Construir arquitecturas hexagonales en AWS

AWS Orientación prescriptiva

AWS Orientación prescriptiva: Construir arquitecturas hexagonales en AWS

Table of Contents

Construir arquitecturas hexagonales en AWS

Furkan Oruc, Dominik Goby, Darius Kunce y Michal Ploski, Amazon Web Services ()AWS

junio de 2022 (historial de documentos)

Esta guía describe un modelo mental y una colección de patrones para desarrollar arquitecturas de software. Estas arquitecturas son fáciles de mantener, ampliar y escalar en toda la organización a medida que crece la adopción de productos. Los hiperescaladores en la nube, como Amazon Web Services (AWS), proporcionan componentes básicos para que las pequeñas y grandes empresas innoven y creen nuevos productos de software. El rápido ritmo de presentación de estos nuevos servicios y funciones hace que las partes interesadas de la empresa esperen que sus equipos de desarrollo puedan crear prototipos de nuevos productos mínimamente viables (MVPs) con mayor rapidez, de modo que las nuevas ideas puedan probarse y verificarse lo antes posible. A menudo, MVPs se adoptan y pasan a formar parte del ecosistema de software empresarial. En el proceso de crearlos MVPs, los equipos a veces abandonan las reglas y las mejores prácticas de desarrollo de software, como los principios de SOLID y las pruebas unitarias. Asumen que este enfoque acelerará el desarrollo y reducirá el tiempo de comercialización. Sin embargo, si no logran crear un modelo fundamental y un marco para la arquitectura del software en todos los niveles, será difícil o incluso imposible desarrollar nuevas funciones para el producto. La falta de certeza y los requisitos cambiantes también pueden ralentizar al equipo durante el proceso de desarrollo.

Esta guía describe una arquitectura de software propuesta, desde una arquitectura hexagonal de bajo nivel hasta una descomposición arquitectónica y organizacional de alto nivel, que utiliza el diseño impulsado por el dominio (DDD) para abordar estos desafíos. El DDD ayuda a gestionar la complejidad empresarial y a ampliar el equipo de ingeniería a medida que se desarrollan nuevas funciones. Permite alinear a las partes interesadas empresariales y técnicas con los problemas empresariales, denominados dominios, mediante el uso de un lenguaje ubicuo. La arquitectura hexagonal posibilita técnicamente este enfoque en un dominio muy específico, denominado contexto limitado. Un contexto acotado es una subárea del problema empresarial muy cohesiva y débilmente acoplada. Le recomendamos que adopte una arquitectura hexagonal para todos sus proyectos de software empresarial, independientemente de su complejidad.

La arquitectura hexagonal alienta al equipo de ingeniería a resolver primero el problema empresarial, mientras que la arquitectura clásica en capas deja de centrarse en el dominio de la ingeniería para centrarse primero en resolver los problemas técnicos. Además, si el software sigue una arquitectura hexagonal, es más fácil adoptar un enfoque de desarrollo basado en pruebas, lo que reduce el ciclo

de retroalimentación que los desarrolladores necesitan para probar los requisitos empresariales. Por último, el uso de comandos y controladores de comandos es una forma de aplicar los principios de responsabilidad única y de apertura y cierre de SOLID. La adhesión a estos principios crea una base de código que los desarrolladores y arquitectos que trabajan en el proyecto pueden entender y entender fácilmente, y reduce el riesgo de introducir cambios importantes en la funcionalidad existente.

Esta guía está dirigida a arquitectos y desarrolladores de software que estén interesados en comprender las ventajas de adoptar la arquitectura hexagonal y la DDD para sus proyectos de desarrollo de software. Incluye un ejemplo del diseño de una infraestructura para su aplicación AWS que sea compatible con una arquitectura hexagonal. Para ver un ejemplo de implementación, consulte Estructurar un proyecto de Python en una arquitectura hexagonal utilizando el AWS Lambda sitio web AWS Prescriptive Guidance.

Descripción general

Diseño impulsado por dominios (DDD)

En el diseño impulsado por dominios (DDD), un dominio es el núcleo del sistema de software. El modelo de dominio se define primero, antes de desarrollar cualquier otro módulo, y no depende de otros módulos de bajo nivel. En cambio, los módulos como las bases de datos, la capa de presentación y APIs los externos dependen del dominio.

En DDD, los arquitectos descomponen la solución en contextos acotados utilizando la descomposición basada en la lógica empresarial en lugar de la descomposición técnica. Los beneficios de este enfoque se analizan en la sección. Resultados empresariales específicos

La DDD es más fácil de implementar cuando los equipos utilizan una arquitectura hexagonal. En la arquitectura hexagonal, el núcleo de la aplicación es el centro de la aplicación. Está desacoplado de otros módulos a través de puertos y adaptadores, y no depende de otros módulos. Esto se alinea perfectamente con la DDD, en la que un dominio es el núcleo de la aplicación que resuelve un problema empresarial. Esta guía propone un enfoque en el que se modela el núcleo de la arquitectura hexagonal como el modelo de dominio de un contexto limitado. La siguiente sección describe la arquitectura hexagonal con más detalle.

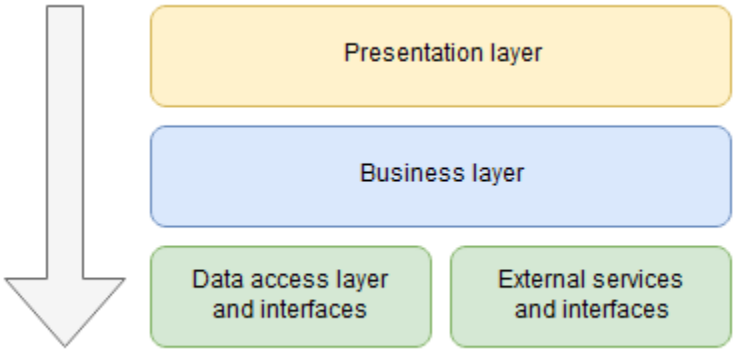
Esta guía no cubre todos los aspectos de la DDD, que es un tema muy amplio. Para comprenderlo mejor, puede revisar los recursos sobre la DDD que figuran en el sitio web de Domain Language.

Arquitectura hexagonal

La arquitectura hexagonal, también conocida como puertos y adaptadores o arquitectura tipo cebolla, es un principio para gestionar la inversión de dependencias en los proyectos de software. La arquitectura hexagonal promueve un fuerte enfoque en la lógica empresarial del dominio central al desarrollar software, y considera los puntos de integración externos como secundarios. La arquitectura hexagonal ayuda a los ingenieros de software a adoptar buenas prácticas, como el desarrollo basado en pruebas (TDD), que, a su vez, promueve la evolución de la arquitectura y ayuda a gestionar dominios complejos a largo plazo.

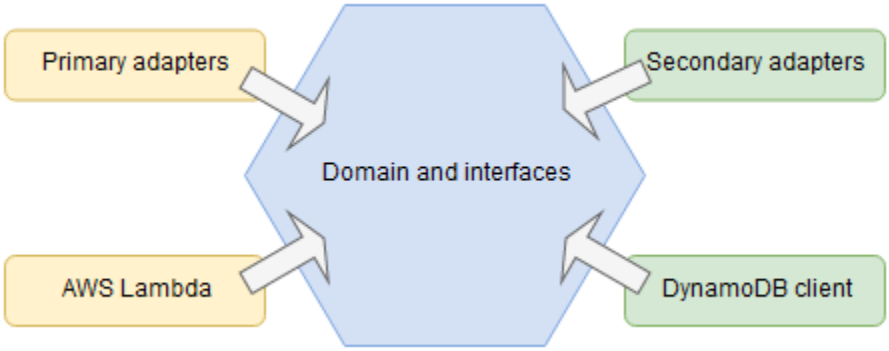
Comparemos la arquitectura hexagonal con la arquitectura en capas clásica, que es la opción más popular para modelar proyectos de software estructurado. Existen diferencias sutiles pero poderosas entre los dos enfoques.

En la arquitectura por capas, los proyectos de software se estructuran en niveles, que representan problemas generales, como la lógica empresarial o la lógica de presentación. Esta arquitectura utiliza una jerarquía de dependencias, en la que las capas superiores dependen de las capas inferiores, pero no al revés. En el siguiente diagrama, la capa de presentación es responsable de las interacciones de los usuarios, por lo que incluye la interfaz de usuario APIs, las interfaces de línea de comandos y componentes similares. La capa de presentación depende de la capa empresarial, que implementa la lógica de dominio. La capa empresarial, a su vez, depende de la capa de acceso a los datos y de varios servicios externos.



La principal desventaja de esta configuración es la estructura de dependencias. Por ejemplo, si el modelo de almacenamiento de datos en la base de datos cambia, esto afecta a la interfaz de acceso a los datos. Cualquier cambio en el modelo de datos también afecta a la capa empresarial, que depende de la interfaz de acceso a datos. Como resultado, los ingenieros de software no pueden realizar ningún cambio en la infraestructura sin afectar a la lógica del dominio. Esto, a su vez, aumenta la probabilidad de que se produzcan errores de regresión.

La arquitectura hexagonal define las relaciones de dependencia de una manera diferente, como se ilustra en el siguiente diagrama. Concentra la toma de decisiones en torno a la lógica empresarial del dominio, que define todas las interfaces. Los componentes externos interactúan con la lógica empresarial a través de interfaces denominadas puertos. Los puertos son abstracciones que definen las interacciones del dominio con el mundo externo. Cada componente de la infraestructura debe implementar esos puertos, de modo que los cambios en esos componentes ya no afecten a la lógica principal del dominio.



Los componentes circundantes se denominan adaptadores. Un adaptador es un proxy entre el mundo externo y el interno e implementa un puerto definido en el dominio. Los adaptadores se pueden clasificar en dos grupos: primarios y secundarios. Los adaptadores principales son los puntos de entrada al componente de software. Permiten a los actores, usuarios y servicios externos interactuar con la lógica central. AWS Lambda es un buen ejemplo de adaptador principal. Se integra con varios AWS servicios que pueden invocar las funciones de Lambda como puntos de entrada. Los adaptadores secundarios son contenedores de bibliotecas de servicios externos que gestionan las comunicaciones con el mundo externo. Un buen ejemplo de adaptador secundario es un cliente Amazon DynamoDB para el acceso a los datos.

Resultados empresariales específicos

La arquitectura hexagonal que se analiza en esta guía le ayuda a lograr los siguientes objetivos:

- Reduzca el tiempo de comercialización mejorando el ciclo de desarrollo
- Mejore la calidad del software
- Adáptese más fácilmente a los cambios

Estos procesos se analizan en detalle en las siguientes secciones.

Mejora del ciclo de desarrollo

El desarrollo de software para la nube presenta nuevos desafíos para los ingenieros de software, ya que es muy difícil replicar el entorno de ejecución de forma local en la máquina de desarrollo. Una forma sencilla de validar el software es implementarlo en la nube y probarlo allí. Sin embargo, este enfoque implica un ciclo de retroalimentación prolongado, especialmente cuando la arquitectura del software contiene varias implementaciones sin servidor. Al mejorar este ciclo de comentarios, se acorta el tiempo de desarrollo de las funciones, lo que reduce considerablemente el tiempo de comercialización.

Pruebas en la nube

Las pruebas directamente en la nube son la única forma de garantizar que los componentes de la arquitectura, como las puertas de enlace de Amazon API Gateway, AWS Lambda las funciones, las tablas de Amazon DynamoDB y los permisos (IAM) AWS Identity and Access Management , estén configurados correctamente. También podría ser la única forma fiable de probar las integraciones de componentes. Si bien algunos AWS servicios (como DynamoDB) se pueden implementar localmente, la mayoría de ellos no se pueden replicar en una configuración local. Al mismo tiempo, es posible que las herramientas de terceros, como Moto, LocalStackque simulan AWS servicios con fines de prueba no reflejen con precisión los contratos de API de servicios reales o que el número de funciones sea limitado.

Sin embargo, la parte más compleja del software empresarial reside en la lógica empresarial, no en la arquitectura de la nube. La arquitectura cambia con menos frecuencia que el dominio, por lo que debe adaptarse a los nuevos requisitos empresariales. Por lo tanto, probar la lógica empresarial en la nube se convierte en un proceso intenso que consiste en realizar un cambio en el código, iniciar una implementación, esperar a que el entorno esté preparado y validar el cambio. Si una implementación tarda tan solo 5 minutos, realizar y probar 10 cambios en la lógica empresarial tardará una hora o más. Si la lógica empresarial es más compleja, las pruebas pueden requerir días y esperar a que se completen las implementaciones. Si tiene varias funciones e ingenieros en el equipo, la empresa se dará cuenta rápidamente de la prolongación del período.

Pruebas a nivel local

Una arquitectura hexagonal ayuda a los desarrolladores a centrarse en el dominio y no en los aspectos técnicos de la infraestructura. Este enfoque utiliza pruebas locales (las herramientas de

pruebas unitarias del marco de desarrollo que elija) para cubrir los requisitos de lógica del dominio. No tendrá que perder tiempo resolviendo problemas de integración técnica ni desplegando su software en la nube para probar la lógica empresarial. Puede ejecutar pruebas unitarias de forma local y reducir el ciclo de retroalimentación de minutos a segundos. Si una implementación tarda 5 minutos, pero las pruebas unitarias se completan en 5 segundos, se reduce considerablemente el tiempo que se tarda en detectar los errores. La Probar primero la lógica empresarial sección que aparece más adelante en esta guía trata este enfoque con más detalle.

Paralelización del desarrollo

El enfoque de arquitectura hexagonal permite a los equipos de desarrollo paralelizar los esfuerzos de desarrollo. Los desarrolladores pueden diseñar e implementar diferentes componentes del servicio de forma individual. Esta paralelización es posible mediante el aislamiento de cada componente y las interfaces definidas entre cada componente.

Plazo de comercialización del producto

Las pruebas unitarias locales mejoran el ciclo de comentarios sobre el desarrollo y reducen el tiempo de comercialización de nuevos productos o funciones, especialmente cuando contienen una lógica empresarial compleja, como se explicó anteriormente. Además, el aumento de la cobertura del código mediante pruebas unitarias reduce considerablemente el riesgo de que se introduzcan errores de regresión al actualizar o refactorizar el código base. La cobertura de las pruebas unitarias también permite refactorizar continuamente la base de código para mantenerla bien organizada, lo que acelera el proceso de incorporación para los nuevos ingenieros. Esto se analiza con más detalle en la Calidad por diseño sección. Por último, si la lógica empresarial está bien aislada y probada, permite a los desarrolladores adaptarse rápidamente a los cambiantes requisitos funcionales y no funcionales. Esto se explica con más detalle en la Adaptarse al cambio sección.

Calidad por diseño

La adopción de una arquitectura hexagonal ayuda a promover la calidad de la base de código desde el principio del proyecto. Es importante crear un proceso que le ayude a cumplir los requisitos de calidad esperados desde el principio, sin ralentizar el proceso de desarrollo.

Cambios localizados y mayor legibilidad

El uso del enfoque de arquitectura hexagonal permite a los desarrolladores cambiar el código de una clase o componente sin afectar a otras clases o componentes. Este diseño promueve la cohesión de los componentes desarrollados. Al desacoplar el dominio de los adaptadores y utilizar interfaces conocidas, puede aumentar la legibilidad del código. Resulta más fácil identificar los problemas y los casos extremos.

Este enfoque también facilita la revisión del código durante el desarrollo y limita la introducción de cambios no detectados o de problemas técnicos.

Probar primero la lógica empresarial

Las pruebas locales se pueden realizar mediante la introducción end-to-end, la integración y las pruebas unitarias en el proyecto. End-to-endlas pruebas cubren todo el ciclo de vida de las solicitudes entrantes. Por lo general, invocan un punto de entrada de la aplicación y comprueban si ha cumplido con los requisitos empresariales. Cada proyecto de software debe tener al menos un escenario de prueba que utilice entradas conocidas y produzca los resultados esperados. Sin embargo, añadir más escenarios extremos puede resultar complejo, ya que cada prueba debe configurarse para enviar una solicitud a través de un punto de entrada (por ejemplo, a través de una API REST o de colas), analizar todos los puntos de integración que requiere la acción empresarial y, a continuación, confirmar el resultado. Configurar el entorno para el escenario de prueba y confirmar los resultados puede llevar mucho tiempo a los desarrolladores.

En la arquitectura hexagonal, se prueba la lógica empresarial de forma aislada y se utilizan pruebas de integración para probar los adaptadores secundarios. Puede utilizar adaptadores simulados o falsos en sus pruebas de lógica empresarial. También puedes combinar las pruebas para casos de uso empresarial con pruebas unitarias para tu modelo de dominio a fin de mantener una alta cobertura con un bajo nivel de acoplamiento. Como buena práctica, las pruebas de integración no deberían validar la lógica empresarial. En su lugar, deben comprobar que el adaptador secundario llama correctamente a los servicios externos.

