La prueba de fuego de la CI/CD: ¿su cartera de proyectos está completamente orientada a la CI/CD?

AWSGuía prescriptiva

AWSGuía prescriptiva: La prueba de fuego de la CI/CD: ¿su cartera de proyectos está completamente orientada a la CI/CD?

Table of Contents

La prueba de fuego de la CI/CD: ¿su cartera de proyectos es totalmente de CI/CD?

Steven Guggenheimer y Ananya Koduri, de Amazon Web Services (AWS)

Agosto de 2023(historial de documentos)

¿Su oleoducto está automatizado? Es una pregunta sencilla, pero muchas organizaciones abordan la respuesta de forma demasiado sencilla. La respuesta es mucho más complicada que unasíono.

Las innovaciones tecnológicas se producen constantemente y, a veces, puede resultar difícil para las organizaciones mantenerse al día. ¿Es esta novedad una moda pasajera o es la próxima gran novedad? ¿Debo revisar mis prácticas actuales o debo esperar? A menudo, cuando queda claro que algo va a ser la próxima gran novedad, te das cuenta de que estás intentando ponerte al día. Integración y entrega continuas(CI/CD) llegó para quedarse, pero no siempre fue así. A muchas personas les llevó mucho tiempo convencerlas, y algunas aún necesitan convencerlas más.

La CI/CD es el proceso de automatización de las etapas de origen, compilación, prueba, puesta en escena y producción del proceso de publicación del software, y se suele describir como una canalización. En la actualidad, el ahorro de costes y la velocidad de las automatizaciones de CI/CD han convencido a la mayoría de las organizaciones de su valor. Sin embargo, la transición a este nuevo enfoque no es una tarea fácil. Debe asegurarse de que su personal tenga la formación adecuada, mejorar algunos recursos y, a continuación, realizar pruebas, pruebas y pruebas. Hay mucho por hacer. En la mayoría de los casos, es recomendable realizar estos cambios de forma gradual para ayudar a la organización a adaptarse.

El propósito de este documento es definir lo que significa tener un proceso completo de CI/CD. Proporciona una herramienta para evaluar sus propios procesos y presenta un camino a seguir para los procesos que aún no están listos. Este camino a seguir rara vez es una conversión de la noche a la mañana. Estos procesos son complejos y dependen de muchos factores, incluidas las habilidades actuales de los empleados y las demandas actuales de infraestructura. Le recomendamos que priorice y realice cambios pequeños e incrementales.

Objetivos

Los beneficios potenciales de implementar las recomendaciones de esta guía son los siguientes:

- Eficiencia— Un proceso de implementación completo de CI/CD puede reducir la complejidad, las cargas de trabajo y las innumerables horas dedicadas a la depuración, la realización de procesos manuales y el mantenimiento. Para obtener más información, consulteVentajas de la entrega continua. Según unTechAhead entrada de blog, la implementación del proceso de CI/CD puede suponer un ahorro estimado del 20% en tiempo, esfuerzo y recursos.
- Reducción de costes— Según unInforme de Forbes Insight, «Tres de cada cuatro ejecutivos están de acuerdo en que la cantidad de tiempo, dinero y recursos que se dedican al mantenimiento y la gestión continuos, en comparación con el desarrollo de nuevos proyectos o nuevas iniciativas, está afectando a la competitividad general de su organización». Cuanto más corto sea el ciclo de desarrollo, mayores serán las probabilidades de que su organización pueda cumplir sus ambiciosos objetivos time-to-market objetivos y aproveche las oportunidades adecuadas en el momento adecuado.
- Velocidad— Por lo general, una cartera completa de CI/CD es capaz de publicar los cambios de software para los clientes en unas pocas horas. La canalización de CI/CD ayuda a mejorar el tiempo medio de recuperación (MTTR), especialmente en los casos en los que los fallos se aíslan rápidamente y se introducen pequeños parches. Para obtener más información, consulteReducir el MTTR.
- Seguridad— Las canalizaciones completas de CI/CD también protegen el proceso de liberación al reducir los posibles puntos de entrada de los ataques y reducir el riesgo de errores humanos. Las mejoras de seguridad que se obtienen con las canalizaciones de CI/CD totalmente automatizadas ayudan a evitar las costosas consecuencias de las filtraciones de datos, las interrupciones del servicio, etc.
- Reducción de la deserción— Los desarrolladores se sienten más satisfechos cuando pueden dedicar más tiempo a crear excelentes funciones y menos tiempo a sumergirse en un ciclo interminable de mantenimiento y depuración. Para las organizaciones, esto significa adquirir y retener a los mejores talentos durante períodos de tiempo más largos.
- Código de calidad superior— Los desarrolladores publican el código en un repositorio compartido en pequeños lotes, lo que les permitepruebas paralelas(BrowserStack entrada de blog). En lugar de trabajar de forma aislada, comparten sus versiones con el equipo con frecuencia y colaboran para identificar los errores críticos. Esto proporciona apoyo a los desarrolladores, lo que ayuda a evitar que el código incorrecto llegue a la fase de producción. El apoyo de colegas desarrolladores contribuye a que los lanzamientos sean de alta calidad e impulsa el crecimiento de la organización.
- Mantenimiento— El mantenimiento y las actualizaciones son una parte crucial de la creación de un gran producto. Sin embargo, no apague el sistema durante las horas de mayor tráfico.

