Introducción al curso

Lección 2 de 27

Este curso para principiantes está orientado a alumnos técnicos de los ámbitos de desarrollo y operaciones que estén interesados en aprender los conceptos básicos de DevOps en Amazon Web Services (AWS). Mediante discusiones, contenido interactivo y demostraciones, aprenderá sobre la cultura, las prácticas y las herramientas utilizadas en un entorno de DevOps. También explorará conceptos para desarrollar y entregar aplicaciones seguras a alta velocidad en AWS.

Al final de este curso, podrá describir cómo AWS ayuda a los equipos a implementar prácticas de DevOps. Estas prácticas se centran en crear e implementar aplicaciones de manera eficiente y más segura y en mejorar el tiempo de comercialización a la vez que se administran los riesgos.

Objetivos de aprendizaje

Al final de este curso, podrá hacer lo siguiente:

- Describir las metodologías de DevOps en torno a cultura, prácticas y herramientas.
- Explicar por qué adoptar una mentalidad que apoye una cultura de DevOps es esencial para implementar DevOps.
- Describir la transformación a DevOps de Amazon.
- Categorizar y describir los servicios clave de AWS DevOps que respaldan el ciclo de vida de las aplicaciones.
- Identificar los servicios de AWS que se utilizan para automatizar el proceso de integración continua y entrega continua (CI/CD).
- Describir cómo crear y controlar una canalización de CI/CD.

Información general sobre el módulo

Lección 3 de 27

En este módulo se presentan los conceptos básicos de DevOps y sus beneficios sobre los enfoques tradicionales del desarrollo de software. Descubra cómo los equipos que adoptan la cultura, las prácticas y las herramientas de DevOps desarrollan y entregan aplicaciones a alta velocidad y responden rápidamente a las necesidades comerciales y de los clientes.

Objetivos de aprendizaje

Al final de este módulo, podrá hacer lo siguiente:

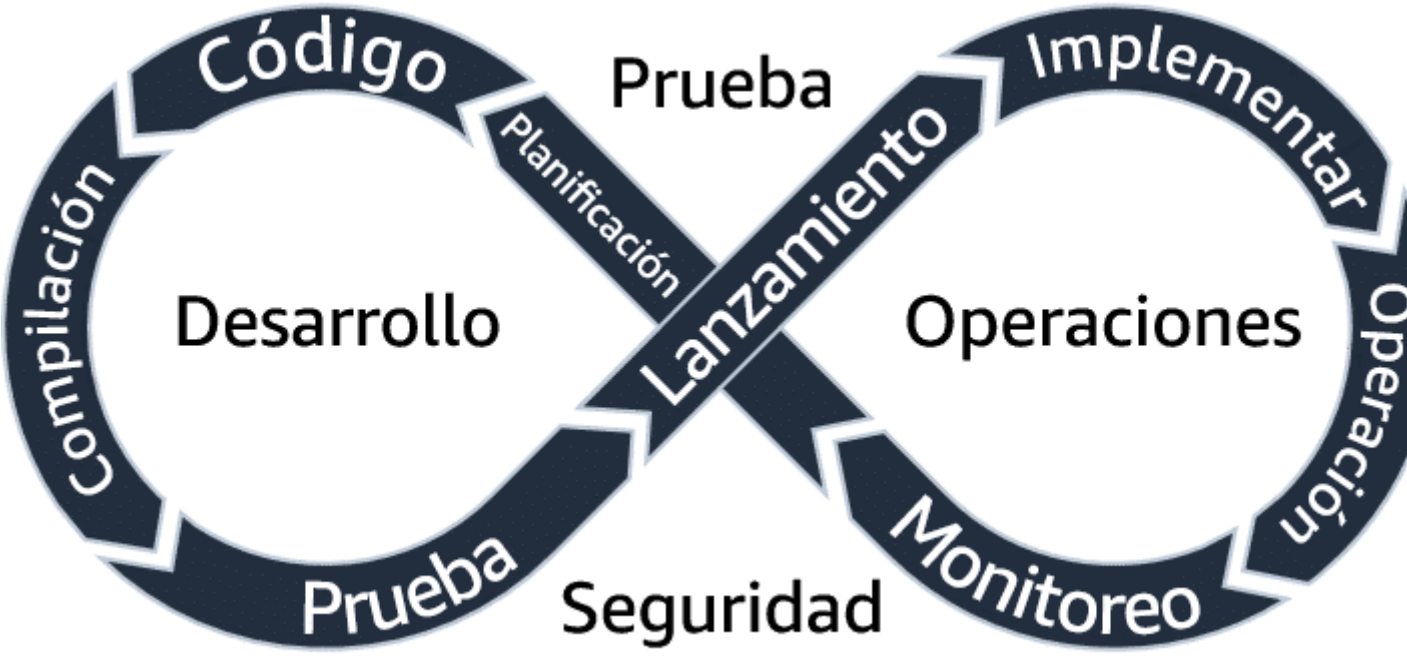
- Describir los desafíos asociados a las prácticas tradicionales de desarrollo de software.
- Enumerar los beneficios de implementar DevOps.

¿Qué es DevOps?

Lección 4 de 27

DevOps es una combinación de filosofías sobre cultura, prácticas y herramientas que incrementan la capacidad de una organización de proporcionar aplicaciones y servicios a gran velocidad y de desarrollar y mejorar productos con mayor rapidez que las organizaciones que utilizan procesos tradicionales de desarrollo de software y administración de infraestructura. Esta velocidad permite a las organizaciones servir mejor a sus clientes y competir de forma más eficaz en el mercado.

DevOps hace hincapié en una mejor colaboración y eficiencia para que los equipos puedan innovar más rápido y ofrecer más valor a empresas y clientes.



Un bucle infinito formado por las etapas del ciclo de vida del software es una representación común de la colaboración constante y las eficiencias que sugiere DevOps.

1. DevOps es la abreviatura de Desarrollo (Development, Dev) y Operaciones (Operations, Ops). Desarrollo refiere a las personas y los procesos que crean software. Operaciones refiere a los equipos y procesos que entregan y monitorean el software.

2. Los desarrolladores cambian las cosas rápidamente, lanzan a menudo y miden el éxito según la tasa de entrega. Las operaciones dependen de la conservación de la estabilidad de la aplicación. Las versiones frecuentes son motivo de preocupación en torno a la estabilidad y fiabilidad de la aplicación en las plataformas admitidas, especialmente durante el tráfico de red elevado.
3. DevOps reúne funciones que antes se agrupaban en roles aislados (desarrollo, operaciones de TI, ingeniería de calidad y seguridad) para optimizar la productividad de los desarrolladores y la confiabilidad de las operaciones.

DevOps es una combinación de:

- Filosofías culturales para eliminar barreras y compartir la responsabilidad integral
- Procesos desarrollados para lograr velocidad y calidad, que agilizan la forma en que las personas trabajan
- Herramientas que se alinean con los procesos y automatizan tareas repetibles, lo que hace que el proceso de lanzamiento sea más eficiente y la aplicación más confiable

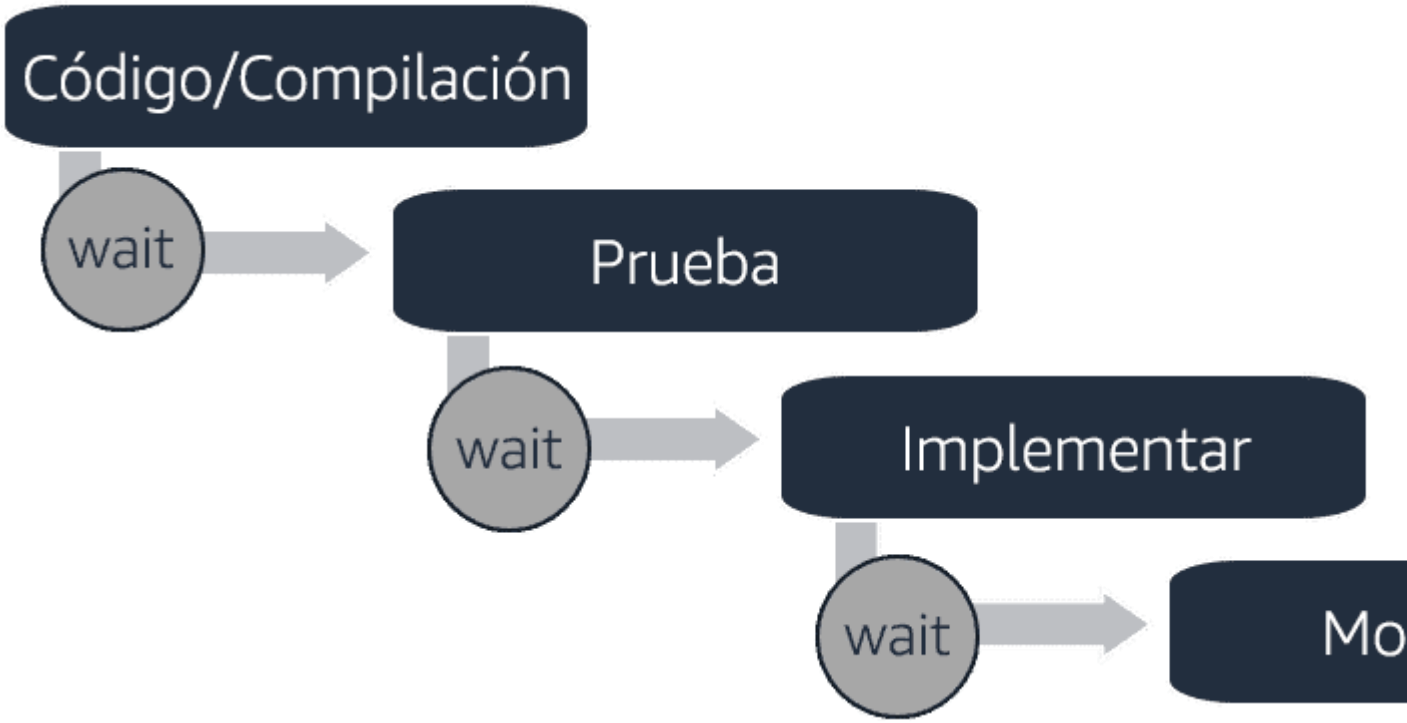
A medida que avance en el curso, tenga en cuenta que los equipos practican DevOps para aumentar la innovación y la agilidad, mejorar la calidad, realizar lanzamientos más rápido y reducir costos. DevOps mejora la eficiencia de entrega y la previsibilidad de la aplicación y los servicios.

