Evolución más rápida con la entrega continua

Mark Mansour

Mejora continua y automatización de software

Hace más de diez años, en Amazon, emprendimos un proyecto para comprender cuán rápido nuestros equipos convertían las ideas en sistemas de producción de alta calidad. Esto nos llevó a medir el rendimiento del software para poder mejorar la velocidad de la ejecución. Descubrimos que nos llevaba, en promedio, 16 días desde el registro del código hasta la producción. En Amazon, los equipos comenzaban con una idea y, por lo general, demoraban un día y medio en escribir el código para dar vida a esa idea. Nos demorábamos menos de una hora en crear e implementar el código nuevo. Pasábamos el resto del tiempo, casi 14 días, esperando que los miembros de los equipos comenzaran con la compilación, realizaran las implementaciones y ejecutaran las pruebas. Al final del proyecto, recomendamos automatizar los procesos posteriores al registro, con el fin de mejorar la velocidad de ejecución. El objetivo era eliminar los retrasos mientras manteníamos, o incluso mejorábamos, la calidad.

En el fondo de esta recomendación, había un programa de mejora continua que tenía el objetivo de aumentar la velocidad de ejecución. Basamos la decisión de mejorar la velocidad de ejecución en el principio de liderazgo de la exigencia de los más altos estándares. Este principio implica tener altos estándares en todo momento, elevar el nivel de exigencia constantemente y ofrecer productos, servicios y procesos de alta calidad. Nuestros principios de liderazgo describen la forma en que Amazon dirige los negocios, cómo los líderes lideran y cómo basamos nuestras decisiones en el cliente.

Amazon ya ha creado herramientas de desarrollo de software para que los ingenieros de software sean más productivos. Creamos nuestro propio sistema de compilación alojado y centralizado, Brazil, que ejecuta una serie de comandos en un servidor con el objetivo de generar un artefacto que se pueda implementar. En ese momento, Brazil no seguía los cambios en el código fuente. Una persona tenía que iniciar la compilación. También teníamos nuestro propio sistema de implementación, Apollo, donde se debía cargar un artefacto de compilación como parte de la iniciación de una implementación. El interés del sector en la entrega continua nos inspiró a crear nuestro propio sistema, Pipelines, para automatizar el proceso de entrega de software entre Brazil y Apollo.

Pipelines: nuestra herramienta de implementación continua

Comenzamos un programa piloto de automatización del proceso de entrega de software para una pequeña cantidad de equipos. Cuando finalizamos, el equipo piloto principal había logrado una reducción del 90 % en el tiempo total que llevaba pasar del registro a la producción.

El proyecto validó el concepto de la canalización como un medio para que los equipos pudieran definir todos los pasos necesarios para poner un sistema de software a disposición de los clientes. El primer paso de la canalización es crear un artefacto. Luego, la canalización ejecuta ese artefacto de compilación a través de una serie de pasos hasta que se lanza al mercado para todos los clientes. Utilizamos canalizaciones para reducir el riesgo de que un cambio nuevo en el código afecte a los clientes. Cada paso de la canalización debe aumentar la confianza en que

el artefacto de compilación no contiene defectos. Si un defecto llega a la producción, queremos que la producción vuelva a un buen estado tan rápido como sea posible.

Cuando lanzamos Pipelines, solo podía modelar un solo proceso de lanzamiento por aplicación. Esta limitación impulsaba la uniformidad, la estandarización y la simplificación de los procesos de lanzamiento de un equipo. Esto dio como resultado menos defectos. Antes de comenzar a usar las canalizaciones, era común que los equipos tuviesen diferentes procesos de lanzamiento para la corrección de errores y el lanzamiento de las características principales. Cuando los otros equipos vieron el éxito de aquellos que habían probado la entrega automatizada, comenzaron a cambiar sus procesos de lanzamiento administrados de manera manual por las canalizaciones, de manera que ellos también pudieran mejorar su uniformidad. Los equipos que solían tener diferentes procesos de lanzamiento pasaron a tener un proceso estandarizado que todos utilizaban. Además, mientras trasladaban sus procesos de lanzamiento a una herramienta, los miembros del equipo a menudo revisaban su enfoque y encontraban formas de simplificar el proceso.