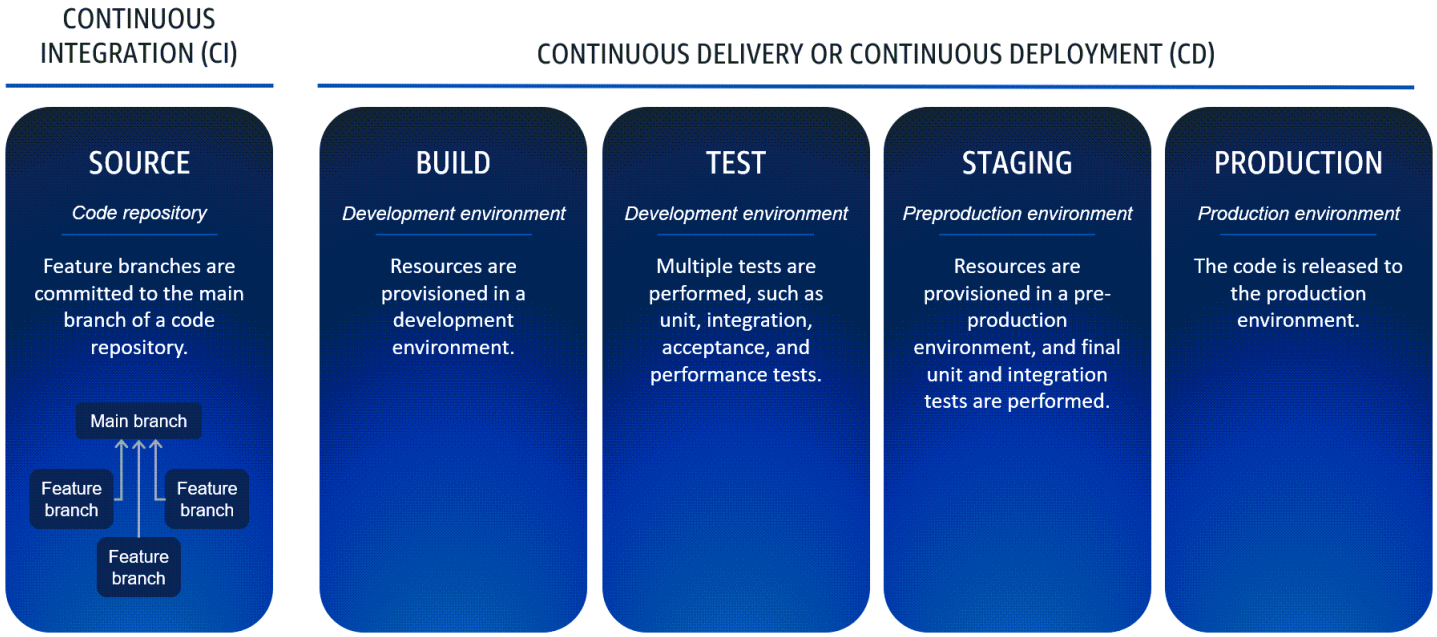
Puede utilizar una canalización de CI/CD para realizar el mantenimiento durante las horas de bajo consumo, lo que minimiza el tiempo de inactividad y el impacto en el rendimiento.

Comprensión de la CI/CD

La integración y la entrega continuas (CI/CD) son el proceso de automatización del ciclo de vida de las versiones de software. En algunos casos, elDen CI/CD también puede significardespliegue. La diferencia entreentrega continuaydespliegue continuose produce cuando se publica un cambio en el entorno de producción. Con la entrega continua, se requiere una aprobación manual antes de promover cambios en la producción. El despliegue continuo ofrece un flujo ininterrumpido a lo largo de todo el proceso y no se requieren aprobaciones explícitas. Como esta estrategia analiza los conceptos generales de CI/CD, las recomendaciones y la información proporcionadas son aplicables tanto a los enfoques de entrega continua como de despliegue continuo.

La CI/CD automatiza gran parte o la totalidad de los procesos manuales que tradicionalmente se requerían para pasar a la producción un nuevo código de un compromiso. Una canalización de CI/CD abarca las etapas de origen, compilación, prueba, puesta en escena y producción. En cada etapa, las canalizaciones de CI/CD proporcionan cualquier infraestructura necesaria para implementar o probar el código. Al utilizar una canalización de CI/CD, los equipos de desarrollo pueden realizar cambios en el código que luego se prueban automáticamente y luego se implementan.

Repasemos el proceso básico de CI/CD antes de analizar algunas de las formas en las que puedes, consciente o inconscientemente, dejar de ser completamente CI/CD. El siguiente diagrama muestra las etapas y actividades de la CI/CD en cada etapa.



Acerca de la integración continua

La integración continua se produce en un repositorio de código, como un repositorio de Git enAWS CodeCommit. Se trata una sola rama principal como fuente de información para el código base y se crean ramas de corta duración para el desarrollo de funciones. Una rama de funciones se integra en la rama principal cuando está listo para implementar la función en los entornos superiores. Las ramas de funciones nunca se implementan directamente en los entornos superiores. Para obtener más información, consulte la sección Enfoque basado en troncos de esta guía.

Proceso de integración continuo

1. El desarrollador crea una nueva sucursal a partir de la rama principal.
2. El desarrollador realiza cambios, compila y prueba localmente.
3. Cuando los cambios estén listos, el desarrollador crea unpull request(GitHub documentación) con la sucursal principal como destino.
4. Se revisa el código.
5. Cuando se aprueba el código, se fusiona con la rama principal.

Acerca de la entrega continua

La entrega continua se produce en entornos aislados, como entornos de desarrollo y entornos de producción. Las acciones que se producen en cada entorno pueden variar. A menudo, una de las primeras etapas se utiliza para realizar actualizaciones en la propia canalización antes de continuar. El resultado final de la implementación es que cada entorno se actualiza con los cambios más recientes. La cantidad de entornos de desarrollo para la creación y las pruebas también varía, pero le recomendamos que utilice al menos dos. En el proceso, cada entorno se actualiza en orden según su importancia y termina con el entorno más importante, el entorno de producción.

Proceso de entrega continuo

La parte del proceso de entrega continua se inicia extrayendo el código de la rama principal del repositorio fuente y pasándolo a la fase de compilación. El documento de infraestructura como código (IaC) del repositorio describe las tareas que se realizan en cada etapa. Si bien el uso de un documento IaC no es obligatorio, un servicio o herramienta del IaC, comoAWS CloudFormationoAWS Cloud Development Kit (AWS CDK), se recomienda encarecidamente. Los pasos más comunes incluyen:

1. Pruebas unitarias
2. Compilación de código
3. Aprovisionamiento de recursos
4. Pruebas de integración

Si se produce algún error o alguna prueba falla en cualquier etapa de la canalización, la etapa actual vuelve a su estado anterior y la canalización finaliza. Los cambios posteriores deben comenzar en el repositorio de código y pasar por todo el proceso de CI/CD.

