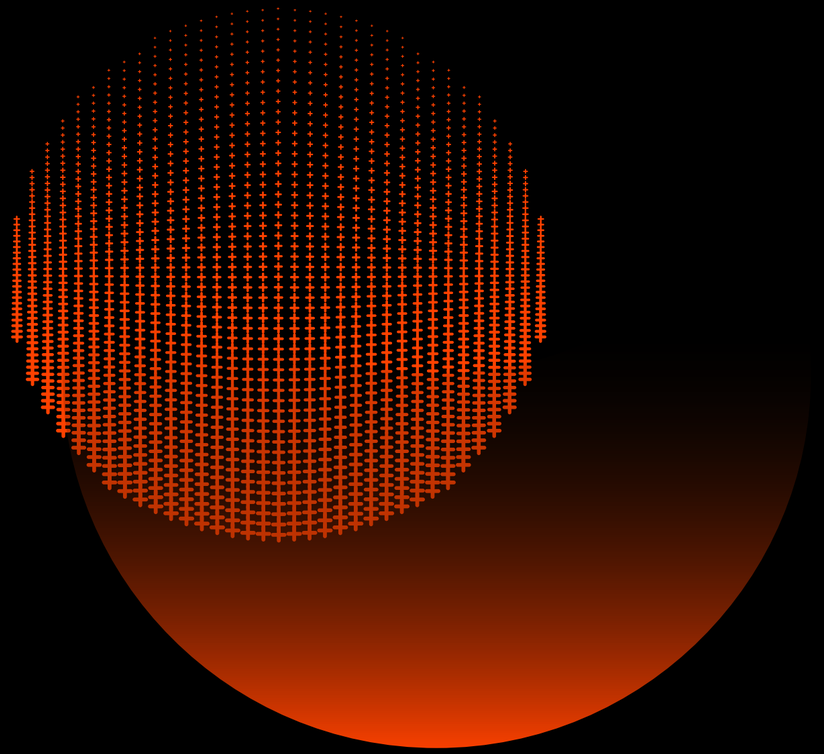
# DevOps

## E-book

Lectura - 20min

Agnostic

# índice




¿Qué es Devops?
Ventajas de la metodología DevOps
Herramientas DevOps
1

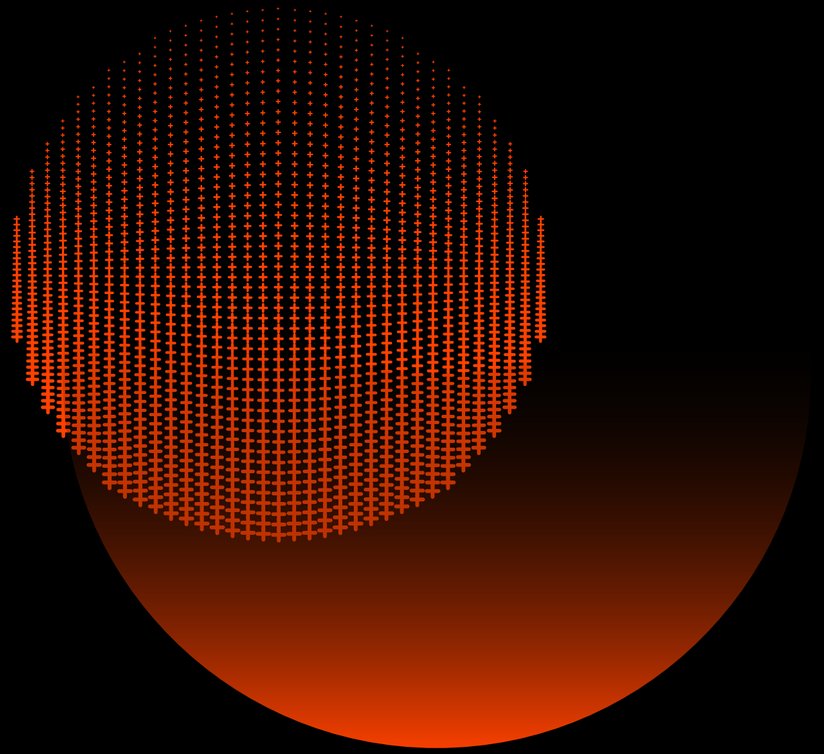
DevOps lo ayuda a crecer a través de la implementación continua
2

Desarrollo continuo
- ¿Por qué entrega continua?
- 5 Principios

1. Calidad de construcción
2. Trabajar en lotes pequeños
3. Las computadoras realizan tareas repetitivas, las personas resuelven problemas
4. Perseguir incansablemente la mejora continua
5. Todos son responsables

3

# índice




¿Qué medir al calibrar un pipeline en CD? 4

- Tiempo de espera
- Tiempo del ciclo
- Tiempo medio de recuperación
- Tiempo de resolución de defectos
- Tasa de aprobación de la prueba

