Integración Contínua

Integración contínua (CI)

Es la **práctica** de **automatizar la integración de los cambios** de código de varios contribuidores **en un único proyecto** de software.

Los desarrolladores **combinan los cambios en el código en un repositorio central** de forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas.

La integración contínua **se refiere** en su mayoría **a la fase de creación o integración del proceso de publicación de software** y conlleva un **componente de automatización y un componente cultural.**

Los **objetivos** clave de la integración continua consisten en **encontrar y arreglar errores con mayor rapidez**, **mejorar la calidad de software** y **reducir el tiempo que se tarda en validar y publicar nuevas actualizaciones** de software.

Integración contínua (CI)

Es una de las **principales prácticas recomendadas de DevOps**, que permite a los desarrolladores fusionar con frecuencia los cambios de código en un repositorio central donde luego se ejecutan las compilaciones y pruebas. Las herramientas automatizadas sirven para verificar que el nuevo código es correcto antes de la integración.

¿Por qué es necesaria la I.C.?

Anteriormente, era común que los **desarrolladores** de un equipo **trabajasen aislados durante un largo periodo de tiempo y solo intentasen combinar los cambios en la versión maestra una vez que habían completado el trabajo.**
Como consecuencia, **la combinación de los cambios en el código resultaba difícil y ardua, además de dar lugar a la acumulación de errores durante mucho tiempo que no se corregían**.
Estos factores hacían que resultase **más difícil proporcionar las actualizaciones a los clientes con rapidez**.

¿En qué consiste la integración continua?

Con la integración continua, **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, los desarrolladores **pueden** 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. La entrega continua amplía la integración continua al implementar todos los cambios en el código en un entorno de pruebas y/o de producción después de la fase de creación.

Beneficios de la integración continua

Mejore la productividad de desarrollo

La integración continua 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.

Beneficios de la integración continua

Encuentre y arregle 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.**

Beneficios de la integración continua

Entregue las actualizaciones con mayor rapidez

La integración continua le permite a su equipo **entregar actualizaciones a los clientes con mayor rapidez y frecuencia.**

Cómo configurar la integración contínua

La I.C. es una práctica propia de las metodologías ágiles y DevOps en la que los desarrolladores integran los cambios de código desde el primer momento y de manera regular en la rama principal o el repositorio del código.

El objetivo es reducir el riesgo de enfrentarse al “infierno de la integración” esperando a que llegue al final de un proyecto o un sprint para fusionar el trabajo de todos los desarrolladores.

Dado que la C.I. automatiza la implementación, ayuda a los equipos a **cumplir los requisitos empresariales**, a **mejorar la calidad del código** y a **incrementar el nivel de seguridad.**

Cómo configurar la integración contínua

Una de las grandes ventajas de adoptar CI es que **te ahorrará tiempo durante el ciclo de desarrollo, al identificar y abordar conflictos desde las etapas más tempranas.**

Es una gran manera de **reducir el tiempo dedicado a la corrección de errores y la regresión,** al poner más **énfasis en contar con un buen conjunto de pruebas.**

Por último, a**yuda a extender el conocimiento del código base y las funciones que estás desarrollando para los clientes.**

El primer paso en tu camino hacia la integración continua es configurar pruebas automatizadas…

Introducción a las pruebas automatizadas

Para disfrutar de todas las ventajas de la IC, **deberás automatizar tus pruebas para poder ejecutarlas para cada cambio realizado en el repositorio principal.**

Se deben ejecutar pruebas **en cada rama del repositorio y no solo centrarse en la rama principal.** De esta manera, podrás detectar incidencias a tiempo y minimizar las interrupciones que pueda sufrir tu equipo.

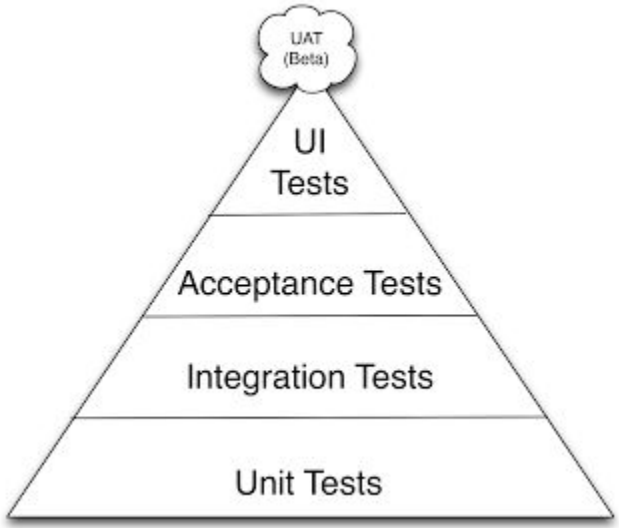
Hay muchos tipos de pruebas implementadas, pero no es necesario hacer todo a la vez si acabas de empezar. P**uedes comenzar con unas pruebas unitarias e ir ampliando la cobertura con el tiempo.**

