Circle CI:** Funciona preferiblemente con GitHub, GitHub Enterprise y Bitbucket. Además, la plataforma ofrece muchas funcionalidades prácticas como la gestión de pedidos, gestión de recursos, docker support, soporte de todos los lenguajes de programación conocidos, almacenamiento seguro en caché, análisis de datos con estadísticas y completos conceptos de seguridad.
- **Microsoft Team Foundation Server:** Herramienta colaborativa para proyectos de software que tienen que planificarse, desarrollarse y finalmente ser gestionados de manera conjunta.
- **Codeship:** La versión gratuita permite realizar un almacenamiento de caché eficiente y pruebas de builds simultáneas en contenedores compartidos y preconfigurados.
- **Concourse:** Sistema open source de automatización constante

Fases del pipeline de continuous delivery

- Cuando se produce una modificación en el código, se activa el pipeline de continuous delivery y se ejecutan las pruebas. Sus fases son:
- **Fase de commit (Commit Stage):** Se ejecutan tests de la versión del software, se desarrollan sus componentes y, cuando es necesario, se compilan. Además, se llevan a cabo las pruebas unitarias pertinentes. Si se superan con éxito todas las pruebas, la fase se da por finalizada.
- **Fase de aceptación (Acceptance Test Stage):** Se ejecutan los tests de aceptación, es decir las pruebas de integración (para comprobar si la interacción entre los componentes funciona) y las pruebas necesarias del sistema (para comprobar si el software funciona del lado del usuario). También se pueden ejecutar pruebas de rendimiento y otros tests que ponen a prueba requisitos no funcionales del software.
- En caso de que se constaten errores o complicaciones durante alguna de las fases comentadas, se creará documentación al respecto y, cuando sea necesario, se deberá enviar feedback al desarrollador.
- Las pruebas manuales se realizan en función de las necesidades concretas de cada caso.
- Si se completan todas las pruebas y el feedback recibido es positivo, ha llegado el momento de instalar el paquete manualmente en el sistema de destino.

Fases del pipeline de continuous delivery



Continuous integration vs. continuous delivery

- En Continuous integration se realizan automatización del proceso de pruebas. Por lo tanto, el pipeline es un componente compartido con el continuous delivery. En cambio, el continuos delivery es un término que abarca más elementos, incluyendo el proceso de liberación del software como un proceso automatizado. La entrega continua complementa al modelo de continuous integration e involucra al usuario final ya que entrega el producto y, simultáneamente, ejecuta las pruebas pertinentes.

Continuous integration (CI)	Continuous delivery (CD)
Proceso de pruebas automatizado que revisa en profundidad cada modificación realizada en el código fuente.	Abarca más que el proceso de pruebas e incluye al proceso de entrega. Las nuevas características y las modificaciones realizadas en el código llegan automáticamente al usuario final.
El equipo tiene que ejecutar pruebas automatizadas cada vez que se incluye una nueva característica, mejora o se produce una modificación en el código.	Las pruebas tienen que ser realmente eficaces en el CD porque los resultados se entregan directamente al usuario final.
Requiere un servidor de integración dedicado y continuo que controle y ejecute las pruebas automatizadas.	La instalación en el sistema de destino también debe ser lo más automatizada posible, lo que plantea mayores exigencias al servidor.
Los desarrolladores tienen que fusionar las modificaciones del código con mucha frecuencia y de forma continua.	Los desarrolladores tienen que mantener una buena comunicación con el cliente y saber explicar de forma clara cómo funciona el software.
Requiere un uso relativamente elevado de recursos si se quiere garantizar la calidad del producto en el momento de la entrega.	En el caso del CD el esfuerzo es aún mayor pero el producto puede ser entregado mucho antes tras haber sido sometido a pruebas “reales”.
El desarrollo en sí es más eficiente, pero es necesario pausarlo más a menudo debido a las liberaciones manuales.	Permite realizar un desarrollo continuo porque el proceso de liberación está automatizado en gran medida.