Casos de uso

Las ventajas del continuous delivery

Beneficios

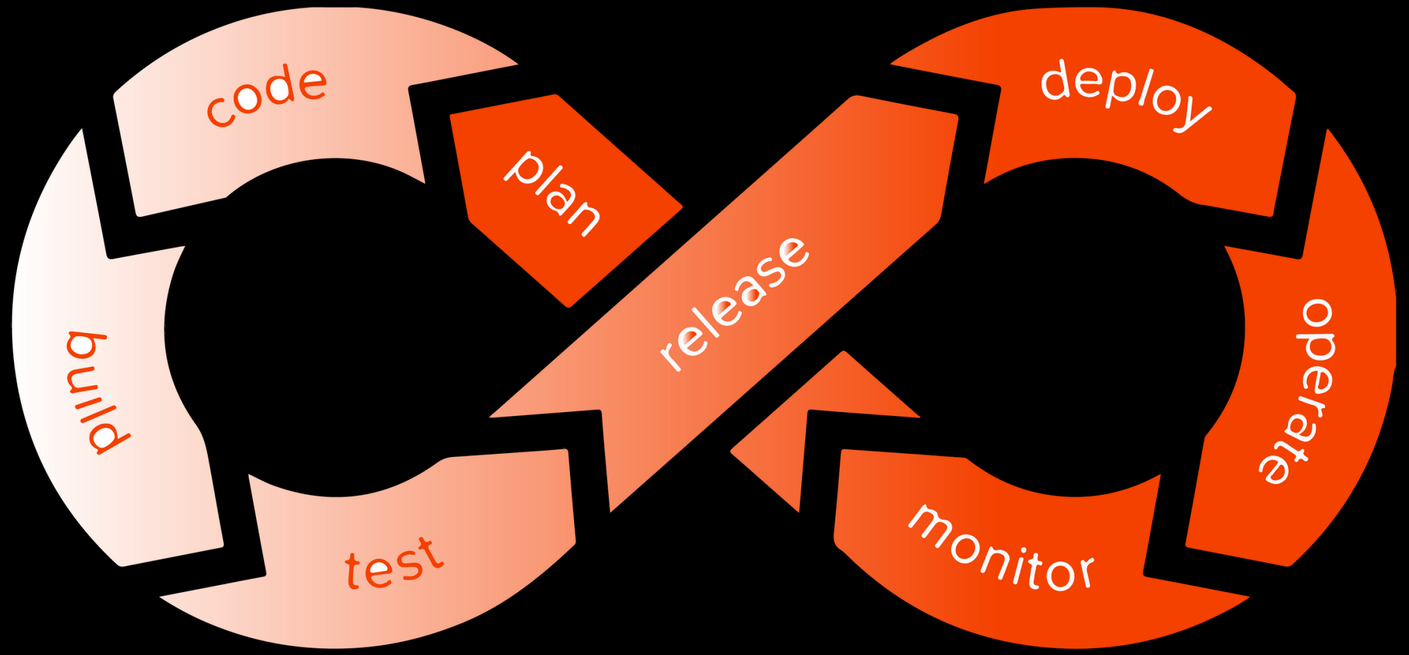
Mejores prácticas para adoptar la entrega continua 5

- Desarrollar objetivos de nivel de servicio (SLO)
- Automatizar la evaluación de SLO con quailty gates
- Automatizar todos los procesos repetibles
- Eliminar la complejidad donde puedas
- Establecer observabilidad y monitoreo continuo

# 1

# ¿Qué es DevOps?

El término "DevOps" es una combinación de las palabras "development" (desarrollo) y "operations" (operaciones), pero representa un conjunto de ideas y prácticas que van más allá de ambos conceptos, ya sea que estén juntos o separados. DevOps incluye sistemas de seguridad, maneras de trabajar en colaboración, análisis de datos, entre otras características.



DevOps describe los enfoques para agilizar los procesos con los que una idea (como una nueva función de software, una solicitud de mejora o una corrección de errores) pasa del desarrollo a la implementación, en un entorno de producción en que puede generar valor para el usuario. Estos enfoques requieren que los equipos de desarrollo y operaciones se comuniquen con frecuencia y aborden su trabajo con empatía hacia sus compañeros de equipo. También es necesario contar con capacidad de ajuste y tener flexibilidad a la hora de preparar los sistemas. Con DevOps, es posible brindar una mayor potencia a quienes más la necesitan gracias al autoservicio y la automatización. Los desarrolladores, que generalmente realizan codificaciones en un entorno de desarrollo estándar, trabajan en estrecha colaboración con los equipos de operaciones de TI para agilizar el diseño, las pruebas y el lanzamiento de los sistemas de software, sin comprometer la confiabilidad.

Por supuesto, esto implica cambios más frecuentes en el código y el uso más dinámico de la infraestructura. Las estrategias tradicionales de gestión no logran satisfacer este aumento de la demanda. Necesitará hacer algunos cambios para obtener una ventaja competitiva.