Lo ideal sería utilizar el desarrollo basado en pruebas (TDD) y empezar a definir entidades de dominio o casos de uso empresarial con las pruebas adecuadas desde el principio del desarrollo. Escribir primero las pruebas te ayuda a crear implementaciones simuladas de las interfaces que requiere el dominio. Cuando las pruebas se realicen correctamente y se cumplan las reglas de la lógica del dominio, podrá implementar los adaptadores propiamente dichos y el software en el entorno de prueba. En este punto, la implementación de la lógica de dominio podría no ser la ideal. A continuación, puede trabajar en la refactorización de la arquitectura existente para hacerla evolucionar introduciendo patrones de diseño o reorganizando el código en general. Al usar este enfoque, puede evitar la introducción de errores de regresión y puede mejorar la arquitectura a medida que el proyecto crece. Al combinar este enfoque con las pruebas automáticas que se ejecutan en el proceso de integración continua, se puede reducir el número de posibles errores antes de que pasen a la fase de producción.

Si utiliza implementaciones sin servidor, puede aprovisionar rápidamente una instancia de la aplicación en su AWS cuenta para la integración y end-to-end las pruebas manuales. Tras estos pasos de implementación, te recomendamos que automatices las pruebas con cada nuevo cambio que se introduzca en el repositorio.

Mantenibilidad

La mantenibilidad se refiere al funcionamiento y la supervisión de una aplicación para garantizar que cumpla con todos los requisitos y minimizar la probabilidad de que se produzca una falla en el sistema. Para que el sistema funcione, debe adaptarlo a los requisitos operativos o de tráfico futuros. También debe asegurarse de que esté disponible y sea fácil de implementar, con un impacto mínimo o nulo en los clientes.

Para comprender el estado actual e histórico de su sistema, debe hacerlo observable. Para ello, puede proporcionar métricas, registros y rastreos específicos que los operadores puedan utilizar para garantizar que el sistema funcione según lo esperado y para realizar un seguimiento de los errores. Estos mecanismos también deberían permitir a los operadores realizar un análisis de la causa raíz sin tener que iniciar sesión en la máquina y leer el código.

Una arquitectura hexagonal tiene como objetivo aumentar la capacidad de mantenimiento de sus aplicaciones web para que su código requiera menos trabajo en general. Al separar los módulos, localizar los cambios y desvincular la lógica empresarial de las aplicaciones de la implementación de los adaptadores, puede generar métricas y registros que ayudan a los operadores a comprender en profundidad el sistema y el alcance de los cambios específicos que se realizan en los adaptadores principales o secundarios.

Adaptarse al cambio

Los sistemas de software tienden a complicarse. Una de las razones de esto podría ser los cambios frecuentes en los requisitos empresariales y el poco tiempo para adaptar la arquitectura del software en consecuencia. Otra razón podría ser la inversión insuficiente para configurar la arquitectura del software al principio del proyecto a fin de adaptarla a los cambios frecuentes. Sea cual sea el motivo, un sistema de software podría complicarse hasta el punto de que sea casi imposible realizar un cambio. Por lo tanto, es importante construir una arquitectura de software fácil de mantener desde el principio del proyecto. Una buena arquitectura de software puede adaptarse fácilmente a los cambios.

En esta sección se explica cómo diseñar aplicaciones fáciles de mantener mediante el uso de una arquitectura hexagonal que se adapte fácilmente a los requisitos empresariales o no funcionales.

Adaptarse a los nuevos requisitos no funcionales mediante el uso de puertos y adaptadores

Como núcleo de la aplicación, el modelo de dominio define las acciones que se requieren del mundo exterior para cumplir con los requisitos empresariales. Estas acciones se definen mediante abstracciones, que se denominan puertos. Estos puertos se implementan mediante adaptadores independientes. Cada adaptador es responsable de la interacción con otro sistema. Por ejemplo, puede tener un adaptador para el repositorio de la base de datos y otro adaptador para interactuar con una API de terceros. El dominio no conoce la implementación del adaptador, por lo que es fácil reemplazar un adaptador por otro. Por ejemplo, la aplicación puede cambiar de una base de datos SQL a una base de datos NoSQL. En este caso, se debe desarrollar un nuevo adaptador para implementar los puertos definidos por el modelo de dominio. El dominio no depende del repositorio de la base de datos y utiliza abstracciones para interactuar, por lo que no sería necesario cambiar nada en el modelo de dominio. Por lo tanto, la arquitectura hexagonal se adapta con facilidad a los requisitos no funcionales.

Se adapta a los nuevos requisitos empresariales mediante el uso de comandos y controladores de comandos

En la arquitectura clásica en capas, el dominio depende de la capa de persistencia. Si desea cambiar el dominio, también tendrá que cambiar la capa de persistencia. En comparación, en la arquitectura

hexagonal, el dominio no depende de otros módulos del software. El dominio es el núcleo de la aplicación y todos los demás módulos (puertos y adaptadores) dependen del modelo de dominio. El dominio utiliza el principio de inversión de dependencias para comunicarse con el mundo exterior a través de los puertos. La ventaja de la inversión de dependencias es que puede cambiar el modelo de dominio libremente sin miedo a descifrar otras partes del código. Como el modelo de dominio refleja el problema empresarial que se intenta resolver, actualizar el modelo de dominio para adaptarlo a los cambiantes requisitos empresariales no es un problema.

Al desarrollar software, la separación de las preocupaciones es un principio importante a seguir. Para lograr esta separación, puede utilizar un patrón de comandos ligeramente modificado. Se trata de un patrón de diseño conductual en el que toda la información necesaria para completar una operación se encapsula en un objeto de comando. A continuación, los controladores de comandos procesan estas operaciones. Los controladores de comandos son métodos que reciben un comando, modifican el estado del dominio y, a continuación, devuelven una respuesta a la persona que llama. Puede usar diferentes clientes, como colas sincrónicas APIs o asíncronas, para ejecutar comandos. Se recomienda utilizar comandos y controladores de comandos para cada operación del dominio. Si sigue este enfoque, puede añadir nuevas funciones mediante la introducción de nuevos comandos y controladores de comandos, sin cambiar su lógica empresarial actual. Por lo tanto, el uso de un patrón de comandos facilita la adaptación a los nuevos requisitos empresariales.

Desacoplar los componentes mediante la fachada de servicio o el patrón CQRS

En la arquitectura hexagonal, los adaptadores principales son responsables de acoplar de forma flexible las solicitudes de lectura y escritura entrantes de los clientes al dominio. Hay dos formas de lograr este acoplamiento flexible: mediante un patrón de fachada de servicio o mediante el patrón de segregación de responsabilidades por consultas de comandos (CQRS).

El patrón de fachada de servicios proporciona una interfaz frontal para atender a los clientes, como la capa de presentación o un microservicio. Una fachada de servicios proporciona a los clientes varias operaciones de lectura y escritura. Se encarga de transferir las solicitudes entrantes al dominio y de mapear la respuesta recibida del dominio a los clientes. Utilizar una fachada de servicios es fácil para los microservicios que tienen una única responsabilidad y varias operaciones. Sin embargo, cuando se utiliza la fachada de servicio, es más difícil seguir los principios de responsabilidad única y de apertura y cierre. El principio de responsabilidad única establece que cada módulo debe ser responsable de una sola funcionalidad del software. El principio de apertura y cierre establece que el código debe estar abierto para su extensión y cerrado para su modificación. A medida que se

extiende la fachada de servicios, todas las operaciones se agrupan en una interfaz, se encapsulan más dependencias y más desarrolladores comienzan a modificar la misma fachada. Por lo tanto, recomendamos utilizar una fachada de servicio solo si está claro que el servicio no se extenderá mucho durante el desarrollo.

Otra forma de implementar los adaptadores principales en una arquitectura hexagonal es usar el patrón CQRS, que separa las operaciones de lectura y escritura mediante consultas y comandos. Como se explicó anteriormente, los comandos son objetos que contienen toda la información necesaria para cambiar el estado del dominio. Los comandos se ejecutan mediante métodos de manejo de comandos. Las consultas, por otro lado, no alteran el estado del sistema. Su único propósito es devolver los datos a los clientes. En el patrón CQRS, los comandos y las consultas se implementan en módulos separados. Esto es especialmente ventajoso para los proyectos que siguen una arquitectura basada en eventos, ya que un comando se puede implementar como un evento que se procesa de forma asíncrona, mientras que una consulta se puede ejecutar de forma sincrónica mediante una API. Una consulta también puede usar una base de datos diferente que esté optimizada para ella. La desventaja del patrón CQRS es que su implementación lleva más tiempo que una fachada de servicio. Recomendamos usar el patrón CQRS para los proyectos que planee escalar y mantener a largo plazo. Los comandos y las consultas proporcionan un mecanismo eficaz para aplicar el principio de responsabilidad única y desarrollar software poco acoplado, especialmente en proyectos a gran escala.

El CQRS ofrece grandes beneficios a largo plazo, pero requiere una inversión inicial. Por este motivo, le recomendamos que evalúe su proyecto detenidamente antes de decidir utilizar el patrón CQRS. Sin embargo, puede estructurar la aplicación mediante comandos y controladores de comandos desde el principio sin separar read/write las operaciones. Esto le ayudará a refactorizar fácilmente su proyecto para el CQRS si decide adoptar ese enfoque más adelante.

Escalamiento organizacional

Una combinación de arquitectura hexagonal, diseño basado en el dominio y (opcionalmente) el CQRS permite a su organización escalar rápidamente su producto. Según la Ley de Conway, las arquitecturas de software tienden a evolucionar para reflejar las estructuras de comunicación de una empresa. Históricamente, esta observación ha tenido connotaciones negativas, ya que las grandes organizaciones suelen estructurar sus equipos en función de conocimientos técnicos, como bases de datos, autobuses de servicios empresariales, etc. El problema de este enfoque es que el desarrollo de productos y funciones siempre implica preocupaciones transversales, como la seguridad y la escalabilidad, que requieren una comunicación constante entre los equipos. La estructuración

de los equipos en función de las características técnicas crea compartimentos innecesarios en la organización, lo que se traduce en una mala comunicación, una falta de implicación y una pérdida de visión del panorama general. Con el tiempo, estos problemas organizativos se reflejan en la arquitectura del software.

La maniobra inversa de Conway, por otro lado, define la estructura organizacional en función de los dominios que promueven la arquitectura del software. Por ejemplo, a los equipos multidisciplinarios se les asigna la responsabilidad de un conjunto específico de contextos acotados, que se identifican mediante la DDD y la tormenta de eventos. Estos contextos acotados pueden reflejar características muy específicas del producto. Por ejemplo, el equipo de cuentas podría ser responsable del contexto de pago. Cada nueva función se asigna a un nuevo equipo que tiene responsabilidades muy cohesionadas y poco coordinadas, por lo que pueden centrarse únicamente en la entrega de esa función y reducir el tiempo de comercialización. Los equipos se pueden escalar en función de la complejidad de las funciones, por lo que las funciones complejas se pueden asignar a más ingenieros.

Prácticas recomendadas

Modele el dominio empresarial

Trabaje desde el dominio empresarial hasta el diseño del software para asegurarse de que el software que está escribiendo se ajuste a las necesidades empresariales.

Utilice metodologías de diseño basado en el dominio (DDD), como la tormenta de eventos, para modelar el ámbito empresarial. Event Storming tiene un formato de taller flexible. Durante el taller, los expertos en dominios y software exploran la complejidad del ámbito empresarial de forma colaborativa. Los expertos en software utilizan los resultados del taller para iniciar el proceso de diseño y desarrollo de los componentes de software.

Escribe y ejecuta pruebas desde el principio

Utilice el desarrollo basado en pruebas (TDD) para verificar la exactitud del software que está desarrollando. El TDD funciona mejor a nivel de pruebas unitarias. El desarrollador diseña un componente de software escribiendo primero una prueba, que invoca ese componente. Ese componente no está implementado al principio, por lo que la prueba falla. Como siguiente paso, el desarrollador implementa la funcionalidad del componente, utilizando dispositivos de prueba con objetos simulados para simular el comportamiento de las dependencias externas o puertos. Cuando la prueba sea exitosa, el desarrollador puede continuar implementando adaptadores reales. Este enfoque mejora la calidad del software y da como resultado un código más legible, ya que los desarrolladores entienden cómo utilizarían los usuarios los componentes. La arquitectura hexagonal apoya la metodología TDD al separar el núcleo de la aplicación. Los desarrolladores escriben pruebas unitarias que se centran en el comportamiento del núcleo del dominio. No tienen que escribir adaptadores complejos para ejecutar sus pruebas; en su lugar, pueden usar dispositivos y objetos simulados simples.

Utilice el desarrollo impulsado por el comportamiento (BDD) para garantizar la end-to-end aceptación a nivel de función. En BDD, los desarrolladores definen escenarios para las características y los verifican con las partes interesadas de la empresa. Para ello, las pruebas de BDD utilizan la mayor cantidad de lenguaje natural posible. La arquitectura hexagonal apoya la metodología BDD con su concepto de adaptadores primarios y secundarios. Los desarrolladores pueden crear adaptadores principales y secundarios que se ejecuten localmente sin necesidad de recurrir a servicios externos. Configuran el conjunto de pruebas de BDD para usar el adaptador principal local para ejecutar la aplicación.

Ejecute automáticamente cada prueba del proceso de integración continua para evaluar constantemente la calidad del sistema.

Defina el comportamiento del dominio

Descomponga el dominio en entidades, objetos de valor y agregados (lea acerca de la implementación del diseño basado en dominios) y defina su comportamiento. Implemente el comportamiento del dominio para que las pruebas que se escribieron al principio del proyecto tengan éxito. Defina comandos que invoquen el comportamiento de los objetos del dominio. Defina los eventos que emiten los objetos de dominio después de completar un comportamiento.

Defina las interfaces que los adaptadores pueden usar para interactuar con el dominio.

Automatice las pruebas y la implementación

Tras una prueba de concepto inicial, le recomendamos que invierta tiempo en implementar DevOps las prácticas. Por ejemplo, las canalizaciones de integración y entrega continuas (CI/CD) y los entornos de prueba dinámicos ayudan a mantener la calidad del código y a evitar errores durante la implementación.

- Ejecute las pruebas unitarias dentro del proceso de CI y pruebe el código antes de fusionarlo.
- Cree un proceso de CD para implementar su aplicación en un dev/test entorno estático o en entornos creados dinámicamente que admitan la integración y end-to-end las pruebas automáticas.
- Automatice el proceso de implementación para entornos dedicados.

Amplíe su producto mediante microservicios y CQRS

Si su producto tiene éxito, amplíe su producto descomponiendo su proyecto de software en microservicios. Utilice la portabilidad que ofrece la arquitectura hexagonal para mejorar el rendimiento. Divida los servicios de consultas y los controladores de comandos en síncronos y asíncronos independientes. APIs Considere la posibilidad de adoptar el patrón de segregación de responsabilidades por consultas de comandos (CQRS) y una arquitectura basada en eventos.

Si recibe muchas solicitudes de funciones nuevas, considere la posibilidad de ampliar su organización en función de los patrones DDD. Estructura tus equipos de forma que posean una

o más funciones como contextos acotados, como se ha explicado anteriormente en esta sección. Escalamiento organizacional Luego, estos equipos pueden implementar la lógica empresarial mediante una arquitectura hexagonal.

Diseñe una estructura de proyecto que se adapte a los conceptos de arquitectura hexagonal

La infraestructura como código (IaC) es una práctica ampliamente adoptada en el desarrollo de la nube. Le permite definir y mantener sus recursos de infraestructura (como redes, balanceadores de carga, máquinas virtuales y puertas de enlace) como código fuente. De esta forma, puede realizar un seguimiento de todos los cambios en su arquitectura mediante un sistema de control de versiones. Además, puede crear y mover la infraestructura fácilmente con fines de prueba. Le recomendamos que mantenga el código de la aplicación y el código de la infraestructura en el mismo repositorio cuando desarrolle sus aplicaciones en la nube. Este enfoque facilita el mantenimiento de la infraestructura de su aplicación.

Se recomienda dividir la aplicación en tres carpetas o proyectos que se adapten a los conceptos de la arquitectura hexagonal: `entrypoints` (adaptadores principales), `domain` (dominio e interfaces) y `adapters` (adaptadores secundarios).

La siguiente estructura de proyecto proporciona un ejemplo de este enfoque al diseñar una API en AWS. El proyecto mantiene el código de aplicación (`app`) y el código de infraestructura (`infra`) en el mismo repositorio, tal y como se recomendó anteriormente.

```
app/  # application code
|--- adapters/  # implementation of the ports defined in the domain
     |--- tests/  # adapter unit tests
|--- entrypoints/  # primary adapters, entry points
     |--- api/  # api entry point
          |--- model/  # api model
          |--- tests/  # end to end api tests
|--- domain/  # domain to implement business logic using hexagonal architecture
     |--- command_handlers/  # handlers used to run commands on the domain
     |--- commands/  # commands on the domain
     |--- events/  # events emitted by the domain
     |--- exceptions/  # exceptions defined on the domain
     |--- model/  # domain model
     |--- ports/  # abstractions used for external communication
     |--- tests/  # domain tests
```

```
infra/  # infrastructure code
```

Como se mencionó anteriormente, el dominio es el núcleo de la aplicación y no depende de ningún otro módulo. Se recomienda estructurar la `domain` carpeta para incluir las siguientes subcarpetas:

- `command handlers`contiene los métodos o clases que ejecutan comandos en el dominio.
- `commands`contiene los objetos de comando que definen la información necesaria para realizar una operación en el dominio.
- `events`contiene los eventos que se emiten a través del dominio y luego se enrutan a otros microservicios.
- `exceptions`contiene los errores conocidos definidos en el dominio.
- `model`contiene las entidades del dominio, los objetos de valor y los servicios del dominio.
- `ports`contiene las abstracciones a través de las cuales el dominio se comunica con las bases de datos u otros componentes externos. APIs
- `tests`contiene los métodos de prueba (como las pruebas de lógica empresarial) que se ejecutan en el dominio.