El equipo de Pipelines tenía objetivos anuales de incrementar el uso a través de una "adopción atractiva". En otras palabras, necesitaban que el producto fuera tan bueno que la gente exigiera usarlo. Medimos la cantidad de equipos que usaban una canalización para implementar su software en la producción y clasificamos las canalizaciones según su nivel de automatización. Observamos que los equipos se proponían usar una canalización para lanzar software y avanzar hacia lanzamientos totalmente automatizados. Sin embargo, nos dimos cuenta de que, en algunas organizaciones, la forma en que medíamos la calidad podía llevar a los equipos a automatizar sus procesos de lanzamiento sin realizar ninguna prueba.

La respuesta a la pregunta "¿cuántas pruebas son suficientes?" es una decisión personal. Requiere que el equipo comprenda el contexto en el que opera. Para resolver esta situación, utilizamos otro principio de liderazgo: la propiedad. Este principio implica pensar en el futuro y no sacrificar el valor a largo plazo por los resultados a corto plazo. Los equipos de software de Amazon tienen altos estándares para las pruebas y ponen mucho empeño en ellas, porque ser propietarios de un producto también implica asumir las consecuencias de cualquier defecto en dicho producto. Si un problema llegara a afectar a los clientes, los miembros de ese pequeño equipo de software con un único hilo serían los encargados de abordar el problema y solucionarlo en tiempo real. La tensión que existe entre aumentar la velocidad de ejecución y responder a los problemas en la producción se traduce en la motivación de los equipos para efectuar las pruebas correspondientes. Sin embargo, si invertimos demasiado en las pruebas, puede que no tengamos éxito porque otros fueron más rápidos que nosotros. Siempre buscamos la manera de mejorar los procesos de lanzamiento de software sin obstaculizar el negocio.

Otro problema que enfrentábamos era que los equipos no aprendían las prácticas recomendadas para el lanzamiento de software de los demás. Se recomienda que los equipos pequeños con un único hilo trabajen de forma autónoma, lo cual implicaba que los ingenieros resolvieran sus problemas de implementación de manera independiente. Cuando encontraban una solución que satisficiera sus necesidades de lanzamiento de software, divulgaban la técnica entre los demás ingenieros a través de listas de correo, reuniones operativas y otros canales de comunicación. Había dos problemas con este estilo de comunicación. Primero, estos canales de

comunicación implican un mayor esfuerzo, lo que significa que no todo el mundo aprendía las técnicas nuevas. Segundo, los líderes que incentivaban a sus equipos a adoptar las nuevas prácticas recomendadas no tenían forma de saber si los equipos habían realizado el trabajo necesario para poder adoptarla. Comprendimos que debíamos ayudar a todos los ingenieros a obtener acceso a las prácticas recomendadas que habíamos aprendido, así como brindar a los líderes la capacidad de identificar las canalizaciones que requerían atención.

La solución fue mecanizar el aprendizaje agregando verificaciones para las prácticas recomendadas en las herramientas que usábamos para crear y lanzar software. Nos percatamos de que una práctica recomendada para una organización podía no serlo para otra, por lo que permitimos que estas verificaciones se configuraran en función de cada organización. Las verificaciones de las prácticas recomendadas a nivel organizacional brindaban a los líderes la capacidad de personalizar los procesos de lanzamiento a fin de satisfacer las necesidades de sus negocios. Los líderes que querían fomentar o imponer una nueva práctica recomendada podían comenzar proporcionando una advertencia en las herramientas que usaban los ingenieros a diario. Si colocaban mensajes en las herramientas, prácticamente garantizaban que los miembros del equipo estuvieran al tanto de la práctica recomendada y cuándo surtiría efecto. Nos dimos cuenta de que, si los equipos tenían tiempo para aprender y debatir acerca de las nuevas prácticas recomendadas, la organización tenía la oportunidad de repetir y mejorar las verificaciones de prácticas recomendadas. Finalmente, esto condujo a una mejora en la calidad de las prácticas recomendadas y un aumento de su adopción por parte de la comunidad de ingenieros.

Identificábamos de manera sistemática las prácticas recomendadas que podíamos aplicar. Un grupo de ingenieros con mayor experiencia enumeraron las razones comunes de las fallas en los lanzamientos. Identificaron los pasos que habrían permitido que el lanzamiento funcionara correctamente. Luego, utilizamos esa lista para diseñar un conjunto de verificaciones de prácticas recomendadas. A través de este proceso, comprendimos que, aunque queríamos que las revisiones nuevas del software llegaran a los clientes de inmediato, sin esfuerzo y sin disminuir la disponibilidad, priorizábamos, en primer lugar, la disponibilidad, luego, la velocidad y, por último, la simplificación para los ingenieros.