# Ventajas de la metodología DevOps

La gran adopción que las empresas están haciendo a día de hoy de DevOps está motivada por el gran número de ventajas que aporta su adopción. Entre todas esas ventajas, enumeramos a continuación las principales:

- DevOps aumenta la cooperación y el entendimiento entre los departamentos de una organización, especialmente entre los departamentos de operaciones y desarrollo.
- Incrementa la producción y el despliegue de los desarrollos software a través de sistemas de producción llamados de despliegue o entrega continua lo que se traduce en una mayor agilidad de desarrollo.
- El aumento de la agilidad se traduce en una mejora de la calidad gracias a la verificación, casi continua, de los productos desde las etapas más tempranas del desarrollo y reduciendo los posibles errores.
- La automatización de los procedimientos de control contra amenazas y vulnerabilidades bajo metodologías DevOps ofrece una mayor seguridad de las aplicaciones desarrolladas.
- La entrega continua permite una mayor flexibilidad y adaptación al entorno que los sistemas de desarrollos tradicionales en los que el tiempo de despliegue es mayor.

# Herramientas DevOps

Las implementaciones eficaces de software bajo la metodología DevOps normalmente se basan en un conjunto integrado de soluciones, o bien, una lista de herramientas independientes que tienen por objetivo la automatización, la reducción del número de errores, el aumento de la agilidad del equipo y su escalado por pequeño que éste sea.

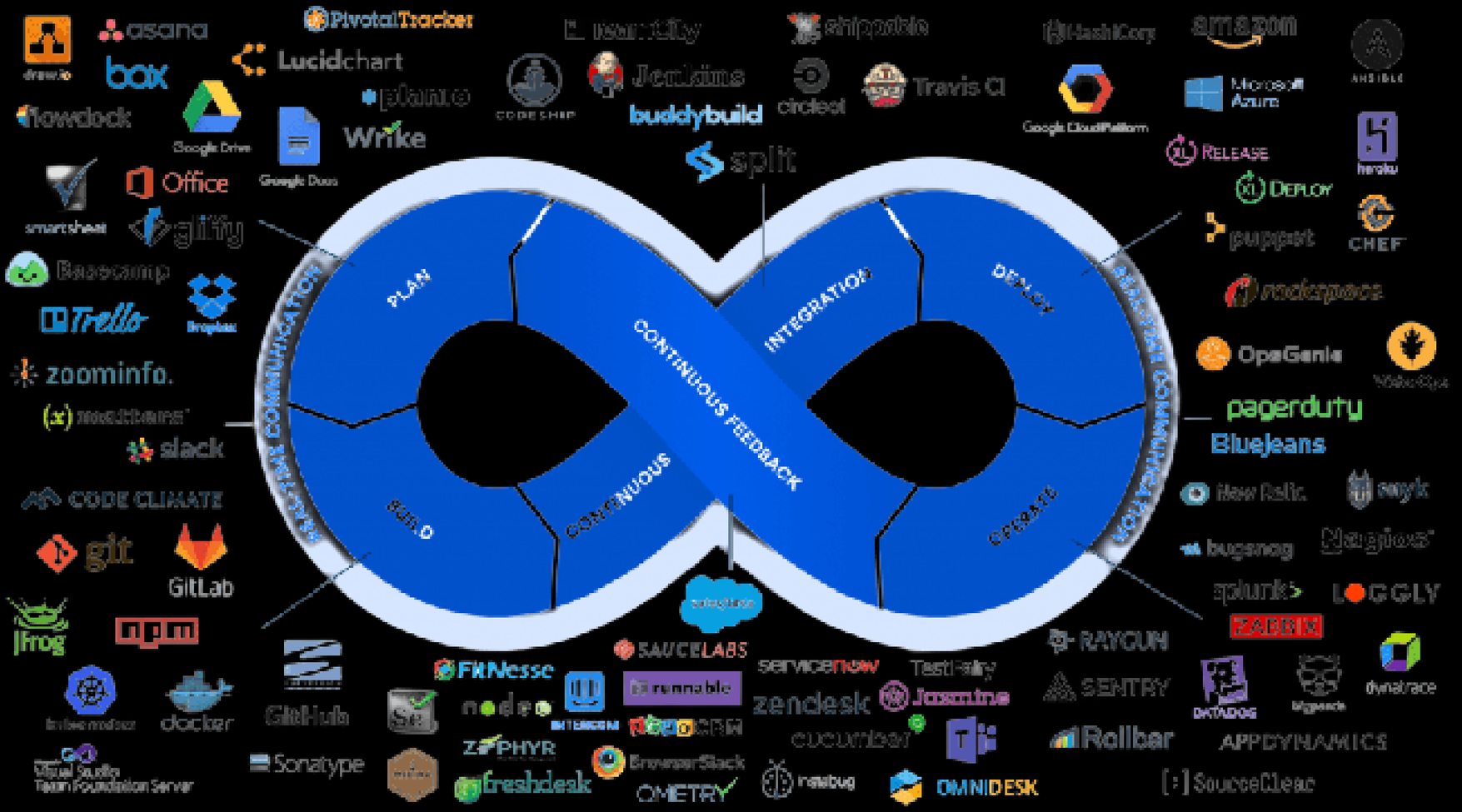
Entre la gran multitud de herramientas existentes en el mercado para implantar metodologías ágiles de desarrollo de software bajo metodología DevOps, destacamos las siguientes:

- Docker: Líder del segmento de herramientas de contenedores que permite automatizar la administración de partes del software como contenedores independientes que pueden ser ejecutados en cualquier entorno.