Introducción a las pruebas automatizadas

- **Las pruebas unitarias** son de **alcance limitado** y **suelen verificar el comportamiento de métodos** o **funciones** individuales.
- Las **pruebas de integración** garantizan que varios **componentes se comporten de forma correcta en conjunto.** Esto **puede involucrar varias clases,** así como pruebas de integración con otros servicios.
- **Las pruebas de aceptación** son similares a las de integración, pero **se centran en los casos de negocio** más que en los componentes en sí.
- **Las pruebas de interfaz de usuario** permiten asegurarte de que **la aplicación funciona correctamente desde la perspectiva del usuario.**

Pirámide de pruebas

No todas las pruebas son iguales, y puedes ver las compensaciones que harás con la pirámide de pruebas.



Pruebas Unitarias vs. Interfaz de Usuario

Las pruebas unitarias son **rápidas y baratas** de implementar, ya que en su mayoría hacen comprobaciones en pequeñas piezas de código.

Por otro lado, **la implementación de las pruebas de interfaz de usuario** será **compleja y su ejecución, lenta.** Esto es así porque **suelen necesitar un entorno completo y múltiples servicios** para emular comportamientos en navegadores o dispositivos móviles.

Por esto, **puede que quieras limitar el número de pruebas de interfaz de usuario complejas y confiar en buenas pruebas unitarias para agilizar la compilación y dar feedback a los desarrolladores lo antes posible.**

Uso de la cobertura de código para encontrar código no sometido a pruebas

La cobertura de código es una m**edida porcentual en las pruebas de software** que mide el grado en que el código fuente de un programa ha sido comprobado.
Es **comúnmente utilizada en pruebas de caja blanca,** como las pruebas unitarias, en las que sí se tiene acceso al código y estructura del software que se está testeando.
Cuando adoptes **pruebas automatizadas**, es recomendable **combinarlas** con una herramienta de **cobertura de pruebas** que te permita **saber qué parte del código base cubre tu conjunto de pruebas.**
Está bien aspirar a una cobertura superior al 80 %, pero hay que tener cuidado con no confundir un porcentaje de cobertura elevado con un buen conjunto de pruebas. **Una herramienta de cobertura de código te ayudará a encontrar código no sometido a pruebas, pero es la calidad de las pruebas la que marcará la diferencia al fin y al cabo.**

Si acabas de empezar, no tengas prisa por alcanzar una cobertura del 100 % del código base. En lugar de eso, **utiliza una herramienta de cobertura que te permita identificar partes críticas de la aplicación que aún no tienen pruebas**, y empieza por ahí.

La refactorización da la oportunidad de añadir pruebas

Si estás a punto de realizar cambios significativos en tu aplicación, deberías empezar escribiendo pruebas de aceptación alrededor de las funciones que pueden verse afectadas.

Esto te dará una red de seguridad para asegurarte de que el comportamiento original no se vea afectado después de refactorizar código o de añadir nuevas funciones.

(Referencia a TDD -primero falla el código, luego corrijo el código y por último refactorizar-)

Adopción de la integración continua

Automatizar las pruebas es una parte importante de la CI, pero con eso no basta.

Es posible que debas cambiar la cultura del equipo para que los desarrolladores no trabajen durante días en una función sin fusionar los cambios en la rama principal; también tendrás que imponer la filosofía de las compilaciones con "luz verde".

Integra temprano y a menudo

Tanto si utilizas el desarrollo por tronco como ramas de funciones, es importante que los desarrolladores integren sus cambios lo antes posible en el repositorio principal. Si dejas que el código se quede en una rama o en la estación de trabajo del desarrollador durante mucho tiempo, corres el riesgo de tener demasiados conflictos cuando por fin decidas hacer la fusión en la rama principal.

Si la integración se hace desde el primer momento, se reduce el alcance de los cambios y será más fácil identificar los conflictos. La otra ventaja es que facilita el uso compartido del conocimiento entre desarrolladores, al hacer más absorbibles los cambios.

