Elegir una estrategia de ramificación de Git para entornos de múltiples cuentas DevOps

AWS Recomendaciones de

AWS Recomendaciones de: Elegir una estrategia de ramificación de Git para entornos de múltiples cuentas DevOps

Table of Contents

Introducción 1
Objetivos 1
Uso de prácticas de CI/CD 2
Comprensión de los DevOps entornos 4
Entorno sandbox 5
Acceso 5
Pasos de construcción 5
Pasos de implementación 5
Expectativas antes de pasar al entorno de desarrollo 6
Entorno de desarrollo 6
Acceso 5
Pasos de construcción 5
Pasos de implementación 5
Expectativas antes de pasar al entorno de pruebas 7
Entorno de pruebas 8
Acceso 5
Pasos de construcción 5
Pasos de implementación 5
Expectativas antes de pasar al entorno de ensayo 9
Entorno de puesta en escena 9
Acceso 5
Pasos de construcción 5
Pasos de implementación 5
Expectativas antes de pasar al entorno de producción 10
Entorno de producción 10
Acceso 5
Pasos de construcción 5
Pasos de implementación 5
Mejores prácticas para el desarrollo basado en Git 12
Estrategias de ramificación de Git 14
Estrategia de ramificación troncal 14
Descripción visual de la estrategia de Trunk 15
Estrategia de ramas en un tronco 16
Ventajas y desventajas de la estrategia Trunk 18

GitHub Estrategia de ramificación de flujos ... 21
Descripción visual de la estrategia GitHub Flow ... 21
Las ramas en una estrategia de GitHub flujo ... 22
Ventajas y desventajas de la estrategia GitHub Flow ... 24
Estrategia de ramificación de Gitflow ... 27
Descripción visual de la estrategia de Gitflow ... 28
Las ramificaciones en una estrategia de Gitflow ... 30
Ventajas y desventajas de la estrategia de Gitflow ... 34
Siguientes pasos ... 36
Recursos ... 37
AWS Guía prescriptiva ... 37
Otra orientación AWS ... 37
Otros recursos ... 37
Colaboradores ... 39
Creación ... 39
Revisando ... 39
Redacción técnica ... 39
Historial de documentos ... 40
Glosario ... 41
... 41
A ... 42
B ... 45
C ... 47
D ... 50
E ... 55
F ... 57
G ... 58
H ... 59
I ... 60
L ... 63
M ... 64
O ... 68
P ... 71
Q ... 74
R ... 74
S ... 77

T 81
U 82
V 83
W 83
Z 85
..... lxxxvi

Elegir una estrategia de ramificación de Git para entornos de múltiples cuentas DevOps

Amazon Web Services (colaboradores)

Febrero de 2024 (historial del documento)

Pasar a un enfoque basado en la nube y ofrecer soluciones de software AWS puede ser transformador. Es posible que requiera cambios en el proceso del ciclo de vida del desarrollo de software. Por lo general, Cuentas de AWS se utilizan varios durante el proceso de desarrollo en el Nube de AWS. Elegir una estrategia de ramificación de Git compatible que combine con tus DevOps procesos es esencial para el éxito. Elegir la estrategia de ramificación de Git adecuada para tu organización te ayuda a comunicar de forma concisa DevOps los estándares y las mejores prácticas entre los equipos de desarrollo. La ramificación de Git puede ser sencilla en un solo entorno, pero puede resultar confusa cuando se aplica en varios entornos, como entornos sandbox, de desarrollo, de pruebas, de puesta en escena y de producción. Tener varios entornos aumenta la complejidad de la implementación. DevOps

Esta guía proporciona diagramas visuales de las estrategias de ramificación de Git que muestran cómo una organización puede implementar un proceso de múltiples cuentas DevOps . Las guías visuales ayudan a los equipos a entender cómo combinar sus estrategias de ramificación de Git con sus DevOps prácticas. El uso de un modelo de ramificación estándar, como GitHub Gitflow, Flow o Trunk, para gestionar el repositorio de código fuente ayuda a los equipos de desarrollo a alinear su trabajo. Estos equipos también pueden utilizar los recursos de formación estándar de Git en Internet para comprender e implementar esos modelos y estrategias.

Para conocer las DevOps mejores prácticas al respecto AWS, consulte la DevOpsGuía de AWS Well-Architected. Al revisar esta guía, utilice la diligencia debida para seleccionar la estrategia de ramificación adecuada para su organización. Es posible que algunas estrategias se adapten mejor a su caso de uso que otras.

Objetivos

Esta guía forma parte de una serie de documentos sobre cómo elegir e implementar estrategias de DevOps ramificación para organizaciones con múltiples Cuentas de AWS sucursales. Esta serie está diseñada para ayudarlo a aplicar la estrategia que mejor se adapte a sus requisitos, objetivos y mejores prácticas desde el principio, a fin de optimizar su experiencia en la Nube de AWS industria.

Esta guía no contiene scripts DevOps ejecutables porque varían según el motor de integración y entrega continuas (CI/CD) y los marcos tecnológicos que utilice su organización.

Esta guía explica las diferencias entre tres estrategias comunes de ramificación de Git: GitHub Flow, Gitflow y Trunk. Las recomendaciones de esta guía ayudan a los equipos a identificar una estrategia de ramificación que se ajuste a sus objetivos organizacionales. Tras revisar esta guía, deberías poder elegir una estrategia de ramificación para tu organización. Tras elegir una estrategia, puedes usar uno de los siguientes patrones para ayudarte a implementar esa estrategia con tus equipos de desarrollo:

- Implemente una estrategia de ramificación troncal para entornos con varias cuentas DevOps
- Implemente una estrategia de ramificación GitHub de Flow para entornos con varias cuentas DevOps
- Implementa una estrategia de ramificación de Gitflow para entornos con varias cuentas DevOps

Es importante tener en cuenta que lo que funciona para una organización, equipo o proyecto puede no ser adecuado para otras. La elección entre las estrategias de ramificación de Git depende de varios factores, como el tamaño del equipo, los requisitos del proyecto y el equilibrio deseado entre la colaboración, la frecuencia de integración y la gestión de las versiones.

Uso de prácticas de CI/CD

AWS recomienda implementar la integración y la entrega continuas (CI/CD), que son el proceso de automatización del ciclo de vida de las versiones de software. Automatiza gran parte o la totalidad de los DevOps procesos manuales que tradicionalmente se requieren para que el código nuevo pase del desarrollo a la producción. Una canalización de CI/CD abarca los entornos sandbox, de desarrollo, de pruebas, de puesta en escena y de producción. En cada entorno, la canalización de CI/CD proporciona cualquier infraestructura necesaria para implementar o probar el código. Mediante el uso de la CI/CD, los equipos de desarrollo pueden realizar cambios en el código que luego se prueban e implementan automáticamente. Las canalizaciones de CI/CD también proporcionan control y protección a los equipos de desarrollo. Imponen la coherencia, los estándares, las mejores prácticas y los niveles mínimos de aceptación para la aceptación y el despliegue de las funciones. Para obtener más información, consulte Practicar la integración continua y la entrega continua en AWS.

Todas las estrategias de ramificación analizadas en esta guía se adaptan bien a las prácticas de CI/CD. La complejidad de la cartera de CI/CD aumenta con la complejidad de la estrategia de

ramificación. Por ejemplo, Gitflow es la estrategia de ramificación más compleja que se analiza en esta guía. Las canalizaciones de CI/CD para esta estrategia requieren más pasos (por ejemplo, por motivos de conformidad) y deben admitir múltiples versiones de producción simultáneas. El uso de la CI/CD también adquiere más importancia a medida que aumenta la complejidad de la estrategia de ramificación. Esto se debe a que la CI/CD establece barreras y mecanismos para los equipos de desarrollo que impiden que los desarrolladores eludan intencional o involuntariamente el proceso definido.

AWS ofrece un conjunto de servicios para desarrolladores diseñados para ayudarle a crear canalizaciones de CI/CD. Por ejemplo, AWS CodePipelinees un servicio de entrega continua totalmente gestionado que le ayuda a automatizar sus procesos de lanzamiento para obtener actualizaciones rápidas y fiables de las aplicaciones y la infraestructura. AWS CodeCommitestá diseñado para alojar de forma segura repositorios Git escalables y AWS CodeBuildcompila el código fuente, ejecuta pruebas y produce paquetes de ready-to-deploy software. Para obtener más información, consulte las Herramientas para desarrolladores en. AWS

Comprensión de los DevOps entornos

Para comprender las estrategias de ramificación, debe comprender el propósito y las actividades que se llevan a cabo en cada entorno. El establecimiento de varios entornos le ayuda a separar las actividades de desarrollo en etapas, a supervisarlas y a evitar el lanzamiento involuntario de funciones no aprobadas. Puede tener uno o más Cuentas de AWS en cada entorno.