Problemas con las prácticas tradicionales de desarrollo

Lección 5 de 27

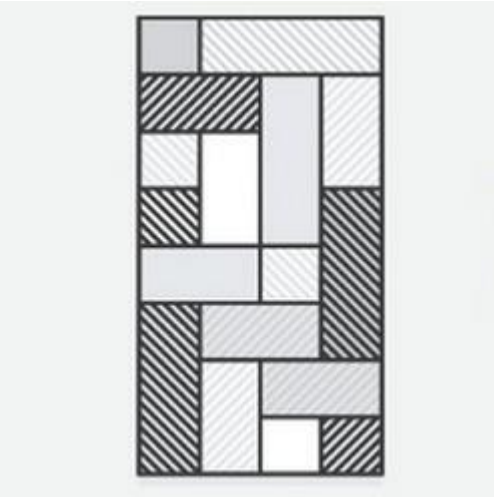
Las formas tradicionales de desarrollar software han demostrado ser lentas e ineficaces y no apoyan los esfuerzos de los equipos para ofrecer rápidamente aplicaciones estables y de alta calidad. Los desafíos del desarrollo en cascada, las aplicaciones monolíticas, los procesos manuales y las estructuras de equipos independientes ocasionan atascos y retrasos.

- Los proyectos de desarrollo en cascada son lentos, no iterativos, resistentes al cambio y tienen largos ciclos de lanzamiento. Algunas razones de esto son:
    - Los requisitos son rígidos, se establecen al comienzo del proyecto y probablemente no cambiarán.
    - Las fases de desarrollo son separadas y cada una comienza una vez que finaliza la anterior. Cada fase está a cargo de equipos altamente especializados. Las entregas de una fase a la otra son largas y a menudo requieren que los equipos cambien de herramientas y dediquen tiempo a aclarar información incompleta o ambigua.
    - Las pruebas y la seguridad vienen después de la implementación, momento en el que las acciones correctivas responden y son costosas.



Las fases de desarrollo con equipos separados y especializados ocasionan

retrasos y son costosas.



La funcionalidad de las aplicaciones monolíticas es altamente dependiente.

- Las aplicaciones monolíticas son difíciles de actualizar e implementar porque:
    - Se desarrollan e implementan como una unidad, por lo que cuando se realizan cambios, se debe volver a desplegar toda la aplicación
    - Su funcionalidad es altamente dependiente y esto quiere decir que si la aplicación es muy grande, el mantenimiento se convierte en un problema porque los desarrolladores tienen dificultad para entender toda la aplicación
    - Se implementan con una sola pila de desarrollo, por lo que cambiar la tecnología es difícil y costoso

- Los procesos manuales a lo largo del ciclo de vida de la aplicación son lentos, inconsistentes y propensos a errores. Por ejemplo, configurar manualmente la infraestructura requiere mucho tiempo. La repetición manual de este proceso no garantiza la omisión de un paso. Otro ejemplo es ***decirles*** a los desarrolladores que se aseguren de que el código se prueba por completo antes de implementarlo. Incluso con las mejores intenciones, este proceso manual es lento y no impide que alguien olvide una o dos pruebas.

- Los equipos que apoyan el ciclo de vida de desarrollo del software han sido tradicionalmente equipos independientes. Especializados en su conjunto de habilidades, equipos como negocios, desarrollo, control de calidad, especialización, mantenimiento y operaciones, han trabajado separados y requieren entregas programadas y rígidas. Aunque estos equipos tienen un objetivo común de entregar y apoyar la aplicación, también tienen sus propias prioridades, herramientas y procesos. Es difícil lograr eficiencias cuando los miembros de un proyecto dependen de diferentes unidades y trabajan con objetivos diferentes.

¿Por qué DevOps?

Lección 6 de 27

106x

Paso más rápido de confirmación a implementación

20

Implement
más frecue

7x

Tasa más baja de error en los cambios

2,6

Mayor agili
la recupera

Fuente: Accelerate State of DevOps 2019, DevOps Research and Assessment (DORA)

Éxito comprobado

Según el informe Accelerate State of DevOps 2019, las organizaciones de cualquier tamaño y de cualquier sector pueden sacar aprovecho de DevOps. Según el informe, los equipos que adoptan con éxito DevOps experimentan ciclos de entrega más breves, tasas de error de cambios más bajas y rendimiento mejorado.

Beneficios de DevOps

Las organizaciones de todos los tamaños, desde empresas emergentes pequeñas hasta empresas grandes, pueden beneficiarse de la adopción de DevOps. A continuación se presentan algunos de los principales beneficios de DevOps.

Para obtener más información sobre ellos, seleccione cada punto.

1.
2.
3.
4.
5.
6.

¿Por qué algunos equipos se resisten inicialmente a adoptar DevOps?

La renuencia a la adopción de DevOps es natural porque DevOps traerá cambios e interrumpirá la forma en que trabaja e interactúa con los demás. DevOps tendrá un impacto a nivel de organización y de equipos. Para superar esta renuencia, es importante comprender el valor de DevOps y establecer expectativas realistas para los equipos. Para tener éxito, es necesaria la colaboración entre la organización y los equipos de desarrollo.

Evaluación de conocimientos

Lección 7 de 27

Seleccione la respuesta que mejor describa DevOps:

- DevOps es la combinación de equipos de desarrollo y operaciones, donde los miembros de cada equipo deben aprender a desarrollar, probar e implementar el software.
- DevOps es una combinación de herramientas, prácticas y filosofías culturales que aumenta la capacidad de una organización para entregar aplicaciones y servicios con mayor rapidez.
- DevOps es un conjunto de herramientas de desarrollo y operaciones que automatizan el proceso de desarrollo de software.
- DevOps es la coordinación entre los equipos de desarrollo y operaciones para mejorar la comunicación entre los equipos para desarrollar software de manera más eficiente.

ENVIAR

REALIZAR DE NUEVO

¿Cuál de las siguientes opciones describe mejor los desafíos asociados a los

métodos tradicionales de desarrollo de software?

- Los métodos tradicionales de desarrollo de software suelen utilizar software y prácticas de desarrollo obsoletas y deben reemplazarse por software DevOps.
- Los métodos tradicionales de desarrollo de software suelen utilizar una arquitectura de microservicios que es difícil de actualizar e implementar, porque cuando se realizan cambios, se debe volver a desplegar toda la aplicación.
- Los métodos tradicionales de desarrollo de software suelen utilizar la integración y la entrega continuas, lo que ralentiza el proceso de desarrollo mediante el uso de tareas repetitivas.
- Los métodos tradicionales de desarrollo de software suelen implicar arquitecturas monolíticas que son difíciles de actualizar e implementar y son métodos de desarrollo no iterativos con ciclos de lanzamiento extensos.

ENVIAR

Resumen

Lección 8 de 27

En este módulo, aprendió que los equipos que adoptan la cultura, las prácticas y las herramientas de DevOps en lugar de las prácticas de desarrollo tradicionales pueden desarrollar y entregar aplicaciones a mayor velocidad y responder más rápido a las necesidades comerciales y de los clientes.

Ahora puede:

- Describir los desafíos asociados con las prácticas tradicionales de desarrollo de software.
- Enumerar los beneficios de implementar DevOps.

Información general sobre el módulo

Lección 9 de 27

La metodología de DevOps aumenta la colaboración a lo largo del ciclo de vida del servicio, desde el diseño del producto y el proceso de desarrollo hasta las operaciones de producción. Reúne a las personas para trabajar y eliminar obstáculos para que puedan alcanzar eficazmente sus objetivos. En este módulo se profundizará en la metodología de DevOps: cultura, procesos y herramientas.

Objetivos de aprendizaje

Al final de este módulo, usted podrá hacer lo siguiente:

- Discutir los desafíos que implica adoptar una cultura de DevOps y describir posibles soluciones.
- Identificar oportunidades de automatización en el desarrollo y mantenimiento de aplicaciones.
- Describir los beneficios de separar servicios o componentes.
- Definir la observabilidad y describir su importancia para DevOps.
- Explicar por qué la seguridad es importante en cada fase de la canalización.
- Explicar cómo AWS se integra con herramientas de terceros para la entrega y las implementaciones automatizadas de código.

Cultura de DevOps

Lección 10 de 27

Un cambio a DevOps requiere crear y nutrir una cultura de DevOps, que es una cultura de transparencia, colaboración eficaz y fluida y objetivos comunes. Es posible que tenga los procesos y herramientas para sustentar DevOps, pero para una adopción exitosa de DevOps, las personas de la organización deben tener la mentalidad correcta para nutrir la cultura de DevOps.

Hay siete principios básicos que pueden ayudarlo a lograr una cultura de DevOps.

Para obtener más información, seleccione el símbolo + junto a cada principio.

Crear un entorno altamente colaborativo

+

DevOps reúne el desarrollo y las operaciones para dejar de lado el trabajo aislado, alinear metas y cumplir objetivos comunes. Todo el equipo (desarrollo, pruebas, seguridad, operaciones y otros) tiene propiedad total del software que lanzan. Trabajan juntos para optimizar la productividad de los desarrolladores y la fiabilidad de las operaciones. Los equipos aprenden de las experiencias de los demás, escuchan preocupaciones y perspectivas y agilizan sus procesos para obtener los resultados requeridos. Esta mayor visibilidad permite unificar y mejorar continuamente los procesos para cumplir los objetivos del negocio. La colaboración también crea una cultura de alta confianza que valora los esfuerzos de cada miembro del equipo y transfiere conocimientos y prácticas recomendadas entre los equipos y la organización.

Automatizar cuando sea posible

+

Con DevOps, las tareas repetibles se automatizan, lo que permite a los equipos centrarse en la innovación. La automatización proporciona los medios para un rápido desarrollo, pruebas e implementación. Identifique oportunidades de automatización en cada fase, como integraciones de código, revisiones, pruebas, seguridad, implementaciones y monitoreo, mediante las herramientas y servicios adecuados. Por ejemplo, la infraestructura como código (IaC) se puede utilizar para entornos predefinidos o aprobados y versionados para crear entornos repetibles y consistentes. También puede definir comprobaciones reglamentarias e incorporarlas en pruebas que se ejecutan continuamente como parte de la canalización de versiones.

Foco en las necesidades del cliente

+

La mentalidad donde el cliente está primero es un factor clave para impulsar el desarrollo. Por ejemplo, con bucles de retroalimentación, los equipos de DevOps se mantienen en contacto con sus clientes y desarrollan software que satisface las necesidades del cliente. Con una arquitectura de microservicios, pueden cambiar rápidamente de dirección y alinear sus esfuerzos a esas necesidades. Los procesos optimizados y la automatización proporcionan las actualizaciones solicitadas más rápidamente y mantienen elevada la satisfacción del cliente. El monitoreo ayuda a los equipos a determinar el éxito de su aplicación y alinear continuamente sus esfuerzos centrados en el cliente.

Desarrollar poco y presentar a menudo

+

Las aplicaciones ya no se están desarrollando como un sistema monolítico con prácticas rígidas de desarrollo, pruebas e implementación. La arquitectura de las aplicaciones está diseñada con componentes más pequeños y poco relacionados. Existen políticas generales (como la compatibilidad con versiones anteriores o la gestión del cambio) que proporcionan gobernanza a los esfuerzos de desarrollo. Los equipos están organizados en función de la arquitectura del sistema. Sienten que sus esfuerzos les pertenecen. La adopción de prácticas de desarrollo modernas, como el lanzamiento pequeño y frecuente de códigos, proporciona a los equipos la agilidad que necesitan para responder a las necesidades de los clientes y los objetivos comerciales.