Los adaptadores principales son los puntos de entrada a la aplicación, representados por la `entrypoints` carpeta. En este ejemplo, se utiliza la `api` carpeta como adaptador principal. Esta carpeta contiene una API`model`, que define la interfaz que necesita el adaptador principal para comunicarse con los clientes. La `tests` carpeta contiene end-to-end las pruebas de la API. Se trata de pruebas superficiales que validan que los componentes de la aplicación estén integrados y funcionen en armonía.

Los adaptadores secundarios, representados por la `adapters` carpeta, implementan las integraciones externas que requieren los puertos del dominio. Un repositorio de base de datos es un excelente ejemplo de adaptador secundario. Cuando el sistema de base de datos cambia, puede escribir un adaptador nuevo mediante la implementación definida por el dominio. No es necesario cambiar el dominio ni la lógica empresarial. La `tests` subcarpeta contiene pruebas de integración externas para cada adaptador.

Ejemplos de infraestructura en AWS

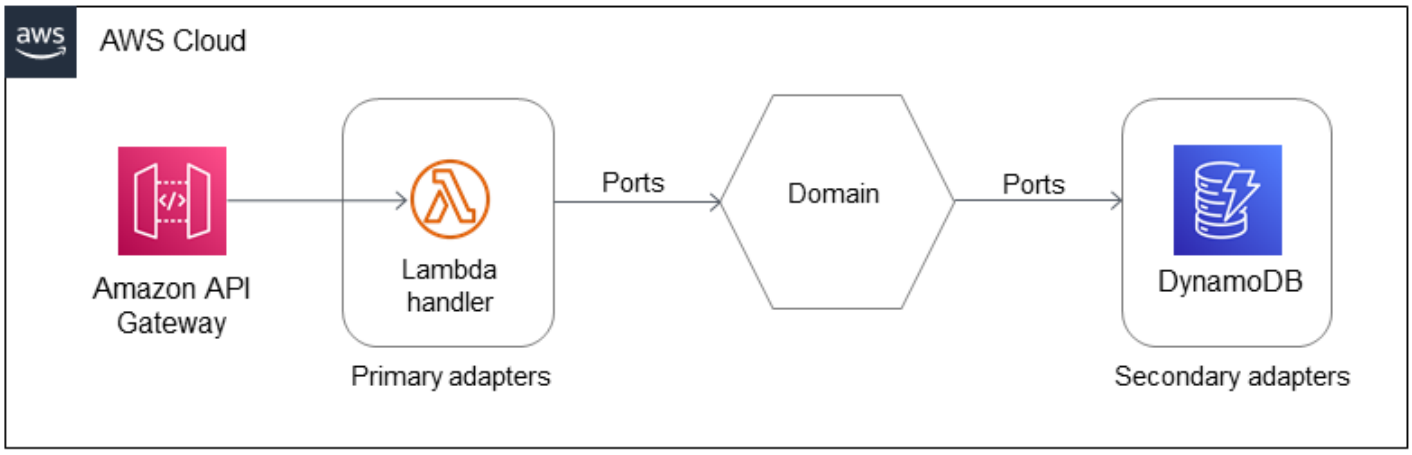
En esta sección se proporcionan ejemplos para diseñar una infraestructura para su aplicación AWS que pueda utilizar para implementar una arquitectura hexagonal. Le recomendamos que comience con una arquitectura sencilla para crear un producto mínimo viable (MVP). La mayoría de los microservicios necesitan un punto de entrada único para gestionar las solicitudes de los clientes, una capa de procesamiento para ejecutar el código y una capa de persistencia para almacenar los datos. Los siguientes AWS servicios son excelentes candidatos para su uso como clientes, adaptadores principales y adaptadores secundarios en una arquitectura hexagonal:

- Clientes: Amazon API Gateway, Amazon Simple Queue Service (Amazon SQS), Elastic Load Balancing, Amazon EventBridge
- Adaptadores principales: AWS Lambda Amazon Elastic Container Service (Amazon ECS), Amazon Elastic Kubernetes Service (Amazon EKS), Amazon Elastic Compute Cloud (Amazon EC2)
- Adaptadores secundarios: Amazon DynamoDB, Amazon Relational Database Service (Amazon RDS), Amazon Aurora, API Gateway, Amazon SQS, EventBridge Elastic Load Balancing, Amazon Simple Notification Service (Amazon SNS) Simple Notification Service (Amazon SNS)

En las siguientes secciones, se analizan estos servicios con más detalle en el contexto de la arquitectura hexagonal.

Comience de forma sencilla

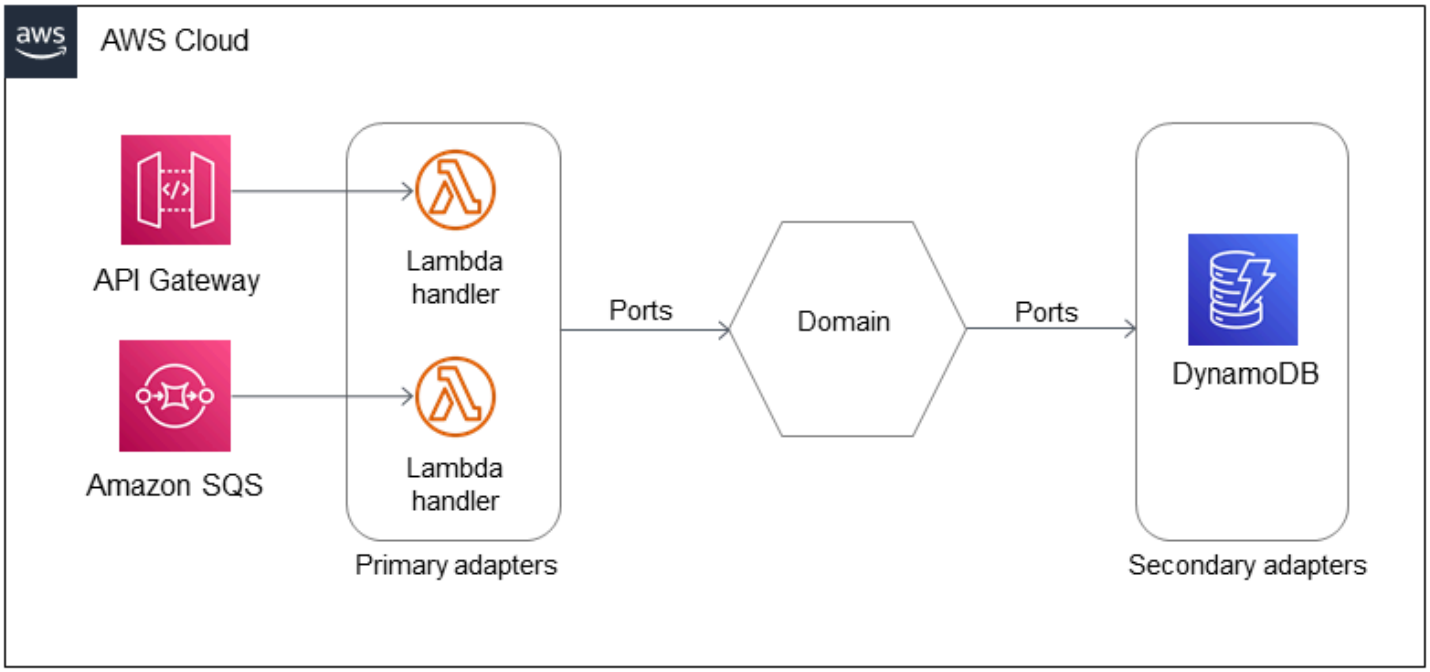
Le recomendamos que comience de forma sencilla cuando diseñe una aplicación mediante una arquitectura hexagonal. En este ejemplo, API Gateway se usa como cliente (API REST), Lambda se usa como adaptador principal (procesamiento) y DynamoDB se usa como adaptador secundario (persistencia). El cliente de puerta de enlace llama al punto de entrada, que, en este caso, es un controlador Lambda.



Esta arquitectura es totalmente libre de servidores y proporciona al arquitecto un buen punto de partida. Le recomendamos que utilice el patrón de comandos en el dominio porque facilita el mantenimiento del código y se adapta a los nuevos requisitos empresariales y no funcionales. Esta arquitectura podría ser suficiente para crear microservicios sencillos con unas pocas operaciones.

Aplique el patrón CQRS

Le recomendamos que cambie al patrón CQRS si quiere aumentar el número de operaciones en el dominio. Puede aplicar el patrón CQRS como una arquitectura totalmente sin servidor AWS mediante el siguiente ejemplo.

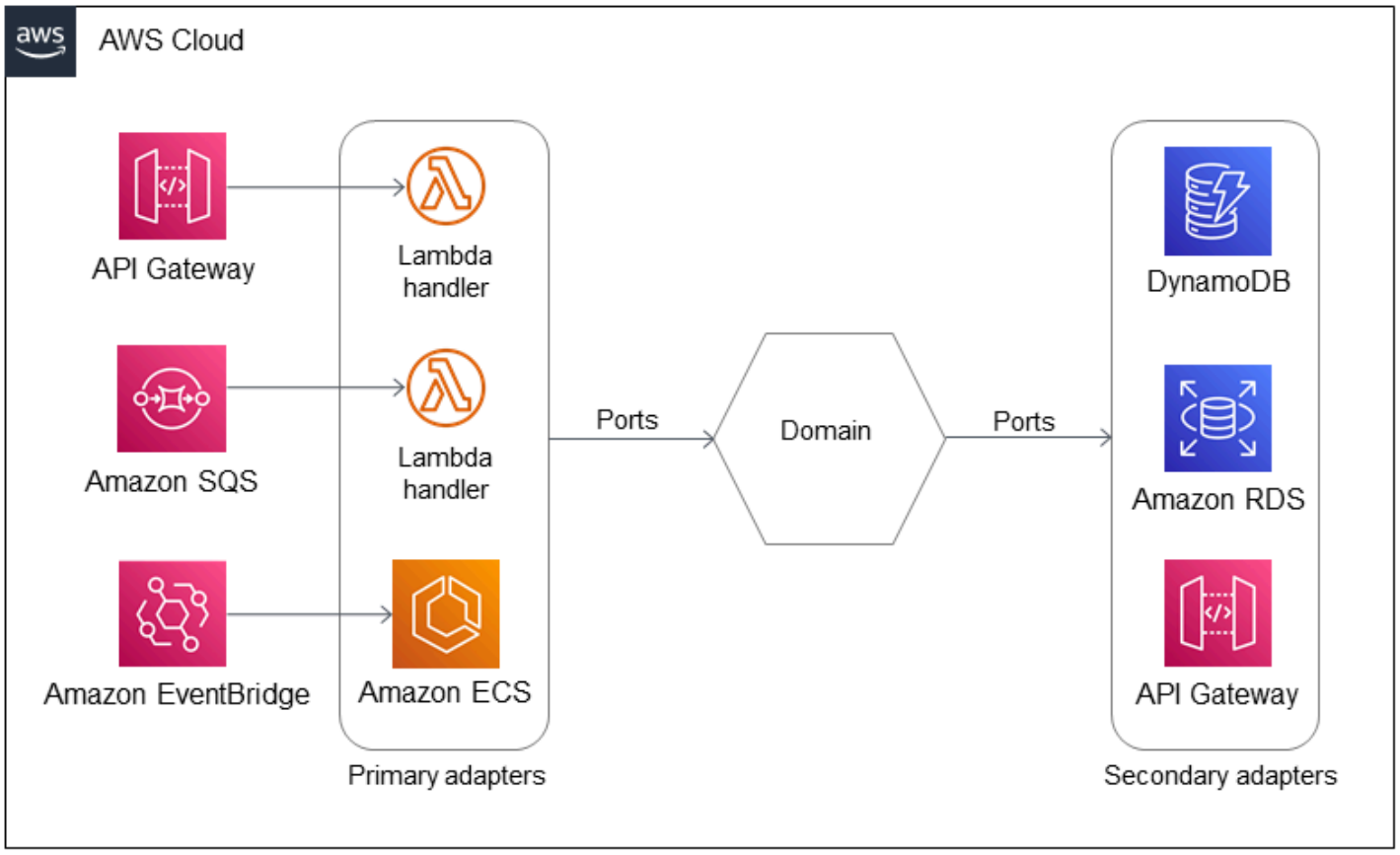


En este ejemplo, se utilizan dos controladores Lambda, uno para consultas y otro para comandos. Las consultas se ejecutan de forma sincrónica utilizando una puerta de enlace de API como cliente. Los comandos se ejecutan de forma asíncrona utilizando Amazon SQS como cliente.

Esta arquitectura incluye varios clientes (API Gateway y Amazon SQS) y varios adaptadores principales (Lambda), a los que se llama mediante sus puntos de entrada correspondientes (controladores Lambda). Todos los componentes pertenecen al mismo contexto limitado, por lo que se encuentran dentro del mismo dominio.

Desarrolle la arquitectura agregando contenedores, una base de datos relacional y una API externa

Los contenedores son una buena opción para tareas de larga duración. Es posible que también desee utilizar una base de datos relacional si tiene un esquema de datos predefinido y desea aprovechar la potencia del lenguaje SQL. Además, el dominio tendría que comunicarse con el externo APIs. Puede desarrollar el ejemplo de arquitectura para que cumpla con estos requisitos, como se muestra en el siguiente diagrama.

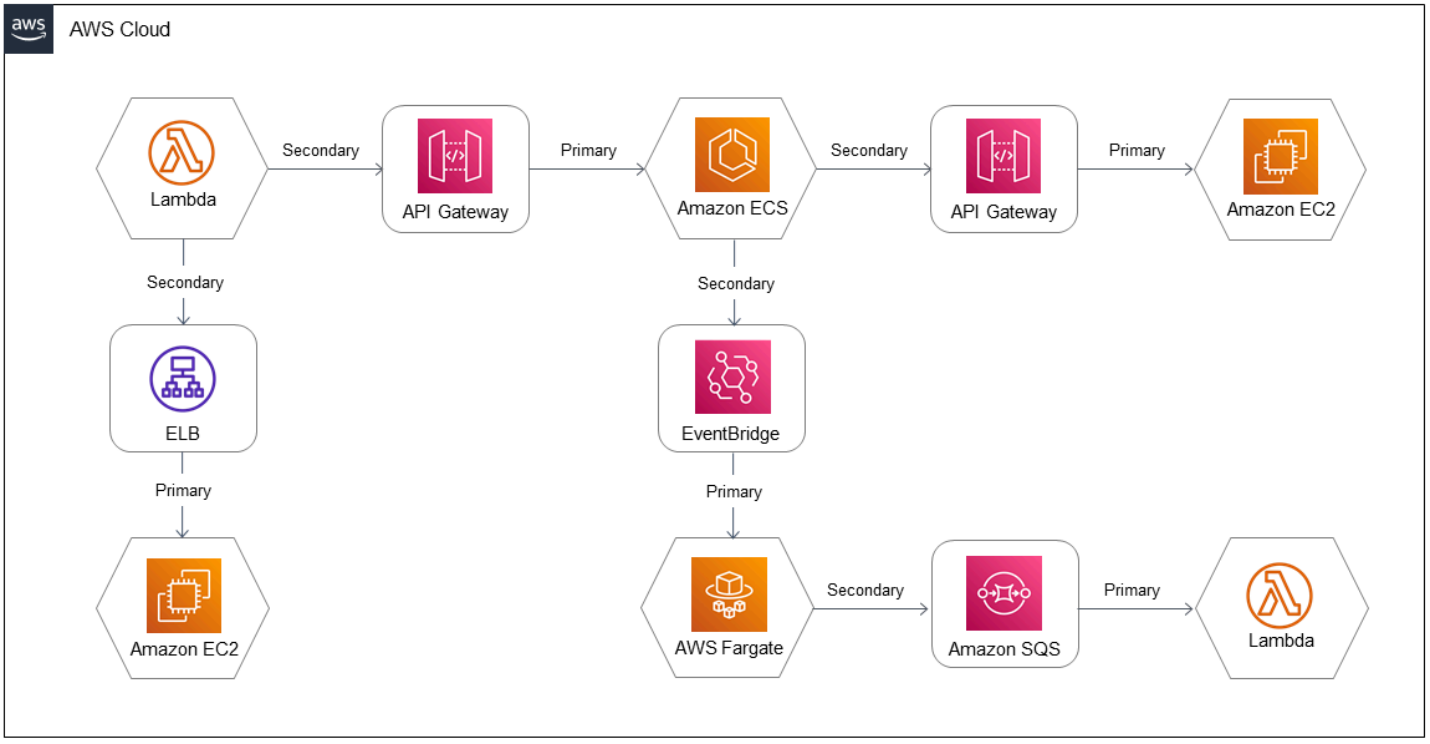


En este ejemplo, se utiliza Amazon ECS como adaptador principal para lanzar tareas de ejecución prolongada en el dominio. Amazon EventBridge (cliente) inicia una tarea de Amazon ECS (punto de entrada) cuando ocurre un evento específico. La arquitectura incluye Amazon RDS como otro adaptador secundario para almacenar datos relacionales. También agrega otra puerta de enlace de API como adaptador secundario para invocar una llamada de API externa. Como resultado, la arquitectura utiliza varios adaptadores principales y secundarios que se basan en diferentes capas informáticas subyacentes en un dominio empresarial.