Pruebas de canalización de CI/CD

Los dos tipos de pruebas automatizadas a los que se hace referencia habitualmente en los procesos de implementación sonpruebas unitariasypruebas de integración. Sin embargo, hay muchos tipos de pruebas que se pueden ejecutar en una base de código y en el entorno de desarrollo. ElAWSLa arquitectura de referencia del canal de desplieguedefine los siguientes tipos de pruebas:

- Prueba unitaria— Estas pruebas compilan y ejecutan el código de la aplicación para verificar que funciona de acuerdo con las expectativas. Simulan todas las dependencias externas que se utilizan en la base de código. Algunos ejemplos de herramientas de pruebas unitarias sonJUnit,Broma, ymás pytest.
- Prueba de integración— Estas pruebas verifican que la aplicación cumple los requisitos técnicos al probarla en un entorno de prueba aprovisionado. Algunos ejemplos de herramientas de pruebas de integración sonPepino,vRest NG, ypruebas integrales(paraAWS CDK).
- Prueba de aceptación— Estas pruebas verifican que la aplicación cumple los requisitos del usuario al probarla en un entorno de prueba aprovisionado. Algunos ejemplos de herramientas de prueba de aceptación sonCiprésySelenio.
- Prueba sintética— Estas pruebas se ejecutan de forma continua en segundo plano para generar tráfico y verificar que el sistema esté en buen estado. Algunos ejemplos de herramientas de prueba sintéticas incluyenAmazon CloudWatch SintéticosyMonitorización sintética Dynatrace.
- Prueba de rendimiento— Estas pruebas simulan la capacidad de producción. Determinan si la aplicación cumple con los requisitos de rendimiento y comparan las métricas con el rendimiento anterior. Algunos ejemplos de herramientas de prueba de rendimiento incluyenApache JMeter,Langosta, yGatling.
- Prueba de resiliencia— También conocido comopruebas de caos, estas pruebas inyectan fallas en los entornos para identificar las áreas de riesgo. A continuación, los períodos en los que se

producen los fallos se comparan con los períodos en los que no se producen los fallos. Algunos ejemplos de herramientas de prueba de resiliencia incluyenAWS Fault Injection ServiceyGremlin.

- Prueba de seguridad de aplicaciones estáticas (SAST)— Estas pruebas analizan el código para detectar infracciones de seguridad, comoInyección de SQLosecuencias de comandos entre sitios (XSS). Algunos ejemplos de herramientas SAST incluyenAmazon CodeGuru,SonarQube, yMarca de verificación.
- Prueba dinámica de seguridad de aplicaciones (DAST)— Estas pruebas también se conocen comopruebas de penetraciónopruebas con lápiz. Identifican vulnerabilidades, como la inyección de SQL o el XSS, en un entorno de prueba aprovisionado. Algunos ejemplos de herramientas de DAST sonZed Attack Proxy (ZAP)yHCL AppScan. Para obtener más información, consultePruebas de penetración.

No todas las canalizaciones completas de CI/CD ejecutan todas estas pruebas. Sin embargo, como mínimo, una canalización debería ejecutar pruebas unitarias y pruebas de SAST en el código base, así como pruebas de integración y aceptación en un entorno de pruebas.

Métricas para canalizaciones de CI/CD

Según elAWSArquitectura de referencia de la canalización de despliegue, debería, como mínimo, realizar un seguimiento de las cuatro métricas siguientes para las canalizaciones de CI/CD:

- Plazo de entrega— El tiempo medio que tarda un solo compromiso en llegar a la fase de producción. Te recomendamos que te enfoques en un plazo de entre 1 hora y 1 día, según corresponda a tu caso de uso.
- Frecuencia de despliegue— El número de despliegues de producción en un período de tiempo determinado. Recomendamos segmentar las frecuencias de despliegue entre varias veces al día y dos veces por semana, según corresponda a su caso de uso.
- Tiempo medio entre fallos (MTBF)— El tiempo medio entre el inicio de una canalización exitosa y el inicio de una canalización fallida. Te recomendamos que optes por un MTBF lo más alto posible. Para obtener más información, consulteIncrementar el MTBF.
- Tiempo medio de recuperación (MTTR)— El tiempo medio entre el inicio de una canalización fallida y el inicio de la siguiente canalización exitosa. Te recomendamos que optes por un MTTR lo más bajo posible. Para obtener más información, consulteReducir el MTTR.

Estas métricas ayudan a los equipos a hacer un seguimiento de su progreso para convertirse plenamente en CI/CD. Los equipos deben mantener conversaciones abiertas con las partes interesadas de la organización sobre cuáles deberían ser los objetivos óptimos. Las situaciones y necesidades varían mucho de una organización a otra, e incluso de un equipo a otro.

Es muy importante recordar que los cambios rápidos y drásticos suelen aumentar el riesgo de que surjan problemas. Establece metas con el objetivo de lograr mejoras pequeñas e incrementales. Un plazo de entrega óptimo habitual para tuberías totalmente de CI/CD es inferior a 3 horas. Un equipo que comience con un plazo de entrega de 5,2 días debería tener como objetivo una reducción de un día cada pocas semanas. Cuando este equipo cumpla un plazo de entrega de un día o menos, podrá permanecer allí durante varios meses y adoptar un plazo de entrega más agresivo solo si el equipo y las partes interesadas de la organización lo consideran necesario.

En qué se diferencian completamente los procesos de CI/CD

Las canalizaciones de CI/CD utilizan un sistema modernoflujo de trabajo basado en enlaces troncales, en el que los desarrolladores combinan actualizaciones pequeñas y frecuentes en una rama principal (otronco) que se crea y prueba a través de la parte CD de la canalización de CI/CD. Este flujo de trabajo ha sustituido alFlujo de trabajo de Gitflow, en el que las ramas de desarrollo y lanzamiento están separadas por un calendario de lanzamiento. En muchas organizaciones, Gitflow sigue siendo un método popular de control y despliegue de versiones. Sin embargo, ahora se considera heredado y su integración en una canalización de CI/CD puede resultar difícil.