Incluir seguridad en cada fase

+

Para admitir la entrega continua, la seguridad debe ser iterativa, incremental, automatizada y existir en cada fase del ciclo de vida de la aplicación, no solo antes de lanzar una versión.

Eduque a los equipos de desarrollo y operaciones para integrar la seguridad en cada paso del ciclo de vida de las aplicaciones. De esta forma, puede identificar y resolver posibles vulnerabilidades antes de que se conviertan en problemas importantes y resulten más costosos de solucionar. Por ejemplo, puede incluir pruebas de seguridad para buscar claves de acceso codificadas o el uso de puertos restringidos.

Experimentar y aprender continuamente

+

La investigación, innovación, aprendizaje y mentoría se fomentan e incorporan en los procesos de DevOps. Los equipos son innovadores y se supervisa su progreso. Con la innovación, habrá fracasos. El liderazgo acepta el fracaso y se anima a los equipos a ver el fracaso como una oportunidad de aprendizaje. Por ejemplo, los equipos utilizan herramientas de DevOps para potenciar entornos bajo demanda, lo que les permite experimentar e innovar, tal vez en el uso de nuevas tecnologías para satisfacer las necesidades de los clientes.

Mejorar continuamente

Las métricas reflexivas ayudan a los equipos a monitorear su progreso, evaluar sus procesos y herramientas y a trabajar hacia objetivos comunes y mejoras continuas. Por ejemplo, los equipos se esfuerzan por mejorar las medidas de cumplimiento del desarrollo, como el rendimiento. También se esfuerzan por aumentar la estabilidad y reducir el tiempo medio de restauración del servicio. Con las herramientas de monitoreo adecuadas, puede establecer puntos de referencia de aplicaciones para comportamientos habituales y controlar continuamente las variaciones.

Prácticas de DevOps

Lección 11 de 27

La cultura de DevOps conduce a prácticas de DevOps orientadas a optimizar y mejorar el ciclo de vida del desarrollo, para ofrecer actualizaciones frecuentes de forma fiable y al mismo tiempo mantener la estabilidad.
Para obtener más información, seleccione el símbolo + junto a cada práctica de DevOps.

Comunicación y colaboración

+

Los equipos de DevOps establecen normas culturales sólidas en torno a la transparencia de la información y la comunicación. Estos equipos multifuncionales tienen la propiedad y, por lo tanto, en lugar de evaluar solo su trabajo, tienen en cuenta las necesidades colectivas del proyecto. Crean empatía por los esfuerzos, las alianzas y la confianza de los demás, al tiempo que colaboran para alcanzar objetivos comunes. Reúnen físicamente los flujos de trabajo tradicionales de desarrollo y operaciones y mejoran sistemáticamente la productividad. Las herramientas de DevOps y la automatización del proceso de entrega soportan estos procesos y flujos de trabajo consolidados, coordinan esfuerzos, automatizan tareas repetitivas y facilitan los bucles de retroalimentación necesarios para una buena comunicación y colaboración.

Monitoreo y observabilidad

+

El monitoreo y la observabilidad se utilizan para evaluar la efectividad de los cambios en la aplicación y la infraestructura y estos cómo afectan el rendimiento y la experiencia general del usuario. Parte de los bucles de retroalimentación de DevOps, el monitoreo y la observabilidad ayuda a los equipos a reaccionar, aprender, planificar y mejorar.
Un sistema observable es un sistema que genera suficientes datos de todos los recursos, aplicaciones y servicios en forma de registros, métricas y trazas para obtener visibilidad operativa de todo el sistema. Los registros informan sobre eventos discretos, tales como advertencias. Las métricas capturan información sobre el estado y el rendimiento, como la tasa de solicitud o el tiempo de respuesta. Los rastreos informan sobre las transacciones y el flujo de datos a través de un sistema distribuido, como uno compuesto por microservicios. Al observar un sistema, puede sacar conclusiones concisas sobre la causa de algo.
El monitoreo le indica lo que sucede con el sistema. Al consolidar y visualizar los datos recopilados por un sistema observable a lo largo del tiempo, los equipos obtienen información sobre el rendimiento, identifican tendencias, pueden establecer alarmas y hacer predicciones sobre los resultados esperados.

Integración continua (CI)

+

La integración continua es una práctica de desarrollo de software de DevOps mediante la cual los desarrolladores combinan los cambios en el código en un repositorio central de

forma periódica, tras lo cual se ejecutan versiones y pruebas automáticas. De esta manera, los equipos pueden resolver los problemas de fusión y los defectos de código de forma temprana, cuando resulten más fáciles y rentables de resolver. La integración continua suele realizarse con mayor frecuencia durante la etapa de compilación o integración del proceso de publicación de software. Requiere tanto un componente de automatización (por ejemplo, una CI o servicio de compilación) como un componente cultural (por ejemplo, aprender a integrar a menudo). Los objetivos clave de la integración continua consisten en encontrar y arreglar errores con mayor rapidez, mejorar la calidad del software y reducir el tiempo que lleva validar y lanzar nuevas actualizaciones de software.

Entrega continua/Implementación continua (CD)

+

La entrega continua es una práctica de desarrollo de software en la que cada cambio de código se construye, prueba y luego se implementa automáticamente en un entorno de prueba o ensayo que no sea de producción. Se requiere la aprobación manual antes de enviar a producción. Cuando se implementan correctamente, los desarrolladores siempre tendrán un artefacto de compilación listo para la implementación que ha pasado a través de un proceso de prueba estandarizado.

La implementación continua es similar a la entrega continua, pero con implementación automática en producción. El código probado no necesita una aprobación explícita antes de enviarse a producción.

Arquitectura de microservicios

+

Una arquitectura de microservicios es un enfoque de diseño que construye una aplicación como un conjunto de servicios vagamente relacionados. Cada servicio está diseñado para un conjunto de funciones y se enfoca en resolver un problema específico. Los servicios no necesitan compartir ninguno de sus códigos o implementaciones con otros servicios. Cualquier comunicación entre componentes individuales ocurre a través de API bien definidas. Estos servicios se pueden asignar a equipos plenamente responsables y desarrollarse, probarse e implementarse de manera independiente de otros servicios.

Según la investigación de DevOps Research and Assessment (DORA), el tipo de arquitectura en el que se asienta el equipo es un indicador directo del éxito que tendrá para

lograr una entrega continua. La naturaleza de los microservicios admite desarrollo, actualizaciones, correcciones e implementaciones más rápidos.

Infraestructura como código

+

El desarrollo, las pruebas y la producción se ejecutan en entornos complejos compuestos por hardware y software. El inicio manual y la configuración de entornos no se escala y son propensos a errores.

Infraestructura como código (IaC) es una práctica en la que la infraestructura se aprovisiona y administra a través de técnicas de desarrollo de código y software, como el control de versiones y la integración continua.

El modelo basado en API de la nube permite a los desarrolladores y administradores de sistemas interactuar con la infraestructura mediante programación y a escala, en lugar de tener que configurar recursos manualmente. Debido a que los entornos están definidos por código, se pueden implementar rápidamente con conformidad obligatoria, actualizar con los parches más recientes, revertir a versiones anteriores o duplicarse de formas repetibles. Además, al reaccionar a los cambios del entorno mediante la modificación de este código, puede realizar un seguimiento de los cambios, optimizar los recursos y mejorar el tiempo de actividad del sistema.

Una canalización de DevOps es un conjunto de etapas que llevan el código desde el origen hasta la implementación. El siguiente gráfico muestra las etapas típicas de una canalización DevOps y representa las fases involucradas en una canalización de CI/CD.

Para obtener más información sobre las etapas del ciclo de vida de DevOps, seleccione cada punto.

1.
2.

3.

4.
5.

6.

Una canalización de CI/CD es un buen ejemplo de cómo los equipos de DevOps utilizan herramientas para optimizar los flujos de trabajo y estandarizar las prácticas. Una canalización de CI/CD garantiza la calidad del código, la seguridad y las implementaciones rápidas y consistentes al progresar repetidamente a través de la canalización. Los equipos de DevOps eliminan iterativamente las superposiciones de procesos, los errores humanos y los atascos mediante la automatización.

Cada equipo de DevOps requiere una canalización CI/CD eficiente y confiable. Una canalización de CI/CD requiere una secuencia de herramientas bien integrada.

Herramientas de DevOps

Lección 12 de 27

Las prácticas de DevOps requieren herramientas de DevOps. Las herramientas hacen que los procesos sean más sencillos, consistentes y predecibles.

A medida que aprende acerca de las herramientas de DevOps de AWS, es importante darse cuenta de que se integran con otras herramientas de terceros. Los socios de AWS se integran con las ofertas de AWS y las amplían. Esto significa que puede usar herramientas como GitHub y Jenkins para construir su secuencia de herramientas DevOps, que apoyará a sus equipos y optimizará sus procesos. Use sus herramientas de terceros y de código abierto preferidas con AWS para crear una solución completa.

AWS ofrece servicios que lo ayudan a practicar DevOps en su empresa. Más adelante en el curso, aprenderá sobre las herramientas y los servicios de DevOps de AWS. Por ahora, aquí están las categorías generales de herramientas que puede necesitar para respaldar sus esfuerzos de DevOps.

Para obtener más información, seleccione el símbolo + junto a cada categoría.

Nube

+

Los equipos de desarrollo necesitan innovar rápidamente y ofrecer aplicaciones confiables y seguras. Los equipos confían en proveedores de plataformas en la nube y recursos de informática en la nube para una variedad de tecnologías que respaldan los esfuerzos de desarrollo de aplicaciones. En lugar de comprar, poseer y mantener centros de datos físicos y servidores, los equipos aprovisionan entornos bajo demanda al utilizar proveedores de nube como AWS.

Desarrollo

+

Los equipos de DevOps necesitan colaborar continuamente con los miembros de su equipo. Existen varias herramientas que ayudan a los equipos a desarrollar y entregar más rápido. Los entornos de desarrollo integrados (IDE) ayudan al cliente a escribir, ejecutar y depurar código para aplicaciones. Los kits de desarrollo de software (SDK) son conjuntos de herramientas que permiten a los programadores desarrollar aplicaciones para una plataforma específica. Los repositorios de código fuente o los sistemas de control de versiones almacenan los archivos del proyecto. Según sea necesario, puede acceder a los documentos y al código, ver el historial de revisiones, comparar los cambios a lo largo del tiempo o volver a versiones anteriores.