·Kubernetes. Orquestador que permite la administración de sistemas distribuidos complejos para desarrolladores que trabajen con contenedores Linux.

- Github: Plataforma de control de versiones de desarrollo que permite alinear los equipos operativos y de desarrollo en la misma herramienta de administración y control de un mismo código fuente.
- Jenkins: Herramienta de integración y distribución continua escrito en Java con una fuerte interfaz de usuario gráfica, que se utiliza para informar cambios, realizar pruebas en vivo y distribuir código en varias máquinas.
- Grafana: Herramienta de monitorización en código abierto que permite conectar con multitud de fuentes de datos para el análisis de datos.
- Prometheus: Conjunto de herramientas de supervisión y alertas para contenedores y microservicios gracias a un sistema altamente personalizable.

# Herramientas DevOps



2

## DevOps lo ayuda a crecer a través de la implementación continua

Uno de los principales resultados de la adopción de DevOps es un canal de integración e implementación continuas (CI/CD). La CI/CD permite distribuir aplicaciones a los clientes con frecuencia y validar la calidad del software con una intervención humana mínima.

En concreto, incorpora la automatización permanente y la supervisión constante a todo el ciclo de vida de las aplicaciones, desde las etapas de integración y prueba hasta las de distribución e implementación, para que pueda identificar y corregir los problemas y los defectos rápidamente. Este conjunto de prácticas se conoce como "canales de CI/CD" y cuenta con el respaldo de los equipos de desarrollo y de operaciones que trabajan en conjunto de manera ágil.

La CI/CD es un método para distribuir las aplicaciones a los clientes con frecuencia mediante el uso de la automatización en las etapas del desarrollo de aplicaciones. Los principales conceptos que se le atribuyen son la integración, la distribución y la implementación continuas. Se trata de una solución para los problemas que puede generar la integración del código nuevo para los equipos de desarrollo y de operaciones (también conocida como "el infierno de la integración").

En concreto, el proceso de integración y distribución continuas incorpora la automatización y la supervisión permanentes en todo el ciclo de vida de las aplicaciones, desde las etapas de integración y prueba hasta las de distribución e implementación. Este conjunto de prácticas se conoce como "canales de CI/CD" y cuenta con el respaldo de los equipos de desarrollo y de operaciones que trabajan en conjunto de manera ágil, con un enfoque de DevOps o de ingeniería de confiabilidad del sitio (SRE).

3

# Desarrollo continuo

La entrega continua es la capacidad de introducir cambios de todo tipo (incluidas nuevas funciones, cambios de configuración, correcciones de errores y experimentos) en producción o en manos de los usuarios, de forma segura, rápida y sostenible.

Nuestro objetivo es hacer que las implementaciones, ya sea de un sistema distribuido a gran escala, un entorno de producción complejo, un sistema integrado o una aplicación, sean asuntos rutinarios y predecibles que se puedan realizar bajo demanda.

Logramos todo esto asegurándonos de que nuestro código esté siempre en un estado desplegable, incluso frente a equipos de miles de desarrolladores que realizan cambios a diario. De este modo, eliminamos por completo las fases de integración, prueba y endurecimiento que tradicionalmente seguían al "desarrollo completo", así como las congelaciones de código.

## ¿Por qué entrega continua?

A menudo se supone que si queremos implementar software con más frecuencia, debemos aceptar niveles más bajos de estabilidad y confiabilidad en nuestros sistemas. De hecho, la investigación revisada por pares muestra que este no es el caso: los equipos de alto rendimiento brindan servicios de manera más rápida y confiable que su competencia de bajo rendimiento. Esto es cierto incluso en dominios altamente regulados, como los servicios financieros y el gobierno. Esta capacidad proporciona una ventaja competitiva increíble para las organizaciones que están dispuestas a invertir el esfuerzo para conseguirla.

## 5 principios

- **Calidad de construcción**

"Dejar de depender de la inspección para lograr la calidad. Elimine la necesidad de una inspección en masa incorporando calidad en el producto en primer lugar".
Es mucho más económico solucionar problemas y defectos si los encontramos de inmediato, idealmente antes de que se registren en el control de versiones, ejecutando pruebas automatizadas localmente. Encontrar defectos aguas abajo a través de la inspección (como las pruebas manuales) lleva mucho tiempo y requiere una clasificación importante. Luego debemos arreglar el defecto, tratando de recordar lo que estábamos pensando cuando presentamos el problema hace días o incluso semanas.
Crear y desarrollar bucles de retroalimentación para detectar problemas lo antes posible es un trabajo esencial e interminable en la entrega continua. Si encontramos un problema en nuestras pruebas exploratorias, no solo debemos solucionarlo, sino también preguntarnos: ¿Cómo pudimos detectar el problema con una prueba de aceptación automatizada? Cuando falla una prueba de aceptación, debemos preguntarnos: ¿Podríamos haber escrito una prueba unitaria para detectar este problema?