Reducción del riesgo de que un defecto llegue a los clientes

Se prevé que todos los ingenieros, en algún momento, introducirán un defecto en alguno de los sistemas. Los procesos de lanzamiento, incluidos los sistemas de canalización e implementación, se deben diseñar para poder identificar esos errores lo más rápido posible y evitar que afecten a los clientes. Debemos asegurarnos de que los procesos de lanzamiento tengan la configuración correcta y que le artefacto de compilación funcione según lo previsto.

Higiene en la implementación: la forma más básica de probar las implementaciones asegura que el artefacto recientemente implementado pueda inicializarse y responder al trabajo. Como parte del flujo de trabajo posterior a la implementación, efectuamos verificaciones rápidas que garantizan que el artefacto que se implementó recientemente ha comenzado a funcionar y abastece el tráfico. Por ejemplo, utilizamos enlaces del ciclo de vida en el archivo AppSpec de

AWS CodeDeploy para desencadenar scripts simples que detengan, inicien y validen la implementación. También verificamos que tengamos suficiente capacidad para abastecer el tráfico de los clientes. Hemos diseñado técnicas, como la de la cantidad mínima de hosts en buen estado en CodeDeploy, con el fin de validar que siempre contamos con la capacidad suficiente para atender a los clientes. Por último, si el motor de implementación puede detectar un error, debe deshacer el cambio para minimizar el tiempo que los clientes están expuestos al defecto.

Pruebas antes de la producción: una de las prácticas recomendadas de Amazon es automatizar las pruebas de unidades, integración y preproducción y agregarlas a la canalización. Insistimos en ejecutar pruebas de carga y seguridad, con un énfasis en agregar estas pruebas a las canalizaciones. Cuando hablamos de pruebas de unidades nos referimos a todas las pruebas que usted podría efectuar a su equipo de compilación, incluidas las verificaciones de estilo, la cobertura y la complejidad del código, entre otras. Consideramos que las pruebas de integración incluyen todas las pruebas fuera del entorno, como la introducción de fallas, las pruebas automatizadas del navegador y otras similares. Existen muchos artículos excelentes acerca de las pruebas de unidades e integración, por lo que no ahondaré más en este tema.

Nuestras pruebas de unidades e integración tienen como objetivo verificar que nuestro artefacto de compilación funcione correctamente. Mientras más validaciones realicemos, menor será el riesgo de exponer un defecto a los clientes. Con el fin de disminuir el tiempo que lleva poner un producto a disposición de los clientes, intentamos detectar los defectos en el proceso de lanzamiento lo más pronto posible. En general, esto significa que si las pruebas son más pequeñas y rápidas, recibirá información acerca de cualquier error en los cambios en menos tiempo.

En Amazon, también utilizamos una técnica que llamamos *prueba de preproducción*. Un entorno de preproducción es el último lugar en el que se efectúan las pruebas antes de implementar nuestros cambios en la producción. En una prueba del entorno de preproducción, se utiliza la configuración de producción del sistema de manera que este actúe exactamente igual a un sistema de producción. Esta estrategia tiene dos beneficios. Primero, los entornos de preproducción evalúan la configuración de producción a fin de asegurarse de que el servicio puede conectarse de manera correcta con todos los recursos de producción, incluidos los almacenes de datos de producción. Segundo, esta estrategia se asegura de que el sistema interactúe de manera correcta con las API de los servicios de producción de los que depende. A los entornos de preproducción solo los usa el equipo propietario de ese servicio. Además, estos entornos nunca reciben tráfico de los clientes. La realización de las pruebas de preproducción incrementa nuestra confianza en que los mismos códigos y la misma configuración funcionarán en la producción.