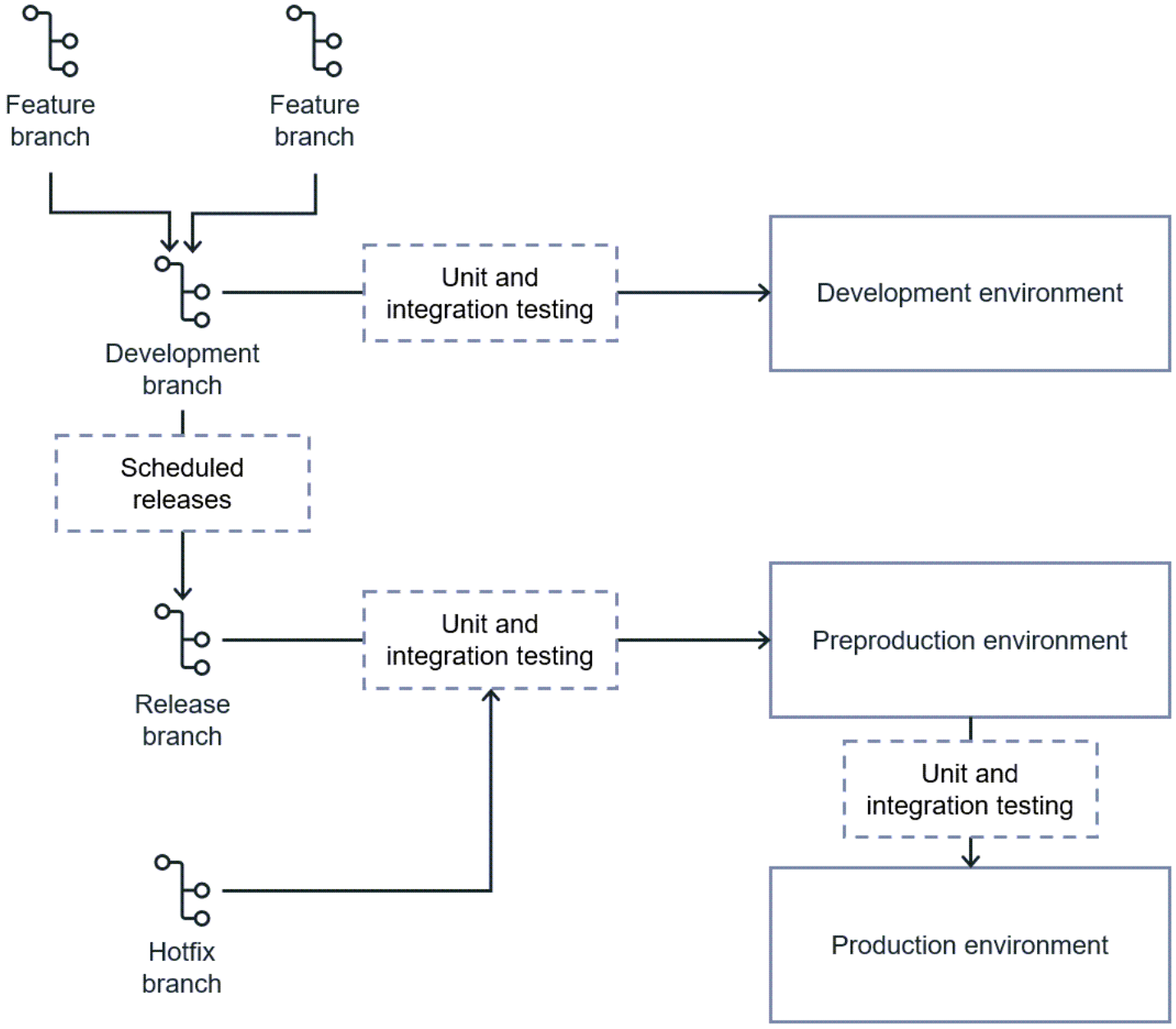
Para muchas organizaciones, la transición de un flujo de trabajo de Gitflow a un flujo de trabajo basado en enlaces troncales ha sido incompleta, y el resultado es que se quedan estancadas en algún momento y nunca migran completamente a la CI/CD. De alguna manera, sus flujos de trabajo acaban aferrándose a algunos remanentes del flujo de trabajo heredado, atrapados en un estado de transición entre el pasado y el presente. Revisa las diferencias entre los flujos de trabajo de Git y, a continuación, descubre cómo el uso de un flujo de trabajo heredado puede afectar a lo siguiente:

- Integridad ambiental
- Versiones
- Seguridad

Para facilitar la identificación de los remanentes de un flujo de trabajo de Git heredado en una configuración moderna, comparemosGitflowa lo moderno,basado en troncosenfoque.

enfoque de Gitflow

La siguiente imagen muestra un flujo de trabajo de Gitflow. El enfoque de Gitflow usa múltiples ramas para rastrear varias versiones diferentes del código al mismo tiempo. Programas el lanzamiento de las actualizaciones de una aplicación para algún momento en el futuro mientras los desarrolladores siguen trabajando en la versión actual del código. Los repositorios basados en enlaces troncales pueden usar marcadores de características para lograrlo, pero está integrado en Gitflow de forma predeterminada.



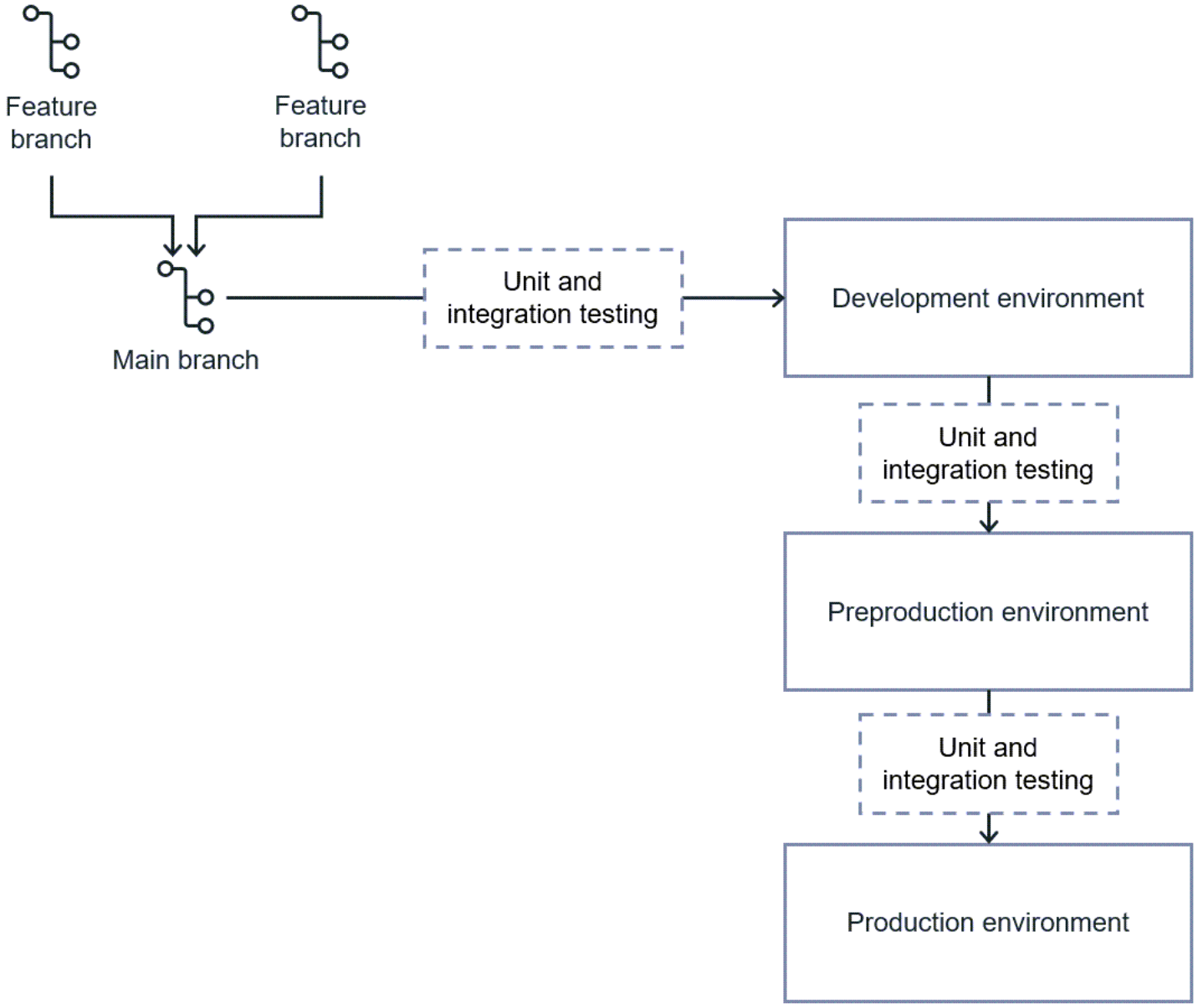
Uno de los resultados del enfoque de Gitflow es que los entornos de las aplicaciones no suelen estar sincronizados. En una implementación estándar de Gitflow, los entornos de desarrollo reflejan el estado actual del código, mientras que los entornos de preproducción y producción permanecen inmóviles en función del estado del código base desde la versión más reciente.

