Engineer

integración continua

integrar a diario con compilaciones y pruebas automáticas

Qué es

CI es la abreviatura de continuous integration, integración continua: una forma de trabajar en la que todas las personas de un equipo incorporan sus cambios a una misma línea de desarrollo muchas veces por semana y una máquina compila y prueba el resultado cada vez, para que un error aparezca cuando quien lo cometió todavía recuerda haberlo hecho. El nombre completo, integración continua, se utiliza tanto como la abreviatura.

La palabra se usa para dos cosas distintas, y separarlas evita mucha confusión. La primera es la práctica, que trata de las personas. El artículo de Martin Fowler sobre el tema, la fuente que la mayoría de los equipos acaba citando, la describe como un conjunto de hábitos: mantener una línea principal, automatizar la compilación, hacer que la compilación se pruebe a sí misma, enviar cambios a la línea principal al menos a diario, compilar cada envío, corregir una compilación fallida de inmediato, mantener la compilación rápida, ocultar el trabajo sin terminar en lugar de conservarlo en una rama y permitir que todo el mundo vea el estado de la compilación. El artículo sitúa el origen del nombre en Extreme Programming y señala que Grady Booch utilizó la expresión antes.

El segundo significado es la máquina. Cuando alguien dice «la CI está en rojo», se refiere al servicio que ejecutó la compilación e informó de un fallo. GitHub Actions, GitLab CI/CD y sus equivalentes comparten la misma estructura. Un archivo del repositorio describe el trabajo: `.github/workflows/*.yml` en GitHub o `.gitlab-ci.yml` en la raíz en GitLab. Un evento lo inicia, como un envío de cambios, una solicitud de incorporación o fusión, una programación horaria o una persona que pulsa un botón. Un runner, una máquina reservada para ello, extrae el commit, ejecuta los comandos del archivo e informa del resultado, correcto o fallido, asociado al commit.

Un servidor por sí solo no es integración continua. Un equipo que conserva trabajo en ramas durante un mes y tiene un servidor que compila cada rama ha automatizado sus compilaciones, algo útil, pero la integración sigue ocurriendo de golpe al final. La práctica es fusionar con frecuencia; la máquina es lo que hace que fusionar con frecuencia sea lo bastante seguro como para seguir haciéndolo.

La entrega continua y el despliegue continuo son los pasos siguientes. La entrega mantiene cada compilación correcta lista para publicarse, y el despliegue publica cada una automáticamente. Ambas dependen de que CI esté ya implantada, de ahí que las tres se escriban juntas tan a menudo como CI/CD.

Utilice el concepto en una decisión

Aplique «¿Qué es la integración continua?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cambios probados, publicados y reversibles, para que lleguen a sus clientes un martes por la tarde en vez de un viernes por la noche.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Despliegue GIS SaaS con GitHub Actions — despliegue, datos e informes automatizados

Despliegue GIS SaaS con GitHub Actions Póngase en contacto →

Por qué importa

El coste de fusionar dos líneas de trabajo crece más deprisa que el tiempo que pasan separadas. Dos personas que modificaron ayer el mismo módulo pueden resolver el conflicto en unos minutos porque ambas recuerdan lo que pretendían. Dos personas cuyas líneas de trabajo se separaron hace seis semanas reconstruyen la intención a partir de diferencias de código. Fusionar fragmentos pequeños a menudo mantiene pequeño cada conflicto, y ahí está la mayor parte del beneficio antes de que intervenga ninguna máquina.

La compilación en cada envío añade otra cosa: una máquina limpia que confirma el resultado. «Funciona en mi portátil» es una afirmación sobre un portátil concreto, con sus propias bibliotecas instaladas y archivos sobrantes. Un runner que parte siempre de una copia de trabajo limpia reproduce el resultado o muestra qué hacía especial a ese portátil.

El valor está en la rapidez. El artículo de Fowler fija un objetivo de unos diez minutos para la compilación principal, y el razonamiento se sostiene en la práctica. Un fallo comunicado diez minutos después de un envío vuelve a quien

lo provocó cuando el cambio aún está en su pantalla. Un fallo comunicado a la mañana siguiente llega a quien mire en ese momento y exige antes una hora de arqueología.

La visibilidad también importa. Cuando todo el mundo ve el estado de la línea principal, nadie empieza trabajo nuevo sobre una compilación fallida sin saberlo, y corregirla se convierte en la primera tarea del equipo en lugar de en el problema de otra persona.

CI tampoco tiene por qué ser un servicio alojado. pre-commit ejecuta las comprobaciones de un proyecto como hooks de Git antes de cada commit o envío, y su documentación muestra cómo ejecutar el mismo conjunto en un servidor con `pre-commit run --all-files`. Task da nombre a cada comprobación en un `Taskfile.yml`, para que una persona y un runner invoquen el mismo comando. Las comprobaciones de este sitio siguen ese modelo. Se ejecutan localmente, donde se escriben las páginas, y antes de cada envío se ejecuta la comprobación completa, sin ningún servicio alojado de compilación entre medias.