El dominio siempre está acoplado de forma flexible a todos los adaptadores principales y secundarios mediante abstracciones denominadas puertos. El dominio define lo que necesita del mundo exterior mediante el uso de puertos. Como es responsabilidad del adaptador implementar el puerto, el cambio de un adaptador a otro no afecta al dominio. Por ejemplo, puede cambiar de Amazon DynamoDB a Amazon RDS escribiendo un adaptador nuevo, sin que ello afecte al dominio.

Añada más dominios (amplíe la imagen)

La arquitectura hexagonal se alinea bien con los principios de una arquitectura de microservicios. Los ejemplos de arquitectura mostrados hasta ahora contenían un único dominio (o contexto limitado). Las aplicaciones suelen incluir varios dominios, que deben comunicarse a través de adaptadores principales y secundarios. Cada dominio representa un microservicio y está estrechamente vinculado a otros dominios.



En esta arquitectura, cada dominio usa un conjunto diferente de entornos de cómputo. (Cada dominio también puede tener varios entornos de procesamiento, como en el ejemplo anterior). Cada dominio define las interfaces necesarias para comunicarse con otros dominios a través de los puertos. Los puertos se implementan mediante adaptadores principales y secundarios. De esta forma, el dominio no se ve afectado si se produce un cambio en el adaptador. Además, los dominios están desacoplados unos de otros.

En el ejemplo de arquitectura que se muestra en el diagrama anterior, Lambda, Amazon EC2 y Amazon ECS AWS Fargate se utilizan como adaptadores principales. API Gateway, Elastic Load Balancing y Amazon SQS se utilizan como adaptadores secundarios. EventBridge

Preguntas frecuentes

¿Por qué debo usar una arquitectura hexagonal?

La arquitectura hexagonal hace que los desarrolladores se centren en la lógica del dominio, simplifica la automatización de las pruebas y mejora la calidad y la adaptabilidad del código. Estas mejoras se traducen en una comercialización más rápida y en una ampliación técnica y organizativa más sencilla.

¿Por qué debo usar un diseño basado en el dominio?

El diseño impulsado por dominios (DDD) le permite crear componentes y construcciones de software utilizando un lenguaje común entre las partes interesadas de la empresa y los ingenieros. El DDD le ayuda a gestionar la complejidad del software y es una estrategia eficaz para mantener los productos de software a largo plazo.

¿Puedo practicar el desarrollo basado en pruebas sin una arquitectura hexagonal?

Sí. El desarrollo basado en pruebas (TDD) no se limita a patrones de diseño de software específicos. Sin embargo, la arquitectura hexagonal facilita la práctica del TDD.

¿Puedo escalar mi producto sin una arquitectura hexagonal ni un diseño basado en el dominio?

Sí. El escalado técnico y organizativo del producto se puede lograr con la mayoría de los patrones de diseño. Sin embargo, la arquitectura hexagonal y la DDD facilitan la escalabilidad y, a largo plazo, son más eficaces para proyectos de gran envergadura.

¿Qué tecnologías debo usar para implementar una arquitectura hexagonal?

La arquitectura hexagonal no se limita a un conjunto de tecnologías específico. Le recomendamos que elija una tecnología que admita la inversión de dependencias y las pruebas unitarias.

Estoy desarrollando un producto mínimo viable. ¿Tiene sentido dedicar tiempo a pensar en la arquitectura del software?

Sí. Le recomendamos que utilice patrones de diseño con los que esté familiarizado MVPs. Le animamos a que intente practicar la arquitectura hexagonal hasta que sus ingenieros se sientan cómodos con ella. Establecer una arquitectura hexagonal para nuevos proyectos no requiere una inversión de tiempo significativamente mayor que empezar sin ninguna arquitectura.

Estoy desarrollando un producto mínimo viable y no tengo tiempo para escribir pruebas.

Si tu MVP contiene lógica empresarial, te recomendamos encarecidamente que escribas pruebas automatizadas para ello. Esto reducirá el ciclo de retroalimentación y ahorrará tiempo.

¿Qué patrones de diseño adicionales puedo usar con la arquitectura hexagonal?

Utilice el patrón CQRS para permitir el escalado de todo el sistema. Utilice el patrón de repositorio para almacenar y restaurar su modelo de dominio. Utilice el patrón de unidad de trabajo para gestionar los pasos del proceso transaccional. Utilice la composición en lugar de la herencia para modelar agregados de dominios, entidades y objetos de valor. No cree jerarquías de objetos complejas.

Pasos a seguir a continuación

- Familiarícese aún más con los conceptos de diseño basados en el dominio leyendo los enlaces recopilados en la sección. Recursos
- Si vas a implementar un proyecto nuevo, usa la plantilla de estructura del proyecto que se proporciona en esta guía e implementa algunas funciones.
- Si estás implementando un proyecto existente, identifica el código que se pueda dividir entre operaciones de solo lectura y de solo escritura. Extrae el código de solo lectura en los servicios de consulta y coloca el código de solo escritura en los controladores de comandos.
- Cuando la estructura base del proyecto esté lista, escriba pruebas unitarias, establezca la integración continua (CI) con la automatización de las pruebas y siga las prácticas de desarrollo basado en pruebas (TDD).

Recursos

Referencias

- Estructure un proyecto de Python en una arquitectura hexagonal utilizando AWS Lambda (patrón de orientación AWS prescriptiva)
- Agile Teams (sitio web de Scaled Agile Framework)
- Patrones de arquitectura con Python, de Harry Percival y Bob Gregory (O'Reilly Media, 31 de marzo de 2020), concretamente los siguientes capítulos:
  - Comandos y controlador de comandos
  - Segregación de responsabilidades entre comandos y consultas (CQRS)
  - Patrón de repositorio
- Event Storming: el enfoque más inteligente para la colaboración más allá de los límites de los silos, por Alberto Brandolini (sitio web de Event Storming)
- Desmitificando la ley de Conway, por Sam Newman (sitio web de Thoughtworks, 30 de junio de 2014)
- Desarrollando una arquitectura evolutiva con AWS Lambda, de James Beswick (Compute Blog, 8 de julio de 2021)AWS
- Lenguaje de dominio: abordando la complejidad en el corazón del software (sitio web sobre lenguaje de dominio)
- Facade, de Dive Into Design Patterns de Alexander Shvets (libro electrónico, 5 de diciembre de 2018)
- GivenWhenThen, de Martin Fowler (21 de agosto de 2013)
- Implementación del diseño basado en dominios, por Vaughn Vernon (Addison-Wesley Professional, febrero de 2013)
- Inverse Conway Maneuver (sitio web de Thoughtworks, 8 de julio de 2014)
- Patrón del mes: Red Green Refactor (DZone sitio web, 2 de junio de 2017)
- Explicación de los principios de diseño sólidos: el principio de inversión de dependencias con ejemplos de código, por Thorben Janssen (sitio web de Stackify, 7 de mayo de 2018)
- Principios sólidos: explicación y ejemplos, por Simon Hoiberg (sitio web de ITNEXT, 1 de enero de 2019)
- El arte del desarrollo ágil: desarrollo basado en pruebas, por James Shore y Shane Warden (O'Reilly Media, 25 de marzo de 2010)

- Los sólidos principios de la programación orientada a objetos explicados en un lenguaje sencillo, por Yigit Kemal Erinc (publicaciones sobre programación orientada a objetos, 20 de agosto de 2020) freeCodeCamp
- ¿Qué es una arquitectura basada en eventos? (sitio web)AWS

AWS servicios

- Amazon API Gateway
- Amazon Aurora
- Amazon DynamoDB
- Amazon Elastic Compute Cloud (Amazon EC2)
- Amazon Elastic Container Service (Amazon ECS)
- Amazon Elastic Kubernetes Service (Amazon EKS)
- Elastic Load Balancing
- Amazon EventBridge
- AWS Fargate
- AWS Lambda
- Amazon Relational Database Service (Amazon RDS)
- Amazon Simple Notification Service (Amazon SNS)
- Amazon Simple Queue Service (Amazon SQS)

Otras herramientas

- Moto
- LocalStack

Historial de documentos

En la siguiente tabla, se describen cambios significativos de esta guía. Si quiere recibir notificaciones de futuras actualizaciones, puede suscribirse a las notificaciones RSS.

Cambio	Descripción	Fecha
Publicación inicial	—	15 de junio de 2022

AWS Glosario de orientación prescriptiva

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

Números

Las 7 R

Siete estrategias de migración comunes para trasladar aplicaciones a la nube. Estas estrategias se basan en las 5 R que Gartner identificó en 2011 y consisten en lo siguiente:

- Refactor/re-architect — Mueva una aplicación y modifique su arquitectura aprovechando al máximo las funciones nativas de la nube para mejorar la agilidad, el rendimiento y la escalabilidad. Por lo general, esto implica trasladar el sistema operativo y la base de datos. Ejemplo: migre su base de datos Oracle local a la PostgreSQL-Compatible edición Amazon Aurora.
- Redefinir la plataforma (transportar y redefinir): traslade una aplicación a la nube e introduzca algún nivel de optimización para aprovechar las capacidades de la nube. Ejemplo: Migrar la base de datos Oracle en las instalaciones a Amazon Relational Database Service (Amazon RDS) para Oracle en la nube de Nube de AWS.
- Recomprar (readquirir): cambie a un producto diferente, lo cual se suele llevar a cabo al pasar de una licencia tradicional a un modelo SaaS. Ejemplo: migre su sistema de gestión de relaciones con los clientes (CRM) a Salesforce.com.
- Volver a alojar (migrar mediante lift-and-shift): traslade una aplicación a la nube sin hacer cambios para aprovechar las funcionalidades de la nube. Ejemplo: Migrar la base de datos de Oracle en las instalaciones a Oracle en una instancia de EC2 en la Nube de AWS.
- Reubicar: (migrar el hipervisor mediante lift and shift): traslade la infraestructura a la nube sin comprar equipo nuevo, reescribir aplicaciones o modificar las operaciones actuales. Los servidores se migran de una plataforma en las instalaciones a un servicio en la nube para la misma plataforma. Ejemplo: migrar una Microsoft Hyper-V aplicación a AWS.
- Retener (revisitar): conserve las aplicaciones en el entorno de origen. Estas pueden incluir las aplicaciones que requieren una refactorización importante, que desee posponer para más adelante, y las aplicaciones heredadas que desee retener, ya que no hay ninguna justificación empresarial para migrarlas.

- Retirar: retire o elimine las aplicaciones que ya no sean necesarias en un entorno de origen.

A

A2A () Agent-to-Agent

Un protocolo completo para la colaboración entre agentes que facilita la delegación de tareas y la transferencia de estados.

ABAC

Consulte control de acceso basado en atributos.

servicios abstractos

Consulte servicios administrados.

ACID

Consulte atomicidad, consistencia, aislamiento, durabilidad.

migración activa-activa

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas (mediante una herramienta de replicación bidireccional o mediante operaciones de escritura doble) y ambas bases de datos gestionan las transacciones de las aplicaciones conectadas durante la migración. Este método permite la migración en lotes pequeños y controlados, en lugar de requerir una transición única. Es más flexible, pero requiere más trabajo que una migración activa-pasiva.

migración activa-pasiva

Método de migración de bases de datos en el que las bases de datos de origen y destino se mantienen sincronizadas, pero solo la de origen gestiona las transacciones de las aplicaciones conectadas, mientras los datos se replican en la de destino. La base de datos de destino no acepta ninguna transacción durante la migración.

Agente

Un sistema de IA que puede razonar, planificar y tomar medidas de forma autónoma utilizando herramientas para alcanzar los objetivos.

Agent Ops

Prácticas operativas para crear, probar, implementar y ejecutar agentes de IA en producción a escala.

función de agregación

Función SQL que actúa en un grupo de filas y calcula un único valor de devolución para el grupo. Entre los ejemplos de funciones de agregación se incluyen `SUM` y `MAX`.

IA

Consulte inteligencia artificial.

AIOps

Consulte operaciones de inteligencia artificial

anonimización

El proceso de eliminar permanentemente la información personal de un conjunto de datos. La anonimización puede ayudar a proteger la privacidad personal. Los datos anonimizados ya no se consideran datos personales.

antipatrones

Una solución que se utiliza con frecuencia para un problema recurrente en el que la solución es contraproducente, ineficaz o menos eficaz que una alternativa.

control de aplicaciones

Enfoque de seguridad que permite usar de manera exclusiva aplicaciones aprobadas para ayudar a proteger un sistema contra el malware.

cartera de aplicaciones

Recopilación de información detallada sobre cada aplicación que utiliza una organización, incluido el costo de creación y mantenimiento de la aplicación y su valor empresarial. Esta información es clave para el proceso de detección y análisis de la cartera y ayuda a identificar y priorizar las aplicaciones que se van a migrar, modernizar y optimizar.

inteligencia artificial (IA)

El campo de la informática que se dedica al uso de tecnologías informáticas para realizar funciones cognitivas que suelen estar asociadas a los seres humanos, como el aprendizaje, la resolución de problemas y el reconocimiento de patrones. Para más información, consulte ¿Qué es la inteligencia artificial?

operaciones de inteligencia artificial (AIOps)

El proceso de utilizar técnicas de machine learning para resolver problemas operativos, reducir los incidentes operativos y la intervención humana, y mejorar la calidad del servicio. Para obtener más información sobre cómo se utiliza AIOps en la estrategia de migración de AWS , consulte la Guía de integración de operaciones.

cifrado asimétrico

Algoritmo de cifrado que utiliza un par de claves, una clave pública para el cifrado y una clave privada para el descifrado. Puede compartir la clave pública porque no se utiliza para el descifrado, pero el acceso a la clave privada debe estar sumamente restringido.

atomicidad, consistencia, aislamiento, durabilidad (ACID)

Conjunto de propiedades de software que garantizan la validez de los datos y la fiabilidad operativa de una base de datos, incluso en caso de errores, cortes de energía u otros problemas.

control de acceso basado en atributos (ABAC)

La práctica de crear permisos detallados basados en los atributos del usuario, como el departamento, el puesto de trabajo y el nombre del equipo. Para obtener más información, consulte ABAC AWS en la documentación AWS Identity and Access Management (IAM).

origen de datos fidedigno

Ubicación en la que se almacena la versión principal de los datos, que se considera la fuente de información más fiable. Puede copiar los datos del origen de datos autorizado a otras ubicaciones con el fin de procesarlos o modificarlos, por ejemplo, anonimizarlos, redactarlos o seudonimizarlos.

Zona de disponibilidad

Una ubicación distinta dentro de una Región de AWS que está aislada de los fallos en otras zonas de disponibilidad y que proporciona una conectividad de red económica y de baja latencia a otras zonas de disponibilidad de la misma región.

AWS Marco de adopción de la nube (AWS CAF)

Un marco de directrices y mejores prácticas AWS para ayudar a las organizaciones a desarrollar un plan eficiente y eficaz para migrar con éxito a la nube. AWS CAF organiza la orientación en seis áreas de enfoque denominadas perspectivas: negocios, personas, gobierno, plataforma, seguridad y operaciones. Las perspectivas empresariales, humanas y de gobernanza se centran en las habilidades y los procesos empresariales; las perspectivas de plataforma, seguridad y

operaciones se centran en las habilidades y los procesos técnicos. Por ejemplo, la perspectiva humana se dirige a las partes interesadas que se ocupan de los Recursos Humanos (RR. HH.), las funciones del personal y la administración de las personas. Desde esta perspectiva, AWS CAF proporciona orientación para el desarrollo, la formación y la comunicación de las personas a fin de preparar a la organización para una adopción exitosa de la nube. Para obtener más información, consulte la Página web de AWS CAF y el Documento técnico de AWS CAF.

AWS Marco de calificación de la carga de trabajo (AWS WQF)

Herramienta que evalúa las cargas de trabajo de migración de bases de datos, recomienda estrategias de migración y proporciona estimaciones de trabajo. AWS WQF se incluye con AWS Schema Conversion Tool ().AWS SCT Analiza los esquemas de bases de datos y los objetos de código, el código de las aplicaciones, las dependencias y las características de rendimiento y proporciona informes de evaluación.