CI/CD

+

Prácticas tales como pruebas continuas, integración continua (CI) y entrega/implementación continuas (CD) son compatibles con herramientas que brindan continuidad sin problemas en todas las fases de desarrollo. Las herramientas de CI/CD automatizan el código integrado de forma continua que los equipos desarrollan, verifican la conformidad de los estándares, ejecutan pruebas con más frecuencia, promueven código en diferentes entornos de prueba e implementan productos en la infraestructura de forma repetida y confiable. Las herramientas de CI/CD deben ayudar a aportar agilidad a los procesos de desarrollo e implementación de aplicaciones, al tiempo que proporcionan continuamente retroalimentación y alertan a los equipos apropiados sobre cualquier problema.

Ejemplos:

- Herramientas de compilación: Jenkins, Travis CI, AWS CodeBuild
- Herramientas de control de código fuente, repositorios: Git, AWS CodeCommit
- Herramientas de implementación: AWS CodeDeploy, AWS CloudFormation
- Herramientas de automatización de canalización: AWS CodePipeline, Jenkins, GitLab

Automatización de la infraestructura

+

Defina su infraestructura mediante programación, incluidas las restricciones, para aprovisionar de forma repetida y consistente sus entornos (entornos de pruebas de desarrollo, pruebas, almacenamiento provisional, producción). Mediante plantillas, puede

implementar servicios informáticos, permisos, dependencias y más. Puede configurar reglas y automatizar correcciones.

Ejemplos:

- herramientas de automatización de infraestructura: AWS CloudFormation, Terraform, AWS Elastic Beanstalk
- Herramientas de administración de la configuración: Chef, Puppet, AWS OpsWorks

Contenedores e informática sin servidor

+

Los contenedores y los servicios informáticos sin servidor permiten a los desarrolladores centrarse en las aplicaciones y no en los detalles del entorno de alojamiento.

- El código de empaquetado de contenedores, ajustes de configuración y las dependencias necesarias para ejecutar la aplicación. De esta manera, la aplicación es portátil y puede ejecutarse en cualquier servidor. Los contenedores son similares a las máquinas virtuales, pero son más ligeros porque se virtualizan a nivel del sistema operativo (SO). Los contenedores ejecutan cualquier cosa, desde microservicios hasta extensas aplicaciones heredadas. Optimizan la forma en que se crean, prueban e implementan aplicaciones en numerosos entornos. Hacen que la aplicación implementada sea más segura porque las políticas de seguridad se pueden implementar en el nivel de contenedor. Sin embargo, los contenedores requieren la orquestación de contenedores para administrar o programar el trabajo de contenedores individuales.
- Los servicios informáticos sin servidor permiten crear y ejecutar código y permiten que la sobrecarga de infraestructura sea administrada por el proveedor de servicios en la nube, como AWS.

Ejemplos:

- Servicios sin servidor: AWS Lambda, AWS Fargate
- Servicios de contenedor:
    - Tiempos de ejecución: Docker, Containerd
    - Orquestación: Amazon Elastic Container Service (Amazon ECS), Kubernetes, Amazon Elastic Kubernetes Service (Amazon EKS)

Monitoreo y observabilidad

+

El monitoreo y la observabilidad son aspectos clave de DevOps, lo que lo ayuda a ser proactivo en la prevención de desafíos. Con las herramientas, puede recopilar métricas sobre el estado y el rendimiento de su aplicación. Puede capturar la frecuencia de implementación, identificar implementaciones exitosas o fallidas, tráfico de uso de aplicaciones y mucho más. Las herramientas pueden ayudarlo a rastrear los flujos de

solicitudes y transacciones de extremo a extremo a través de un sistema distribuido. Con las herramientas, puede visualizar y analizar registros, métricas y trazas para descubrir nuevos conocimientos sobre el estado, el rendimiento y la disponibilidad del sistema. Con entendimiento, puede optimizar los procesos, mejorar el rendimiento del sistema y reducir el tiempo de inactividad.

Ejemplos: AWS X-Ray, Amazon CloudWatch, AWS Config, AWS CloudTrail

A medida que decida sobre las herramientas, es importante utilizar aquellas que funcionen para su equipo.

Evaluación de conocimientos

Lección 13 de 27

Su equipo de administración ha decidido que desea pasar de su modelo tradicional de desarrollo de software actual y adoptar un modelo DevOps.

¿Qué cambios necesitarán implementar para pasar a DevOps?

- La transición a DevOps requiere contratar profesionales de DevOps e invertir en nuevas herramientas.
- La transición a DevOps requiere un cambio cultural, la implementación de prácticas de DevOps y la utilización de las herramientas adecuadas para automatizar los procesos.
- La transición a DevOps requiere contratar profesionales de DevOps, adoptar nuevos procesos e invertir en nuevas herramientas.
- La transición a DevOps requiere crear nuevos equipos, cambiar todos los procesos y seleccionar las herramientas adecuadas.

Su gerente está preocupado por pasar a DevOps porque cree que todas las herramientas existentes deberán reemplazarse por herramientas de AWS. ¿Qué le puede decir para aliviar sus preocupaciones?

- Asegúrele a su gerente que se debería utilizar la filosofía de DevOps de AWS en todas las empresas. Hágale saber que AWS tiene todas las herramientas más actuales que se necesitan para soportar DevOps. Y que las herramientas actuales que la empresa utiliza son antiguas y necesitan reemplazarse.

- Dígale a su gerente que DevOps es una metodología y no un producto de AWS. Hágale saber que solo se deben reemplazar las herramientas que no estén optimizadas para funcionar en un entorno de DevOps. AWS cuenta con excelentes herramientas, pero también admite herramientas de muchos proveedores líderes del sector.

- Hágale saber a su gerente que DevOps es una metodología y no un producto de AWS. Hágale saber que solo se deben reemplazar las herramientas que no estén optimizadas para funcionar en un entorno de DevOps. AWS tiene excelentes herramientas, pero también soporta todas las herramientas que la empresa elija utilizar.

- Dígale a su gerente que todas las empresas deben utilizar la filosofía de DevOps de AWS. Hágale saber que solo se deben reemplazar las herramientas que no estén optimizadas para funcionar en un entorno de DevOps. AWS cuenta con excelentes herramientas, pero también admite herramientas de proveedores líderes del sector.

Resumen

Lección 14 de 27

En este módulo, aprendió que adoptar una metodología de DevOps requiere crear y nutrir una cultura de DevOps, incorporar prácticas de DevOps y usar las herramientas de DevOps apropiadas.

Ahora puede: Analizar los desafíos que

- Implica adoptar una cultura de DevOps y describir posibles soluciones.
- Identificar oportunidades de automatización en el desarrollo y mantenimiento de aplicaciones.
- Describir los beneficios de desacoplar servicios o componentes.
- Definir la observabilidad y describir su importancia para DevOps.
- Explicar por qué la seguridad es importante en cada fase de la canalización.
- Explicar cómo se integra AWS con herramientas de terceros para la entrega y las implementaciones automatizadas de código.

Información general sobre el módulo

Lección 15 de 27

Obtenga información sobre la transformación de Amazon en DevOps, que comenzó a partir del deseo de responder a las necesidades de los clientes y ofrecer valor más rápido.

Objetivos de aprendizaje

Al final de este módulo, podrá hacer lo siguiente:

- Describir la transición de un enfoque monolítico a una arquitectura de microservicios.
- Describir los beneficios de los equipos más pequeños y autónomos.

Transformación de DevOps de Amazon

Lección 16 de 27

Amazon es un gran ejemplo de cómo una empresa que utilizó prácticas de desarrollo tradicionales se transformó hacia prácticas de entrega de software más flexibles y eficientes con DevOps.

Hoy, Amazon automatiza la entrega de software para lograr más de 150 millones de implementaciones al año. Está configurado para DevOps desde el punto de vista cultural y organizacional. Miles de equipos de software independientes trabajan en paralelo para entregar software de forma rápida, segura, fiable y con tolerancia cero a interrupciones. Pero este no siempre fue el caso.

A principios de la década de 2000, el sitio web amazon.com se desarrollaba y actualizaba. Amazon realizaba prácticas tradicionales de desarrollo. La estructura organizativa era jerárquica, compuesta por equipos de desarrollo, pruebas y operaciones separados. Las prácticas de desarrollo eran monolíticas y la aplicación se desplegaba como una sola unidad. Las prácticas de desarrollo de Amazon tenían todas las dificultades típicas de dependencias de código, procesos manuales de control de calidad e implementaciones engorrosas. Amazon rápidamente se dio cuenta de que se ralentizaban debido a la arquitectura de desarrollo y la estructura organizativa.

El viaje de DevOps de Amazon

Al igual que muchos otros esfuerzos de desarrollo de aplicaciones, el desarrollo temprano en Amazon se llevó a cabo con prácticas tradicionales ineficientes de desarrollo. Finalmente se hizo evidente que había que cambiar algo para que Amazon aumentara la velocidad de desarrollo y la velocidad de implementación para centrarse en las necesidades de los clientes. Este proceso no ocurrió de la noche a la mañana y se dio en muchas etapas. A los fines de simplicidad, lo hemos dividido en tres etapas principales.

Desde las prácticas tradicionales de desarrollo

El sitio web de Amazon.com era un gran monolito arquitectónico. Se trataba de una arquitectura de varios niveles y tenía miles de componentes en ella. Por lo tanto, consistió en una enorme base de código y progresivamente se convirtió en un gran sistema complejo.

Un gran número de desarrolladores trabajaban en una pequeña parte de ese sistema o en nuevas mejoras. Liberar incluso pequeños cambios requería coordinación y esperar hasta el siguiente ciclo de lanzamiento.

Los equipos independientes y especializados soportaban el ciclo de vida del desarrollo que requería entregas de largas transferencias.

La estructura organizativa original de Amazon, la arquitectura monolítica, los equipos especializados y los rígidos ciclos de lanzamiento dificultaban la agilidad y retrasaban el tiempo de comercialización.

Para el desarrollo separado y dos equipos de pizza
Amazon desacopló el desarrollo y avanzó hacia una arquitectura orientada a servicios. También crearon pequeños equipos funcionales de *dos pizzas*, compuestos desde 8 hasta 10 personas. Los equipos *de dos pizzas* estaban alineados con estos servicios, se les dio la propiedad y podían desarrollarse de forma independiente. Los equipos encontraron y eliminaron sistemáticamente redundancias dentro de sus procesos. Esto aceleró el desarrollo, pero Amazon sabía que podían hacer más. Se dieron cuenta de que los procesos manuales, las entregas y los ciclos de liberación comunes seguían causando retrasos.

A procesos optimizados y de automatización
La arquitectura monolítica se desacopló completamente en servicios de un solo propósito y pronto se convirtió en una solución de microservicios. Amazon creó y adoptó herramientas para visualizar y automatizar el proceso de lanzamiento de software, desde el registro de código, las pruebas y la producción. El monitoreo durante todo el desarrollo y luego del lanzamiento, dio confianza a los equipos. Pronto, los equipos se lanzaron de forma independiente, más rápida y fiable.