Cómo se aplica en la práctica

Se recuperaron aproximadamente 4.000 horas cuando un equipo dejó de publicar a mano. Entregar software había dependido de pasos manuales que cada persona realizaba de una forma ligeramente distinta, hasta que la compilación, las pruebas y el despliegue automatizados los sustituyeron. La ruta desde un cambio hasta producción pasó a ser estándar en lugar de improvisada, y las publicaciones se hicieron más rápidas y fiables.

En un proyecto con una base de datos de larga vida, el esquema se describe dos veces: una mediante la configuración inicial y otra mediante 1.022 migraciones, numeradas de 036 a 1102. Un trabajo de CI construye la base de datos de ambas formas en cada cambio, una a partir del esquema para una base nueva y otra reproduciendo todas las migraciones en orden, y compara la estructura y los datos de inicialización. Cuando las dos construcciones difieren, el trabajo pasa a rojo e imprime las diferencias, de modo que un entorno nuevo y uno de larga vida son la misma base de datos por comprobación, no por suposición.

CI también puede mantener de acuerdo las dos mitades de una aplicación. En otro proyecto, la descripción de la API y el cliente del frontend se generan en una cadena que comienza en la base de datos, y solo el cliente tiene 44.076 líneas de TypeScript tipado. Una comprobación de CI contrasta lo que envía el frontend

con lo que espera la API y hace fallar la compilación cuando divergen. Para eso existe CI: un campo que cambia solo en un lado falla en una compilación minutos después del cambio, en lugar de llegar a un usuario como un error en el navegador.

Dónde surgen los problemas

La mayoría de las configuraciones de CI se deteriora en lugar de romperse de golpe. Cada uno de estos problemas empieza siendo pequeño y termina con una compilación que el equipo ha aprendido a sortear.

La compilación tarda más que el trabajo. Cuando un envío tarda cuarenta minutos en devolver un resultado, las personas dejan de esperarlo, agrupan varios cambios en un envío y envían con menos frecuencia: lo contrario de la práctica. Mida en qué se va el tiempo antes de añadir hardware. Almacene en caché las dependencias, ejecute primero las comprobaciones rápidas y traslade las baterías lentas a una segunda etapa o a una ejecución programada, para que la compilación principal se mantenga cerca de los diez minutos.

La línea principal permanece en rojo todo un día. Una compilación fallida que nadie corrige se convierte en lo normal, y un segundo fallo llega inadvertido detrás del primero. La regla para evitarlo es clara: quien la rompió la corrige ahora o revierte el cambio y lo arregla por su cuenta. Una línea principal correcta es la base sobre la que trabaja el resto.

Las ramas viven semanas mientras el servidor compila cada una. Todas aparecen en verde y los problemas llegan de golpe el día de fusionarlas. Eso es compilación automatizada sin integración. Integre los cambios en la línea principal al menos a diario y mantenga desactivadas las funciones sin terminar en lugar de mantenerlas separadas.

El servidor comprueba una lista y el portátil otra. Dos listas de comandos mantenidas a mano se distancian, y un cambio pasa localmente y falla en el servidor, o al revés, hasta que nadie confía en ninguna. Dé a las comprobaciones un único punto de entrada e invóquelo desde ambos lugares. Con pre-commit, el servidor puede ejecutar exactamente los mismos hooks que cada desarrollador:

```
# .gitlab-ci.yml
check:
    image: python:3.13
    script:
        - pip install pre-commit
        - pre-commit run --all-files --show-diff-on-failure
```

Una prueba inestable enseña a pulsar «reintentar». Una prueba que falla una ejecución de cada veinte por motivos ajenos al cambio enseña al equipo a repetirla en lugar de leer. Después, un fallo real recibe el mismo tratamiento. Aísle la prueba, corríjala o elimínela, y trate un botón de reintento que se pulsa a menudo como un hallazgo por sí mismo.

Lecturas adicionales

- Integración continua — el extenso artículo de Martin Fowler: cada práctica, la compilación de diez minutos y el origen del nombre.
- Comprender GitHub Actions — workflows, eventos, trabajos y runners según los define GitHub.
- GitLab CI/CD — el archivo `.gitlab-ci.yml`, pipelines, etapas, trabajos y runners en los términos de GitLab.
- pre-commit — comprobaciones declaradas una vez en un archivo de configuración, ejecutadas como hooks de Git y de nuevo como etapa de un servidor de CI.
- Task — un `Taskfile.yml` que da un nombre a cada comprobación, para que una persona y un runner escriban el mismo comando.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es la integración continua?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/ci/