La mayoría de las organizaciones tienen varios entornos descritos para su uso. Sin embargo, la cantidad de entornos puede variar según la organización y según las políticas de desarrollo de software. En esta serie de documentación se supone que tiene los siguientes cinco entornos comunes que abarcan su proceso de desarrollo, aunque es posible que tengan nombres diferentes:

- Sandbox: un entorno en el que los desarrolladores escriben código, cometen errores y realizan trabajos de prueba de concepto.
- Desarrollo: un entorno en el que los desarrolladores integran su código para confirmar que todo funciona como una aplicación única y cohesiva.
- Pruebas: un entorno en el que se llevan a cabo equipos de control de calidad o pruebas de aceptación. Los equipos suelen realizar pruebas de rendimiento o integración en este entorno.
- Preparación: un entorno de preproducción en el que se valida que el código y la infraestructura funcionan según lo esperado en circunstancias equivalentes a las de producción. Este entorno está configurado para ser lo más parecido posible al entorno de producción.
- Producción: un entorno que gestiona el tráfico de los usuarios finales y los clientes.

En esta sección se describe cada entorno en detalle. También describe los pasos de creación, los pasos de implementación y los criterios de salida de cada entorno para que pueda continuar con el siguiente. La siguiente imagen muestra estos entornos en secuencia.

Temas de esta sección:

- Entorno sandbox
- Entorno de desarrollo

- Entorno de pruebas
- Entorno de puesta en escena
- Entorno de producción

Entorno sandbox

El entorno sandbox es el lugar donde los desarrolladores escriben código, cometen errores y realizan trabajos de prueba de concepto. Puede realizar la implementación en un entorno sandbox desde una estación de trabajo local o mediante un script en una estación de trabajo local.

Acceso

Los desarrolladores deben tener acceso completo al entorno sandbox.

Pasos de construcción

Los desarrolladores ejecutan la compilación manualmente en sus estaciones de trabajo locales cuando están preparados para implementar cambios en el entorno sandbox.

1. Usa git-secrets (GitHub) para buscar información confidencial
2. Borra el código fuente
3. Compila y compila el código fuente, si corresponde
4. Realice pruebas unitarias
5. Realice un análisis de cobertura de código
6. Realizar un análisis de código estático
7. Cree la infraestructura como código (IaC)
8. Realice un análisis de seguridad de IaC
9. Extraiga licencias de código abierto
10. Publica artefactos de construcción

Pasos de implementación

Si utilizas los modelos Gitflow o Trunk, los pasos de implementación se inician automáticamente cuando una `feature` rama se crea correctamente en el entorno sandbox. Si utilizas el modelo

GitHub Flow, debes realizar manualmente los siguientes pasos de despliegue. Los siguientes son los pasos de implementación en el entorno sandbox:

1. Descargue los artefactos publicados
2. Realice el versionado de la base de datos
3. Realice el despliegue de iAC
4. Realice pruebas de integración

Expectativas antes de pasar al entorno de desarrollo

- Construcción exitosa de la `feature` sucursal en un entorno sandbox
- Un desarrollador implementó y probó la función manualmente en el entorno sandbox

Entorno de desarrollo

El entorno de desarrollo es donde los desarrolladores integran su código para garantizar que todo funcione como una aplicación cohesiva. En Gitflow, el entorno de desarrollo contiene las últimas funciones incluidas en la solicitud de fusión y están listas para su lanzamiento. En las estrategias GitHub Flow y Trunk, el entorno de desarrollo se considera un entorno de pruebas y el código base puede ser inestable e inadecuado para su despliegue en producción.

Acceso

Asigne los permisos de acuerdo con el principio de privilegios mínimos. El privilegio mínimo es la práctica recomendada de seguridad que consiste en conceder los permisos mínimos necesarios para realizar una tarea. Los desarrolladores deberían tener menos acceso al entorno de desarrollo que al entorno sandbox.

Pasos de construcción

Al crear una solicitud de fusión para la `develop` rama (Gitflow) o la `main` rama (Trunk o GitHub Flow), se inicia automáticamente la compilación.

1. Usa git-secrets (GitHub) para buscar información confidencial
2. Borra el código fuente

3. Compila y compila el código fuente, si corresponde
4. Realice pruebas unitarias
5. Realice un análisis de cobertura de código
6. Realizar un análisis de código estático
7. Construye iAC
8. Realice un análisis de seguridad de IaC
9. Extraiga licencias de código abierto

Pasos de implementación

Si utilizas el modelo de Gitflow, los pasos de despliegue se inician automáticamente cuando una `develop` rama se ha creado correctamente en el entorno de desarrollo. Si utilizas el modelo GitHub Flow o el modelo Trunk, los pasos de despliegue se inician automáticamente cuando se crea una solicitud de fusión en la `main` sucursal. Los siguientes son los pasos de implementación en el entorno de desarrollo:

1. Descargue los artefactos publicados desde los pasos de construcción
2. Realice el versionado de la base de datos
3. Realice el despliegue de iAC
4. Realice pruebas de integración

Expectativas antes de pasar al entorno de pruebas

- Creación e implementación satisfactorios de la `develop` rama (Gitflow) o la `main` sucursal (Trunk o GitHub Flow) en el entorno de desarrollo
- Las pruebas unitarias se aprueban al 100%
- Construcción exitosa de iAC
- Los artefactos de despliegue se crearon correctamente
- Un desarrollador ha realizado una verificación manual para confirmar que la función funciona según lo esperado

Entorno de pruebas

El personal de control de calidad (QA) utiliza el entorno de pruebas para validar las características. Aprueban los cambios una vez finalizadas las pruebas. Cuando lo aprueban, la sucursal pasa al siguiente entorno, el de puesta en escena. En Gitflow, este entorno y otros anteriores solo están disponibles para su implementación desde `release` sucursales. Una `release` sucursal se basa en una `develop` rama que contiene las funciones planificadas.

Acceso

Asigne los permisos de acuerdo con el principio de privilegios mínimos. Los desarrolladores deberían tener menos acceso al entorno de pruebas que al entorno de desarrollo. El personal de control de calidad necesita permisos suficientes para probar la función.

Pasos de construcción

El proceso de compilación en este entorno solo se aplica a las correcciones de errores cuando se utiliza la estrategia de Gitflow. Al crear una solicitud de fusión para la `bugfix` sucursal, se inicia automáticamente la compilación.

1. Usa git-secrets (GitHub) para buscar información confidencial
2. Borra el código fuente
3. Compila y compila el código fuente, si corresponde
4. Realice pruebas unitarias
5. Realice un análisis de cobertura de código
6. Realizar un análisis de código estático
7. Construye iAC
8. Realice un análisis de seguridad de IaC
9. Extraiga licencias de código abierto

Pasos de implementación

Inicie automáticamente el despliegue de la `release` sucursal (Gitflow) o de la `main` sucursal (Trunk o GitHub Flow) en el entorno de prueba tras el despliegue en el entorno de desarrollo. Los siguientes son los pasos de implementación en el entorno de pruebas:

1. Implemente la `release` rama (Gitflow) o la `main` rama (Trunk o GitHub Flow) en el entorno de prueba
2. Haga una pausa para la aprobación manual por parte del personal designado
3. Descarga los artefactos publicados
4. Realice el versionado de la base de datos
5. Realice el despliegue de iAC
6. Realice pruebas de integración
7. Realice pruebas de rendimiento
8. Aprobación de control de calidad

Expectativas antes de pasar al entorno de ensayo

- Los equipos de desarrollo y control de calidad han realizado suficientes pruebas para satisfacer los requisitos de su organización.
- El equipo de desarrollo ha resuelto los errores descubiertos a través de una `bugfix` sucursal.

Entorno de puesta en escena

El entorno de ensayo está configurado para ser el mismo que el entorno de producción. Por ejemplo, la configuración de los datos debe ser similar en alcance y tamaño a las cargas de trabajo de producción. Utilice el entorno de ensayo para comprobar que el código y la infraestructura funcionan según lo esperado. Este entorno también es la opción preferida para los casos de uso empresarial, como las vistas previas o las demostraciones para clientes.