B

bot malicioso

Bot destinado a causar interrupciones o daños a personas u organizaciones.

BCP

Consulte planificación de la continuidad del negocio.

gráfico de comportamiento

Una vista unificada e interactiva del comportamiento de los recursos y de las interacciones a lo largo del tiempo. Puede utilizar un gráfico de comportamiento con Amazon Detective para examinar los intentos de inicio de sesión fallidos, las llamadas sospechosas a la API y acciones similares. Para obtener más información, consulte Datos en un gráfico de comportamiento en la documentación de Detective.

sistema big-endian

Un sistema que almacena primero el byte más significativo. Consulte también endianidad.

clasificación binaria

Un proceso que predice un resultado binario (una de las dos clases posibles). Por ejemplo, es posible que su modelo de ML necesite predecir problemas como “¿Este correo electrónico es spam o no es spam?” o “¿Este producto es un libro o un automóvil?”.

filtro de floración

Estructura de datos probabilística y eficiente en términos de memoria que se utiliza para comprobar si un elemento es miembro de un conjunto.

blue/green despliegue

Estrategia de implementación en la que se crean dos entornos separados, pero idénticos. La versión actual de la aplicación se ejecuta en un entorno (azul) y la nueva versión de la aplicación se ejecuta en el otro entorno (verde). Esta estrategia lo ayuda a hacer reversiones rápidas con un impacto mínimo.

bot

Aplicación de software que ejecuta tareas automatizadas a través de Internet y simula la actividad o interacción humana. Algunos bots son útiles o beneficiosos, como los rastreadores web que indexan la información de Internet. Otros bots, conocidos como bots maliciosos, tienen como objetivo causar interrupciones o daños a personas u organizaciones.

botnet

Redes de bots infectadas por malware y que están bajo el control de una sola parte, conocida como pastor de bots u operador de bots. Las botnets son el mecanismo más conocido para escalar los bots y su impacto.

branch

Área contenida de un repositorio de código. La primera rama que se crea en un repositorio es la rama principal. Puede crear una rama nueva a partir de una rama existente y, a continuación, desarrollar características o corregir errores en la rama nueva. Una rama que se genera para crear una característica se denomina comúnmente rama de característica. Cuando la característica se encuentra lista para su lanzamiento, se vuelve a combinar la rama de característica con la rama principal. Para obtener más información, consulte Acerca de las sucursales (GitHub documentación).

acceso de emergencia

En circunstancias excepcionales y mediante un proceso aprobado, es una forma rápida de que un usuario pueda acceder a un Cuenta de AWS sitio al que normalmente no tiene permisos de acceso. Para obtener más información, consulte el indicador de implementación de procedimientos rompe-cristales en la AWS Well-Architected guía.

estrategia de implementación sobre infraestructura existente

La infraestructura existente en su entorno. Al adoptar una estrategia de implementación sobre infraestructura existente para una arquitectura de sistemas, se diseña la arquitectura en función de las limitaciones de los sistemas y la infraestructura actuales. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

caché de búfer

El área de memoria donde se almacenan los datos a los que se accede con más frecuencia.

capacidad empresarial

Lo que hace una empresa para generar valor (por ejemplo, ventas, servicio al cliente o marketing). Las arquitecturas de microservicios y las decisiones de desarrollo pueden estar impulsadas por las capacidades empresariales. Para obtener más información, consulte la sección Organizado en torno a las capacidades empresariales del documento técnico Ejecutar microservicios en contenedores en AWS.

planificación de la continuidad del negocio (BCP)

Plan que aborda el posible impacto de un evento disruptivo, como una migración a gran escala en las operaciones y permite a la empresa reanudar las operaciones rápidamente.

C

CAF

Consulte AWS Cloud Adoption Framework.

implementación canario

Lanzamiento lento e incremental de una versión para los usuarios finales. Cuando tenga mayor confianza en la nueva versión, la implementa y reemplaza la versión actual en su totalidad.

CCoE

Consulte Centro de excelencia en la nube.

CDC

Consulte captura de datos de cambios.

captura de datos de cambio (CDC)

Proceso de seguimiento de los cambios en un origen de datos, como una tabla de base de datos, y registro de los metadatos relacionados con el cambio. Puede utilizar los CDC para diversos fines, como auditar o replicar los cambios en un sistema de destino para mantener la sincronización.

ingeniería del caos

Introducción intencionada de fallos o eventos disruptivos para poner a prueba la resiliencia de un sistema. Puedes usar AWS Fault Injection Service (AWS FIS) para realizar experimentos que estresen tus AWS cargas de trabajo y evalúen su respuesta.

CI/CD

Consulte integración continua y entrega continua.

clasificación

Un proceso de categorización que permite generar predicciones. Los modelos de ML para problemas de clasificación predicen un valor discreto. Los valores discretos siempre son distintos entre sí. Por ejemplo, es posible que un modelo necesite evaluar si hay o no un automóvil en una imagen.

Desarrollador ciudadano

Un usuario empresarial que crea aplicaciones de IA utilizando plataformas sin code/low código sin conocimientos técnicos especializados.

cifrado del cliente

Cifrado de datos localmente, antes de que el objetivo los Servicio de AWS reciba.

Centro de excelencia en la nube (CCoE)

Equipo multidisciplinario que impulsa los esfuerzos de adopción de la nube en toda la organización, incluido el desarrollo de las prácticas recomendadas en la nube, la movilización de recursos, el establecimiento de plazos de migración y la dirección de la organización durante las transformaciones a gran escala. Para obtener más información, consulte las publicaciones de CCoE en el blog de estrategia Nube de AWS empresarial.

computación en la nube

La tecnología en la nube que se utiliza normalmente para la administración de dispositivos de IoT y el almacenamiento de datos de forma remota. La computación en la nube suele estar relacionada con la tecnología de computación de periferia.

modelo operativo en la nube

En una organización de TI, el modelo operativo que se utiliza para crear, madurar y optimizar uno o más entornos de nube. Para obtener más información, consulte Creación de su modelo operativo de nube.

etapas de adopción de la nube

Las siguientes son las cuatro fases por las que suelen pasar las empresas cuando migran a la Nube de AWS:

- Proyecto: ejecución de algunos proyectos relacionados con la nube con fines de prueba de concepto y aprendizaje
- Fundamento: realización de inversiones fundamentales para escalar la adopción de la nube (p. ej., crear una zona de aterrizaje, definir un CCoE, establecer un modelo de operaciones)
- Migración: migración de aplicaciones individuales
- Re-invention — Optimizar los productos y servicios e innovar en la nube

Stephen Orban definió estas etapas en la entrada del blog The Journey Toward Cloud-First & the Stages of Adoption del blog Nube de AWS Enterprise Strategy. Para obtener información sobre su relación con la estrategia de AWS migración, consulte la guía de preparación para la migración.

CMDB

Consulte base de datos de administración de configuración.

repositorio de código

Una ubicación donde el código fuente y otros activos, como documentación, muestras y scripts, se almacenan y actualizan mediante procesos de control de versiones. Algunos repositorios en la nube comunes son GitHub o Bitbucket Cloud. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio se encuentra dedicado a una única funcionalidad. Una sola CI/CD canalización puede utilizar varios repositorios.

caché en frío

Una caché de búfer que está vacía no está bien poblada o contiene datos obsoletos o irrelevantes. Esto afecta al rendimiento, ya que la instancia de la base de datos debe leer desde la memoria principal o el disco, lo que es más lento que leer desde la memoria caché del búfer.

datos fríos

Datos a los que se accede con poca frecuencia y que suelen ser históricos. Al consultar este tipo de datos, normalmente se aceptan consultas lentas. Trasladar estos datos a niveles o clases de almacenamiento de menor rendimiento y menos costosos puede reducir los costos.

visión artificial (CV)

Campo de la IA que utiliza el machine learning para analizar y extraer información de formatos visuales, como imágenes y videos digitales. Por ejemplo, Amazon SageMaker AI proporciona algoritmos de procesamiento de imágenes para CV.

deriva de configuración

En el caso de una carga de trabajo, un cambio en la configuración con respecto al estado esperado. Podría provocar que la carga de trabajo deje de cumplir las normas y, por lo general, es gradual e involuntaria.

base de datos de administración de configuración (CMDB)

Repositorio que almacena y administra información sobre una base de datos y su entorno de TI, incluidos los componentes de hardware y software y sus configuraciones. Por lo general, los datos de una CMDB se utilizan en la etapa de detección y análisis de la cartera de productos durante la migración.

paquete de conformidad

Un conjunto de AWS Config reglas y medidas correctivas que puede reunir para personalizar sus controles de conformidad y seguridad. Puede implementar un paquete de conformidad como una entidad única en una región Cuenta de AWS y, o en una organización, mediante una plantilla YAML. Para obtener más información, consulta los paquetes de conformidad en la documentación. AWS Config

integración y entrega continuas () CI/CD

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

CV

Consulte visión artificial.

D

datos en reposo

Datos que están estacionarios en la red, como los datos que se encuentran almacenados.

clasificación de datos

Un proceso para identificar y clasificar los datos de su red en función de su importancia y sensibilidad. Es un componente fundamental de cualquier estrategia de administración de riesgos de ciberseguridad porque lo ayuda a determinar los controles de protección y retención adecuados para los datos. La clasificación de los datos es un componente del pilar de seguridad del AWS Well-Architected Framework. Para obtener más información, consulte Clasificación de datos.

deriva de datos

Una variación significativa entre los datos de producción y los datos que se utilizaron para entrenar un modelo de machine learning, o un cambio significativo en los datos de entrada a lo largo del tiempo. La deriva de datos puede reducir la calidad, la precisión y la imparcialidad generales de las predicciones de los modelos de machine learning.

datos en tránsito

Datos que se mueven de forma activa por la red, por ejemplo, entre los recursos de la red.

malla de datos

Marco de arquitectura que proporciona una propiedad de datos distribuida y descentralizada con una administración y una gobernanza centralizadas.

minimización de datos

El principio de recopilar y procesar solo los datos estrictamente necesarios. Practicar la minimización de los datos Nube de AWS puede reducir los riesgos de privacidad, los costos y la huella de carbono de la analítica.

perímetro de datos

Un conjunto de barreras preventivas en su AWS entorno que ayudan a garantizar que solo las identidades confiables accedan a los recursos confiables desde las redes esperadas. Para obtener más información, consulte Crear un perímetro de datos sobre. AWS

preprocesamiento de datos

Transformar los datos sin procesar en un formato que su modelo de ML pueda analizar fácilmente. El preprocesamiento de datos puede implicar eliminar determinadas columnas o filas y corregir los valores faltantes, incoherentes o duplicados.

procedencia de los datos

El proceso de rastrear el origen y el historial de los datos a lo largo de su ciclo de vida, por ejemplo, la forma en que se generaron, transmitieron y almacenaron los datos.

titular de los datos

Persona cuyos datos se recopilan y procesan.

almacenamiento de datos

Sistema de administración de datos que respalda la inteligencia empresarial, como los análisis. Los almacenes de datos suelen contener grandes cantidades de datos históricos y, por lo general, se utilizan para las consultas y los análisis.

lenguaje de definición de datos (DDL)

Instrucciones o comandos para crear o modificar la estructura de tablas y objetos de una base de datos.

lenguaje de manipulación de datos (DML)

Instrucciones o comandos para modificar (insertar, actualizar y eliminar) la información de una base de datos.

DDL

Consulte lenguaje de definición de bases de datos.

conjunto profundo

Combinar varios modelos de aprendizaje profundo para la predicción. Puede utilizar conjuntos profundos para obtener una predicción más precisa o para estimar la incertidumbre de las predicciones.

aprendizaje profundo

Un subcampo del ML que utiliza múltiples capas de redes neuronales artificiales para identificar el mapeo entre los datos de entrada y las variables objetivo de interés.

defensa en profundidad

Un enfoque de seguridad de la información en el que se distribuyen cuidadosamente una serie de mecanismos y controles de seguridad en una red informática para proteger la confidencialidad, la integridad y la disponibilidad de la red y de los datos que contiene. Al adoptar esta estrategia AWS, se añaden varios controles en diferentes capas de la AWS Organizations estructura para ayudar a proteger los recursos. Por ejemplo, un enfoque de defensa en profundidad podría combinar la autenticación multifactor, la segmentación de la red y el cifrado.

administrador delegado

En AWS Organizations, un servicio compatible puede registrar una cuenta de AWS miembro para administrar las cuentas de la organización y gestionar los permisos de ese servicio. Esta cuenta se denomina administrador delegado para ese servicio. Para obtener más información y una lista de servicios compatibles, consulte Servicios que funcionan con AWS Organizations en la documentación de AWS Organizations .

Implementación

El proceso de hacer que una aplicación, características nuevas o correcciones de código se encuentren disponibles en el entorno de destino. La implementación abarca implementar cambios en una base de código y, a continuación, crear y ejecutar esa base en los entornos de la aplicación.

entorno de desarrollo

Consulte entorno.

control de detección

Un control de seguridad que se ha diseñado para detectar, registrar y alertar después de que se produzca un evento. Estos controles son una segunda línea de defensa, ya que lo advierten sobre los eventos de seguridad que han eludido los controles preventivos establecidos. Para obtener más información, consulte Controles de detección en Implementación de controles de seguridad en AWS.

asignación de flujos de valor para el desarrollo (DVSM)

Proceso que se utiliza para identificar y priorizar las restricciones que afectan negativamente a la velocidad y la calidad en el ciclo de vida del desarrollo de software. DVSM amplía el proceso de asignación del flujo de valor diseñado originalmente para las prácticas de fabricación ajustada. Se centra en los pasos y los equipos necesarios para crear y transferir valor a través del proceso de desarrollo de software.

gemelo digital

Representación virtual de un sistema del mundo real, como un edificio, una fábrica, un equipo industrial o una línea de producción. Los gemelos digitales son compatibles con el mantenimiento predictivo, la supervisión remota y la optimización de la producción.

tabla de dimensiones

En un esquema en estrella, tabla más pequeña que contiene los atributos de datos sobre los datos cuantitativos en una tabla de hechos. Los atributos de la tabla de dimensiones suelen ser campos de texto o números discretos que se comportan como texto. Estos atributos se suelen utilizar para restringir consultas, filtrarlas y etiquetar los conjuntos de resultados.

desastre

Un evento que impide que una carga de trabajo o un sistema cumplan sus objetivos empresariales en su ubicación principal de implementación. Estos eventos pueden ser desastres naturales, fallos técnicos o el resultado de acciones humanas, como una configuración incorrecta involuntaria o un ataque de malware.

recuperación de desastres (DR)

Estrategia y proceso que utiliza para minimizar el tiempo de inactividad y la pérdida de datos a causa de un desastre. Para obtener más información, consulte Recuperación de cargas de trabajo ante desastres en AWS: Recuperación en la nube en el AWS Well-Architected marco.

DML

Consulte lenguaje de manipulación de bases de datos.

diseño basado en el dominio

Un enfoque para desarrollar un sistema de software complejo mediante la conexión de sus componentes a dominios en evolución, o a los objetivos empresariales principales, a los que sirve cada componente. Eric Evans introdujo este concepto en su libro Domain-Driven Design: Tackling Complexity in the Heart of Software (Boston: Addison-Wesley Professional, 2003). Para obtener información sobre cómo utilizar el diseño basado en dominios con el patrón de higos estranguladores, consulte Modernización gradual de los servicios web antiguos de ASP.NET Microsoft (ASMX) mediante contenedores y Amazon API Gateway.

DR

Consulte recuperación ante desastres.

Detección de desviaciones

Seguimiento de las desviaciones con respecto a una configuración con línea de base. Por ejemplo, puedes usarlo AWS CloudFormation para detectar desviaciones en los recursos del sistema o puedes usarlo AWS Control Tower para detectar cambios en tu landing zone que puedan afectar al cumplimiento de los requisitos de gobierno.

DVSM

Consulte asignación de flujos de valor para el desarrollo.

E

EDA

Consulte análisis de datos de tipo exploratorio.