Esto complica las cosas cuando aparece un defecto en el entorno de producción, ya que el código base con el que trabajan los desarrolladores no se puede fusionar con el entorno de producción sin dejar al descubierto funciones inéditas. La forma en que Gitflow maneja esta situación es mediante un hotfix. Se crea una rama de hotfix a partir de la rama de lanzamiento y luego se despliega

directamente en los entornos superiores. Luego, la rama de hotfix se fusiona con la rama de desarrollo para mantener el código actualizado.

Enfoque basado en troncos

La siguiente imagen muestra un flujo de trabajo basado en enlaces troncales. En un flujo de trabajo basado en enlaces troncales, los desarrolladores crean y prueban las funciones de forma local en una rama de funciones y, a continuación, combinan esos cambios en la rama principal. Luego, la rama principal se adapta a los entornos de desarrollo, preproducción y producción, de forma secuencial. Las pruebas unitarias y de integración se realizan entre cada entorno.



Con este flujo de trabajo, todos los entornos funcionan con la misma base de código. No es necesaria una rama de revisiones para los entornos superiores, ya que se pueden implementar cambios en la rama principal sin exponer funciones inéditas. Siempre se supone que la rama principal es estable, está libre de defectos y está lista para su lanzamiento. Esto le ayuda a integrarla como fuente para una canalización de CI/CD, que puede probar e implementar automáticamente su base de código en todos los entornos de la canalización.

Los beneficios de un enfoque basado en enlaces troncales para la integridad del entorno

Como muchos desarrolladores saben, un cambio en el código a veces puede crear unaefecto mariposa(artículo de American Scientist), donde una pequeña desviación aparentemente no relacionada desencadena una reacción en cadena que provoca resultados inesperados. Luego, los desarrolladores deben investigar a fondo para descubrir la causa raíz.

Cuando los científicos realizan un experimento, separan a los sujetos de prueba en dos grupos: el grupo experimental y el grupo de control. La intención es hacer que el grupo experimental y el grupo de control sean completamente idénticos, excepto en lo que respecta a lo que se está probando en el experimento. Cuando ocurre algo en el grupo experimental que no ocurre en el grupo de control, la única causa puede ser lo que se está probando.

Piense en los cambios de un despliegue como en el grupo experimental y piense en cada entorno como grupos de control independientes. Los resultados de las pruebas en un entorno inferior solo son fiables cuando los controles son los mismos que en un entorno superior. Cuanto más se desvíen los entornos, mayor será la probabilidad de descubrir defectos en los entornos superiores. En otras palabras, si los cambios de código van a fallar en producción, preferimos que fallen primero en la versión beta para que nunca lleguen a la fase de producción. Por este motivo, se debe hacer todo lo posible para mantener sincronizados todos los entornos, desde el entorno de prueba más bajo hasta el de producción en sí. Esto se denominaintegridad ambiental.

El objetivo de cualquier proceso completo de CI/CD es descubrir los problemas lo antes posible. Preservar la integridad del entorno mediante un enfoque basado en enlaces troncales puede eliminar prácticamente la necesidad de revisiones. En un flujo de trabajo basado en enlaces troncales, es poco frecuente que un problema aparezca por primera vez en el entorno de producción.

En un enfoque de Gitflow, una vez que una revisión se implementa directamente en los entornos superiores, se añade a la rama de desarrollo. Esto preserva la corrección para futuras versiones. Sin

embargo, la revisión se desarrolló y probó directamente a partir del estado actual de la aplicación. Incluso si la revisión funciona perfectamente en producción, existe la posibilidad de que surjan problemas cuando interactúe con las funciones más nuevas de la rama de desarrollo. Como no suele ser deseable implementar una revisión para una revisión, esto hace que los desarrolladores dediquen más tiempo a intentar adaptar la revisión al entorno de desarrollo. En muchos casos, esto puede generar una importante deuda técnica y reducir la estabilidad general del entorno de desarrollo.

Cuando se produce un error en un entorno, se revierten todos los cambios para que el entorno vuelva a su estado anterior. Cualquier cambio en una base de código debería volver a iniciar la canalización desde la primera etapa. Cuando surge un problema en el entorno de producción, la solución también debería pasar por todo el proceso. El tiempo adicional que se tarda en pasar por los entornos más bajos suele ser insignificante en comparación con los problemas que se evitan con este enfoque. Como el único propósito de los entornos inferiores es detectar los errores antes de que lleguen a producción, evitar estos entornos mediante un enfoque de Gitflow es un riesgo ineficiente e innecesario.

Libere los beneficios de un enfoque basado en enlaces troncales

Una de las razones por las que suele ser necesaria una revisión es que, en un flujo de trabajo heredado, el estado de la aplicación en la que están trabajando los desarrolladores puede contener varias funciones inéditas que aún no están en fase de producción. El entorno de producción y el entorno de desarrollo solo se sincronizan cuando se produce una versión programada y, de inmediato, comienzan a diferir de nuevo hasta la siguiente versión programada.