Amazon quería agilidad y la transformación se convirtió en la base para adoptar DevOps.

- **Descomponer para lograr agilidad**
  Amazon cambió su estructura organizativa y la arquitectura monolítica de la aplicación. Empezaron a implementar una arquitectura orientada a servicios, que ahora se conoce como arquitectura de microservicios. Estos servicios se desacoplaron de la aplicación general, tenían interfaces estándar y se comunicaban entre sí a través de interfaces estándar. Se establecieron reglas, como la compatibilidad con versiones anteriores para garantizar la estabilidad del sistema. Se desarrollaron equipos de *dos pizzas*pequeños, multifuncionales, autónomos e independientes. *Los equipos de dos pizzas* consisten desde 8 hasta 10 personas. Estos equipos tienen la propiedad integral de estos servicios. Las metas organizativas compartidas y

la comunicación permiten que estos pequeños equipos se ajusten, desarrollen e implementen dinámicamente tan rápido como lo necesiten.

- **Usar herramientas y automatizar**
  Los equipos*de dos pizzas*pronto se dieron cuenta de que para maximizar sus esfuerzos y aprovechar mejor los recursos del equipo, tenían que implementar las prácticas recomendadas para la implementación y tenían que aprovechar la automatización. Las métricas y el monitoreo respaldaron sus decisiones. Se pusieron en marcha plantillas, procesos y herramientas para apoyar a los equipos. La automatización se utilizó en las fases de preproducción y postproducción. Por ejemplo: se crearon directivas de seguridad para impedir que los equipos implementaran código con riesgos de seguridad conocidos. Las pruebas se realizaron antes y con mayor frecuencia, para detectar errores al principio del desarrollo. La automatización comenzó a aparecer con las tareas de origen y compilación, en las pruebas y en la fase de implementación y pronto se convirtió en CI/CD completo. La automatización aumentó la velocidad y redujo los errores por parte del equipo y en el producto final.
- **Transformar la cultura**
  La transformación de DevOps no ocurrió de la noche a la mañana, sino más bien a través de pequeños incrementos durante un largo periodo de tiempo. Tuvo un impacto en toda la empresa. Cambió la forma en que las personas piensan sobre su trabajo. Se dio valor a la innovación, a la mejora y a los resultados empresariales respaldados por métricas. En Amazon, los equipos pasaron de ser equipos separados a equipos multifuncionales. La educación y la colaboración pasaron a ser un foco y se incorporó el talento. A pesar de que existe la especialización en conocimientos y la diversidad de equipos, cada miembro es consciente de cómo sus esfuerzos afectan los objetivos generales del equipo y aprenden intentándolo. Los equipos pequeños, la pertenencia, la utilización de talentos internos, la planificación ascendente, la comunicación, la colaboración, la automatización y la mejora continua a través de la innovación y el monitoreo, se han convertido en una fuerza impulsora de la cultura en Amazon.

Amazon y otras organizaciones han pasado de prácticas de desarrollo tradicionales bien establecidas a DevOps. Con DevOps, la organización puede innovar, ofrecer productos y escalar a un ritmo más rápido que las organizaciones que utilizan procesos tradicionales de desarrollo de software y de administración de infraestructura. De esta manera, pueden servir mejor a sus clientes y competir de manera más eficaz en el mercado.

La Amazon Builders' Library, escrita por los líderes técnicos superiores e ingenieros de Amazon, se ha creado para ayudarlo con sus esfuerzos de desarrollo. Es una colección de artículos existentes que describen de qué manera Amazon desarrolla, diseña, lanza y opera la tecnología.

Resumen

Lección 17 de 27

En este módulo, aprendió acerca de la transformación de Amazon a DevOps.

Ahora puede:

- Describir la transición de un enfoque monolítico a una arquitectura de microservicios.
- Describir los beneficios de los equipos más pequeños y autónomos.

Información general sobre el módulo

Lección 18 de 27

En este módulo se presentan los recursos de infraestructura y herramientas de AWS para los profesionales de DevOps. AWS ofrece un conjunto de servicios específicos de DevOps, que abarcan desde el aprovisionamiento y administración de la infraestructura hasta la implementación de código de aplicación, la automatización de la implementación con integración continua y entrega continua (CI/CD) y la supervisión.

Objetivos de aprendizaje

Al final de este módulo, podrá:

- Enumerar los servicios de AWS disponibles para implementar una metodología de DevOps exitosa.
- Identificar los servicios de AWS utilizados para automatizar la integración continua y el proceso de entrega continua.

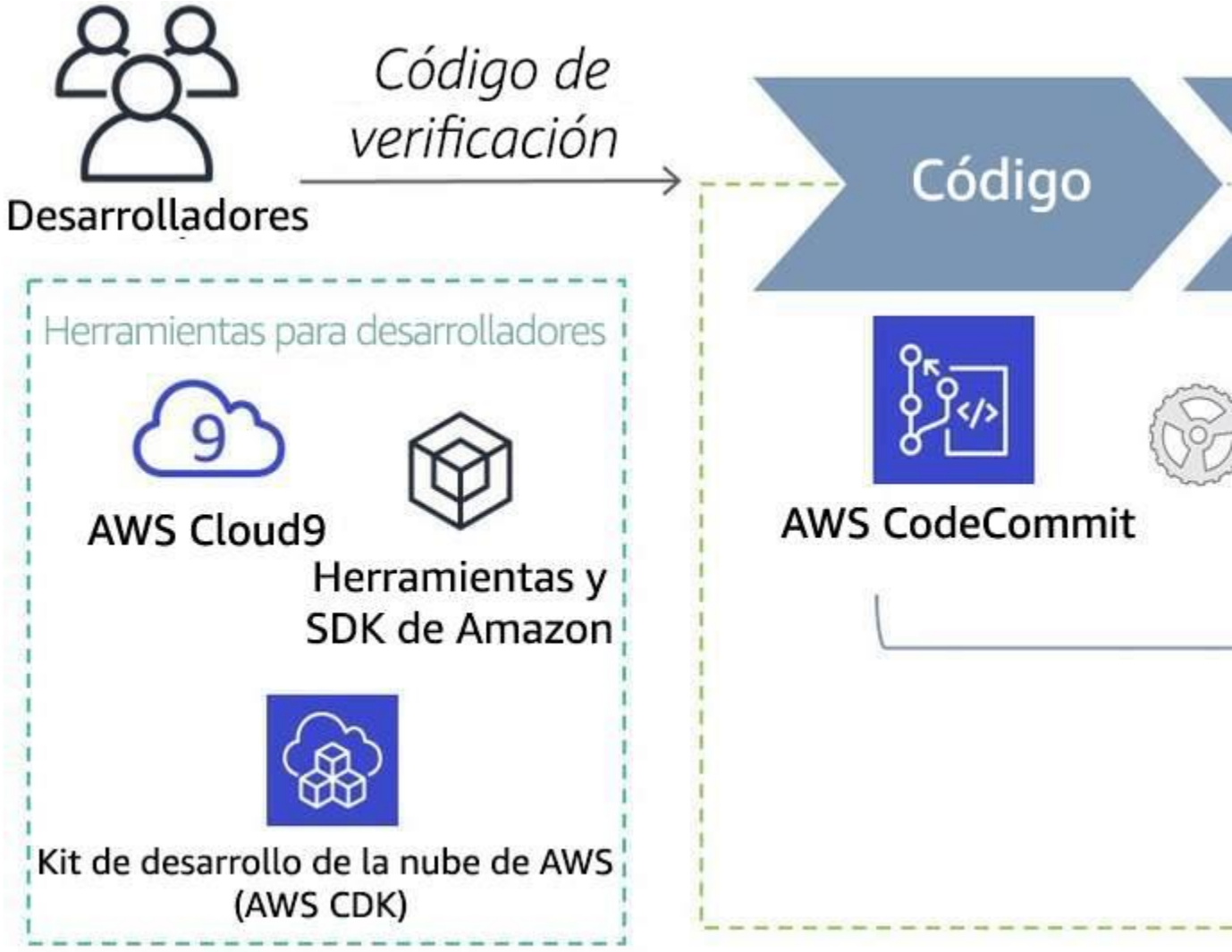
Herramientas de DevOps de AWS

Lección 19 de 27

Para utilizar DevOps con éxito, necesita herramientas eficaces que apoyen a su equipo y a sus flujos de trabajo y que además lo ayuden a ofrecer valor más rápido. AWS admite DevOps con una serie de servicios que los desarrolladores utilizan en cada etapa del ciclo de vida de la aplicación. A medida que obtenga más información sobre los servicios de AWS, recuerde que su solución DevOps podría incluir servicios de AWS y soluciones de terceros.

El siguiente diagrama muestra una canalización de DevOps creada con algunos de los servicios de AWS que admiten esas fases. La canalización comienza cuando un desarrollador registra el código.

Para obtener más información, seleccione cada punto.



1.
2.
3.
4.
5.
6.
7.

Para obtener más información sobre cada servicio, seleccione la pestaña correspondiente.

AWS CODEPIPELINE **AWS CODECOMMIT** **AWS CODEBUILD** **AWS CODEDEPLOY**

AWS CodePipeline es un servicio de entrega continua que permite modelar, visualizar y automatizar los pasos necesarios para lanzar un software.

¿Por qué usarlo?

Con CodePipeline puede:

- Capturar y visualizar su canalización, ejecutarla, ver el estado en tiempo real y reintentar acciones fallidas.
- Automatizar procesos de lanzamiento, eliminar errores humanos, acelerar la entrega y mejorar la calidad de la versión.
- Establecer consistencia en la versión.
- Incorporar las herramientas de origen, compilación e implementación.
- Ver los detalles del historial de canalización.
- Integrar con herramientas de terceros y de AWS para crear, probar e implementar el código cuando se notifique un cambio de código.

Monitoreo
Puede monitorear la canalización de varias maneras a fin de garantizar su rendimiento, confiabilidad y disponibilidad, y para encontrar formas de mejorarla. Puede monitorear la canalización directamente desde la consola de AWS CodePipeline, la interfaz de línea de comandos (CLI), el uso de Amazon EventBridge o AWS CloudTrail.