Acceso

Asigne los permisos de acuerdo con el principio de privilegios mínimos. Los desarrolladores deben tener el mismo acceso al entorno de ensayo que al entorno de producción.

Pasos de construcción

Ninguna. Los mismos artefactos que se usaron en el entorno de prueba se reutilizan en el entorno de ensayo.

Pasos de implementación

Inicie automáticamente el despliegue de la `release` rama (Gitflow) o la `main` sucursal (Trunk o GitHub Flow) en el entorno de ensayo tras la aprobación y el despliegue en el entorno de prueba. Los siguientes son los pasos de implementación en el entorno provisional:

1. Implemente la `release` rama (Gitflow) o la `main` rama (Trunk o GitHub Flow) en el entorno de ensayo
2. Haga una pausa para la aprobación manual por parte del personal designado
3. Descarga los artefactos publicados
4. Realice el versionado de la base de datos
5. Realice el despliegue de iAC
6. (Opcional) Realice pruebas de integración
7. (Opcional) Realice pruebas de carga
8. Obtenga la aprobación de los responsables de desarrollo, control de calidad, productos o empresas necesarios

Expectativas antes de pasar al entorno de producción

- Se implementó correctamente una versión equivalente a la producción en el entorno de ensayo
- (Opcional) Las pruebas de integración y carga se realizaron correctamente

Entorno de producción

El entorno de producción respalda el producto lanzado y gestiona datos reales de clientes reales. Se trata de un entorno protegido al que se le asigna el acceso con el mínimo privilegio y el acceso elevado solo debe permitirse mediante un proceso de excepción auditado durante un período de tiempo limitado.

Acceso

En el entorno de producción, los desarrolladores deben tener acceso limitado y de solo lectura a la consola de administración de AWS. Por ejemplo, los desarrolladores deberían poder acceder a los datos de registro para day-to-day las operaciones. Todas las versiones para producción deben estar sujetas a un paso de aprobación antes de su implementación.

Pasos de construcción

Ninguna. Los mismos artefactos que se usaron en los entornos de prueba y puesta en escena se reutilizan en el entorno de producción.

Pasos de implementación

Inicie automáticamente el despliegue de la `release` sucursal (Gitflow) o la `main` sucursal (Trunk o GitHub Flow) en el entorno de producción tras la aprobación y el despliegue en el entorno provisional. Los siguientes son los pasos de implementación en el entorno de producción:

1. Implemente la `release` sucursal (Gitflow) o la `main` rama (Trunk o GitHub Flow) en el entorno de producción
2. Haga una pausa para la aprobación manual por parte del personal designado
3. Descarga los artefactos publicados
4. Realice el versionado de la base de datos
5. Realice el despliegue de iAC

Mejores prácticas para el desarrollo basado en Git

Para adoptar con éxito el desarrollo basado en Git, es importante seguir un conjunto de mejores prácticas que promuevan la colaboración, mantengan la calidad del código y respalden la integración y la entrega continuas (CI/CD). Además de las mejores prácticas de esta guía, consulte la Guía de AWS DevOps Well-Architected. Las siguientes son algunas de las mejores prácticas clave para el desarrollo basado en Git en AWS:

- Mantenga los cambios pequeños y frecuentes: anime a los desarrolladores a realizar cambios o funciones pequeños e incrementales. Esto reduce el riesgo de conflictos de fusión y facilita la identificación y la solución rápida de los problemas.
- Utilice los conmutadores de funciones: para gestionar la publicación de funciones incompletas o experimentales, utilice las opciones o los indicadores de funciones. Esto te ayuda a ocultar, activar o desactivar funciones específicas en producción sin que ello afecte a la estabilidad de la rama principal.
- Mantenga un conjunto de pruebas sólido: un conjunto de pruebas completo y bien mantenido es crucial para detectar los problemas de manera temprana y verificar que la base de código se mantenga estable. Invierta en la automatización de las pruebas y priorice la corrección de las pruebas fallidas.
- Adopte la integración continua: utilice herramientas y prácticas de integración continua para crear, probar e integrar automáticamente los cambios de código en la `develop` rama (Gitflow) o en la `main` rama (Trunk o GitHub Flow). Esto le ayuda a detectar los problemas de forma temprana y agiliza el proceso de desarrollo.
- Realice revisiones del código: fomente las revisiones del código por pares para mantener la calidad, compartir conocimientos y detectar posibles problemas antes de que se integren en la `main` rama. Usa solicitudes de extracción u otras herramientas de revisión de código para facilitar este proceso.
- Supervisa y corrige compilaciones defectuosas: cuando una compilación se interrumpe o las pruebas fallan, prioriza la solución del problema lo antes posible. Esto mantiene la `develop` rama (Gitflow) o la `main` rama (Trunk o GitHub Flow) en un estado liberable y minimiza el impacto en otros desarrolladores.
- Comunícate y colabora: promueve la comunicación abierta y la colaboración entre los miembros del equipo. Asegúrese de que los desarrolladores estén al tanto del trabajo en curso y de los cambios que se están realizando en el código base.

- Refactoriza de forma continua: refactoriza periódicamente el código base para mejorar su mantenibilidad y reducir la deuda técnica. Anime a los desarrolladores a dejar el código en un estado mejor del que lo encontraron.
- Usa ramas de corta duración para tareas complejas: para tareas más grandes o complejas, usa ramas de corta duración (también conocidas como ramas de tareas) para trabajar en los cambios. Sin embargo, asegúrate de que la vida útil de las ramas sea corta, normalmente menos de un día. Vuelve a combinar los cambios en la `develop` rama (Gitflow) o en la `main` rama (Trunk o GitHub Flow) lo antes posible. Las fusiones y revisiones más pequeñas y frecuentes son más fáciles de procesar para un equipo que una solicitud de fusión grande.
- Capacite y apoye al equipo: brinde capacitación y apoyo a los desarrolladores que son nuevos en el desarrollo basado en Git o que requieren orientación para adoptar sus mejores prácticas.

Estrategias de ramificación de Git

En orden de menor a mayor complejidad, esta guía describe en detalle las siguientes estrategias de ramificación basadas en Git:

- Troncal: el desarrollo basado en enlaces troncales es una práctica de desarrollo de software en la que todos los desarrolladores trabajan en una sola rama, normalmente denominada rama o. `trunk` `main` La idea detrás de este enfoque es mantener el código base en un estado de publicación continua integrando los cambios de código con frecuencia y confiando en las pruebas automatizadas y la integración continua.
- GitHub Flow: GitHub Flow es un flujo de trabajo ligero y basado en ramas que fue desarrollado por. GitHub Se basa en la idea de ramas de corta duración`feature`. Cuando una función está completa y lista para su implementación, la función se fusiona en la `main` rama.
- Gitflow: con un enfoque de Gitflow, el desarrollo se completa en ramas de funciones individuales. Tras la aprobación, se fusionan `feature` las ramas en una rama de integración que suele tener un nombre. `develop` Cuando se han acumulado suficientes funciones en la `develop` rama, se crea una `release` rama para implementar las funciones en los entornos superiores.

Cada estrategia de ramificación tiene ventajas y desventajas. Si bien todas utilizan los mismos entornos, no todas utilizan las mismas sucursales ni los mismos pasos de aprobación manual. En esta sección de la guía, revise cada estrategia de ramificación en detalle para familiarizarse con sus matices y poder evaluar si se ajusta al caso de uso de su organización.

Temas de esta sección:

- Estrategia de ramificación troncal
- GitHub Estrategia de ramificación de flujos
- Estrategia de ramificación de Gitflow

Estrategia de ramificación troncal

El desarrollo basado en enlaces troncales es una práctica de desarrollo de software en la que todos los desarrolladores trabajan en una sola rama, normalmente denominada rama o. `trunk` `main` La idea detrás de este enfoque es mantener el código base en un estado de publicación continua

integrando los cambios de código con frecuencia y confiando en las pruebas automatizadas y la integración continua.

En el desarrollo basado en enlaces troncales, los desarrolladores envían sus cambios a la `main` rama varias veces al día, con el objetivo de realizar actualizaciones pequeñas e incrementales. Esto permite ciclos de retroalimentación rápidos, reduce el riesgo de conflictos de fusión y fomenta la colaboración entre los miembros del equipo. La práctica enfatiza la importancia de un conjunto de pruebas bien mantenido porque se basa en pruebas automatizadas para detectar posibles problemas de manera temprana y garantizar que la base de código permanezca estable y liberable.