Es posible programar los lanzamientos dentro de un proceso completo de CI/CD. Puede retrasar el lanzamiento del código a producción mediante el uso de indicadores de función. Sin embargo, un proceso de CI/CD completo permite una mayor flexibilidad al hacer innecesarias las publicaciones programadas. Después de todo,continuaes una palabra clave en CI/CD, y eso sugiere que los cambios se publican a medida que están listos. Evite mantener un entorno de lanzamiento independiente que casi siempre no esté sincronizado con los entornos de prueba inferiores.

Si una canalización no es completamente de CI/CD, la divergencia entre los entornos superior e inferior suele producirse a nivel de sucursal. Los desarrolladores trabajan en una rama de desarrollo y mantienen una rama de publicación independiente que solo se actualiza cuando llega el momento de una publicación programada. A medida que la rama de lanzamiento y la rama de desarrollo divergen, pueden surgir otras complicaciones.

Además de que los entornos no están sincronizados, a medida que los desarrolladores trabajan en la rama de desarrollo y se van acostumbrando a un estado de las aplicaciones muy superior al de producción, deben reajustarse al estado de producción cada vez que surge un problema en ese entorno. El estado de la rama de desarrollo podría consistir en muchas funciones antes de la producción. Cuando los desarrolladores trabajan en esa rama todos los días, es difícil recordar qué se ha lanzado al mercado y qué no. Esto añade el riesgo de que se introduzcan nuevos errores mientras se están corrigiendo otros errores. El resultado es un ciclo aparentemente interminable de correcciones que prolonga los plazos y retrasa el lanzamiento de las funciones durante semanas, meses o incluso años.

Ventajas de seguridad de un enfoque basado en enlaces troncales

Un proceso de CI/CD completamente automatizado proporciona un enfoque de implementación totalmente automatizado y con una única fuente de información. La tubería tiene un único punto de entrada. Las actualizaciones de software entran en la canalización desde el principio y se transmiten tal cual de un entorno a otro. Si se descubre un problema en cualquier fase del proceso, los cambios de código que lo solucionen deben pasar por el mismo proceso y empezar en la primera fase. Al reducir los puntos de entrada en una canalización, también se reducen las posibles formas en que se pueden introducir vulnerabilidades en la canalización.

Además, dado que el punto de entrada es el punto más alejado posible del entorno de producción, se reduce drásticamente la probabilidad de que las vulnerabilidades lleguen a la producción. Si se implementa un proceso de aprobación manual en un proceso íntegramente relacionado con la CI/CD, aún puede dejar de tomar decisiones sobre si los cambios se deben promover o no en el siguiente entorno. El responsable de la toma de decisiones no es necesariamente la misma persona que implementa los cambios. Esto separa las responsabilidades de quien implementa los cambios de código y de quien los aprueba. También hace que sea más factible que un líder de la organización menos técnico desempeñe la función de aprobador.

Por último, el punto de entrada único permite limitar el acceso de escritura a la consola de interfaz de usuario (UI) del entorno de producción a unos pocos usuarios o incluso a ninguno. Al reducir el número de usuarios que pueden realizar cambios manuales en la consola, se reduce el riesgo de que se produzcan problemas de seguridad. La capacidad de gestionar manualmente la consola en el entorno de producción es mucho más necesaria en los flujos de trabajo tradicionales que en un enfoque automatizado de CI/CD. Estos cambios manuales son más difíciles de rastrear, revisar y probar. Por lo general, se realizan para ahorrar tiempo, pero a la larga, añaden una importante deuda técnica al proyecto.

Los problemas de seguridad de las consolas no se deben necesariamente a personas malas. Muchos de los problemas que se producen en la consola son accidentales. La exposición accidental a la seguridad es muy común y ha llevado al surgimiento del modelo de seguridad de confianza cero. Este modelo postula, en parte, que los accidentes de seguridad son menos probables cuando incluso el personal interno tiene el menor acceso posible, lo que también se conoce comopermisos de mínimo privilegio. Preservar la integridad del entorno de producción mediante la restricción de todos los procesos a una canalización automatizada prácticamente elimina el riesgo de problemas de seguridad relacionados con la consola.

Prueba de fuego para tuberías de CI/CD

En química, el papel tornasol es una tira delgada de papel tratada con un colorante rojo o azul especial que se utiliza para determinar la acidez de una sustancia. Un ácido convierte el papel tornasol azul en rojo, una base hace que el papel tornasol rojo se vuelva azul y las sustancias neutras no afectan en absoluto al color del papel.

La forma en que el papel tornasol determina la acidez es midiendo el nivel de pH de una sustancia. Si un nivel de pH es superior a 8, es ácido; si está por debajo de 5, es básico; y si está entre 5 y 8, es neutro. Del mismo modo, elPrueba de fuego CI/CDle ayuda a medir el nivel de CI/CD de su canalización.

Para comprobar si su canalización es totalmente CI/CD

1. Comience con una puntuación de 0.
2. Responda a cada una de las siguientes preguntas y añada 1 a su puntuación cada vez que respondasí:

    - ¿Cada uno de nuestros repositorios tiene exactamente una rama principal que se utiliza para implementar en los entornos?
    - ¿Enviamos el código a la rama principal con frecuencia y evitamos que las ramas de funciones se ejecuten durante mucho tiempo?
    - ¿Nuestro oleoducto tiene un único punto de entrada? En otras palabras, ¿nuestra canalización extrae el código de cada repositorio exactamente una vez?
    - ¿Tenemos más de un entorno de implementación?
    - Cuando la canalización no está en ejecución, ¿nuestros entornos superior e inferior están generalmente sincronizados?
    - ¿Realizamos pruebas en el código antes de implementarlo?
    - ¿Realizamos pruebas en un entorno antes de pasar al siguiente entorno?
    - ¿Nuestra canalización se revierte por completo y se cierra después de un fallo?
    - ¿Nuestra canalización se reinicia desde el primer paso cuando se recupera de un error?
    - ¿Seguimos el mismo proceso para corregir errores en la producción que para lanzar funciones a producción?
    - ¿Usamos algún tipo de infraestructura como plantillas de código (IaC) para implementar el código?

3. Responda a cada una de las siguientes preguntas y añada 1 a su puntuación cada vez que respondano:

   - ¿Alguna vez realizamos la implementación directamente en un entorno de implementación desde sucursales distintas de la sucursal principal?
   - ¿Alguna vez realizamos el despliegue directamente desde cualquier sucursal a un entorno superior o de producción?
   - ¿Encontramos a menudo errores en los entornos superiores que no estaban presentes en los entornos inferiores?
   - ¿Alguna vez evitamos los entornos inferiores durante una implementación?
   - ¿Tenemos que esperar hasta la fecha de lanzamiento programada para implementarla en producción?
   - ¿Realizamos actualizaciones periódicas en la consola del entorno de producción?
   - ¿Hay algún paso de implementación manual que deba realizarse en la consola del entorno de producción para completar la implementación?
   - ¿Tiene más de una persona acceso de escritura al entorno de producción?
   - ¿Tienen más de cinco personas acceso de escritura al entorno de producción?
4. Divida su puntuación entre 2. Esta es la puntuación de CI/CD de tu pipeline.
5. Compara la puntuación de CI/CD de tu canalización con la siguiente tabla para determinar el nivel de CI/CD de tu canalización.