Validación en la producción: cuando lanzamos un código a nuestros clientes, no lo hacemos para todos al mismo tiempo. El alcance del impacto que causaría lanzar un defecto a todos los clientes al mismo tiempo sería demasiado grande. En cambio, implementamos los lanzamientos en celdas, una instancia completamente independiente de un servicio. Cuando implementamos los cambios en nuestro primer conjunto de clientes de nuestra primera celda, somos cuidadosos en extremo. Solo dejamos que una pequeña cantidad de clientes observen el nuevo cambio, y reunimos información acerca de si el nuevo código funciona o no. Monitoreamos la cantidad de errores que nuestros servicios emiten luego de una

implementación Canary. Si el índice de errores se eleva, automáticamente revertimos el cambio. Por ejemplo, podemos esperar a registrar 3000 puntos de datos positivos sin ningún punto de datos negativo antes de continuar con la implementación.

Es posible que surja una complicación si las pruebas automatizadas se pierden un caso de uso. Nos esforzamos en detectar todos los errores con nuestras pruebas estructuradas y repetibles, ya sean automáticas o manuales. Sin embargo, aun cuando nos esforzamos al máximo, siempre se puede escapar un defecto. Para evaluar nuestras pruebas, dejamos el nuevo cambio en la producción por un periodo de tiempo determinado para observar si un miembro que no sea del equipo encuentra algún problema. Hemos pasado mucho tiempo debatiendo si deberíamos dejar que los cambios se queden en la producción o cuánto tiempo deberíamos esperar luego de efectuar una implementación Canary antes de implementarla en el resto del grupo de implementación. Muchos de nuestros equipos han decidido esperar un periodo de tiempo determinado, además de reunir los puntos de datos positivos, para poder continuar con nuestra rutina de implementación. La cantidad de tiempo que una canalización espera depende en gran medida del equipo. Algunos equipos esperan durante horas y, otros, durante minutos. Mientras mayor sea el impacto y el tiempo que se requiera para solucionar el problema, más lento será el proceso de lanzamiento.

Luego de tomar confianza con la primera celda, iremos exponiendo el nuevo cambio de código a más y más clientes hasta que se complete el lanzamiento. Tal como hicimos con la implementación Canary, esperamos a ganar confianza en la implementación efectuada en la primera celda para recién seguir con la celda siguiente. A medida que ganamos más confianza en el artefacto de compilación, vamos reduciendo el tiempo que dedicamos para verificar el cambio de código. Esto da como resultado un patrón en el que nuestro objetivo es llegar desde registro del código al primer cliente en la producción lo más rápido posible. Sin embargo, cuando ya estamos en la etapa de producción, vamos lanzando de a poco el nuevo código a los clientes, trabajando para ganar confianza adicional a medida que, de manera gradual, aceleramos el resto de nuestras implementaciones.

Para asegurarnos de que nuestros sistemas de producción siguen atendiendo las solicitudes de los clientes, generamos tráfico sintético en nuestros sistemas. Queremos obtener información rápida si nuestro servicio no funciona de forma correcta, por lo que ejecutamos nuestras pruebas sintéticas al menos a cada minuto. Diseñamos las pruebas sintéticas para asegurarnos de que nuestros procesos de ejecución funcionen correctamente y de que se prueben todas las dependencias, lo que a menudo implica evaluar todas las API con acceso público.

Control del momento en que se lanza el software: para controlar la seguridad de los lanzamientos de software, hemos creado mecanismos que nos permiten controlar la velocidad con la que se mueven los cambios por la canalización. Usamos métricas, ventanas de tiempo y verificaciones de seguridad para controlar cuándo se lanza el software.

Las canalizaciones se pueden configurar para evitar una implementación cuando se enciende una alarma debido a un cambio en las métricas. Utilizamos métricas de forma generalizada, y tenemos alarmas acerca del estado de nuestros sistemas, del estado de nuestras celdas, de las zonas de disponibilidad y de las regiones y de casi todo lo demás que se pueda imaginar. Configuramos nuestras canalizaciones para detener la implementación del código cuando una

métrica importante disparare alguna alarma. Sin embargo, a veces un equipo necesita implementar una corrección para abordar la alarma del sistema. En este caso, permitimos que nuestros equipos anulen las alarmas con el fin de evitar que los cambios se muevan a través de la canalización.