El desarrollo basado en enlaces troncales suele contrastarse con el desarrollo basado en funciones (también conocido como ramificación de funciones o desarrollo impulsado por funciones), en el que cada nueva función o corrección de errores se desarrolla en su propia rama dedicada, separada de la rama principal. La elección entre el desarrollo troncal y el desarrollo basado en funciones depende de factores como el tamaño del equipo, los requisitos del proyecto y el equilibrio deseado entre la colaboración, la frecuencia de integración y la gestión de las versiones.

Para obtener más información sobre la estrategia de ramificación de Trunk, consulta los siguientes recursos:

- Implemente una estrategia de ramificación troncal para DevOps entornos con varias cuentas (AWS orientación prescriptiva)
- Introducción al desarrollo basado en troncos (sitio web de desarrollo basado en troncos)

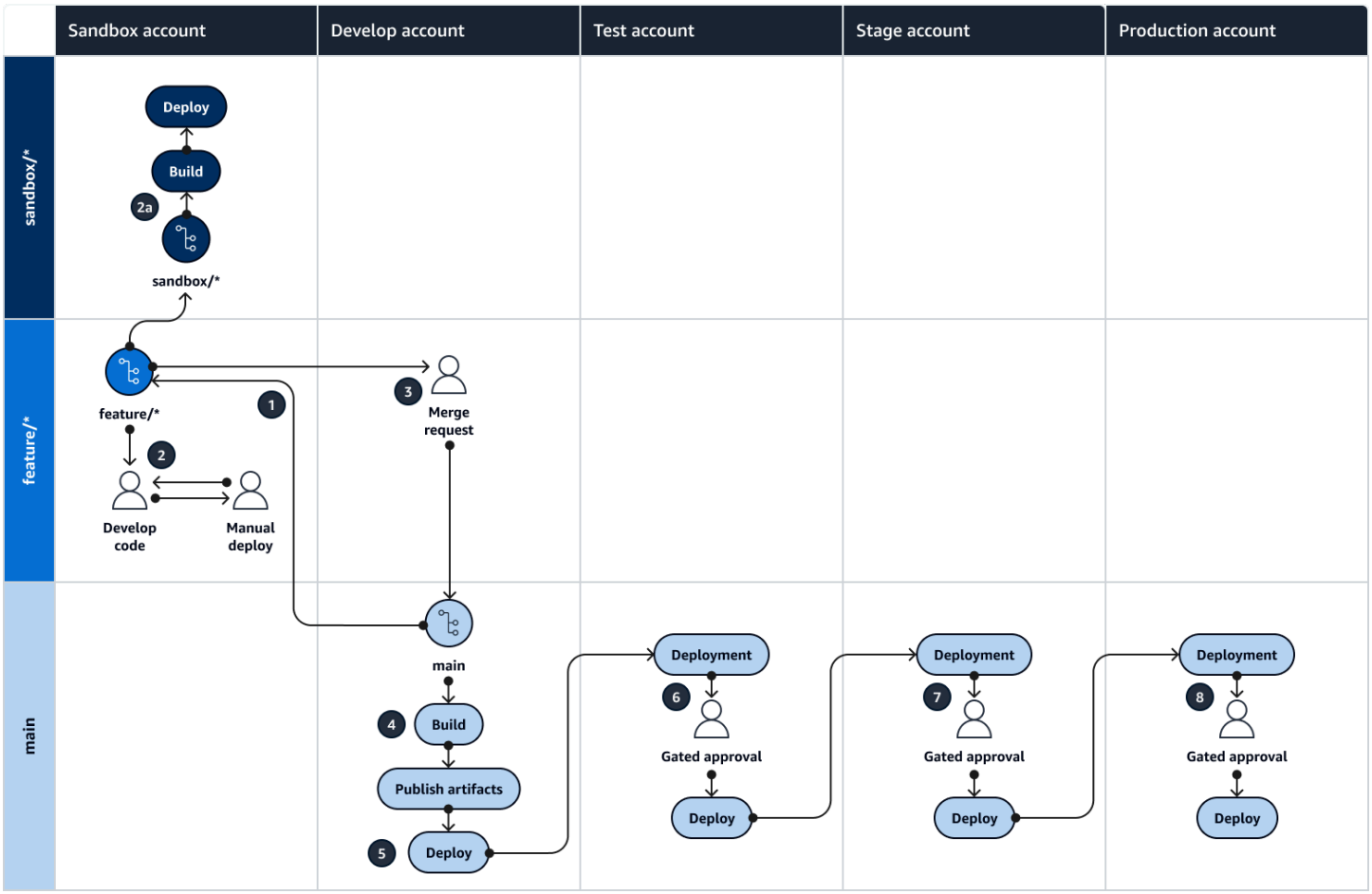
Temas de esta sección:

- Descripción visual de la estrategia de Trunk
- Estrategia de ramas en un tronco
- Ventajas y desventajas de la estrategia Trunk

Descripción visual de la estrategia de Trunk

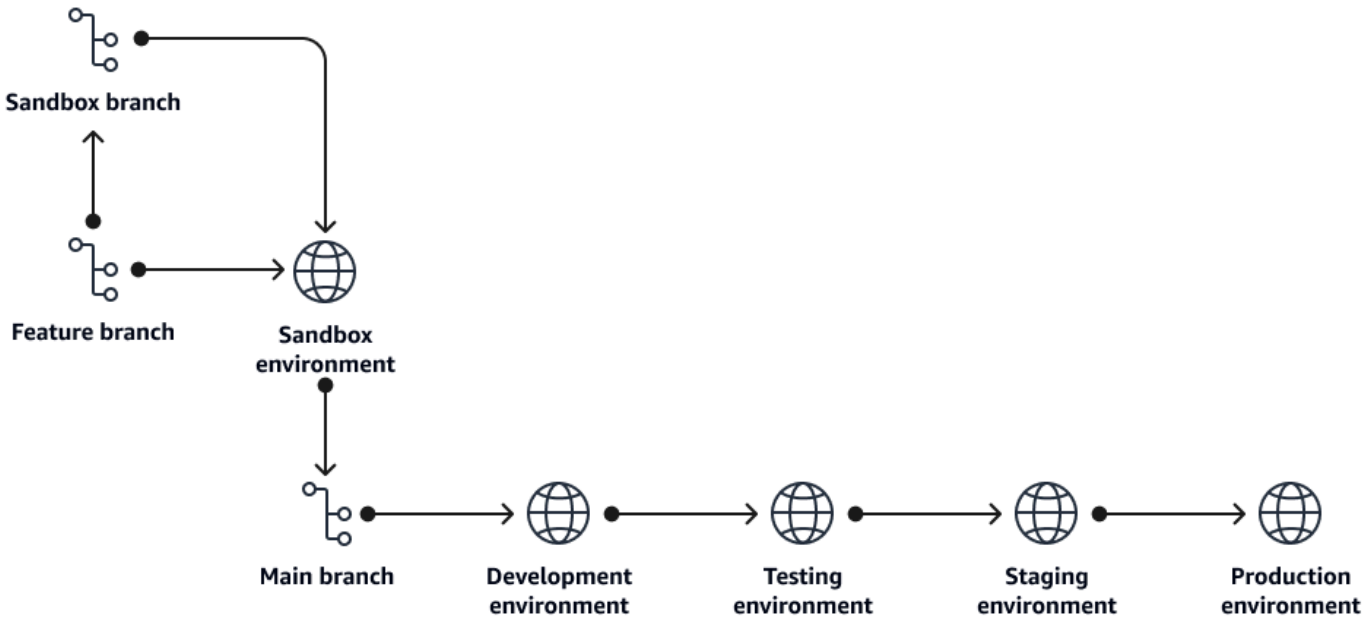
El siguiente diagrama se puede utilizar como un cuadro de Punnett (Wikipedia) para entender la estrategia de ramificación de Trunk. Alinee las ramas del eje vertical con los AWS entornos del eje horizontal para determinar qué acciones realizar en cada escenario. Los números encerrados en un círculo te guían por la secuencia de acciones representada en el diagrama. Este diagrama muestra el flujo de trabajo de desarrollo de una estrategia de ramificación troncal, desde una `feature`

sucursal en un entorno sandbox hasta la versión de producción de la rama. `main` Para obtener más información sobre las actividades que se producen en cada entorno, consulte DevOps los entornos en esta guía.