## 5 principios

- **Trabajar en lotes pequeños**

formados por decenas o cientos de personas.

En la entrega continua, adoptamos el enfoque opuesto y tratamos de llevar cada cambio en el control de versiones lo más posible hacia el lanzamiento, obteniendo comentarios completos lo más rápido posible.

Trabajar en lotes pequeños tiene muchos beneficios. Reduce el tiempo que se tarda en recibir comentarios sobre nuestro trabajo, facilita la clasificación y solución de problemas, aumenta la eficiencia y la motivación, y evita que sucumbamos a la falacia de los costes irrecuperables.

La razón por la que trabajamos en lotes grandes es por el gran costo fijo de entregar los cambios. Un objetivo clave de la entrega continua es cambiar la economía del proceso de entrega de software para que sea económicamente viable trabajar en lotes pequeños para que podamos obtener los muchos beneficios de este enfoque.

## 5 principios

- **Las computadoras realizan tareas repetitivas, las personas resuelven problemas**

  Una de las primeras ideas filosóficas de la tradición de Toyota es jidoka, a veces traducida como “automatización con un toque humano”. El objetivo es que las computadoras realicen tareas simples y repetitivas, como pruebas de regresión, para que los humanos puedan concentrarse en la resolución de problemas. Así, las computadoras y las personas se complementan entre sí.

  A muchas personas les preocupa que la automatización los deje sin trabajo. Este no es el objetivo. Nunca habrá escasez de trabajo en una empresa exitosa. Más bien, las personas se liberan del trabajo pesado sin sentido para centrarse en actividades de mayor valor. Esto también tiene el beneficio de mejorar la calidad, ya que los humanos son más propensos a errores cuando realizan tareas sin sentido.

- **Perseguir incansablemente la mejora continua**

  La mejora continua, o kaizen en japonés, es otra idea clave del movimiento Lean. Taiichi Ohno, una figura clave en la historia de la empresa Toyota, dijo una vez:

  Las oportunidades Kaizen son infinitas. No creas que has hecho las cosas mejor que antes y quédate tranquilo... Esto sería como el estudiante que se enorgullece porque superó a su maestro dos veces de tres en esgrima. Una vez que recoge los brotes de las ideas kaizen, es importante tener la actitud en nuestro trabajo diario de que justo debajo de una idea kaizen hay otra.

  No trate la transformación como un proyecto que se debe emprender y luego completar para que podamos volver a la normalidad. Las mejores organizaciones son aquellas en las que todo el mundo trata el trabajo de mejora como una parte esencial de su trabajo diario y en las que nadie está satisfecho con el statu quo.

## 5 principios

- **Todos son responsables**

En las organizaciones de alto desempeño, nada es “problema de otra persona”. Los desarrolladores son responsables de la calidad y la estabilidad del software que construyen. Los equipos de operaciones son responsables de ayudar a los desarrolladores a incorporar la calidad. Todos trabajan juntos para lograr los objetivos de nivel organizacional, en lugar de optimizar lo que es mejor para su equipo o departamento.

Cuando las personas realizan optimizaciones locales que reducen el rendimiento general de la organización, a menudo se debe a problemas sistémicos, como sistemas de gestión deficientes, como ciclos presupuestarios anuales, o incentivos que recompensan los comportamientos incorrectos. Un ejemplo clásico es recompensar a los desarrolladores por aumentar su velocidad o escribir más código, y recompensar a los evaluadores según la cantidad de errores que encuentren.

La mayoría de las personas quieren hacer lo correcto, pero adaptarán su comportamiento en función de cómo sean recompensados. Por lo tanto, es muy importante crear ciclos rápidos de retroalimentación de las cosas que realmente importan: cómo reaccionan los clientes a lo que construimos para ellos y el impacto en nuestra organización.

4

## ¿Qué medir al calibrar un pipeline en CD?

Cada pipeline en CD es único para la organización a la que sirve y varía según las arquitecturas, los entornos informáticos, los conjuntos de herramientas y los requisitos corporativos, industriales y reglamentarios. Aquí hay algunas métricas comunes que los equipos deben rastrear para cada canalización en CD para ayudarlo a evaluar su eficacia.

## Tiempo de espera

El tiempo entre el momento en que se identifica un requisito y el momento en que se lanza la funcionalidad final. Al evaluar los plazos de entrega de cada incremento de software, puede determinar si debe dividir los entregables en paquetes más pequeños y reducir aún más el tiempo de lanzamiento.