Compilaciones con "luz verde" en todo momento

Si un desarrollador rompe la compilación de la rama principal, arreglarla es prioritario. Cuantos más cambios entren en la compilación mientras está rota, más difícil será determinar qué es lo que la ha roto (y también correrás el riesgo de introducir más fallos).

Merece la pena dedicar tiempo al conjunto de pruebas, para que los fallos salgan rápidamente y poder dar feedback lo antes posible al desarrollador que envió los cambios. Puedes dividir las pruebas para que las más rápidas (las pruebas unitarias, por ejemplo) se ejecuten antes que las de larga ejecución. Si tu conjunto de pruebas siempre tarda mucho tiempo en fallar, perderás tiempo de los desarrolladores, ya que tendrán que cambiar de contexto para retomar un trabajo anterior y resolver los errores.

Recuerda configurar notificaciones para que los desarrolladores reciban alertas en cuanto se rompa la compilación. Además, puedes dar un paso más allá y mostrar el estado de tus ramas principales en un panel que todos puedan ver.

Escribe pruebas dentro de tus historias

Deberás asegurarte de que cada función que se desarrolle tenga pruebas automatizadas. Puede parecerte que esto ralentiza el desarrollo, pero, en realidad, reducirá drásticamente el tiempo que el equipo dedica a corregir regresiones o errores introducidos con cada iteración. También podrás hacer cambios en tu código base con confianza, ya que tu conjunto de pruebas podrá verificar rápidamente que todas las funciones desarrolladas anteriormente funcionan como se esperaba.

Para escribir buenas pruebas, tendrás que asegurarte de que los desarrolladores participan desde el principio en la definición de las historias de usuario. Es una manera excelente de conocer mejor los requisitos empresariales y facilitar la relación con los gestores de productos. Incluso puedes comenzar a escribir las pruebas antes de implementar el código al que se aplicarán.

Cuando corrijas errores, escribe pruebas

Si tienes ya un código base como si acabas de empezar, seguro que tendrás errores en las publicaciones. Recuerda añadir pruebas cuando los resuelvas para evitar que vuelvan a ocurrir.

Con la CI, los ingenieros de control de calidad podrán escalar la calidad

Otra función que cambiará con la adopción de la CI y de la automatización es la de los ingenieros de control de calidad. **Ya no tendrán que probar a mano capacidades poco relevantes de la aplicación y podrán dedicar más tiempo a proporcionar herramientas que respalden a los desarrolladores y a ayudarles a adoptar las estrategias de prueba adecuadas.**

En cuanto empieces a adoptar la integración continua, los ingenieros de control de calidad **podrán centrarse en facilitar pruebas con mejores herramientas y conjuntos de datos, y en ayudar a los desarrolladores a mejorar su capacidad de escribir mejor código**.

Seguirá habiendo pruebas exploratorias para casos de uso complejos, pero esto debería ser una parte menos relevante de su trabajo.

Integración continua en cinco pasos

1. **Empieza escribiendo pruebas para las partes críticas de tu código base.**

2. **Obtén un servicio de CI que ejecute esas pruebas automáticamente cada vez que se hace un envío al repositorio principal.**

3. **Asegúrate de que tu equipo integra los cambios a diario.**

4. **Corrige la compilación en cuanto se rompa.**

5. **Escribe pruebas para cada nueva historia que implementes.**

Integración Contínua Vs. Entrega Contínua

Los desarrolladores que emplean la integración continua vuelven a fusionar sus cambios en la rama principal con la mayor frecuencia posible. Los cambios del desarrollador se validan creando una compilación y sometiéndola a pruebas automatizadas. Al hacerlo, se evitan los retos que supone la integración y que pueden surgir al esperar al día de la publicación para fusionar los cambios en la rama de publicación.

La **integración continua** hace especial hincapié en la **automatización de pruebas para comprobar que la aplicación no genera errores** cuando se integran nuevas confirmaciones en la rama principal.

La **entrega continua** es una extensión de la integración continua, ya que **implementa automáticamente todos los cambios de código en un entorno de pruebas o producción** después de la fase de compilación.

Esto significa que, además de las pruebas automatizadas, cuentas con un proceso de publicación automatizado y puedes implementar la aplicación en cualquier momento haciendo clic en un botón.