Estrategia de ramas en un tronco

Una estrategia de ramificación troncal suele tener las siguientes ramas.



rama de característica

Desarrollas funciones o creas una revisión en una `feature` rama. Para crear una `feature` rama, se ramifica desde la `main` rama. Los desarrolladores iteran, confirman y prueban el código de una `feature` rama. Cuando una función está completa, el desarrollador la promociona. Solo hay dos caminos hacia adelante desde una `feature` rama:

- Incorpórate a la `sandbox` rama
- Crea una solicitud de fusión en la `main` sucursal

Convención de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

rama sandbox

Esta rama es una rama troncal no estándar, pero es útil para el desarrollo de canalizaciones de CI/CD. La `sandbox` rama se utiliza principalmente para los siguientes fines:

- Realice un despliegue completo en el entorno sandbox mediante las canalizaciones de CI/CD
- Desarrolle y pruebe una canalización antes de enviar solicitudes de fusión para realizar pruebas completas en un entorno inferior, como el de desarrollo o las pruebas.

`Sandbox`las sucursales son de naturaleza temporal y están destinadas a ser de corta duración. Deben suprimirse una vez finalizadas las pruebas específicas.

Convención de nomenclatura:	`sandbox/<story number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`sandbox/123456_MS_Test_Pipeline_Deploy`

rama principal

La `main` rama siempre representa el código que se está ejecutando en producción. El código se ramifica, se desarrolla y`main`, a continuación, se fusiona nuevamente con`main`. Las implementaciones desde `main` pueden dirigirse a cualquier entorno. Para evitar que se eliminen, habilite la protección de sucursales para la `main` sucursal.

Convención de nomenclatura:	`main`

rama de hotfix

No hay una `hotfix` rama dedicada en un flujo de trabajo basado en enlaces troncales. Los hotfixes utilizan ramas. `feature`

Ventajas y desventajas de la estrategia Trunk

La estrategia de ramificación de Trunk es ideal para equipos de desarrollo más pequeños y maduros que tienen sólidas habilidades de comunicación. También funciona bien si tiene versiones continuas y continuas de funciones para la aplicación. No es adecuada si sus equipos de desarrollo son grandes o están fragmentados o si tiene programadas versiones de funciones amplias. En este modelo se producirán conflictos de fusión, así que tenga en cuenta que la resolución de conflictos

de fusión es una habilidad clave. Todos los miembros del equipo deben estar capacitados en consecuencia.

Ventajas

El desarrollo basado en enlaces troncales ofrece varias ventajas que pueden mejorar el proceso de desarrollo, agilizar la colaboración y mejorar la calidad general del software. Las siguientes son algunas de las principales ventajas:

- Bucles de retroalimentación más rápidos: con el desarrollo basado en enlaces troncales, los desarrolladores integran sus cambios de código con frecuencia, a menudo varias veces al día. Esto permite obtener información más rápida sobre posibles problemas y ayuda a los desarrolladores a identificar y solucionar los problemas con mayor rapidez de lo que lo harían en un modelo de desarrollo basado en funciones.
- Reducción de los conflictos de fusión: en el desarrollo basado en enlaces troncales, el riesgo de conflictos de fusión grandes y complicados se minimiza porque los cambios se integran de forma continua. Esto ayuda a mantener una base de código más limpia y reduce el tiempo dedicado a resolver conflictos. La resolución de conflictos puede llevar mucho tiempo y ser propensa a errores en el desarrollo basado en funciones.
- Colaboración mejorada: el desarrollo basado en Trunk alienta a los desarrolladores a trabajar juntos en la misma sucursal, lo que promueve una mejor comunicación y colaboración dentro del equipo. Esto puede conducir a una resolución de problemas más rápida y a una dinámica de equipo más cohesiva.
- Revisiones de código más sencillas: dado que los cambios de código son más pequeños y más frecuentes en el desarrollo basado en troncos, puede resultar más fácil realizar revisiones exhaustivas del código. Los cambios más pequeños suelen ser más fáciles de entender y revisar, lo que permite identificar de forma más eficaz los posibles problemas y mejoras.
- Integración y entrega continuas: el desarrollo basado en Trunk respalda los principios de integración y entrega continuas (CI/CD). Al mantener el código base en un estado que se pueda publicar e integrar los cambios con frecuencia, los equipos pueden adoptar con mayor facilidad las prácticas de CI/CD, lo que se traduce en ciclos de implementación más rápidos y en una mejor calidad del software.
- Calidad del código mejorada: con la integración, las pruebas y las revisiones del código frecuentes, el desarrollo basado en troncos puede contribuir a mejorar la calidad general del código. Los desarrolladores pueden detectar y solucionar los problemas con mayor rapidez, lo que reduce la probabilidad de que la deuda técnica se acumule con el tiempo.

- Estrategia de ramificación simplificada: el desarrollo basado en troncales simplifica la estrategia de ramificación al reducir el número de sucursales duraderas. Esto puede facilitar la administración y el mantenimiento del código base, especialmente para proyectos o equipos grandes.

Desventajas

El desarrollo basado en troncos tiene algunas desventajas, que pueden afectar al proceso de desarrollo y a la dinámica del equipo. Los siguientes son algunos inconvenientes notables:

- Aislamiento limitado: dado que todos los desarrolladores trabajan en la misma rama, todos los miembros del equipo pueden ver sus cambios de forma inmediata. Esto puede provocar interferencias o conflictos, provocar efectos secundarios no deseados o interrumpir la compilación. Por el contrario, el desarrollo basado en funciones aísla mejor los cambios para que los desarrolladores puedan trabajar de forma más independiente.
- Mayor presión sobre las pruebas: el desarrollo basado en Trunk se basa en la integración continua y las pruebas automatizadas para detectar los problemas rápidamente. Sin embargo, este enfoque puede ejercer una gran presión sobre la infraestructura de pruebas y requiere un conjunto de pruebas bien mantenido. Si las pruebas no son exhaustivas o fiables, pueden producirse problemas no detectados en la rama principal.
- Menor control sobre las versiones: el desarrollo basado en Trunk tiene como objetivo mantener el código base en un estado de publicación continua. Si bien esto puede ser ventajoso, puede que no siempre sea adecuado para proyectos con calendarios de lanzamiento estrictos o aquellos que requieren que funciones específicas se publiquen juntas. El desarrollo basado en funciones proporciona un mayor control sobre cuándo y cómo se lanzan las funciones.
- Pérdida de código: dado que los desarrolladores integran constantemente los cambios en la rama principal, el desarrollo basado en enlaces troncales puede provocar una mayor pérdida de código. Esto puede dificultar que los desarrolladores realicen un seguimiento del estado actual del código base y puede generar confusión al intentar comprender el efecto de los cambios recientes.
- Requiere una cultura de equipo sólida: el desarrollo basado en Trunk exige un alto nivel de disciplina, comunicación y colaboración entre los miembros del equipo. Esto puede ser difícil de mantener, especialmente en equipos más grandes o cuando se trabaja con desarrolladores que tienen menos experiencia con este enfoque.
- Retos de escalabilidad: a medida que crece el tamaño del equipo de desarrollo, la cantidad de cambios de código que se integran en la rama principal puede aumentar rápidamente. Esto puede

provocar interrupciones de compilación y errores en las pruebas más frecuentes, lo que dificulta mantener el código base en un estado que pueda publicarse.

GitHub Estrategia de ramificación de flujos

GitHub Flow es un flujo de trabajo ligero y basado en sucursales que fue desarrollado por. GitHub GitHubFlow se basa en la idea de ramificaciones de funciones de corta duración que se fusionan en la rama principal cuando la función está completa y lista para su implementación. Los principios clave de GitHub Flow son:

- La ramificación es ligera: los desarrolladores pueden crear ramas de funciones para su trabajo con solo unos pocos clics, lo que mejora la capacidad de colaborar y experimentar sin afectar a la rama principal.
- Despliegue continuo: los cambios se implementan tan pronto como se fusionan en la rama principal, lo que permite una rápida retroalimentación e iteración.
- Solicitudes de fusión: los desarrolladores utilizan las solicitudes de fusión para iniciar un proceso de discusión y revisión antes de fusionar sus cambios en la rama principal.