## Tiempo del ciclo

CD permite lanzamientos más rápidos y frecuentes, y los eventos dentro de este proceso se miden en tiempos de ciclo, el tiempo que tarda un proceso en completarse. Puede medir el tiempo del ciclo para todo el ciclo de vida de la entrega de software o para procesos más pequeños y tácticos en su canalización. El seguimiento de los tiempos de ciclo le permite descubrir formas de mejorar la eficiencia. Agrupe características similares o complementarias y planifique de una manera que reduzca el tiempo del ciclo.

## Tiempo medio de recuperación

El tiempo medio de recuperación (MTTR) es el tiempo que tarda un sistema en revertir las actualizaciones. Si una actualización falla, este es el tiempo que se tarda en volver a un producto funcional o factible. Para acortar el MTTR, los desarrolladores necesitan datos para solucionar problemas. El CD puede reducir el tiempo que lleva restaurar el servicio al proporcionar datos sobre su compilación rápidamente, creando los ciclos de retroalimentación rápidos que los equipos de DevOps necesitan para recuperarse rápidamente.

## Tiempo de resolución de defectos

El tiempo de resolución de defectos mide el período desde que se descubre un problema hasta que los desarrolladores lo resuelven. Si su ciclo de resolución de defectos es demasiado largo, puede afectar significativamente su tasa de abandono de clientes. Esta medida puede ayudarlo a optimizar su proceso de CD para mejorar los tiempos de respuesta.

## Tasa de aprobación de la prueba

Medir la tasa de aprobación de la prueba le da una idea de la calidad de su producto en función del porcentaje de casos de prueba que pasan. También ayuda a medir qué tan bien funcionan sus pruebas automatizadas y con qué frecuencia los cambios en el código están causando fallas en sus pruebas. El CD depende de las pruebas automatizadas, y la tasa de aprobación de las pruebas lo ayuda a afinar los métodos de prueba y mejorar la eficacia del ciclo de comentarios de su desarrollador.

## Casos de uso

- Lanzamientos de bajo riesgo. El objetivo principal de la entrega continua es hacer que las implementaciones de software sean eventos sencillos y de bajo riesgo que se puedan realizar en cualquier momento y bajo demanda. Mediante la aplicación de patrones como implementaciones azul-verde, es relativamente sencillo lograr implementaciones sin tiempo de inactividad que son indetectables para los usuarios.

- Tiempo de comercialización más rápido. No es raro que la fase de integración y prueba/corrección del ciclo de vida tradicional de entrega de software por fases consuma semanas o incluso meses. Cuando los equipos trabajan juntos para automatizar los procesos de compilación e implementación, aprovisionamiento del entorno y pruebas de regresión, los desarrolladores pueden incorporar pruebas de integración y regresión en su trabajo diario y eliminar por completo estas fases. También evitamos las grandes cantidades de reelaboración que plagan el enfoque por etapas.

- Mejor calidad. Cuando los desarrolladores tienen herramientas automatizadas que descubren regresiones en cuestión de minutos, los equipos tienen la libertad de centrar su esfuerzo en la investigación de usuarios y actividades de prueba de nivel superior, como pruebas exploratorias, pruebas de usabilidad y pruebas de rendimiento y seguridad. Mediante la creación de una canalización de implementación, estas actividades se pueden realizar de forma continua a lo largo del proceso de entrega, lo que garantiza que la calidad se incorpore a los productos y servicios desde el principio

## Casos de uso

- Costos mas bajos. Cualquier producto o servicio de software exitoso evolucionará significativamente a lo largo de su vida. Al invertir en compilación, prueba, implementación y automatización del entorno, reducimos sustancialmente el costo de realizar y entregar cambios incrementales al software al eliminar muchos de los costos fijos asociados con el proceso de lanzamiento.

- Mejores productos. La entrega continua hace que sea económico trabajar en lotes pequeños. Esto significa que podemos obtener comentarios de los usuarios a lo largo del ciclo de vida de entrega en función del software en funcionamiento. Las técnicas como las pruebas A/B nos permiten adoptar un enfoque basado en hipótesis para el desarrollo de productos mediante el cual podemos probar ideas con los usuarios antes de desarrollar funciones completas. Esto significa que podemos evitar los 2/3 de las características que construimos y que brindan valor cero o negativo a nuestros negocios.

- Equipos más felices. La investigación revisada por pares ha demostrado que la entrega continua hace que los lanzamientos sean menos dolorosos y reduce el agotamiento del equipo. Además, cuando lanzamos con más frecuencia, los equipos de entrega de software pueden interactuar más activamente con los usuarios, aprender qué ideas funcionan y cuáles no, y ver de primera mano los resultados del trabajo que han realizado. Al eliminar las actividades dolorosas de bajo valor asociadas con la entrega de software, podemos centrarnos en lo que más nos importa: deleitar continuamente a nuestros usuarios.

## Las ventajas del continuous delivery

Antes, el desarrollo de software funcionaba de una manera diferente: el producto final solo se entregaba si todas las funcionalidades estaban totalmente desarrolladas, funcionaban a la perfección y no se detectaban fallos importantes cuando se realizaban las pruebas de calidad. Así que el desarrollador tenía que entregar posteriormente parches o actualizaciones cada cierto tiempo.