EDI

Consulte intercambio electrónico de datos.

computación en la periferia

La tecnología que aumenta la potencia de cálculo de los dispositivos inteligentes en la periferia de una red de IoT. En comparación con la computación en la nube, la computación de periferia puede reducir la latencia de la comunicación y mejorar el tiempo de respuesta.

intercambio electrónico de datos (EDI)

Intercambio automatizado de documentos comerciales entre organizaciones. Para más información, consulte ¿Qué es el intercambio electrónico de datos?

cifrado

Proceso de computación que transforma datos de texto plano, que son legibles por humanos, en texto cifrado.

clave de cifrado

Cadena criptográfica de bits aleatorios que se genera mediante un algoritmo de cifrado. Las claves pueden variar en longitud y cada una se ha diseñado para ser impredecible y única.

endianidad

El orden en el que se almacenan los bytes en la memoria del ordenador. Big-endian los sistemas almacenan primero el byte más significativo. Little-endian los sistemas almacenan primero el byte menos significativo.

punto de conexión

Consulte punto de conexión de servicio.

servicio de punto de conexión

Servicio que puede alojar en una nube privada virtual (VPC) para compartir con otros usuarios. Puede crear un servicio de punto final con AWS PrivateLink entidades principales Cuentas de AWS o AWS Identity and Access Management (de IAM) y conceder permisos a ellas. Estas cuentas o entidades principales pueden conectarse a su servicio de punto de conexión de forma privada mediante la creación de puntos de conexión de VPC de interfaz. Para obtener más información, consulte Creación de un servicio de punto de conexión en la documentación de Amazon Virtual Private Cloud (Amazon VPC).

planificación de recursos empresariales (ERP)

Sistema que automatiza y administra los procesos empresariales clave (como la contabilidad, MES y la administración de proyectos) de una empresa.

cifrado de sobre

El proceso de cifrar una clave de cifrado con otra clave de cifrado. Para obtener más información, consulte el cifrado de sobres en la documentación de AWS Key Management Service (AWS KMS).

entorno

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

- entorno de desarrollo: instancia de una aplicación en ejecución que solo se encuentra 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 un CI/CD proceso, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios que no sean 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.

epopeya

En las metodologías ágiles, son categorías funcionales que ayudan a organizar y priorizar el trabajo. Las epopeyas brindan una descripción detallada de los requisitos y las tareas de implementación. Por ejemplo, las epopeyas AWS de seguridad de CAF incluyen la gestión de identidades y accesos, los controles de detección, la seguridad de la infraestructura, la protección de datos y la respuesta a incidentes. Para obtener más información sobre las epopeyas en la estrategia de migración de AWS , consulte la Guía de implementación del programa.

ERP

Consulte planificación de recursos empresariales.

análisis de datos de tipo exploratorio (EDA)

El proceso de analizar un conjunto de datos para comprender sus características principales. Se recopilan o agregan datos y, a continuación, se realizan las investigaciones iniciales para encontrar patrones, detectar anomalías y comprobar las suposiciones. El EDA se realiza mediante el cálculo de estadísticas resumidas y la creación de visualizaciones de datos.

F

tabla de hechos

Tabla central de un esquema en estrella. Almacena datos cuantitativos sobre operaciones empresariales. Por lo general, una tabla de hechos contiene dos tipos de columnas: las que contienen medidas y las que contienen una clave externa para una tabla de dimensiones.

Fail Fast

Filosofía que utiliza pruebas frecuentes e incrementales para reducir el ciclo de vida del desarrollo. Es una parte fundamental de los enfoques ágiles.

límite de aislamiento de errores

En el Nube de AWS, un límite, como una zona de disponibilidad Región de AWS, un plano de control o un plano de datos, que limita el efecto de una falla y ayuda a mejorar la resiliencia de las cargas de trabajo. Para más información, consulte AWS Fault Isolation Boundaries.

rama de característica

Consulte rama.

características

Los datos de entrada que se utilizan para hacer una predicción. Por ejemplo, en un contexto de fabricación, las características pueden ser imágenes que se capturan periódicamente desde la línea de fabricación.

importancia de las características

La importancia que tiene una característica para las predicciones de un modelo. Por lo general, esto se expresa como una puntuación numérica que se puede calcular mediante diversas técnicas, como las explicaciones aditivas de Shapley (SHAP) y los gradientes integrados. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

transformación de funciones

Optimizar los datos para el proceso de ML, lo que incluye enriquecer los datos con fuentes adicionales, escalar los valores o extraer varios conjuntos de información de un solo campo de datos. Esto permite que el modelo de ML se beneficie de los datos. Por ejemplo, si divide la fecha del “27 de mayo de 2021 00:15:37” en “jueves”, “mayo”, “2021” y “15”, puede ayudar al algoritmo de aprendizaje a aprender patrones matizados asociados a los diferentes componentes de los datos.

peticiones con pocos pasos

Proporcionar a un LLM una pequeña cantidad de ejemplos que demuestren la tarea y el resultado deseado antes de pedirle que lleve a cabo una tarea similar. Esta técnica es una aplicación del aprendizaje contextual, en el que los modelos aprenden a partir de ejemplos (tomas) integrados en las instrucciones. Few-shot Las indicaciones pueden ser eficaces para tareas que requieren un formato, un razonamiento o un conocimiento del dominio específicos. Consulte también peticiones desde cero.

FGAC

Consulte control de acceso detallado.

control de acceso preciso (FGAC)

El uso de varias condiciones que tienen por objetivo permitir o denegar una solicitud de acceso.

migración relámpago

Método de migración de bases de datos que utiliza la replicación continua de datos mediante la captura de datos de cambio para migrar los datos en el menor tiempo posible, en lugar de utilizar un enfoque gradual. El objetivo es reducir al mínimo el tiempo de inactividad.

FM

Consulte modelo fundacional.

Modelo fundacional (FM)

Gran red neuronal de aprendizaje profundo que se entrenó con conjuntos de datos masivos de datos generalizados y no etiquetados. Los FM pueden hacer una amplia variedad de tareas generales, como comprender el lenguaje, generar texto e imágenes y conversar en lenguaje natural. Para más información, consulte ¿Qué son los modelos fundacionales?

Puerta de enlace FM

Un intermediario centralizado que controla y normaliza el acceso a los modelos básicos. También se conoce como puerta de enlace LLM.

G

IA generativa

Subconjunto de modelos de IA que se entrenaron con grandes cantidades de datos y que pueden utilizar una simple petición de texto para crear contenido y artefactos nuevos, como imágenes, videos, texto y audio. Para más información, consulte ¿Qué es la IA generativa?

bloqueo geográfico

Consulte restricciones geográficas.

restricciones geográficas (bloqueo geográfico)

En Amazon CloudFront, una opción para impedir que los usuarios de países específicos accedan a las distribuciones de contenido. Puede utilizar una lista de permitidos o bloqueados para especificar los países aprobados y prohibidos. Para obtener más información, consulta Restringir la distribución geográfica del contenido en la CloudFront documentación.

Flujo de trabajo de Gitflow

: Un enfoque en el que los entornos inferiores y superiores utilizan diferentes ramas en un repositorio de código fuente. El flujo de trabajo de Gitflow se considera heredado, mientras que el flujo de trabajo basado en enlaces troncales es el enfoque moderno preferido.

imagen dorada

: Instantánea de un sistema o software que se usa como plantilla para implementar nuevas instancias de ese sistema o software. Por ejemplo, en la fabricación, una imagen dorada se puede utilizar para aprovisionar software en varios dispositivos y ayuda a mejorar la velocidad, la escalabilidad y la productividad de las operaciones de fabricación de dispositivos.

estrategia de implementación desde cero

: La ausencia de infraestructura existente en un entorno nuevo. Al adoptar una estrategia de implementación desde cero para una arquitectura de sistemas, puede seleccionar todas las tecnologías nuevas sin que estas deban ser compatibles con una infraestructura existente, lo que también se conoce como implementación sobre infraestructura existente. Si está ampliando la infraestructura existente, puede combinar las estrategias de implementación sobre infraestructuras existentes y de implementación desde cero.

barrera de protección

: Una regla de alto nivel que ayuda a regular los recursos, las políticas y la conformidad en todas las unidades organizativas (OU). Las barreras de protección preventivas aplican políticas para garantizar la alineación con los estándares de conformidad. Se implementan mediante políticas de control de servicios y límites de permisos de IAM. Las barreras de protección de detección detectan las vulneraciones de las políticas y los problemas de conformidad, y generan alertas para su corrección. Se implementan mediante Amazon AWS Config AWS Security Hub CSPM GuardDuty AWS Trusted Advisor, Amazon Inspector y AWS Lambda cheques personalizados.

barandas (AI)

: Mecanismos de seguridad que filtran, validan y restringen las entradas y salidas de los agentes para ayudar a garantizar un comportamiento responsable y seguro de la IA.

H

HA

: Consulte alta disponibilidad.

migración heterogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que utilice un motor de base de datos diferente (por ejemplo, de Oracle a Amazon Aurora). La migración heterogénea suele ser parte de un esfuerzo de rediseño de la arquitectura y convertir el esquema puede ser una tarea compleja. AWS ofrece AWS SCT, lo cual ayuda con las conversiones de esquemas.

alta disponibilidad (HA)

La capacidad de una carga de trabajo para funcionar de forma continua, sin intervención, en caso de desafíos o desastres. Los sistemas de alta disponibilidad están diseñados para realizar una conmutación por error automática, ofrecer un rendimiento de alta calidad de forma constante y gestionar diferentes cargas y fallos con un impacto mínimo en el rendimiento.

modernización histórica

Un enfoque utilizado para modernizar y actualizar los sistemas de tecnología operativa (TO) a fin de satisfacer mejor las necesidades de la industria manufacturera. Un histórico es un tipo de base de datos que se utiliza para recopilar y almacenar datos de diversas fuentes en una fábrica.

datos de reserva

Parte de los datos históricos etiquetados que se ocultan de un conjunto de datos que se utiliza para entrenar un modelo de machine learning. Puede utilizar los datos de reserva para evaluar el rendimiento del modelo mediante la comparación de las predicciones del modelo con los datos de reserva.

human-in-the-loop (HiTL)

Un patrón de flujo de trabajo en el que la ejecución de los agentes se detiene para su revisión y aprobación por parte de una persona en los puntos de decisión críticos.

migración homogénea de bases de datos

Migración de la base de datos de origen a una base de datos de destino que comparte el mismo motor de base de datos (por ejemplo, Microsoft SQL Server a Amazon RDS para SQL Server). La migración homogénea suele formar parte de un esfuerzo para volver a alojar o redefinir la plataforma. Puede utilizar las utilidades de bases de datos nativas para migrar el esquema.

datos recientes

Datos a los que se accede con frecuencia, como datos en tiempo real o datos traslacionales recientes. Por lo general, estos datos requieren un nivel o una clase de almacenamiento de alto rendimiento para proporcionar respuestas rápidas a las consultas.

hotfix

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.

periodo de hiperatención

Periodo, inmediatamente después de la transición, durante el cual un equipo de migración administra y monitorea las aplicaciones migradas en la nube para solucionar cualquier problema. Por lo general, este periodo dura de 1 a 4 días. Al final del periodo de hiperatención, el equipo de migración suele transferir la responsabilidad de las aplicaciones al equipo de operaciones en la nube.

I

IaC

Consulte infraestructura como código.

políticas basadas en identidades

Política asociada a uno o más directores de IAM que define sus permisos en el entorno. Nube de AWS

aplicación inactiva

Aplicación que utiliza un promedio de CPU y memoria de entre 5 y 20 por ciento durante un periodo de 90 días. En un proyecto de migración, es habitual retirar estas aplicaciones o mantenerlas en las instalaciones.

IIoT

Consulte Internet de las cosas industrial.

infraestructura inmutable

Modelo que implementa una nueva infraestructura para las cargas de trabajo de producción en lugar de actualizar o modificar la infraestructura existente o aplicarle revisiones. Las infraestructuras inmutables son de manera intrínseca más coherentes, fiables y predecibles que las infraestructuras mutables. Para obtener más información, consulte las mejores prácticas del Framework para implementar con una infraestructura inmutable. AWS Well-Architected

VPC entrante (de entrada)

En una arquitectura de AWS cuentas múltiples, una VPC que acepta, inspecciona y enruta las conexiones de red desde fuera de una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

migración gradual

Estrategia de transición en la que se migra la aplicación en partes pequeñas en lugar de realizar una transición única y completa. Por ejemplo, puede trasladar inicialmente solo unos pocos microservicios o usuarios al nuevo sistema. Tras comprobar que todo funciona correctamente, puede trasladar microservicios o usuarios adicionales de forma gradual hasta que pueda retirar su sistema heredado. Esta estrategia reduce los riesgos asociados a las grandes migraciones.

Industria 4.0

Un término que Klaus Schwab introdujo en 2016 para referirse a la modernización de los procesos de fabricación mediante avances en la conectividad, los datos en tiempo real, la automatización, el análisis y. AI/ML

infraestructura

Todos los recursos y activos que se encuentran 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. La IaC se ha diseñado para ayudarlo a centralizar la administración de la infraestructura, estandarizar los recursos y escalar con rapidez a fin de que los entornos nuevos sean repetibles, fiables y consistentes.

Internet de las cosas industrial (IIoT)

El uso de sensores y dispositivos conectados a Internet en los sectores industriales, como el productivo, el eléctrico, el automotriz, el sanitario, el de las ciencias de la vida y el de la agricultura. Para obtener más información, consulte Creación de una estrategia de transformación digital del Internet de las cosas industrial (IIoT).

VPC de inspección

En una arquitectura de AWS cuentas múltiples, una VPC centralizada que gestiona las inspecciones del tráfico de red entre las VPC (iguales o Regiones de AWS diferentes), Internet y las redes locales. La Arquitectura de referencia de seguridad de AWS recomienda configurar su

cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

Internet de las cosas (IoT)

Red de objetos físicos conectados con sensores o procesadores integrados que se comunican con otros dispositivos y sistemas a través de Internet o de una red de comunicación local. Para obtener más información, consulte ¿Qué es IoT?.

interpretabilidad

Característica de un modelo de machine learning que describe el grado en que un ser humano puede entender cómo las predicciones del modelo dependen de sus entradas. Para obtener más información, consulte Interpretabilidad del modelo de aprendizaje automático con. AWS

IoT

Consulte Internet de las cosas.

biblioteca de información de TI (ITIL)

Conjunto de prácticas recomendadas para ofrecer servicios de TI y alinearlos con los requisitos empresariales. La ITIL proporciona la base para la ITSM.

administración de servicios de TI (ITSM)

Actividades asociadas con el diseño, la implementación, la administración y el soporte de los servicios de TI para una organización. Para obtener información sobre la integración de las operaciones en la nube con las herramientas de ITSM, consulte la Guía de integración de operaciones.

ITIL

Consulte biblioteca de información de TI.

ITSM

Consulte administración de servicios de TI.

L

control de acceso basado en etiquetas (LBAC)

Una implementación del control de acceso obligatorio (MAC) en la que a los usuarios y a los propios datos se les asigna explícitamente un valor de etiqueta de seguridad. La intersección

entre la etiqueta de seguridad del usuario y la etiqueta de seguridad de los datos determina qué filas y columnas puede ver el usuario.

zona de aterrizaje

Una landing zone es un AWS entorno multicuenta bien diseñado, escalable y seguro. Este es un punto de partida desde el cual las empresas pueden lanzar e implementar rápidamente cargas de trabajo y aplicaciones con confianza en su entorno de seguridad e infraestructura. Para obtener más información sobre las zonas de aterrizaje, consulte Configuración de un entorno de AWS seguro y escalable con varias cuentas.

modelo de lenguaje de gran tamaño (LLM)

Modelo de IA de aprendizaje profundo que se entrenó previamente con una gran cantidad de datos. Un LLM puede llevar a cabo varias tareas, como responder preguntas, resumir documentos, traducir textos a otros idiomas y completar oraciones. Para más información, consulte ¿Qué es un LLM (modelo de lenguaje de gran tamaño)?

migración grande

Migración de 300 servidores o más.

LBAC

Consulte control de acceso basado en etiquetas.

privilegio mínimo

La práctica recomendada de seguridad que consiste en conceder los permisos mínimos necesarios para realizar una tarea. Para obtener más información, consulte Aplicar permisos de privilegio mínimo en la documentación de IAM.

migrar mediante lift-and-shift

Consulte Las 7 R.

sistema little-endian

Un sistema que almacena primero el byte menos significativo. Consulte también endianidad.

LLM

Consulte modelo de lenguaje de gran tamaño.

entornos inferiores

Consulte entorno.

M

machine learning (ML)

Un tipo de inteligencia artificial que utiliza algoritmos y técnicas para el reconocimiento y el aprendizaje de patrones. El ML analiza y aprende de los datos registrados, como los datos del Internet de las cosas (IoT), para generar un modelo estadístico basado en patrones. Para más información, consulte Machine learning.

rama principal

Consulte rama.

malware

Software diseñado para comprometer la seguridad o la privacidad de la computadora. El malware podría interrumpir los sistemas informáticos, filtrar información confidencial u obtener acceso no autorizado. Algunos ejemplos de malware son los virus, los gusanos, el ransomware, los troyanos, el spyware y los registradores de pulsaciones de teclas.

Servicios administrados

Servicios de AWS en el que AWS opera la capa de infraestructura, el sistema operativo y las plataformas, y se accede a los puntos finales para almacenar y recuperar datos. Amazon Simple Storage Service (Amazon S3) y Amazon DynamoDB son ejemplos de servicios administrados. También se conocen como servicios abstractos.

sistema de ejecución de fabricación (MES)

Sistema de software para seguir, supervisar, documentar y controlar los procesos de producción que convierten las materias primas en productos acabados en la zona de producción.

MAP

Consulte Programa de aceleración de la migración.

MCP

Consulte Model Context Protocol.