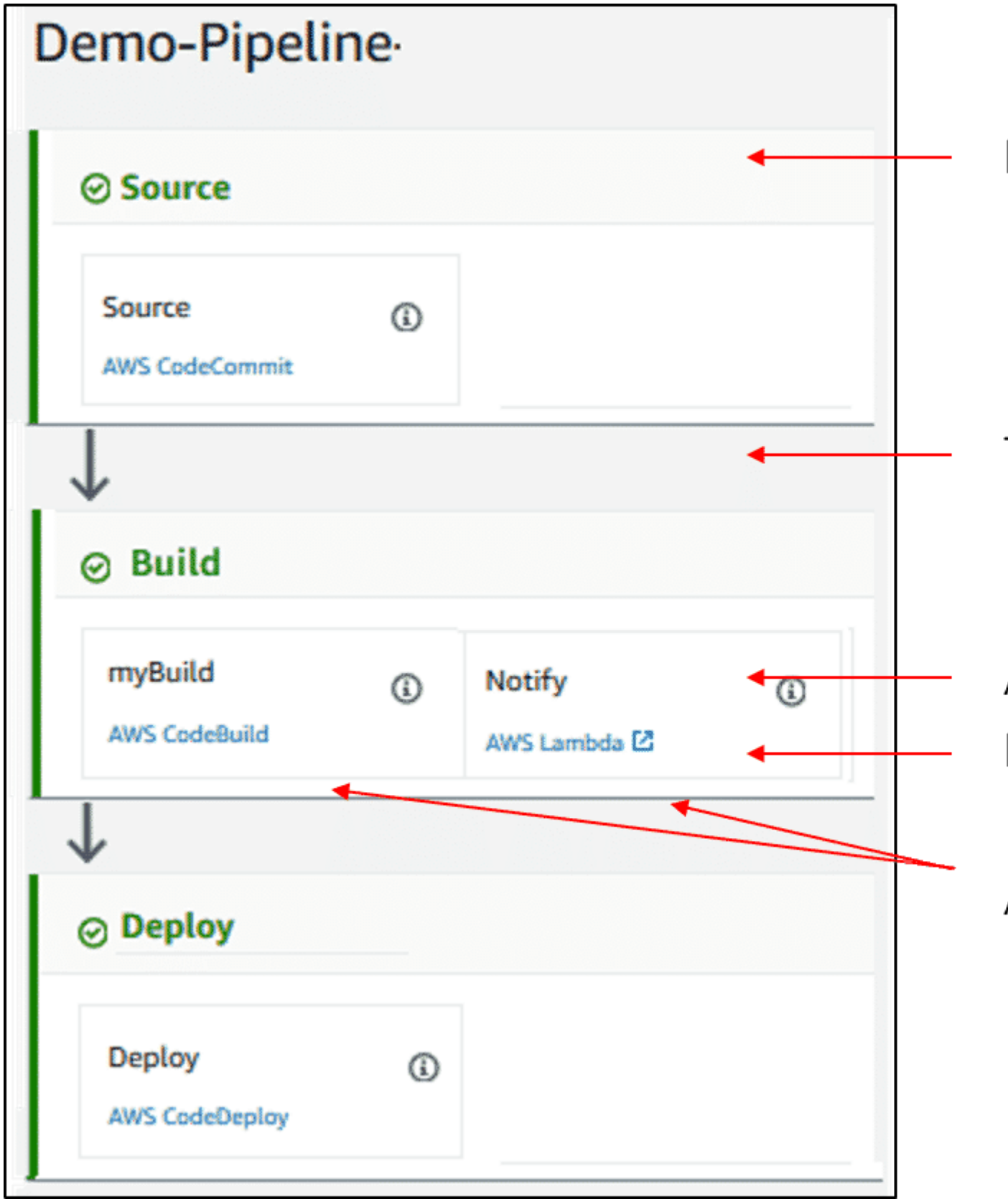
Seguridad
La seguridad es una parte importante de cualquier canalización. CodePipeline admite permisos a nivel de recursos, lo que le permite especificar qué usuario puede realizar qué acción en la canalización. Algunos usuarios pueden tener acceso solo de lectura a la canalización, mientras que otros pueden tener acceso a una etapa o acción determinada dentro de una etapa. Para obtener más información acerca de la seguridad, consulte Seguridad en AWS CodePipeline.

Cómo funciona
Con CodePipeline, puede modelar su proceso de lanzamiento con una serie de etapas y acciones.
CodePipeline divide el flujo de trabajo de la versión en una serie de etapas, como código, compilación y prueba. Cada etapa puede tener una serie de acciones que deben realizarse.

- Las acciones son tareas que se pueden ejecutar en secuencia o en paralelo entre sí. Están asociadas a un proveedor de servicios que ejecuta la acción o pueden requerir la intervención del usuario. Los proveedores de servicios pueden ser servicios de AWS (como CodeBuild, Amazon Simple Storage Service (Amazon S3), AWS Lambda y AWS CloudFormation) o servicios de terceros (como Jenkins y TeamCity).
- Los tipos de acción incluyen:
    - Origen (donde se almacena el origen)
    - Compilación (cómo construir la aplicación)
    - Prueba (cómo probar la aplicación)
    - Implementación (cómo implementar la aplicación)
    - Aprobación (aprobación manual y notificaciones)

- Invocación (invocar una función personalizada).

A continuación, se presenta un ejemplo de una canalización de tres etapas:



Servicios adicionales que debe conocer

Servicios de entorno de desarrollo integrado (IDE)

+

Un entorno de desarrollo integrado (IDE) mejora la productividad de un desarrollador con características como el resaltado de sintaxis, autocompletado y sugerencias para la sintaxis, etc. Los IDE se pueden instalar localmente o basados en la nube y acceder desde un explorador web.

- **AWS Cloud9** es el IDE basado en la nube de Amazon, que puede utilizar para escribir, ejecutar y depurar el código con solo un navegador. Puede utilizar AWS Cloud9 para realizar cambios de código en un repositorio de CodeCommit. También puede utilizar AWS Cloud9 con los kits de desarrollo de software (SDK) específicos de AWS.

Servicios de administración de infraestructuras

+

Manejar la infraestructura como código (IaC) aumenta la agilidad y eficiencia del equipo, hace que la infraestructura sea más sólida y disminuye el costo general. Esto se debe a que con IaC, la implementación y actualización de la infraestructura es consistente y repetible.

- **AWS CloudFormation** es un servicio de infraestructura como código. Mediante las plantillas, puede describir los recursos necesarios y las dependencias, de modo que pueda aprovisionarlos y administrarlos de forma rápida y consistente a lo largo de los ciclos de vida. Cree, actualice y elimine una pila entera como una sola unidad, con la frecuencia que necesite, en lugar de administrar recursos individualmente. Puede administrar y aprovisionar pilas en varias cuentas de AWS y regiones de AWS.
- **AWS OpsWorks** es un servicio de administración de configuración. Proporciona instancias administradas de plataformas de automatización de Chef y Puppet y lo ayuda a automatizar cómo se configuran, implementan y administran los servidores en las instancias de Amazon EC2 o entornos informáticos en las instalaciones.

Contenedores y servicios sin servidor

+

AWS ofrece una serie de servicios que respaldan los esfuerzos de desarrollo y mejoran la canalización, entre ellos:

- **AWS Lambda** es un servicio informático sin servidor que permite escribir código y ejecutarlo cuando se invoque. Con las funciones de Lambda, puede

personalizar la canalización de CI/CD. Por ejemplo, puede aprobar acciones de lanzamiento, detener el flujo de liberación y controlar el flujo de tráfico durante las implementaciones, etc.

- **Amazon Elastic Container service (Amazon ECS)** es un servicio de administración de contenedores altamente escalable y de alto rendimiento que admite contenedores Docker. Permite ejecutar aplicaciones fácilmente en un clúster administrado de instancias de Amazon Elastic Compute Cloud (Amazon EC2). ECS elimina la necesidad de instalar, operar y escalar software de administración de contenedores.

Servicios de monitoreo

+

El monitoreo es una parte clave de DevOps. Ayuda al usuario a ser proactivo en la prevención de desafíos antes de que ocurran. Aquí hay algunos servicios de monitoreo que debe conocer.

- **AWS X-Ray** es un sistema de rastreo distribuido. Lo ayuda a analizar y depurar las aplicaciones distribuidas, como las creadas con arquitectura de microservicios o sin servidor. Proporciona una vista completa de las solicitudes a medida que viajan por la aplicación y muestra un mapa de los componentes subyacentes de la aplicación.
- **Amazon CloudWatch** es un servicio de monitoreo y administración que proporciona datos e información útil para aplicaciones y recursos de infraestructura de AWS, híbridos y locales. Con CloudWatch, puede realizar un seguimiento de los recursos y el rendimiento de las aplicaciones, recopilar y monitorear archivos de registro, solucionar problemas y establecer notificaciones de alarma.
- **AWS Config** es un servicio que permite examinar, auditar y evaluar las configuraciones de los recursos de AWS. AWS Config monitorea y registra continuamente las configuraciones de recursos de AWS. Permite automatizar la evaluación de configuraciones grabadas en comparación con las configuraciones deseadas.
- **AWS CloudTrail** es un servicio que permite la gobernanza, la conformidad, la auditoría operativa y la auditoría de riesgos de la cuenta de AWS. Con CloudTrail, puede registrar, monitorear y conservar continuamente la actividad de la cuenta relacionada con acciones en la infraestructura de AWS. CloudTrail proporciona el historial de eventos de la actividad de la cuenta de AWS, incluidas las acciones realizadas a través de AWS Management Console, los SDK de AWS, las herramientas de línea de comandos y otros servicios de AWS. Este historial de eventos simplifica el análisis de seguridad, el seguimiento de cambios de recursos y la solución de problemas. Además, puede utilizar CloudTrail para detectar actividades

inusuales en las cuentas de AWS. Estas capacidades ayudan a simplificar el análisis operativo y la solución de problemas.

Para obtener más información, seleccione el símbolo + junto a cada categoría.

A medida que decida sobre las herramientas, es importante utilizar aquellas que funcionen para su equipo.

Para obtener más información sobre los servicios de AWS mencionados en este curso o si desea obtener más información sobre otros servicios de AWS, consulte la documentación de AWS.

Evaluación de conocimientos

Lección 20 de 27

Categorizar la cartera de servicios de AWS DevOps

Seleccione cada servicio de AWS y póngalo en la categoría adecuada.

0/8 Corrección de tarjetas

REPRODUCIR OTRA VEZ

AWS CodeCommit

AWS X-Ray

AWS CodePipeline

Amazon CloudWatch

AWS CodeBuild

AWS CodeDeploy

AWS Lambda

AWS CloudFormation

Herramienta CI/CD

Infraestructura como código

Monitoreo y observabilidad

Contenedores e informática sin servidor

¿Qué servicio de monitoreo y observabilidad sería más útil para depurar y rastrear aplicaciones distribuidas, como las creadas con una arquitectura de microservicios? (Seleccione la mejor respuesta.)

- AWS X-Ray
- Amazon CloudWatch
- AWS Config
- AWS CloudTrail

Forma parte de un equipo de DevOps que planea configurar, aplicar parches y mantener el software del servidor de compilación. El equipo no desea poseer la administración o la creación de hardware o software. Desea una herramienta que escale automáticamente para satisfacer el volumen de compilación, procese inmediatamente cada compilación que envíe y pueda ejecutar compilaciones separadas simultáneamente. ¿Cuál de los siguientes servicios proporcionará al equipo la funcionalidad necesaria?