Para obtener más información sobre GitHub Flow, consulta los siguientes recursos:

- Implemente una estrategia GitHub de ramificación de Flow para DevOps entornos con varias cuentas (guíaAWS prescriptiva)
- GitHub Flow Quickstart (documentación) GitHub
- ¿Por qué GitHub Flow? (Sitio web GitHub de Flow)

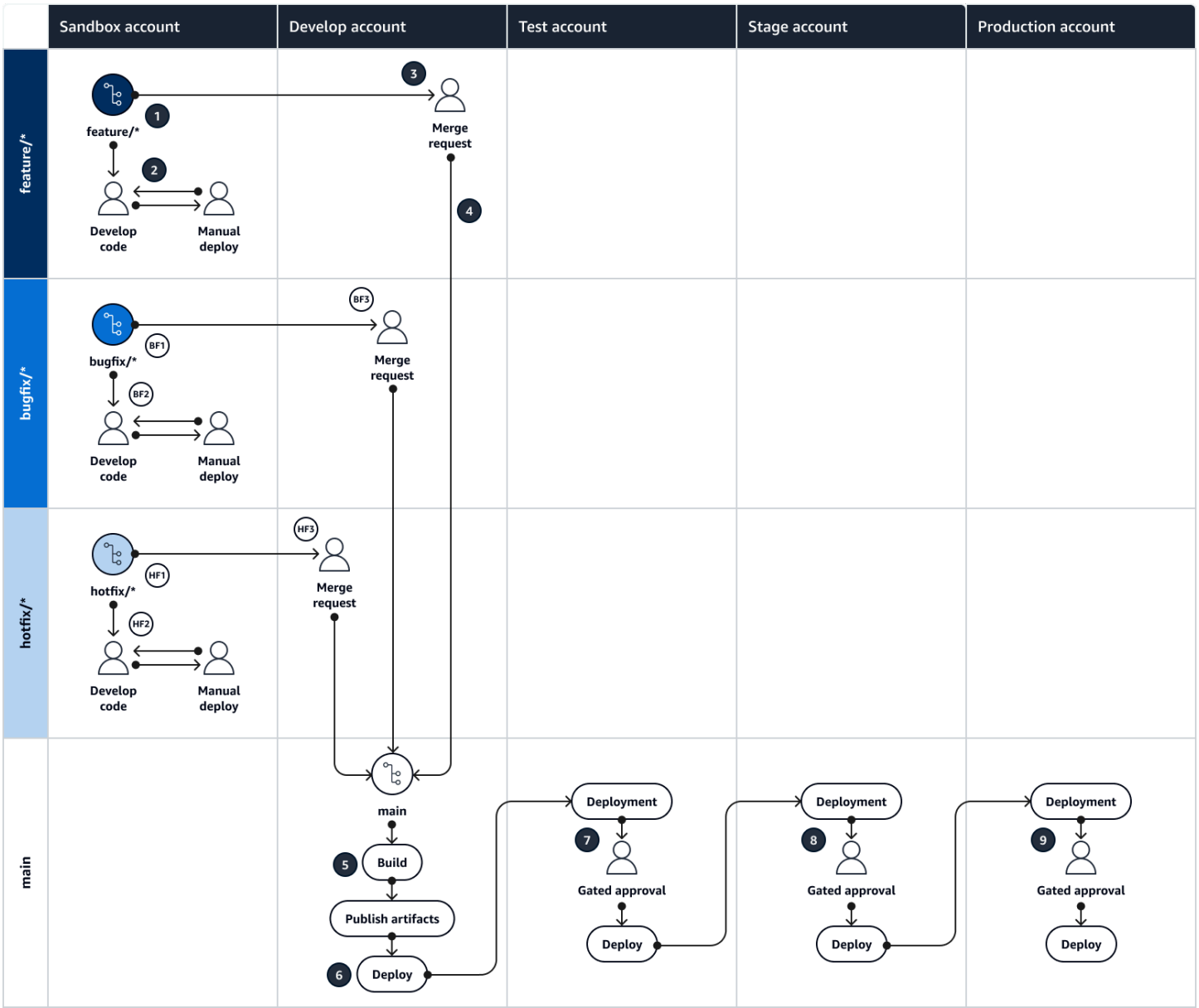
Temas de esta sección:

- Descripción visual de la estrategia GitHub Flow
- Las ramas en una estrategia de GitHub flujo
- Ventajas y desventajas de la estrategia GitHub Flow

Descripción visual de la estrategia GitHub Flow

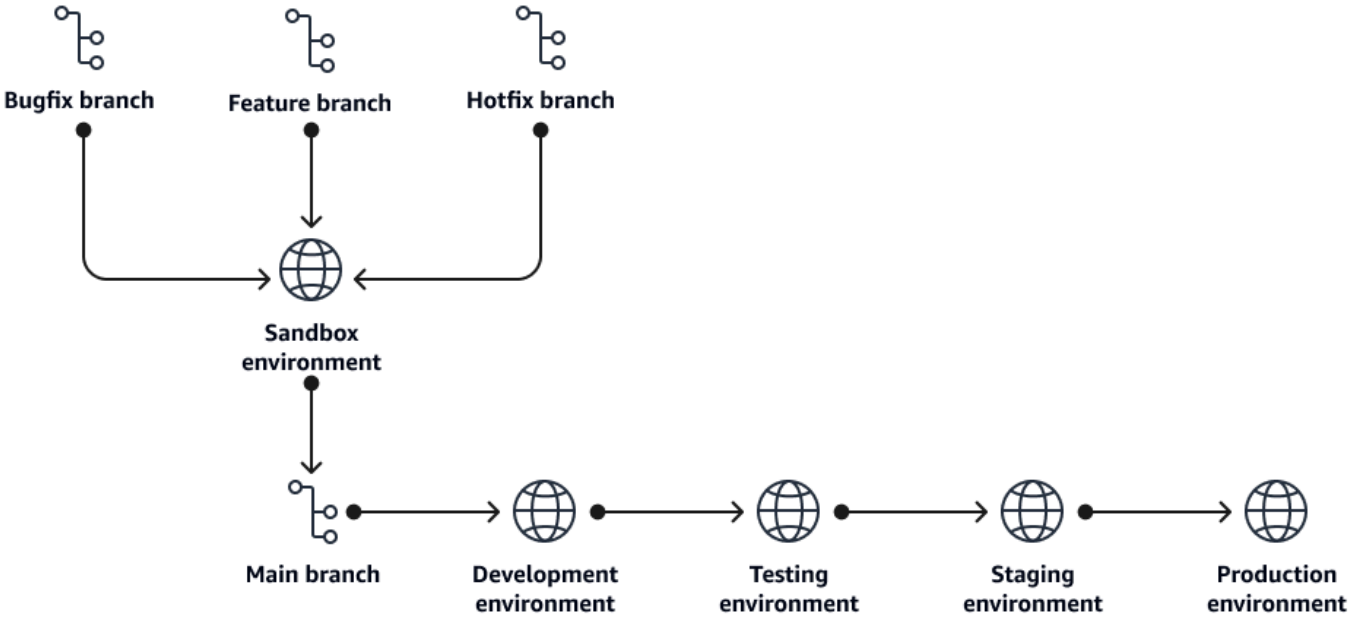
El siguiente diagrama se puede utilizar como un cuadro de Punnett para entender la estrategia de ramificación de GitHub Flow. Alinee las ramas del eje vertical con los AWS entornos del eje

horizontal para determinar qué acciones realizar en cada escenario. Los números encerrados en un círculo te guían por la secuencia de acciones representada en el diagrama. Este diagrama muestra el flujo de trabajo de desarrollo de una estrategia de ramificación de GitHub Flow, desde una rama de funciones en un entorno sandbox hasta la versión de producción de la rama principal. Para obtener más información sobre las actividades que se producen en cada entorno, consulte DevOps los entornos en esta guía.



Las ramas en una estrategia de GitHub flujo

Una estrategia GitHub de ramificación de Flow suele tener las siguientes ramas.



rama de característica

Desarrollas funciones en las `feature` sucursales. Para crear una `feature` rama, se ramifica a partir de la `main` rama. Los desarrolladores iteran, confirman y prueban el código de la `feature` rama. Cuando una función está completa, el desarrollador la promociona creando una solicitud de fusión para`main`.

Convención de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

rama de corrección de errores

La `bugfix` rama se usa para solucionar problemas. Estas ramas se ramifican a partir de la `main` rama. Una vez que la corrección del error se haya probado en entornos aislados o en cualquiera de los entornos inferiores, se puede promover a entornos superiores fusionándola mediante una solicitud de fusión. `main` Esta es una convención de nomenclatura sugerida para la organización y el seguimiento. Este proceso también se puede gestionar mediante una rama de funciones.