Protocolo de contexto para modelos (MCP)

Un protocolo sin estado para la comunicación entre el agente y la herramienta.

Servidor MCP

Un servicio que expone una o más herramientas a través del protocolo Model Context.

mecanismo

Proceso completo mediante el que se crea una herramienta, se impulsa su adopción y, a continuación, se inspeccionan los resultados para hacer ajustes. Un mecanismo es un ciclo que se refuerza y mejora por sí mismo a medida que funciona. Para obtener más información, consulte Creación de mecanismos en el AWS Well-Architected marco.

cuenta de miembro

Todas las Cuentas de AWS demás cuentas, excepto la de administración, que forman parte de una organización AWS Organizations. Una cuenta no puede pertenecer a más de una organización a la vez.

MES

Consulte sistema de ejecución de fabricación.

Message Queuing Telemetry Transport (MQTT)

Un protocolo de comunicación ligero de máquina a máquina (M2M), basado en el publish/subscribepatrón, para dispositivos de IoT con recursos limitados.

microservicio

Un servicio pequeño e independiente que se comunica a través de API bien definidas y que, por lo general, es propiedad de equipos pequeños e independientes. Por ejemplo, un sistema de seguros puede incluir microservicios que se adapten a las capacidades empresariales, como las de ventas o marketing, o a subdominios, como las de compras, reclamaciones o análisis. Los beneficios de los microservicios incluyen la agilidad, la escalabilidad flexible, la facilidad de implementación, el código reutilizable y la resiliencia. Para obtener más información, consulte Integrar microservicios mediante servicios sin servidor. AWS

arquitectura de microservicios

Un enfoque para crear una aplicación con componentes independientes que ejecutan cada proceso de la aplicación como un microservicio. Estos microservicios se comunican a través de una interfaz bien definida mediante API ligeras. Cada microservicio de esta arquitectura se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación. Para obtener más información, consulte Implementación de microservicios en. AWS

Programa de aceleración de la migración (MAP)

Un AWS programa que proporciona soporte de consultoría, formación y servicios para ayudar a las organizaciones a crear una base operativa sólida para migrar a la nube y para ayudar a

compensar el costo inicial de las migraciones. El MAP incluye una metodología de migración para ejecutar las migraciones antiguas de forma metódica y un conjunto de herramientas para automatizar y acelerar los escenarios de migración más comunes.

migración a escala

Proceso de transferencia de la mayoría de la cartera de aplicaciones a la nube en oleadas, con más aplicaciones desplazadas a un ritmo más rápido en cada oleada. En esta fase, se utilizan las prácticas recomendadas y las lecciones aprendidas en las fases anteriores para implementar una fábrica de migración de equipos, herramientas y procesos con el fin de agilizar la migración de las cargas de trabajo mediante la automatización y la entrega ágil. Esta es la tercera fase de la estrategia de migración de AWS.

fábrica de migración

Cross-functional equipos que agilizan la migración de las cargas de trabajo mediante enfoques ágiles y automatizados. Los equipos de las fábricas de migración suelen estar compuestos por analistas y propietarios de operaciones, ingenieros de migración, desarrolladores y DevOps profesionales que trabajan a pasos agigantados. Entre el 20 y el 50 por ciento de la cartera de aplicaciones empresariales se compone de patrones repetidos que pueden optimizarse mediante un enfoque de fábrica. Para obtener más información, consulte la discusión sobre las fábricas de migración y la Guía de fábricas de migración a la nube en este contenido.

metadatos de migración

Información sobre la aplicación y el servidor que se necesita para completar la migración. Cada patrón de migración requiere un conjunto diferente de metadatos de migración. Algunos ejemplos de metadatos de migración son la subred de destino, el grupo de seguridad y AWS la cuenta.

patrón de migración

Tarea de migración repetible que detalla la estrategia de migración, el destino de la migración y la aplicación o el servicio de migración utilizados. Ejemplo: rehospede la migración a Amazon EC2 AWS con Application Migration Service.

Migration Portfolio Assessment (MPA)

Herramienta en línea que proporciona información a fin de validar los argumentos comerciales necesarios para migrar a la Nube de AWS. La MPA ofrece una evaluación detallada de la cartera (adecuación del tamaño de los servidores, precios, comparaciones del costo total de propiedad, análisis de los costos de migración), así como una planificación de la migración (análisis y recopilación de datos de aplicaciones, agrupación de aplicaciones, priorización de la migración y

planificación de oleadas). La herramienta MPA (requiere iniciar sesión) está disponible de forma gratuita para todos los AWS consultores y consultores de los socios de APN.

Evaluación de la preparación para la migración (MRA)

Proceso que consiste en obtener información sobre el estado de preparación de una organización para la nube, identificar sus puntos fuertes y débiles y elaborar un plan de acción para cerrar las brechas identificadas mediante el AWS CAF. Para obtener más información, consulte la Guía de preparación para la migración. La MRA es la primera fase de la estrategia de migración de AWS.

estrategia de migración

Enfoque utilizado para migrar una carga de trabajo a la Nube de AWS. Para más información, consulte la entrada Las 7 R de este glosario y también Mobilize your organization to accelerate large-scale migrations.

ML

Consulte machine learning.

modernización

Transformar una aplicación obsoleta (antigua o monolítica) y su infraestructura en un sistema ágil, elástico y de alta disponibilidad en la nube para reducir los gastos, aumentar la eficiencia y aprovechar las innovaciones. Para más información, consulte Strategy for modernizing applications in the Nube de AWS.

evaluación de la preparación para la modernización

Evaluación que ayuda a determinar la preparación para la modernización de las aplicaciones de una organización; identifica los beneficios, los riesgos y las dependencias; y determina qué tan bien la organización puede soportar el estado futuro de esas aplicaciones. El resultado de la evaluación es un esquema de la arquitectura objetivo, una hoja de ruta que detalla las fases de desarrollo y los hitos del proceso de modernización y un plan de acción para abordar las brechas identificadas. Para más información, consulte Evaluating modernization readiness for applications in the Nube de AWS.

aplicaciones monolíticas (monolitos)

Aplicaciones que se ejecutan como un único servicio con procesos estrechamente acoplados. Las aplicaciones monolíticas presentan varios inconvenientes. Si una característica de la aplicación experimenta un aumento en la demanda, se debe escalar toda la arquitectura. Agregar o mejorar las características de una aplicación monolítica también se vuelve más complejo a medida que crece la base de código. Para solucionar problemas con la aplicación, puede utilizar

una arquitectura de microservicios. Para obtener más información, consulte Descomposición de monolitos en microservicios.

MPA

Consulte Migration Portfolio Assessment.

MQTT

Consulte Message Queuing Telemetry Transport.

clasificación multiclase

Un proceso que ayuda a generar predicciones para varias clases (predice uno de más de dos resultados). Por ejemplo, un modelo de ML podría preguntar "¿Este producto es un libro, un automóvil o un teléfono?" o "¿Qué categoría de productos es más interesante para este cliente?".

infraestructura mutable

Modelo que actualiza y modifica la infraestructura actual para las cargas de trabajo de producción. Para mejorar la coherencia, la confiabilidad y la previsibilidad, el AWS Well-Architected Marco recomienda el uso de una infraestructura inmutable como práctica recomendada.

O

OAC

Consulte control de acceso de origen.

OAI

Consulte identidad de acceso de origen.

OCM

Consulte administración del cambio organizacional.

migración fuera de línea

Método de migración en el que la carga de trabajo de origen se elimina durante el proceso de migración. Este método implica un tiempo de inactividad prolongado y, por lo general, se utiliza para cargas de trabajo pequeñas y no críticas.

OI

Consulte integración de operaciones.

OLA

Consulte acuerdo de nivel operativo.

migración en línea

Método de migración en el que la carga de trabajo de origen se copia al sistema de destino sin que se desconecte. Las aplicaciones que están conectadas a la carga de trabajo pueden seguir funcionando durante la migración. Este método implica un tiempo de inactividad nulo o mínimo y, por lo general, se utiliza para cargas de trabajo de producción críticas.

OPC-UA

Consulte Open Process Communications: arquitectura unificada.

Comunicaciones de proceso abierto: arquitectura unificada () OPC-UA

Un protocolo de comunicación de máquina a máquina (M2M) para la automatización industrial. OPC-UA proporciona un estándar de interoperabilidad con esquemas de cifrado, autenticación y autorización de datos.

acuerdo de nivel operativo (OLA)

Acuerdo que aclara lo que los grupos de TI operativos se comprometen a ofrecerse entre sí, para respaldar un acuerdo de nivel de servicio (SLA).

revisión de la preparación operativa (ORR)

Lista de comprobación de preguntas y prácticas recomendadas asociadas que son útiles para comprender, evaluar, prevenir o reducir el alcance de los incidentes y posibles errores. Para obtener más información, consulte las revisiones de preparación operativa (ORR) en el AWS Well-Architected marco.

tecnología operativa (TO)

Sistemas de hardware y software que funcionan con el entorno físico para controlar las operaciones, los equipos y la infraestructura industriales. En el sector de la fabricación, la integración de los sistemas de TO y tecnología de la información (TI) es un enfoque clave para las transformaciones de la industria 4.0.

integración de operaciones (OI)

Proceso de modernización de las operaciones en la nube, que implica la planificación de la preparación, la automatización y la integración. Para obtener más información, consulte la Guía de integración de las operaciones.

registro de seguimiento organizativo

Un registro creado por y AWS CloudTrail que registra todos los eventos Cuentas de AWS de una organización AWS Organizations. Este registro de seguimiento se crea en cada Cuenta de AWS que forma parte de la organización y realiza un seguimiento de la actividad en cada cuenta. Para obtener más información, consulte Crear un registro para una organización en la CloudTrail documentación.

administración del cambio organizacional (OCM)

Marco para administrar las transformaciones empresariales importantes y disruptivas desde la perspectiva de las personas, la cultura y el liderazgo. La OCM ayuda a las empresas a prepararse para nuevos sistemas y estrategias y a realizar la transición a ellos, al acelerar la adopción de cambios, abordar los problemas de transición e impulsar cambios culturales y organizacionales. En la estrategia de AWS migración, este marco se denomina aceleración de personal, debido a la velocidad de cambio que requieren los proyectos de adopción de la nube. Para obtener más información, consulte la Guía de OCM.

control de acceso de origen (OAC)

En CloudFront, una opción mejorada para restringir el acceso y proteger el contenido del Amazon Simple Storage Service (Amazon S3). El OAC admite todos los buckets de S3 Regiones de AWS, el cifrado del lado del servidor con AWS KMS (SSE-KMS) y `DELETE` las solicitudes `PUT` y dinámicas al bucket de S3.

identidad de acceso de origen (OAI)

En CloudFront, una opción para restringir el acceso y proteger el contenido de Amazon S3. Cuando utiliza OAI, CloudFront crea un principal con el que Amazon S3 puede autenticarse. Los directores autenticados solo pueden acceder al contenido de un bucket de S3 a través de una distribución específica. CloudFront Consulte también el OAC, que proporciona un control de acceso más detallado y mejorado.

ORR

Consulte revisión de la preparación operativa.

OT

Consulte tecnología operativa.

VPC saliente (de salida)

En una arquitectura de AWS cuentas múltiples, una VPC que gestiona las conexiones de red que se inician desde una aplicación. La Arquitectura de referencia de seguridad de AWS recomienda

configurar su cuenta de red con VPC entrantes, salientes y de inspección para proteger la interfaz bidireccional entre su aplicación e Internet en general.

P

límite de permisos

Una política de administración de IAM que se adjunta a las entidades principales de IAM para establecer los permisos máximos que puede tener el usuario o el rol. Para obtener más información, consulte Límites de permisos en la documentación de IAM.

información de identificación personal (PII)

Información que, vista directamente o combinada con otros datos relacionados, puede utilizarse para deducir de manera razonable la identidad de una persona. Algunos ejemplos de información de identificación personal son los nombres, las direcciones y la información de contacto.

PII

Consulte información de identificación personal.

manual de estrategias

Conjunto de pasos predefinidos que capturan el trabajo asociado a las migraciones, como la entrega de las funciones de operaciones principales en la nube. Un manual puede adoptar la forma de scripts, manuales de procedimientos automatizados o resúmenes de los procesos o pasos necesarios para operar un entorno modernizado.

PLC

Consulte controlador lógico programable.

PLM

Consulte administración del ciclo de vida del producto.

policy

Objeto que puede definir permisos (consulte política basada en identidad), especificar las condiciones de acceso (consulte política basada en recursos) o definir los permisos máximos para todas las cuentas de una organización de AWS Organizations (consulte política de control de servicio).

persistencia políglota

Elegir de forma independiente la tecnología de almacenamiento de datos de un microservicio en función de los patrones de acceso a los datos y otros requisitos. Si sus microservicios tienen la misma tecnología de almacenamiento de datos, pueden enfrentarse a desafíos de implementación o experimentar un rendimiento deficiente. Los microservicios se implementan más fácilmente y logran un mejor rendimiento y escalabilidad si utilizan el almacén de datos que mejor se adapte a sus necesidades.

evaluación de cartera

Proceso de detección, análisis y priorización de la cartera de aplicaciones para planificar la migración. Para obtener más información, consulte la Evaluación de la preparación para la migración.

predicate

Condición de consulta que devuelve `true` o `false`. En general, se encuentra en una cláusula `WHERE`.

inserción de predicados

Técnica de optimización de consultas en bases de datos que filtra los datos de la consulta antes de transferirlos. Esta técnica reduce la cantidad de datos de la base de datos relacional que se tienen que recuperar y procesar. Además, mejora el rendimiento de las consultas.

control preventivo

Un control de seguridad diseñado para evitar que ocurra un evento. Estos controles son la primera línea de defensa para evitar el acceso no autorizado o los cambios no deseados en la red. Para obtener más información, consulte Controles preventivos en Implementación de controles de seguridad en AWS.

entidad principal

Una entidad AWS que puede realizar acciones y acceder a los recursos. Esta entidad suele ser un usuario raíz para un Cuenta de AWS rol de IAM o un usuario. Para obtener más información, consulte Entidad principal en Términos y conceptos de roles en la documentación de IAM.

Privacidad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la privacidad durante todo el proceso de desarrollo.

zonas alojadas privadas

Contenedor que aloja información acerca de cómo desea que responda Amazon Route 53 a las consultas de DNS de un dominio y sus subdominios en una o varias VPC. Para obtener más información, consulte Uso de zonas alojadas privadas en la documentación de Route 53.

control proactivo

Control de seguridad que se diseñó para evitar la implementación de recursos que no cumplan con la normativa. Estos controles analizan los recursos antes de aprovisionarlos. Si el recurso no cumple con los requisitos del control, no se aprovisiona. Para obtener más información, consulte la guía de referencia de controles en la AWS Control Tower documentación y consulte Controles proactivos en Implementación de controles de seguridad en AWS.

administración del ciclo de vida del producto (PLM)

Administración de los datos y los procesos de un producto a lo largo de todo su ciclo de vida, desde el diseño, el desarrollo y el lanzamiento, pasando por el crecimiento y la madurez, hasta la reducción de su uso y su retirada.

entorno de producción

Consulte entorno.

controlador lógico programable (PLC)

En el sector de la fabricación, computadora adaptable y altamente fiable que supervisa las máquinas y automatiza los procesos de fabricación.

encadenamiento de peticiones

Uso de la salida de una petición de LLM como entrada para la siguiente petición a fin de generar mejores respuestas. Esta técnica se utiliza para dividir una tarea compleja en tareas secundarias o para refinar o ampliar de forma iterativa una respuesta preliminar. Ayuda a mejorar la precisión y la relevancia de las respuestas de un modelo y permite obtener resultados más detallados y personalizados.

seudonimización

El proceso de reemplazar los identificadores personales de un conjunto de datos por valores de marcadores de posición. La seudonimización puede ayudar a proteger la privacidad personal. Los datos seudonimizados siguen considerándose datos personales.

publish/subscribe (pub/sub)

Patrón que permite establecer comunicaciones asíncronas entre microservicios para mejorar la escalabilidad y la capacidad de respuesta. Por ejemplo, en un MES basado en microservicios, un microservicio puede publicar mensajes de eventos en un canal al que se pueden suscribir otros microservicios. El sistema puede agregar nuevos microservicios sin cambiar el servicio de publicación.

Q

plan de consulta

Serie de pasos, como instrucciones, que se utilizan para acceder a los datos de un sistema de base de datos relacional SQL.

regresión del plan de consulta

El optimizador de servicios de la base de datos elige un plan menos óptimo que antes de un cambio determinado en el entorno de la base de datos. Los cambios en estadísticas, restricciones, configuración del entorno, enlaces de parámetros de consultas y actualizaciones del motor de base de datos PostgreSQL pueden provocar una regresión del plan.

R

Matriz RACI

Consulte responsable, fiable, consultada e informada (RACI).

RAG

Consulte generación aumentada por recuperación.

ransomware

Software malicioso que se ha diseñado para bloquear el acceso a un sistema informático o a los datos hasta que se efectúe un pago.

Matriz RASCI

Consulte responsable, fiable, consultada e informada (RACI).