- AWS CodePipeline
- Plugin para Jenkins
- AWS CodeBuild
- AWS CodeDeploy

Su empresa es una empresa global de TI que recientemente adoptó DevOps. El equipo ha elegido AWS CodeDeploy como herramienta de implementación para gestionar las implementaciones de aplicaciones. ¿Cuál de las siguientes características lo convierte en una opción correcta para la automatización? (Seleccione CUATRO.)

- Puede implementarse en instancias en las instalaciones y de Amazon EC2.
- Puede ampliar las implementaciones a varias regiones.
- El agente solo funciona en entornos Windows y Linux.
- Puede realizar implementaciones de tiempo de inactividad cero.

- Funciona con una variedad de sistemas de administración de configuración, integración continua, control de origen e implementación.
- Es fácil para la empresa alojar repositorios Git privados seguros y altamente escalables.
- Lo ayuda a evitar el tiempo de inactividad durante la implementación.

Resumen

Lección 21 de 27

Aproveche las prácticas recomendadas de DevOps y automatice las líneas de lanzamiento, parcial o totalmente. Los servicios de AWS proporcionan a los equipos de DevOps la capacidad de implementar canalizaciones de CI/CD. AWS proporciona servicios para crear, almacenar, implementar y monitorear aplicaciones.

Ahora puede:

- Enumerar los servicios de AWS disponibles para implementar una metodología de DevOps exitosa.
- Identifique los servicios de AWS utilizados para automatizar la integración continua y el proceso de entrega continua.

Información general sobre el módulo

Lección 22 de 27

Este módulo se basa en una demostración. A través de la demostración, usted examina cómo DevOps y los servicios de AWS pueden mejorar las implementaciones al hacer que el proceso de implementación sea coherente y repetible. La demostración muestra cómo se pueden utilizar los servicios de AWS para crear y ejecutar una canalización de CI/CD de DevOps a fin de automatizar la implementación de una aplicación en funcionamiento en varias regiones de AWS.

Una región es un concepto de AWS que representa una ubicación física en todo el mundo donde AWS aloja un clúster de centros de datos.

Objetivos de aprendizaje

Al final de este módulo, podrá:

- Explicar cómo utilizar AWS CodeDeploy para implementar aplicaciones web en servidores de Amazon Elastic Compute Cloud (Amazon EC2) en varias regiones.
- Explique cómo utilizar AWS CodePipeline para establecer una integración continua y
  realizar una implementación continua en varias regiones.

Demostración: Crear y controlar una canalización de CI/CD

Lección 23 de 27

En esta demostración, verá cómo se utilizan los servicios de AWS DevOps para crear y controlar una canalización de integración y entrega continuas (CI/CD). Esta canalización de versión automatizará la implementación de una aplicación en funcionamiento en varias regiones. La aplicación que se encuentra en implementación es una página web sencilla. Verá cómo controlar el flujo de trabajo y acelerar la canalización al eliminar las intervenciones manuales.

La canalización CI/CD se crea mediante AWS CodePipeline. La canalización utiliza AWS CodeCommit, AWS CodeDeploy y AWS Lambda para automatizar y controlar la implementación de código nuevo en entornos aprovisionados.

Información general de alto nivel de la demostración

+

Estos pasos se han completado antes de la demostración:

- La infraestructura necesaria para que la aplicación se ejecute se aprovisionó en dos regiones de AWS.
- Junto con el origen de la aplicación, se ha creado un archivo appspec.yml y scripts auxiliares que se han proporcionado con el código.
- Se ha creado una función Lambda. Cuando se invoca, comprueba algo de texto en un sitio web.

Durante la demostración, la canalización se creará mediante los siguientes pasos:

1. La demostración comienza con una revisión rápida de la infraestructura aprovisionada y el archivo de plantilla que se utilizó para crearlas.
2. AWS CodeCommit está configurado con el fin de contener el código y es el servicio de integración continua que inicia la canalización con cada cambio de código.
3. AWS CodeDeploy se configura con detalles sobre cada región de implementación. CodeDeploy es el servicio de implementación continua que instala la aplicación en la infraestructura.
4. AWS CodePipeline utiliza los servicios configurados para crear canalizaciones. Primero, se crea una canalización de dos etapas que se implementa automáticamente en la región 1. Luego, se avanza sobre la canalización y agregamos dos nuevas etapas. Esta canalización de versión implementará la aplicación web simple en dos regiones de AWS distintas. La canalización implementa automáticamente los cambios de código en la región 1, pero requiere aprobación manual antes de implementar en la región 2. Finalmente, se utiliza una función Lambda para acelerar la canalización y mantener el control. Sustituye la puerta de aprobación manual con automatización.

Servicios citados en la demostración

+

En la demostración se muestran los siguientes servicios:

- AWS Cloud9 se utiliza como IDE con el objetivo de desarrollar y realizar cualquier cambio de código.
- AWS CodeCommit se utiliza a fin de crear un repositorio de control de código fuente para alojar el código de demostración.
- AWS CodeDeploy se utiliza con el propósito de implementar el paquete de aplicación de demostración en servidores de Amazon EC2 en varias regiones de AWS (primero en us-west-2 y luego en us-east-1).
- AWS CodePipeline se utiliza para organizar las fases de la canalización de CI/CD. La canalización utiliza Amazon CloudWatch Events a fin de iniciar automáticamente la canalización cuando se envían cambios de origen al repositorio.
- AWS CloudFormation se utilizó antes de la demostración, con el propósito de crear la infraestructura y los recursos de apoyo para el entorno de aplicaciones de demostración, como los servidores de Amazon EC2, el grupo de seguridad de manera que se pueda controlar el tráfico a esas instancias, implementar scripts y grupos de implementación con el fin de alojar la Web en varias regiones.
- Amazon EC2 utiliza instancias de Amazon Linux para alojar la página web. Amazon EC2 es un servicio web que proporciona una capacidad informática segura y de tamaño variable en la nube.

Para ver el video, seleccione el botón de reproducción.

Transcripción del video

[Yianna] Hola, habla Yianna. En esta demostración, verá cómo utilizar las herramientas de AWS DevOps para crear y controlar una integración continua y una canalización de implementación continua, que automatiza la implementación de una aplicación web sencilla en varias regiones de AWS.

Aquí hay una descripción rápida de lo que verá en la demostración.
En primer lugar, revisaremos el entorno ya aprovisionado en el que se implementará la aplicación. Mediante un archivo de plantilla con parámetros que detalla el ajuste y la configuración de los recursos necesarios, AWS CloudFormation aprovisionó entornos en la región 1, EE. UU. Oeste, Oregón y la región 2, EE. UU. Este, Norte de Virginia. La puesta en marcha y la reducción de los recursos del entorno en función de las condiciones podría ser parte de la canalización de CI/CD, pero lo hicimos fuera de la canalización.
Luego configuramos los servicios necesarios para la canalización de CI/CD. Mediante AWS CodeCommit, se crea un repositorio basado en Git de código

fuente y es donde los desarrolladores insertarán los cambios de código. En esta demostración, el desarrollador utiliza AWS Cloud9 como entorno de desarrollo, clona el repositorio, edita el código e introduce cambios en AWS CodeCommit. AWS CodeDeploy se utiliza para crear la configuración de implementación de la aplicación, que detalla dónde y cómo implementar la aplicación. Mediante AWS CodePipeline, se crea una canalización de CI/CD que utiliza estos servicios y automatiza la implementación de la aplicación en los entornos aprovisionados.
Finalmente, verá cómo controlar el flujo de trabajo de canalización al agregar aprobaciones e identificar oportunidades para eliminar intervenciones manuales.

Comencemos.
Echemos un vistazo a esos entornos de destino aprovisionados. Al iniciar sesión en la consola de administración de AWS, tenga en cuenta que pertenecemos a la región EE. UU. Oeste, Oregón. En la Management Console podemos localizar servicios específicos y navegar a las consolas específicas de servicio. Seleccione AWS CloudFormation para abrir la consola del servicio. AWS CloudFormation aborda la infraestructura como código. Esta pila se creó a partir de una plantilla. Veamos los detalles.

En la pestaña Resource (Recurso), se crearon, entre otras cosas, dos instancias EC2 y el grupo de seguridad para controlar el tráfico a esas instancias. En la pestaña Parameter (Parámetro), vemos los parámetros de la plantilla y los valores proporcionados; esta pila aprovisiona dos instancias EC2 y etiqueta cada una de ellas con tagKey, Name, TagValue y My-Demo-Web-App. Nos referiremos a estas etiquetas nuevamente en CodeDeploy. En la pestaña Template (Plantilla), verá la plantilla utilizada para crear la pila. Al examinar la sección de recursos, verá el tipo de recursos que se declaran, por ejemplo EC2 y las propiedades de esos recursos. La sección UserData se utiliza para realizar tareas comunes de configuración de automatización e incluso para ejecutar scripts después de iniciar la instancia. En esta sección, por ejemplo, se especifica la instalación del agente AWS CodeDeploy. Se utilizó la misma plantilla para crear recursos en la región 2.

Al cambiar a la región 2, vemos la pila. Al mirar los detalles, en la pestaña Parameters (Parámetros), vemos que la región 2 tiene tres instancias EC2. Recuerde que la región 1 solo tenía dos instancias EC2. Al navegar a la consola de servicio de Amazon EC2, vemos las tres instancias de EC2 en la región 2 y al cambiar a la región 1 vemos las dos instancias. Se aprovisionaron los entornos de destino para nuestra implementación de aplicaciones.

Ahora, configuraremos los servicios necesarios para la canalización de CI/CD: AWS CodeCommit como servicio de integración continua y AWS CodeDeploy como servicio de implementación continua. Comencemos a configurar AWS CodeCommit.

Vaya a la consola de AWS CodeCommit y, para crear el repositorio de código fuente, seleccione Create repository (Crear repositorio). Asigne un nombre al repositorio, myAppRepo y seleccione Create (Crear). Se crea un repositorio basado en Git. Copie la URL HTTP proporcionada, ya que la necesitaremos cuando clonemos el repositorio desde AWS Cloud9.

Vaya a la consola de AWS Cloud9 y abra el IDE existente. En el margen izquierdo, verá los archivos del proyecto. Aquí puede editar los archivos. En el terminal de AWS Cloud9, puede utilizar comandos Git para trabajar con el repositorio de AWS CodeCommit. Use la clonación de Git con la URL que copió antes. Tenga en cuenta la carpeta creada. En la ventana del terminal, cambie al directorio. Cree una ramificación principal para el trabajo.

El código de la aplicación de demostración ya se ha creado. Al arrastrar y soltar los archivos estos se cargan en el IDE. La aplicación de demostración se compone de datos de usuario en línea, para que un servidor de Amazon EC2 cree una página web estática y muestre metadatos de instancia en la página web. Lo veremos cuando implementemos. Al utilizar git add, git commit y git push, los archivos se registran en el repositorio CodeCommit.