Gracias al continuous delivery, el cliente recibe el producto en una fase más temprana del desarrollo en la que todavía no ha sido terminado. Esta pre-entrega suele incluir la funcionalidad estructural del software para que el cliente pueda probarla en un entorno real. De esta manera, el propio cliente (o el tester de software) juega un papel muy importante dentro del proceso de control de calidad.

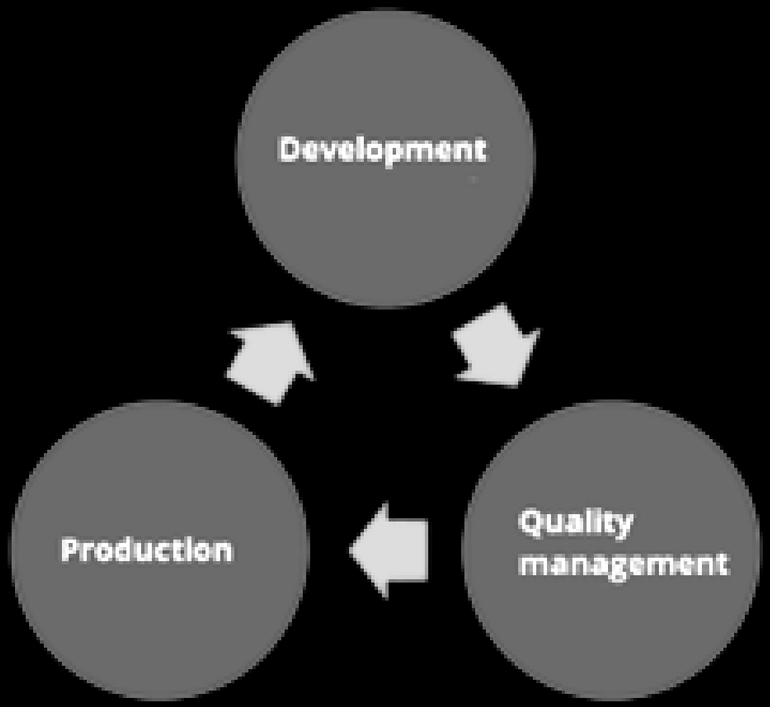
Gracias al feedback recibido, el desarrollador puede mejorar las funcionalidades del producto durante la fase de desarrollo. Además, recibe información muy valiosa que le aporta una idea clara sobre qué funcionalidad debería desarrollar a continuación.

Antes de que existiera el continuous delivery, este proceso era realmente tedioso y, a menudo, dejaba descontentas a las dos partes: por un lado, el cliente normalmente espera que un producto final cumpla con sus requisitos y deseos y, por otro, el desarrollador todavía no sabe exactamente cuáles son esos requisitos y deseos. Si se entabla una comunicación sobre el estado de desarrollo del producto en una fase más temprana, es más fácil tener en cuenta los requisitos del cliente y no cometer errores. Es posible ilustrar en forma de ciclo este principio

# Las ventajas del continuous delivery

Como muestra la imagen, las tres áreas que incluyen el desarrollo, el control de calidad y la producción no se reemplazan por un único proceso, sino que están constantemente interconectadas. De esta forma, un producto pasa por cada una de las fases individuales repetidamente y recibe mejoras continuas. Cuando tenemos que trabajar con muchos clientes al mismo tiempo, es imposible conseguir algo así si no contamos con procesos automatizados. Y en este punto es precisamente en el que interviene la entrega continua ya que se encarga de automatizar todo el proceso.

Gracias al continuous delivery, es posible comprobar los procesos y mejoras implementados sobre el software (es decir, todos los cambios realizados sobre el código fuente) en tiempo real con el fin de conseguir un feedback. Si un cambio genera efectos secundarios no deseados, será posible detectarlos rápidamente, lo que te permitirá implementar las acciones necesarias en una fase temprana del desarrollo. Este punto es realmente una mejora importante porque facilita, por ejemplo, la detección de bugs dentro del código. Sin la entrega continua, detectar dónde se encuentra el error se convierte en un trabajo realmente tedioso.



## Las ventajas del continuous delivery

Hasta que llega al cliente, el software está en una fase intermedia, el pipeline de continuous delivery. En este pipeline se llevan a cabo pruebas manuales y automáticas. Cada fase de pruebas conlleva la aparición de una nueva versión del software (que suele llamarse “versión beta” o en algunas ocasiones “Nightly Build”, es decir la última versión creada de forma automática) que, a su vez, entra en el pipeline. Hasta que se hayan pasado todas las pruebas y se reciba un feedback positivo, no se crea una versión “estable” y no se libera oficialmente el producto (este proceso se conoce como “liberación” y también incluye a la propia aplicación publicada). Así aumentan considerablemente las probabilidades de que el cliente reciba un producto sin bugs.