Integración Contínua Vs. Entrega Contínua

Con la entrega continua puedes publicar cada día, cada semana, cada quincena o lo que se adapte a tus necesidades empresariales. Sin embargo, si realmente quieres aprovechar las ventajas de la entrega continua, debes implementar la producción lo antes posible para asegurarte de publicar lotes pequeños cuyos problemas sean fáciles de solucionar.

La implementación continua va un paso más allá que la entrega continua. Mediante esta práctica, los cambios que pasan por todas las fases de tu canalización de producción se publican para tus clientes. No hay intervención humana y solo una prueba fallida evitará implementar un nuevo cambio en la producción.

Despliegue continuo

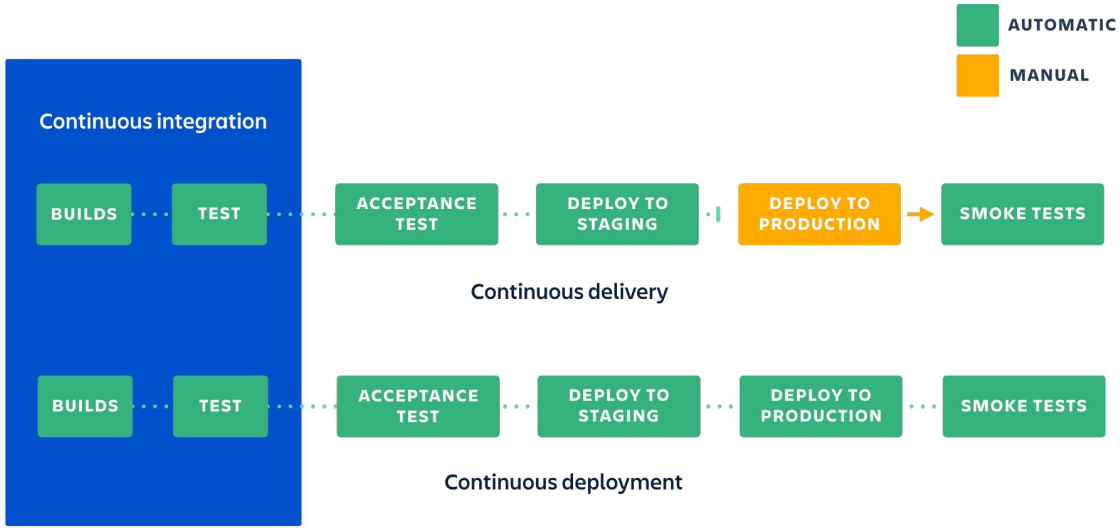
La implementación continua es una excelente manera de acelerar el ciclo de feedback con tus clientes y aliviar la presión del equipo, puesto que ya no hay un día de publicación.

Los desarrolladores pueden centrarse en crear software y ver los resultados minutos después de haber terminado.

Dicho en pocas palabras, la integración continua forma parte tanto de la entrega continua como de la implementación continua.

Lla implementación continua es como la entrega continua, salvo en que las versiones que se publican de forma automática.

Despliegue continuo



¿Cuáles son las ventajas de cada práctica?

Integración continua

Qué se necesita (coste)

- Tu equipo tendrá que **escribir pruebas automatizadas** para cada nueva función, mejora o corrección de errores.
- Se necesita un **servidor de integración continua** que pueda monitorizar el repositorio principal y ejecutar las pruebas automáticamente para las nuevas confirmaciones enviadas.
- Los desarrolladores deben **fusionar sus cambios con la mayor frecuencia** posible, al menos, una vez al día.

Integración continua

Qué se gana

- La producción recibe **menos errores a medida que las pruebas automatizadas van capturando regresiones de forma anticipada.**
- Como todas **las incidencias de integración se han resuelto** pronto, es **fácil compilar la publicación**.
- **Menos cambio de contexto**, ya que los desarrolladores reciben un aviso en cuanto se rompe la compilación y pueden trabajar en la resolución del problema antes de pasar a la siguiente tarea.
- Los **costes** de las pruebas **se reducen** drásticamente: **el servidor de CI puede ejecutar cientos de pruebas en cuestión de segundos.**
- Tu **equipo de control de calidad dedica menos tiempo a las pruebas** y puede centrarse en mejoras significativas de la política corporativa de calidad.