Si cambia a la pestaña del servicio AWS CodeCommit, tenga en cuenta que el repositorio ahora tiene los archivos dentro de la ramificación principal. Acaba de ver cómo crear un repositorio basado en Git en AWS CodeCommit y cómo los desarrolladores pueden utilizar AWS Cloud9 para desarrollar y confirmar el código de forma segura en ese repositorio.

Continuemos la configuración de AWS CodeDeploy y definamos cómo implementar la aplicación.
En el panel de navegación de las herramientas para desarrolladores, vaya a la consola de AWS CodeDeploy, seleccione Deploy (Implementar) y luego, Applications (Aplicaciones). Seleccione Create an application name it CodeDeploy-Web-App (Crear un nombre de aplicación CodeDeploy-Web-App). La aplicación se creará mediante el uso de la plataforma informática EC2/en las instalaciones. Cree un grupo de implementación y asígnele el nombre CodeDeploy-Web-App-Group-Region1. El servicio de implementación requiere permisos para implementar la aplicación. Esto se definió con la plantilla CloudFormation. Utilice la clave del par de claves de etiqueta, nombre y valor, My-Demo-Web-App, para localizar las instancias EC2 para la región 1. Se localizaron dos instancias EC2 para la región 1. AWS CodeDeploy requiere el agente AWS CodeDeploy cuando se implementa en instancias EC2, pero ya se instaló durante el aprovisionamiento. No utilizamos un balanceador de carga, así que desactívelo. Finalice la creación de la agrupación de implementación.

Ahora AWS CodeDeploy está configurado con detalles sobre cómo implementar la aplicación en la región 1. Del mismo modo, CodeDeploy debe configurarse para la región 2. Por razones de tiempo, esto ya se hizo.

Es hora de crear la canalización. AWS CodePipeline es un servicio de entrega continua que lo ayuda a definir y automatizar los pasos necesarios para lanzar el software. La canalización se ajustará a los procesos, los flujos de trabajo. Una vez que cree la canalización, puede utilizarla de nuevo para actualizar de forma rápida y fiable la infraestructura y lanzar la aplicación.

Construyamos nuestra canalización en etapas mediante el uso de los servicios ya configurados. Desde las herramientas para desarrolladores, vaya a AWS CodePipeline. Seleccione Create pipeline (Crear canalización) para que lo guíen a través de la creación de canalizaciones. Asigne el nombre MultiRegionDeploy a la canalización. AWS CodeCommit es nuestro proveedor de origen, pero tenga en cuenta que puede utilizar Amazon S3 o GitHub. Especifique el repositorio. MyAppRepo es el que creamos antes. Nos centramos en la ramificación principal. Los Amazon CloudWatch Events se utilizarán para iniciar automáticamente la canalización cuando se produzca un cambio, como enviar código. La aplicación web no tiene una etapa de compilación, pero si la tiene, estas son algunas opciones. Para la etapa de implementación, el proveedor es AWS CodeDeploy. Seleccione el nombre de la aplicación y el grupo de implementación. Haga clic en Next (Siguiente). Revise la configuración y cree la canalización.

La canalización comienza a ejecutarse, a medida que esta avanza cada etapa informa el estado, mucho antes de que se implemente la aplicación.

Al hacer clic en los detalles de la etapa de implementación, puede ver que la aplicación se implementó correctamente en ambas instancias EC2 en la región 1. Desde aquí, podemos navegar a la aplicación implementada. Observe los detalles de la región. Esta era una canalización fácil para la implementación continua. La mejoraremos.

Hasta ahora, la canalización de implementación continua que consta de la etapa de origen, administrada por el servicio AWS CodeCommit y una etapa de implementación, administrada por AWS CodeDeploy. Ahora tenemos que implementar en la segunda región.

Para demostrar cómo puede agregar control al flujo de trabajo de canalización de CI/CD, agreguemos una puerta de aprobación antes de implementar en la segunda región. Editemos esta canalización.

Haga clic en Add stage (Agregar etapa). Asígnele el nombre Puerta de aprobación. Agregue una acción denominada Aprobación. Esta es una acción de aprobación manual. Los tableros de mensajes y otras opciones podrían configurarse aquí, pero las omitimos por ahora. Ahora agregue una

etapa de implementación que se desplegará en la segunda región. Asígnele el nombre Deploy-Region 2. Agregue una acción para esta etapa denominada Deploy. El proveedor de servicios es AWS CodeDeploy. En la etapa de implementación anterior, usted implementó en EE. UU. Oeste, Oregón. Ahora, implementará en EE. UU. Este, Norte de Virginia. Continúa con el uso del mismo origen, el nombre de la aplicación de configuración CodePipeline es CodeDeploy-Web-App y el grupo de implementación es CodeDeploy-Web-App-Group-Region2. La etapa ahora está definida. Guarde la canalización y ejecútela.

Se ejecutan las dos primeras etapas. A continuación, el sistema espera la aprobación. En cuanto a la página web, el encabezado indica que es una región principal. Rechazaremos esta implementación y solicitemos actualizaciones para el encabezado. Hagámoslo más global. Desde que rechazamos, la etapa final de la canalización no se ejecuta.

Veamos la canalización en acción. El desarrollador actualizará el código. Cuando se registra este código, arrancará la canalización, que implementará el código en la región 1 y esperará una aprobación. Una vez confirmado y aprobado el cambio, tendrá lugar la implementación en la segunda región.

De vuelta en AWS Cloud9, el desarrollador edita la aplicación. El desarrollador actualizó el código, los cambios se registran con los comandos, git add, git commit y git push. Los cambios de código inician la canalización.

Las etapas de origen e implementación se ejecutan y, una vez más, la canalización espera la aprobación. La página web está abierta desde antes. Actualícemosla y verifiquemos las actualizaciones. Con las actualizaciones en marcha, se ve mucho mejor. Aprobemos la canalización para que pueda continuar.

Con la aprobación, se inicia la etapa final y la aplicación se implementa en la segunda región. Desplácese a una de las máquinas y vea la página web. Se ve bien.

Mejoremos nuestra canalización de DevOps al agregar más automatización. Sustituimos la aprobación manual, que ralentiza las cosas, por una comprobación automatizada realizada por una función Lambda.

Edite la canalización y agregue una etapa, Region-Readiness-Check. La acción para esta etapa, automatic-approval-check es proporcionada por una función Lambda. AWS Lambda es un servicio informático sin servidor que ejecuta el código en respuesta a eventos. La función ya se ha creado y se denomina Region-approval-test. Copie la dirección de una de las páginas web, esta será la entrada; el lugar donde la función Lambda se verificará. Para guardar los cambios, seleccione Done (Listo). Ahora elimine la Puerta de aprobación manual definida anteriormente ya que Region-Readiness-Check la reemplaza. Guarde la canalización actualizada y seleccione Release changes (Aplicar cambios).

La canalización se ejecuta. La prueba de AWS Lambda no encontró ningún problema y la canalización pudo implementarse de forma consistente en ambas regiones.

Esto concluye la demostración, donde vio cómo crear y controlar una canalización CI/CD de DevOps mediante el uso de los servicios de AWS. Gracias por ver este video.

Resumen

Lección 24 de 27

En este módulo, vio cómo DevOps y los servicios de AWS pueden mejorar las implementaciones mediante un proceso de implementación coherente y repetible.

Ahora puede hacer lo siguiente:

- Explicar cómo utilizar AWS CodeDeploy para implementar aplicaciones web en servidores de Amazon EC2 en diferentes regiones.
- Explicar cómo utilizar AWS CodePipeline para establecer una integración continua y Realizar una implementación continua en varias regiones.

Evaluación de conocimientos

Lección 25 de 27

¿Cuáles son los principales beneficios de implementar DevOps? (Seleccione

SEIS)

- Mayor velocidad
- Funcionalidad altamente dependiente
- Colaboración mejorada
- Equipos altamente especializados
- Seguridad mejorada
- Entrega rápida
- Requisitos rígidos
- Escalado rápido
- Entrega fiable de aplicaciones de calidad

A continuación, una las características que describen las prácticas de automatización de integración continua, entrega continua e implementación continua. Arrastre los elementos para ordenarlos en la categoría adecuada.

Integración continua
Entrega continua
Implementación continua

El cliente ha decidido adoptar DevOps. El equipo de DevOps elige las herramientas adecuadas para el proceso de control de fuente. Desearían saber cómo se diferencia AWS CodeCommit de otros sistemas de control de código fuente basados en Git. ¿Qué les dirá para ayudarlos a entender las diferencias? (Seleccione CUATRO)

- No es necesario alojar, mantener y hacer copias de seguridad de sus propios servidores de control fuente.
- Cifra automáticamente los archivos en tránsito y en reposo.
- Se basa en servicios de AWS de alta disponibilidad, redundantes y duraderos.
- Permite almacenar cualquier número de archivos, pero el repositorio tiene un límite de tamaño.
- Aumenta la velocidad y la frecuencia del ciclo de vida de desarrollo al mantener los repositorios cerca de las compilaciones.
- No ofrece la capacidad de configurar notificaciones.
- Es un servicio de compilación totalmente administrado que permite crear, probar e integrar código con mayor frecuencia.

Su empresa de TI ha adoptado recientemente DevOps y desea automatizar el proceso de entrega de software. Desea modelar el proceso de lanzamiento completo para crear código, implementarlo en entornos de preproducción, probar la aplicación y lanzarlo a producción. ¿Cuál de los siguientes servicios de AWS mejorará la capacidad para integrarse aún más con otros productos de AWS y de terceros?

- GitLab
- AWS CodeDeploy
- AWS CodePipeline
- Jenkins

Resumen del curso

Lección 26 de 27

Felicitaciones, llegó al final del curso.

En este curso, aprendió acerca de la cultura, las prácticas y las herramientas de DevOps que ayudan a los equipos a entregar aplicaciones y servicios de forma fiable a alta velocidad. También ha aprendido acerca de los servicios clave de AWS diseñados para ayudarlo a practicar DevOps.

Ahora puede hacer lo siguiente:

- Describir las metodologías de DevOps en torno a cultura, prácticas y herramientas.
- Explicar por qué adoptar una mentalidad que apoye una cultura de DevOps es esencial para implementar DevOps.
- Describir la transformación de Amazon a DevOps.
- Categorizar y describir los servicios clave de AWS DevOps que respaldan el ciclo de vida de las aplicaciones.
- Identificar los servicios de AWS que se utilizan para automatizar el proceso de integración y entrega continuas (CI/CD).
- Describir cómo crear y controlar una canalización de CI/CD.

Comentarios

Lección 27 de 27

Sus comentarios y satisfacción son importantes para nosotros. La información que recopilamos nos ayuda a mejorar las iteraciones futuras de nuestros cursos.

Por favor, tómese un momento para completar esta breve encuesta.
Para ayudarnos a mejorar la calidad de nuestros cursos y satisfacer sus expectativas, seleccione SURVEY (Encuesta).

ENCUESTA

¿Tiene correcciones y preguntas? Seleccione CONTACT US (Contacte con nosotros).

CONTÁCTENOS