Convención de nomenclatura:	`bugfix/<ticket number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`bugfix/123456_MS_Fix_Problem_A`

rama de hotfix

La `hotfix` sucursal se utiliza para resolver problemas críticos de alto impacto con una demora mínima entre el personal de desarrollo y el código implementado en producción. Estas sucursales se ramifican fuera de la `main` sucursal. Una vez que la revisión se haya probado en entornos aislados o en alguno de los entornos inferiores, se puede ascender a entornos superiores si se fusiona mediante una solicitud de fusión. `main` Esta es una convención de nomenclatura sugerida para la organización y el seguimiento. Este proceso también se puede gestionar mediante una rama de funciones.

Convención de nomenclatura:	`hotfix/<ticket number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`hotfix/123456_MS_Fix_Problem_A`

rama principal

La `main` rama siempre representa el código que se está ejecutando en producción. El código se fusiona en la `main` rama desde `feature` las ramas mediante solicitudes de combinación. Para evitar que se eliminen y para evitar que los desarrolladores envíen el código directamente a ella`main`, habilita la protección de la `main` rama en cuestión.

Convención de nomenclatura:	`main`

Ventajas y desventajas de la estrategia GitHub Flow

La estrategia de ramificación de Github Flow es ideal para equipos de desarrollo más pequeños y maduros que tienen sólidas habilidades de comunicación. Esta estrategia es adecuada para los equipos que desean implementar la entrega continua y está bien respaldada por los motores de CI/

CD más comunes. GitHub Flow es ligero, no tiene demasiadas reglas y es capaz de soportar equipos que se mueven rápidamente. No es adecuado si tus equipos tienen que seguir procesos estrictos de cumplimiento o publicación. Los conflictos de fusión son comunes en este modelo y es probable que ocurran con frecuencia. La resolución de conflictos de fusión es una habilidad clave, y debes capacitar a todos los miembros del equipo en consecuencia.

Ventajas

GitHub Flow ofrece varias ventajas que pueden mejorar el proceso de desarrollo, agilizar la colaboración y mejorar la calidad general del software. Los siguientes son algunos de los beneficios clave:

- Flexible y ligero: GitHub Flow es un flujo de trabajo ligero y flexible que ayuda a los desarrolladores a colaborar en proyectos de desarrollo de software. Permite una iteración y experimentación rápidas con una complejidad mínima.
- Colaboración simplificada: GitHub Flow proporciona un proceso claro y simplificado para gestionar el desarrollo de funciones. Fomenta cambios pequeños y específicos que se pueden revisar y combinar rápidamente, lo que mejora la eficiencia.
- Control de versiones claro: con GitHub Flow, cada cambio se realiza en una rama independiente. Esto establece un historial de control de versiones claro y rastreable. Esto ayuda a los desarrolladores a rastrear y comprender los cambios, revertirlos si es necesario y mantener una base de código confiable.
- Integración continua perfecta: GitHub Flow se integra con herramientas de integración continua. La creación de solicitudes de cambios puede iniciar procesos automatizados de pruebas e implementación. Las herramientas de CI te ayudan a probar minuciosamente los cambios antes de integrarlos en la `main` rama, lo que reduce el riesgo de introducir errores en el código base.
- Retroalimentación rápida y mejora continua: GitHub Flow fomenta un ciclo de retroalimentación rápido al promover revisiones frecuentes del código y debates a través de solicitudes de selección de información. Esto facilita la detección temprana de problemas, promueve el intercambio de conocimientos entre los miembros del equipo y, en última instancia, conduce a una mayor calidad del código y a una mejor colaboración dentro del equipo de desarrollo.
- Reversiones y reversiones simplificadas: en el caso de que un cambio de código introduzca un error o un problema inesperado, GitHub Flow simplifica el proceso de revertir o revertir el cambio. Al tener un historial claro de confirmaciones y ramificaciones, es más fácil identificar y revertir los cambios problemáticos, lo que ayuda a mantener una base de código estable y funcional.

- Curva de aprendizaje ligera: GitHub Flow puede ser más fácil de aprender y adoptar que Gitflow, especialmente para los equipos que ya están familiarizados con Git y los conceptos de control de versiones. Su sencillez y su intuitivo modelo de ramificación hacen que sea accesible para desarrolladores con distintos niveles de experiencia, lo que reduce la curva de aprendizaje asociada a la adopción de nuevos flujos de trabajo de desarrollo.
- Desarrollo continuo: GitHub Flow permite a los equipos adoptar un enfoque de implementación continua al permitir la implementación inmediata de cada cambio tan pronto como se incorpore a la `main` sucursal. Este proceso simplificado elimina las demoras innecesarias y garantiza que las últimas actualizaciones y mejoras estén disponibles rápidamente para los usuarios. Esto se traduce en un ciclo de desarrollo más ágil y con mayor capacidad de respuesta.

Desventajas

Si bien GitHub Flow ofrece varias ventajas, también es importante tener en cuenta sus posibles desventajas:

- Aptitud limitada para proyectos grandes: es posible que GitHub Flow no sea tan adecuado para proyectos a gran escala con bases de código complejas y múltiples ramas de funciones a largo plazo. En esos casos, un flujo de trabajo más estructurado, como Gitflow, podría proporcionar un mejor control sobre la gestión simultánea del desarrollo y las versiones.
- Falta de una estructura de publicación formal: GitHub Flow no define explícitamente un proceso de publicación ni admite funciones como el control de versiones, las revisiones o las ramas de mantenimiento. Esto puede ser una limitación para los proyectos que requieren una gestión estricta de las versiones o que necesitan soporte y mantenimiento a largo plazo.
- Soporte limitado para la planificación del lanzamiento a largo plazo: GitHub Flow se centra en las ramas de funciones de corta duración, que pueden no adaptarse bien a los proyectos que requieren una planificación del lanzamiento a largo plazo, como aquellos con hojas de ruta estrictas o amplias dependencias de funciones. Administrar calendarios de lanzamiento complejos puede resultar difícil debido a las limitaciones de Flow. GitHub
- Potencial de conflictos de fusión frecuentes: dado que GitHub Flow fomenta las ramificaciones y fusiones frecuentes, existe la posibilidad de que surjan conflictos de fusión, especialmente en proyectos con mucha actividad de desarrollo. Resolver estos conflictos puede llevar mucho tiempo y puede requerir un esfuerzo adicional por parte del equipo de desarrollo.
- Falta de fases de flujo de trabajo formalizadas: GitHub Flow no define fases explícitas para el desarrollo, como las fases alfa, beta o candidata a una versión. Esto puede dificultar la

comunicación del estado actual del proyecto o del nivel de estabilidad de las distintas ramas o versiones.

- Impacto de los cambios importantes: dado que GitHub Flow fomenta la fusión frecuente de los cambios en la `main` rama, existe un mayor riesgo de introducir cambios importantes que afecten a la estabilidad del código base. Las prácticas estrictas de revisión y prueba del código son cruciales para mitigar este riesgo de manera efectiva.

Estrategia de ramificación de Gitflow

Gitflow es un modelo de ramificación que implica el uso de múltiples ramas para mover el código del desarrollo a la producción. Gitflow funciona bien para los equipos que tienen ciclos de lanzamiento programados y que necesitan definir un conjunto de funciones como versión. El desarrollo se completa en ramas de funciones individuales que se fusionan, previa aprobación, en una rama de desarrollo, que se utiliza para la integración. Las funciones de esta rama se consideran listas para su producción. Cuando todas las funciones planificadas se han acumulado en la rama de desarrollo, se crea una rama de lanzamiento para las implementaciones en los entornos superiores. Esta separación mejora el control sobre qué cambios se trasladan a cada entorno determinado según un cronograma definido. Si es necesario, puede acelerar este proceso y convertirlo en un modelo de implementación más rápido.