## Beneficios

- Es mucho más fácil encontrar y eliminar los errores presentes en el software durante la etapa de desarrollo.
- En el pasado, el desarrollo de software requería muchísimo esfuerzo. Gracias al continuous delivery, los desarrolladores pueden concentrarse exclusivamente en el desarrollo.
- Gracias al pipeline de continuous delivery es mucho más sencillo para los desarrolladores llevar a cabo la resolución de problemas.
- Otros procesos de prueba (como las pruebas alfa y beta) conllevan más costes asociados.
- Se pueden dedicar más recursos a la etapa conceptual que a la técnica durante el control de calidad para mejorar el software.
- 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.
- 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.
- 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.

5

## Mejores prácticas para adoptar la entrega continua

La creación de un proceso de liberación rápido y confiable requiere la implementación de controles de calidad, prácticas de registro y soluciones de monitoreo.
Estas son algunas de las mejores prácticas a tener en cuenta para automatizar la entrega de manera efectiva.

## Desarrollar objetivos de nivel de servicio (SLO)

Los objetivos de nivel de servicio (SLO) son un conjunto de criterios específicos que el software debe cumplir para satisfacer los requisitos de las partes interesadas. Se definen dentro de un acuerdo de nivel de servicio (SLA) basado en indicadores de nivel de servicio (SLI). Las organizaciones pueden entregar versiones de mayor calidad más rápido una vez que se establecen los SLO y se prueban continuamente en cada etapa del ciclo de vida del desarrollo.
Establezca un entorno de múltiples etapas con puertas de calidad integradas basadas en SLO que pueden orquestar su flujo de trabajo de CD e integrar sus herramientas de prueba para pruebas de rendimiento, pruebas de caos y más. Cuando puede estar seguro de que su código resiste las evaluaciones de calidad basadas en SLO, puede implementarlo con confianza.

## Mejores prácticas para adoptar la entrega continua

Automatizar la evaluación de SLO con quailty gates

Una vez que se implementan los SLO, tiene un marco para comenzar a automatizar la evaluación de pruebas. Una excelente manera de implementar esta automatización es diseñar puertas de calidad, que establecen criterios de éxito específicos que el software debe cumplir antes de pasar a la siguiente etapa en la tubería de entrega. Las quality gates toman datos de múltiples herramientas de prueba, como pruebas de rendimiento, pruebas de integración y datos de observabilidad, y los evalúan según los criterios especificados por sus SLO. Esto crea un proceso consistente que es fácilmente repetible y fácil de ajustar. La asistencia de IA ayuda a identificar rápidamente por qué una prueba puede haber fallado y cómo solucionarlo.

## Automatizar todos los procesos repetibles

Un error común para cualquier persona nueva en DevOps es pasar por alto uno o dos pasos manuales al automatizar la canalización de entrega, lo que requiere alguna acción antes, durante o después de implementar una versión. Esto puede suceder por muchas razones, incluida la dificultad percibida o el costo inicial, o el hábito de involucrar a un miembro clave del personal para tomar una decisión. Sin embargo, para que los procesos de CD mejoren y escalen continuamente, es importante que automatice todos los procesos repetibles durante el desarrollo, las pruebas y la implementación del software.

A medida que evalúe sus procesos, asegúrese de automatizar todas las pruebas, cambios de configuración, controles de calidad y recopilación de dependencias, prestando especial atención a cualquier proceso repetible, incluso si se trata de aprobaciones o pasos tradicionalmente manuales.

## Eliminar la complejidad donde puedas

La complejidad de una canalización crece con la cantidad de herramientas que utiliza. La gestión del proceso requiere la racionalización de las numerosas tareas implicadas en la comprobación del producto final.
En muchos casos, la automatización se construye gradualmente y según sea necesario. Como resultado, el código de automatización a menudo se distribuye entre varias herramientas, lo que dificulta su uso y supervisión. Una solución que puede ofrecer un marco de orquestación basado en SLO es útil para ayudar a las organizaciones a optimizar y automatizar el proceso de entrega continua. Tener un plano de control central para canalizaciones de CD permite a los equipos mantener todo su código de automatización en un solo lugar.

## Establecer observabilidad y monitoreo continuo

Establecer la observabilidad de extremo a extremo para una canalización de CD nativa de la nube altamente dinámica es crucial para que los equipos de DevOps y SRE entreguen aplicaciones que cumplan con los SLO acordados. Para evitar puntos ciegos y establecer un análisis eficiente de la causa raíz de los problemas en cualquier etapa de la canalización, los equipos necesitan una telemetría confiable que comience con métricas, registros y seguimientos, agregue datos de la experiencia del usuario y también incluya el contexto completo de los procesos ascendentes y descendentes. . La capacidad de capturar detalles a nivel de código también es fundamental para la depuración y la resolución de problemas.
A medida que las aplicaciones y los servicios se distribuyen más y se basan en más tecnologías de código abierto, esta telemetría puede provenir de muchas fuentes dispares, que requieren su propia instrumentación. Las fuentes de monitoreo deben ser automáticas y continuas, y lo suficientemente flexibles para manejar actualizaciones continuas.

Agnostic

www.agnosticit.com

Impulsando la
transformación digital
de las empresas