Nuestras canalizaciones pueden especificar una ventana de tiempo en la que se permite que los cambios avancen a través de una canalización. Los equipos pueden encontrar sus propias ventanas de tiempo para restringir cuándo los cambios deben alcanzar a los clientes. Los equipos de AWS prefieren lanzar el software cuando cuentan con la cantidad suficiente de personas que pueden responder a un problema causado por una implementación y mitigarlo de manera rápida. Para que esto se cumpla, los equipos, por lo general, configuran sus ventanas de tiempo de manera que realicen implementaciones únicamente durante las horas de trabajo. Otros equipos de Amazon prefieren lanzar el software cuando el tráfico de los clientes es bajo. Estas ventanas de tiempo se pueden anular, si es necesario.

También tenemos la capacidad de detener una canalización según los contenidos del artefacto de compilación. Por ejemplo, podemos bloquear un artefacto de compilación que contenga un paquete defectuoso conocido o una referencia de Git específica. Hemos utilizado esta característica cuando descubrimos que un cambio en un paquete implicaba un retroceso en el rendimiento. Si solo eliminábamos el paquete de nuestro repositorio de paquetes, las canalizaciones que ya contenían el paquete defectuoso seguirían implementando ese cambio con errores en los sistemas de los clientes.

¿Cómo abordamos la velocidad de nuestra ejecución?

Descubrimos que los equipos están ansiosos por aprovechar la automatización. A todos nos motiva la idea de crear y lanzar funciones que mejoren la vida de nuestros clientes, y la entrega continua lo hace posible. Hemos visto que la automatización permite a los ingenieros recuperar el tiempo al eliminar el frustrante trabajo manual, que es laborioso y propenso a los errores. Hemos mostrado que la implementación continua tiene un efecto positivo en la calidad. También hemos visto que la automatización permite a los equipos lanzar, con frecuencia, un cambio a la vez, lo que facilita la identificación de las regresiones.

Cuando los sistemas son nuevos, el área que se debe evaluar suele ser comprensible por la mayoría de los miembros del equipo, lo que hace que algunas pruebas manuales sean manejables. Sin embargo, a medida que los sistemas se vuelven más complejos y los miembros del equipo cambian, el valor de la automatización aumenta. Nos gusta automatizar nuestros sistemas de modo que podamos enfocarnos en agregar valor para los clientes en lugar de administrar manualmente los procesos para eliminar esos cambios de los sistemas de los clientes.

Durante muchos años, Amazon ha llevado a cabo programas de mejora continua que se enfocan en la velocidad con la que se lanza el software para los clientes y en la seguridad de esos lanzamientos. No comenzamos con todas las verificaciones y pruebas de riesgos sobre las que he escrito en este artículo. Con el tiempo, hemos diseñado formas de identificar y mitigar los riesgos.

Los líderes empresariales de diferentes niveles de la organización son los encargados de realizar los programas de mejora continua. Esto permite que cada líder ajuste sus procesos de

lanzamiento de software para que se ajusten a los riesgos de sus negocios y a los efectos que puedan producirse en ellos. Algunos de nuestros programas de mejora continua se llevan a cabo a lo largo de grandes sectores de Amazon y, algunas veces, los líderes de organizaciones más pequeñas ejecutan sus propios programas. Sabemos que siempre existen excepciones a la regla. Nuestros sistemas han deshabilitado algunos mecanismos con el fin de no ralentizar a los equipos que necesitan una exención permanente o temporaria. Por último, nuestros equipos son propietarios del comportamiento de su software y también son responsables por invertir de forma apropiada en sus procesos de lanzamiento de software.

Comenzamos por medir dónde estaba el problema, abordarlo y repetir el proceso. Para que este trabajo fuera sostenible, debíamos hacerlo de forma gradual y disfrutar de las mejoras a lo largo del tiempo. Cuando comenzamos a utilizar canalizaciones por primera vez en Amazon, muchos equipos no confiaban en que la implementación continua funcionaría correctamente. Para que los equipos comenzaran a trabajar en ello, les recomendamos que codificaran su proceso de lanzamiento actual, incluso los pasos manuales, en una canalización. Para muchos equipos, la canalización sirvió como una interfaz visual de su proceso de lanzamiento sin promover automáticamente los artefactos de compilación en todo el proceso de lanzamiento. A medida que crecía su confianza, iniciaron la automatización poco a poco en diferentes etapas de la canalización hasta llegar al punto en que no necesitaban activar ningún paso de forma manual.

Avancemos rápido al presente. En la actualidad, los equipos de Amazon aspiran a la automatización completa cuando escriben código nuevo. Para nosotros, la automatización es la única forma la que el negocio puede seguir evolucionando.