Qué se necesita (coste)

- Necesitas una **base sólida en integración continua** y tu conjunto de pruebas debe cubrir lo suficiente de tu base de código.
- Las implementaciones deben automatizarse. **El desencadenador sigue siendo manual**, pero, una vez iniciada la implementación, no es necesaria la intervención humana.
- Lo más probable es que tu equipo tenga que **adoptar marcas de función para que las funciones incompletas no afecten a los clientes en la producción.**

Entrega continua

Qué se gana

- Se ha eliminado la complejidad de implementar software. Los equipos ya no tienen que estar días preparándose para una publicación.
- Puedes publicar más a menudo, de modo que se acelera el ciclo de feedback con tus clientes.
- Hay mucha menos presión sobre las decisiones cuando se trata de cambios pequeños, lo cual facilita las iteraciones más rápidas.

Despliegue continuo

Qué se necesita (coste)

- Tu cultura de pruebas debe ser la mejor. La calidad de tu conjunto de pruebas determinará la calidad de tus versiones.
- Tu proceso de documentación deberá seguir el ritmo de las implementaciones.
- Las marcas de función se convierten en una parte inherente del proceso de publicación de cambios significativos para asegurar que puedes coordinarte con otros departamentos (soporte, marketing, RR. PP., etc.).

Despliegue continuo

Qué se gana

- Puedes desarrollar más rápido al no tener que detener el desarrollo de las publicaciones. Las canalizaciones de implementaciones se desencadenan automáticamente con cada cambio.
- Las publicaciones son menos arriesgadas y más fáciles de corregir en caso de que surjan problemas al implementar pequeños lotes de cambios.
- Los clientes ven una corriente continua de mejoras, y la calidad aumenta cada día, en lugar de cada mes, trimestre o año.

La integración continua conlleva una mejora de la calidad del software

Calidad de proceso

En primer lugar, con la integración continua se le da más visibilidad al proceso de desarrollo en general, a todos los pasos que se siguen desde que se empieza a programar un requisito del cliente hasta que está en producción.

Así, todo el mundo sabe las fases por las que va pasando el código, y el estado del software en cada momento (si compila, si pasa las pruebas, en qué entorno está cada versión, qué versión se está probando etc).

También le damos visibilidad a la estrategia de gestión de configuración, política de ramas del control de versiones y tagueos, entre otras cosas.

Si todo el equipo no conocía esto bien, o las estrategias estaban poco definidas, con la integración continua habrá que solucionarlo.

Y esto ya es un avance importante, porque en muchas empresas a las que vamos, todas estas cosas imprescindibles no son conocidas por todo el equipo, ni están claras ni bien definidas.

La integración continua conlleva una mejora de la calidad del software

Calidad de producto

Por otro lado, también se mejora la calidad del producto software.

El principal objetivo de la integración continua es detectar los errores lo más pronto posible, en fases tempranas del desarrollo, para poder solucionarlos rápidamente.

Así se introducen varios tipos de pruebas y comprobaciones, minimizando los riesgos, y haciendo que el software tenga menos bugs que si no realizáramos integración continua.

Por otra parte, en fases ya avanzadas de la integración continua se suelen lanzar inspecciones continuas de código, análisis periódicos para detectar problemas de calidad en él.

Los desarrolladores tendrán que mejorar esas deficiencias, e incluso en ciertas ocasiones, se puede impedir que los desarrolladores suban el código al control de versiones si no cumplen los estándares de calidad definidos por la empresa.

La integración continua conlleva una mejora de la calidad del software

Calidad de las personas

Por último, también se mejora la calidad del equipo. Si no sabía, el equipo acaba aprendiendo a hacer distintos tipos de pruebas (unitarias, de integración), mejores prácticas de programación y en general a desarrollar código de mayor calidad.

Además, tener todo este proceso de desarrollo claro y tener todas estas comprobaciones para minimizar los riesgos, le da más confianza al equipo.

Así es capaz de afrontar nuevos retos, mejoras y cambios y está más motivado. Además de que le ahorra muchísimo tiempo, porque automatizamos procesos repetitivos (por ejemplo ciertas pruebas de regresión), dejando más tiempo para hacer otras cosas.

Y todo esto, al final se acaba reflejando en una mejora de cara al cliente y por supuesto en términos económicos para la empresa.

¿Y cuáles son esas buenas prácticas, esos componentes que intervienen en la integración continua?

1. **“Práctica de desarrollo software”**

2. **“los miembros del equipo integran su trabajo frecuentemente”**

3. **“cada integración se verifica con un build automático”**

4. **“que incluye la ejecución de pruebas”**

5. **“para detectar errores de integración tan pronto como sea posible”**