Puntuación de CI/CD	Nivel de CI/CD
9.5 o superior	Totalmente CI/CD
8—9	Principalmente CI/CD
5—7	Neutral
Por debajo de 5	No es CI/CD

Si obtuvo una puntuación inferior a 8, le recomendamos que se fije una meta para avanzar gradualmente hacia el siguiente nivel. Cuando se logre ese objetivo, las partes interesadas en el producto deberán evaluar si es necesario establecer un nuevo objetivo y cuándo. La intención de este ejercicio no es necesariamente abogar por un cambio en su cartera, sino más bien dar a

conocer cómo es un proceso de despliegue integral de la CI/CD y cuál es la posición actual de sus proyectos en ese espectro.

Mejores prácticas para canalizaciones de CI/CD

Las siguientes son las mejores prácticas para canalizaciones totalmente de CI/CD:

- Proteja el entorno de producción— Dado que es posible realizar prácticamente todo lo necesario para el mantenimiento de la cuenta y el entorno mediante el uso de iAC, es importante hacer todo lo posible por proteger el entorno de producción limitando el acceso programático y de la consola. Recomendamos limitar el acceso a unos pocos usuarios, o incluso a ninguno. Al implementar iAC medianteAWS CloudFormation, el usuario necesita permisos limitados. La mayoría de los permisos se asignan a CloudFormation servicio a través de un rol de servicio. Para obtener más información, consulteFunción de servicioen el CloudFormation documentación yImplementar políticas de permisos con privilegios mínimos paraAWS CloudFormation.
- Cree cuentas separadas para cada entorno— Al dedicar una cuenta independiente a cada entorno, puede simplificar el proceso de implementación y crear controles de acceso detallados a nivel de cuenta. Cuando varios entornos comparten recursos, se reduce la integridad del entorno como unidad aislada. Es mejor mantener los entornos sincronizados y diferenciados. Esto es aún más importante para el entorno de producción, ya que todo lo que contiene esa cuenta debe tratarse como un recurso de producción.
- Restrinja la información de identificación personal (PII) al entorno de producción— Tanto por motivos de seguridad como de protección contra los riesgos de responsabilidad, proteja la PII en la medida de lo posible. Cuando sea posible en entornos más bajos, utilice datos anónimos o de muestra en lugar de copiar datos potencialmente confidenciales del entorno de producción.
- Revise el código en los repositorios— Un proceso completo de CI/CD reduce los puntos de entrada de una canalización a un solo punto, y ese único punto debe estar protegido. Por este motivo, se recomienda revisar el código varias veces antes de fusionar las ramas de funciones con la rama principal. Estas revisiones de código las puede llevar a cabo cualquier miembro cualificado del equipo, pero al menos un miembro sénior debería revisarlas. El revisor debe probar rigurosamente el código. Después de todo, la mejor manera de solucionar los problemas de una canalización es evitar introducirlos en ella. Además, es importante resolver todos los comentarios realizados por cualquier revisor antes de fusionarlos. Esta resolución podría ser simplemente una explicación de por qué no es necesario realizar cambios, pero abordar todos los comentarios es una medida adicional importante para evitar que se introduzcan problemas en el proceso.
- Realice fusiones pequeñas y frecuentes— Para aprovechar al máximo la integración continua, también es una buena idea impulsar los cambios locales de forma continua. Al fin y al cabo, es

mucho más beneficioso para los entornos de desarrollo mantenerse sincronizados si los entornos locales también se mantienen al día con ellos.

Para obtener más prácticas recomendadas para las canalizaciones de CI/CD, consulteResumen de las mejores prácticasenPracticando la integración continua y la entrega continua enAWS.

Preguntas frecuentes

¿Cuáles son algunos de los indicadores clave de que mi proceso de implementación no es completamente CI/CD?

El indicador más común es cuando hay varias ramas de repositorio que representan entornos independientes en una canalización. En un proceso totalmente de CI/CD, los repositorios utilizan un flujo de trabajo basado en enlaces troncales, en el que una rama actúa como la única fuente de información para las implementaciones de ese repositorio. Para obtener más información, consulte Enfoque basado en troncos. Otros indicadores incluyen los pasos de implementación manual distintos de las simples decisiones de aceptar o no, el uso de revisiones y las versiones programadas.

¿Qué sucede si quiero utilizar un proceso completo de CI/CD, pero aun así quiero programar el lanzamiento de determinadas funciones para momentos específicos?

Por lo general, esto se hace con indicadores de características. En este proceso, las implementaciones se siguen realizando de forma continua, pero algunas funciones se ocultan mediante cierres condicionales en el código hasta que llegue el momento de publicarlas.

¿Qué sucede si algunos pasos de mi proceso de implementación no se pueden automatizar?

Uno de los objetivos de una cartera completa de CI/CD es minimizar la necesidad de procesos manuales, pero no cabe duda de que hay posibles casos de uso en los que pueden ser necesarios los procesos manuales. De hecho, los procesos de solo lectura, como la consulta de los registros de las aplicaciones, suelen realizarse en entornos de producción con un riesgo mínimo. Sin embargo, se recomienda encarecidamente que las acciones de escritura manual en producción sean el último recurso.

¿Qué sucede si mi personal técnico se siente más cómodo con los flujos de trabajo tradicionales que con un proceso completo de CI/CD?

Es común que el personal técnico se resista a los cambios importantes, especialmente cuando algo que solía ser una buena práctica se reemplaza por algo más nuevo. La tecnología avanza rápido y las mejoras se descubren constantemente. Si bien un cierto grado de escepticismo es una buena cualidad para el personal técnico, es igual de importante que estén abiertos a los cambios. No actúes demasiado rápido con el personal escéptico, ya que necesitan gestionar los cambios en el sistema antes de implementarlos. La clave es evitar que los escépticos permanezcan estáticos para siempre.

¿Qué pasa si mis entornos están en varias cuentas? ¿Puedo seguir utilizando un proceso de CI/CD completo?