RCAC

Consulte control de acceso por filas y columnas.

réplica de lectura

Una copia de una base de datos que se utiliza con fines de solo lectura. Puede enrutar las consultas a la réplica de lectura para reducir la carga en la base de datos principal.

rediseñar

Consulte Las 7 R.

objetivo de punto de recuperación (RPO)

La cantidad de tiempo máximo aceptable desde el último punto de recuperación de datos. Esto determina qué se considera una pérdida de datos aceptable entre el último punto de recuperación y la interrupción del servicio.

objetivo de tiempo de recuperación (RTO)

La demora máxima aceptable entre la interrupción del servicio y el restablecimiento del servicio.

refactorizar

Consulte Las 7 R.

Region

Conjunto de AWS recursos en un área geográfica. Cada uno Región de AWS está aislado e independiente de los demás para proporcionar tolerancia a las fallas, estabilidad y resiliencia. Para más información, consulte Specify which Regiones de AWS your account can use.

regresión

Una técnica de ML que predice un valor numérico. Por ejemplo, para resolver el problema de "¿A qué precio se venderá esta casa?", un modelo de ML podría utilizar un modelo de regresión lineal para predecir el precio de venta de una vivienda en función de datos conocidos sobre ella (por ejemplo, los metros cuadrados).

volver a alojar

Consulte Las 7 R.

versión

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

reubicar

Consulte Las 7 R.

redefinir la plataforma

Consulte Las 7 R.

recomprar

Consulte Las 7 R.

resiliencia

Capacidad de una aplicación para resistir interrupciones o recuperarse de ellas. Al planificar la resiliencia en la Nube de AWS, la alta disponibilidad y la recuperación ante desastres son consideraciones comunes. Para más información, consulte Resiliencia en la Nube de AWS.

política basada en recursos

Una política asociada a un recurso, como un bucket de Amazon S3, un punto de conexión o una clave de cifrado. Este tipo de política especifica a qué entidades principales se les permite el acceso, las acciones compatibles y cualquier otra condición que deba cumplirse.

matriz responsable, confiable, consultada e informada (RACI)

Una matriz que define las funciones y responsabilidades de todas las partes involucradas en las actividades de migración y las operaciones de la nube. El nombre de la matriz se deriva de los tipos de responsabilidad definidos en la matriz: responsable (R), contable (A), consultado (C) e informado (I). El tipo de soporte (S) es opcional. Si incluye el soporte, la matriz se denomina matriz RASCI y, si la excluye, se denomina matriz RACI.

control receptivo

Un control de seguridad que se ha diseñado para corregir los eventos adversos o las desviaciones con respecto a su base de seguridad. Para obtener más información, consulte Controles receptivos en Implementación de controles de seguridad en AWS.

retain

Consulte Las 7 R.

retirar

Consulte Las 7 R.

Generación aumentada de recuperación (RAG)

Tecnología de IA generativa mediante la que un LLM hace referencia a un origen de datos autorizado que se encuentra fuera de sus orígenes de datos de entrenamiento antes de generar una respuesta. Por ejemplo, un modelo de RAG podría hacer una búsqueda semántica en la base de conocimientos o en los datos personalizados de una organización. Para más información, consulte ¿Qué es RAG (generación aumentada por recuperación)?

rotación

Proceso mediante el que periódicamente se actualiza un secreto para que resulte más difícil que un atacante pueda acceder a las credenciales.

control de acceso por filas y columnas (RCAC)

El uso de expresiones SQL básicas y flexibles que tienen reglas de acceso definidas. El RCAC consta de permisos de fila y máscaras de columnas.

RPO

Consulte objetivo de punto de recuperación.

RTO

Consulte objetivo de tiempo de recuperación.

manual de procedimientos

Conjunto de procedimientos manuales o automatizados necesarios para realizar una tarea específica. Por lo general, se diseñan para agilizar las operaciones o los procedimientos repetitivos con altas tasas de error.

S

SAML 2.0

Un estándar abierto que utilizan muchos proveedores de identidad (IdPs). Esta función permite el inicio de sesión único (SSO) federado, de modo que los usuarios pueden iniciar sesión Consola de administración de AWS o llamar a las operaciones de la AWS API sin tener que crear un usuario en IAM para todos los miembros de la organización. Para obtener más información sobre la federación basada en SAML 2.0, consulte Acerca de la federación basada en SAML 2.0 en la documentación de IAM.

SCADA

Consulte control de supervisión y adquisición de datos.

SCP

Consulte política de control de servicio.

secreta

En AWS Secrets Manager, información confidencial o restringida, como una contraseña o credenciales de usuario, que se almacena de forma cifrada. Se compone del valor del secreto y de sus metadatos. El valor del secreto puede ser binario, una sola cadena o varias cadenas. Para más información, consulte What's in a Secrets Manager secret? en la documentación de Secrets Manager.

seguridad desde el diseño

Enfoque de ingeniería de sistemas que tiene en cuenta la seguridad durante todo el proceso de desarrollo.

control de seguridad

Barrera de protección técnica o administrativa que impide, detecta o reduce la capacidad de un agente de amenazas para aprovechar una vulnerabilidad de seguridad. Existen cuatro tipos de controles de seguridad principales: preventivos, de detección, de respuesta y proactivos.

refuerzo de la seguridad

Proceso de reducir la superficie expuesta a ataques para hacerla más resistente a los ataques. Esto puede incluir acciones, como la eliminación de los recursos que ya no se necesitan, la implementación de prácticas recomendadas de seguridad consistente en conceder privilegios mínimos o la desactivación de características innecesarias en los archivos de configuración.

sistema de información sobre seguridad y administración de eventos (SIEM)

Herramientas y servicios que combinan sistemas de administración de información sobre seguridad (SIM) y de administración de eventos de seguridad (SEM). Un sistema de SIEM recopila, monitorea y analiza los datos de servidores, redes, dispositivos y otras fuentes para detectar amenazas y brechas de seguridad y generar alertas.

automatización de la respuesta de seguridad

Acción predefinida y programada que está diseñada para responder automáticamente a un evento de seguridad o corregirlo. Estas automatizaciones sirven como controles de seguridad

preventivos o adaptables que le ayudan a implementar las mejores prácticas AWS de seguridad. La modificación de un grupo de seguridad de VPC, la aplicación de revisiones a una instancia de Amazon EC2 o la rotación de credenciales son algunos ejemplos de acciones de respuesta automatizadas.

cifrado del servidor

Cifrado de los datos en su destino, por parte de Servicio de AWS quien los recibe.

política de control de servicio (SCP)

Una política que proporciona un control centralizado de los permisos de todas las cuentas de una organización en AWS Organizations. Las SCP definen barreras de protección o establecen límites a las acciones que un administrador puede delegar en los usuarios o roles. Puede utilizar las SCP como listas de permitidos o rechazados, para especificar qué servicios o acciones se encuentra permitidos o prohibidos. Para obtener más información, consulte las políticas de control de servicios en la AWS Organizations documentación.

punto de enlace de servicio

La URL del punto de entrada de un Servicio de AWS. Para conectarse mediante programación a un servicio de destino, puede utilizar un punto de conexión. Para obtener más información, consulte Puntos de conexión de Servicio de AWS en Referencia general de AWS.

acuerdo de nivel de servicio (SLA)

Acuerdo que aclara lo que un equipo de TI se compromete a ofrecer a los clientes, como el tiempo de actividad y el rendimiento del servicio.

indicador de nivel de servicio (SLI)

Medición de un aspecto del rendimiento de un servicio, como la tasa de errores, la disponibilidad o el rendimiento.

objetivo de nivel de servicio (SLO)

Métrica objetivo que representa el estado de un servicio medido mediante un indicador de nivel de servicio.

modelo de responsabilidad compartida

Un modelo que describe la responsabilidad con AWS la que compartes la seguridad y el cumplimiento de la nube. AWS es responsable de la seguridad de la nube, mientras que usted es responsable de la seguridad en la nube. Para obtener más información, consulte el Modelo de responsabilidad compartida.

Shadow AI

Aplicaciones de IA no autorizadas creadas o utilizadas fuera de los canales regulados dentro de una organización.

SIEM

Consulte sistema de administración de eventos e información de seguridad.

único punto de error (SPOF)

Error en un único componente crítico de una aplicación que puede interrumpir el sistema.

SLA

Consulte acuerdo de nivel de servicio.

SLI

Consulte indicador de nivel de servicio.

SLO

Consulte objetivo de nivel de servicio.

modelo de dividir y sembrar

Un patrón para escalar y acelerar los proyectos de modernización. A medida que se definen las nuevas funciones y los lanzamientos de los productos, el equipo principal se divide para crear nuevos equipos de productos. Esto ayuda a ampliar las capacidades y los servicios de su organización, mejora la productividad de los desarrolladores y apoya la innovación rápida. Para más información, consulte Phased approach to modernizing applications in the Nube de AWS.

SPOF

Consulte único punto de error.

esquema en estrella

Estructura organizativa de una base de datos que utiliza una tabla de hechos de gran tamaño para almacenar datos transaccionales o medidos y una o varias tablas dimensionales más pequeñas para almacenar los atributos de los datos. Esta estructura está diseñada para utilizarse en un almacén de datos o con fines de inteligencia empresarial.

patrón de higo estrangulador

Un enfoque para modernizar los sistemas monolíticos mediante la reescritura y el reemplazo gradual de las funciones del sistema hasta que se pueda desmantelar el sistema heredado.

Este patrón utiliza la analogía de una higuera que crece hasta convertirse en un árbol estable y, finalmente, se apodera y reemplaza a su host. El patrón fue presentado por Martin Fowler como una forma de gestionar el riesgo al reescribir sistemas monolíticos. Para ver un ejemplo de cómo aplicar este patrón, consulte Modernización gradual de los servicios web antiguos de Microsoft ASP.NET (ASMX) mediante contenedores y Amazon API Gateway.

subred

Un intervalo de direcciones IP en la VPC. Una subred debe residir en una sola zona de disponibilidad.

control de supervisión y adquisición de datos (SCADA)

En el sector de la fabricación, sistema que utiliza hardware y software para supervisar los activos físicos y las operaciones de producción.

cifrado simétrico

Un algoritmo de cifrado que utiliza la misma clave para cifrar y descifrar los datos.

pruebas sintéticas

Prueba de un sistema de manera que simule las interacciones de los usuarios para detectar posibles problemas o supervisar el rendimiento. Puede usar Amazon CloudWatch Synthetics para crear estas pruebas.

petición del sistema

Técnica para proporcionar contexto, instrucciones o pautas a un LLM para dirigir su comportamiento. Las peticiones del sistema ayudan a establecer el contexto y las reglas para las interacciones con los usuarios.

T

etiquetas

Key-value pares que actúan como metadatos para organizar sus AWS recursos. Las etiquetas pueden ayudar a administrar, identificar, organizar, buscar y filtrar recursos de . Para obtener más información, consulte Etiquetado de los recursos de AWS.

variable de destino

El valor que intenta predecir en el ML supervisado. Esto también se conoce como variable de resultado. Por ejemplo, en un entorno de fabricación, la variable objetivo podría ser un defecto del producto.

lista de tareas

Herramienta que se utiliza para hacer un seguimiento del progreso mediante un manual de procedimientos. La lista de tareas contiene una descripción general del manual de procedimientos y una lista de las tareas generales que deben completarse. Para cada tarea general, se incluye la cantidad estimada de tiempo necesario, el propietario y el progreso.

entorno de prueba

Consulte entorno.

entrenamiento

Proporcionar datos de los que pueda aprender su modelo de ML. Los datos de entrenamiento deben contener la respuesta correcta. El algoritmo de aprendizaje encuentra patrones en los datos de entrenamiento que asignan los atributos de los datos de entrada al destino (la respuesta que desea predecir). Genera un modelo de ML que captura estos patrones. Luego, el modelo de ML se puede utilizar para obtener predicciones sobre datos nuevos para los que no se conoce el destino.

herramienta

Una función o API que un agente puede invocar para realizar operaciones en sistemas externos.

puerta de enlace de tránsito

Centro de tránsito de red que puede utilizar para interconectar las VPC y las redes en las instalaciones. Para obtener más información, consulte Qué es una pasarela de tránsito en la AWS Transit Gateway documentación.

flujo de trabajo basado en enlaces troncales

Un enfoque en el que los desarrolladores crean y prueban características de forma local en una rama de característica 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.

acceso de confianza

Otorgar permisos a un servicio que especifique para realizar tareas en su organización AWS Organizations y en sus cuentas en su nombre. El servicio de confianza crea un rol vinculado al servicio en cada cuenta, cuando ese rol es necesario, para realizar las tareas de administración por usted. Para obtener más información, consulte AWS Organizations Utilización con otros AWS servicios en la AWS Organizations documentación.

ajuste

Cambiar aspectos de su proceso de formación a fin de mejorar la precisión del modelo de ML. Por ejemplo, puede entrenar el modelo de ML al generar un conjunto de etiquetas, incorporar etiquetas y, luego, repetir estos pasos varias veces con diferentes ajustes para optimizar el modelo.

equipo de dos pizzas

Un DevOps equipo pequeño al que puedes alimentar con dos pizzas. Un equipo formado por dos integrantes garantiza la mejor oportunidad posible de colaboración en el desarrollo de software.

U

incertidumbre

Un concepto que hace referencia a información imprecisa, incompleta o desconocida que puede socavar la fiabilidad de los modelos predictivos de ML. Hay dos tipos de incertidumbre: la incertidumbre epistémica se debe a datos limitados e incompletos, mientras que la incertidumbre aleatoria se debe al ruido y la aleatoriedad inherentes a los datos.

tareas indiferenciadas

También conocido como tareas arduas, es el trabajo que es necesario para crear y operar una aplicación, pero que no proporciona un valor directo al usuario final ni proporciona una ventaja competitiva. Algunos ejemplos de tareas indiferenciadas son la adquisición, el mantenimiento y la planificación de la capacidad.

entornos superiores

Consulte entorno.

V

succión

Una operación de mantenimiento de bases de datos que implica limpiar después de las actualizaciones incrementales para recuperar espacio de almacenamiento y mejorar el rendimiento.

control de versión

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

Emparejamiento de VPC

Conexión entre dos VPC que permite enrutar el tráfico mediante direcciones IP privadas. Para obtener más información, consulte ¿Qué es una interconexión de VPC? en la documentación de Amazon VPC.

vulnerabilidad

Defecto de software o hardware que pone en peligro la seguridad del sistema.

W

caché caliente

Un búfer caché que contiene datos actuales y relevantes a los que se accede con frecuencia. La instancia de base de datos puede leer desde la caché del búfer, lo que es más rápido que leer desde la memoria principal o el disco.

datos templados

Datos a los que el acceso es infrecuente. Al consultar este tipo de datos, normalmente se aceptan consultas moderadamente lentas.

función de ventana

Función SQL que hace un cálculo en un grupo de filas que se relacionan de alguna manera con el registro actual. Las funciones de ventana son útiles para las tareas de procesamiento, como calcular una media móvil o acceder al valor de las filas en función de la posición relativa de la fila actual.

carga de trabajo

Conjunto de recursos y código que ofrece valor comercial, como una aplicación orientada al cliente o un proceso de backend.

flujo de trabajo

Grupos funcionales de un proyecto de migración que son responsables de un conjunto específico de tareas. Cada flujo de trabajo es independiente, pero respalda a los demás flujos de trabajo del proyecto. Por ejemplo, el flujo de trabajo de la cartera es responsable de priorizar las aplicaciones, planificar las oleadas y recopilar los metadatos de migración. El flujo de trabajo de la cartera entrega estos recursos al flujo de trabajo de migración, que luego migra los servidores y las aplicaciones.

WORM

Consulte escritura única y lectura múltiple.

WQF

Consulte AWS Workload Qualification Framework.

escritura única y lectura múltiple (WORM)

Modelo de almacenamiento que escribe los datos una sola vez y evita que se eliminen o modifiquen. Los usuarios autorizados pueden leer los datos tantas veces como sea necesario, pero no los pueden cambiar. Esta infraestructura de almacenamiento de datos se considera inmutable.

Z

ataque de día cero

Ataque, normalmente de malware, que se aprovecha de una vulnerabilidad de día cero.

vulnerabilidad de día cero

Un defecto o una vulnerabilidad sin mitigación en un sistema de producción. Los agentes de amenazas pueden usar este tipo de vulnerabilidad para atacar el sistema. Los desarrolladores suelen darse cuenta de la vulnerabilidad a raíz del ataque.

peticiones desde cero

Proporcionar a un LLM instrucciones para llevar a cabo una tarea, pero sin ejemplos (pasos) que puedan ayudar a guiarlo. El LLM debe usar los conocimientos del entrenamiento previo para

llevar a cabo la tarea. La eficacia de la petición desde cero depende de la complejidad de la tarea y de la calidad de la petición. Consulte también peticiones con pocos pasos.

aplicación zombi

Aplicación que utiliza un promedio de CPU y memoria menor al 5 por ciento. En un proyecto de migración, es habitual retirar estas aplicaciones.

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.