Sí, de hecho, se recomienda utilizar una cuenta independiente para cada entorno. Para obtener más información sobre una canalización que activa etapas en diferentes cuentas, consultaCree una canalización en CodePipeline que utilice recursos de otroCuenta de AWS.

Pasos siguientes

Utilice elPrueba de fuego para tuberías de CI/CDsección para evaluar la DevOpslos procesos de su organización. Determine si sus procesos son completamente de CI/CD. Si no lo son, decida si es necesario mejorarlos para aprovechar al máximo los beneficios de las implementaciones de CI/CD.

¿Cómo sabe que ha terminado? Bueno, la respuesta es que muchas organizaciones en realidad nunca terminan. Se detienen en algún punto del camino, en un lugar adecuado para su caso de uso. Si bien el mejor escenario posible es una cartera completa de CI/CD, depende en gran medida de la situación de la organización y de las partes interesadas que respaldan la decisión. Las partes interesadas deben decidir qué etapa de la implementación de la CI/CD funciona mejor para su caso de uso y cuál es la mejor manera de planificar el progreso hacia las siguientes fases.

Para obtener más información sobre el diseño y la creación de canalizaciones de CI/CD, consulteRecursos.

Recursos

AWSdocumentación y referencias

- AWSArquitectura de referencia para la canalización de despliegue
- ¿Qué es la entrega continua?
- Practicar la integración continua y la entrega continua enAWS(AWSLibro blanco)
- Configure una canalización de CI/CD enAWS(AWSTutorial práctico)
- Crea una canalización enRegiones de AWSque no admitenAWS CodePipeline(AWS(Guía prescriptiva)
- UsoAWS CodeCommityAWS CodePipelinepara implementar una canalización de CI/CD en múltiplesCuentas de AWS(AWS(Guía prescriptiva)
- Canalizaciones de despliegue, arquitectura de referencia e implementaciones de referencia(AWSentrada de blog)

Servicios y herramientas

- La prueba de fuego de la CI/CD
- AWS Cloud Development Kit (AWS CDK)
- AWS CloudFormation
- AWS CodeCommit
- AWS CodePipeline

Historial de documentos

En la siguiente tabla se describen los cambios más importantes en esta guía. Si desea recibir notificaciones sobre futuras actualizaciones, puede suscribirse a unaFuente RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	25 de agosto de 2023

AWSGlosario de orientación prescriptiva

Los siguientes son términos de uso común en las estrategias, guías y patrones proporcionados por AWS Prescriptive Guidance. Para sugerir entradas, utilice el enlace Enviar comentarios al final del glosario.

DevOps términos

branch

Un área contenida de un repositorio de código. La primera rama que se crea en un repositorio es la rama principal. Puedes crear una rama nueva a partir de una rama existente y, a continuación, desarrollar funciones o corregir errores en la nueva rama. Una rama que se crea para crear una entidad se denomina normalmente rama de entidad. Cuando la función esté lista para su publicación, debe volver a fusionar la rama de la entidad con la rama principal. Para obtener más información, consulte Acerca de las sucursales (GitHub documentación).

repositorio de código

Un lugar donde el código fuente y otros activos, como documentación, muestras y scripts, se almacenan y actualizan mediante procesos de control de versiones. Los repositorios en la nube más comunes incluyen GitHub oAWS CodeCommit. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio está dedicado a una única funcionalidad. Una sola canalización de CI/CD puede utilizar varios repositorios.

integración continua y entrega continua (CI/CD)

Proceso de automatización de las etapas de origen, creación, prueba, puesta en escena y producción del proceso de publicación del software. La CI/CD se describe comúnmente como una canalización. La CI/CD puede ayudarlo a automatizar los procesos, mejorar la productividad, mejorar la calidad del código y entregar con mayor rapidez. Para obtener más información, consulte Beneficios de la entrega continua. CD también puede significar despliegue continuo. Para obtener más información, consulte Entrega continua frente a implementación continua.

implementación

El proceso de hacer que una aplicación, nuevas funciones o correcciones de código estén disponibles en el entorno de destino. La implementación implica implementar cambios en una

base de código y, a continuación, crear y ejecutar esa base de código en los entornos de la aplicación.

entorno de desarrollo

Consulte environment.

environment

Una instancia de una aplicación en ejecución. Los siguientes son los tipos de entornos más comunes en la computación en nube:

- entorno de desarrollo: instancia de una aplicación en ejecución que solo está disponible para el equipo principal responsable del mantenimiento de la aplicación. Los entornos de desarrollo se utilizan para probar los cambios antes de promocionarlos a los entornos superiores. Este tipo de entorno a veces se denomina entorno de prueba.
- entornos inferiores: todos los entornos de desarrollo de una aplicación, como los que se utilizan para las compilaciones y pruebas iniciales.
- entorno de producción: instancia de una aplicación en ejecución a la que pueden acceder los usuarios finales. En una canalización de CI/CD, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios distintos del equipo de desarrollo principal. Esto puede incluir un entorno de producción, entornos de preproducción y entornos para las pruebas de aceptación por parte de los usuarios.

Flujo de trabajo de Gitflow

Un enfoque en el que los entornos inferior y superior utilizan diferentes ramas en un repositorio de código fuente. El flujo de trabajo de Gitflow se considera heredado y flujo de trabajo basado en troncos es el enfoque moderno preferido.

rama de características

Consulte branch.

revisión

Una solución urgente para un problema crítico en un entorno de producción. Debido a su urgencia, una revisión suele realizarse fuera del flujo de trabajo habitual de las DevOps versiones.

infraestructura

Todos los recursos y activos contenidos en el entorno de una aplicación.

infraestructura como código (IaC)

Proceso de aprovisionamiento y administración de la infraestructura de una aplicación mediante un conjunto de archivos de configuración. El IaC está diseñado para ayudarlo a centralizar la administración de la infraestructura, estandarizar los recursos y escalar rápidamente para que los nuevos entornos sean repetibles, confiables y consistentes. Para obtener más información, consulte La infraestructura como código (documento técnico). AWS

entornos más bajos

Consulte environment.

rama principal

Consulte branch.

entorno de producción

Consulte environment.

versión

En un proceso de despliegue, el acto de promover cambios en un entorno de producción.

entorno de prueba

Consulte environment.

flujo de trabajo basado en troncos

Un enfoque en el que los desarrolladores crean y prueban funciones de forma local en una rama de funciones y, a continuación, combinan esos cambios en la rama principal. Luego, la rama principal se adapta a los entornos de desarrollo, preproducción y producción, de forma secuencial.

entornos superiores

Consulte environment.

control de versión

Procesos y herramientas que rastrean los cambios, como los cambios en el código fuente de un repositorio.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.