Para obtener más información sobre la estrategia de ramificación de Gitflow, consulta los siguientes recursos:

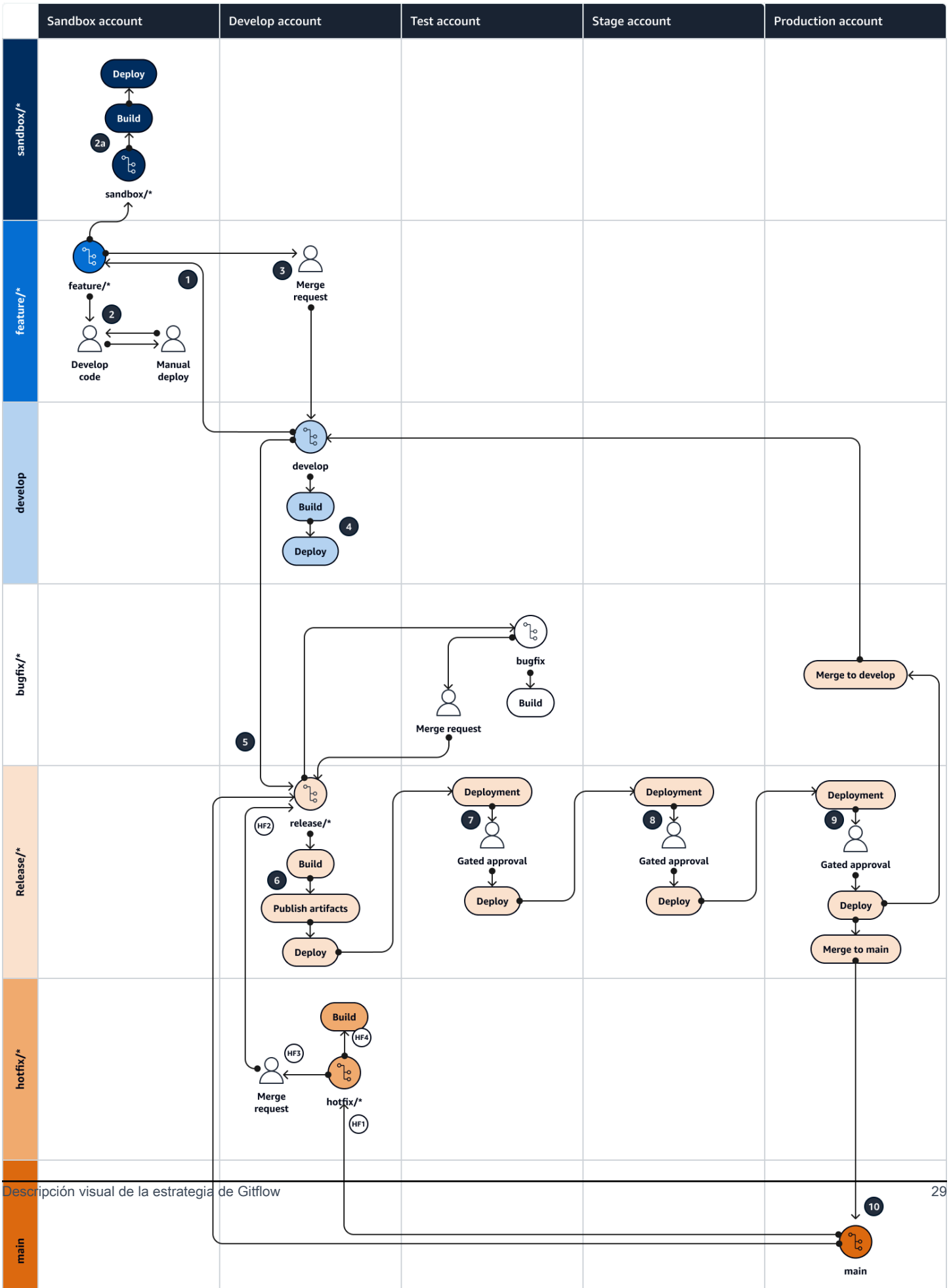
- Implementa una estrategia de ramificación de Gitflow para entornos con varias cuentas DevOps (guía prescriptiva)AWS
- El blog original de Gitflow (entrada del blog de Vincent Driessen)
- Flujo de trabajo de Gitflow (Atlassian)

Temas de esta sección:

- Descripción visual de la estrategia de Gitflow
- Las ramificaciones en una estrategia de Gitflow
- Ventajas y desventajas de la estrategia de Gitflow

Descripción visual de la estrategia de Gitflow

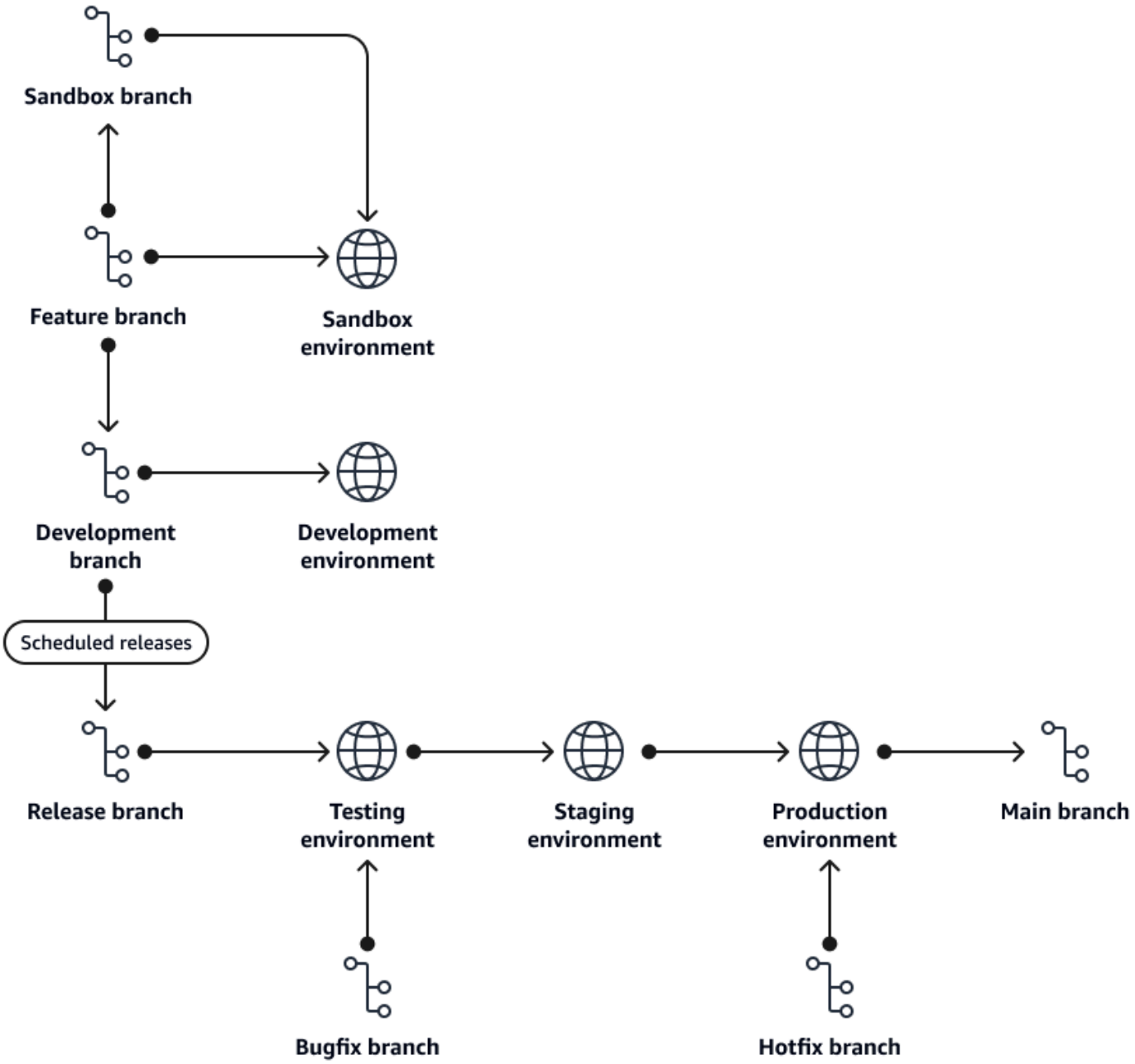
El siguiente diagrama puede usarse como un cuadro de Punnett para entender la estrategia de ramificación de Gitflow. Alinee las ramas del eje vertical con los AWS entornos del eje horizontal para determinar qué acciones realizar en cada escenario. Los números encerrados en un círculo te guían por la secuencia de acciones representada en el diagrama. Para obtener más información sobre las actividades que se llevan a cabo en cada entorno, consulte DevOps los entornos en esta guía.



Descripción visual de la estrategia de Gitflow

Las ramificaciones en una estrategia de Gitflow

Una estrategia de ramificación de Gitflow suele tener las siguientes ramas.



rama de característica

`Feature`las sucursales son ramas a corto plazo en las que se desarrollan funciones. La `feature` rama se crea al ramificarse a partir de la `develop` rama. Los desarrolladores iteran, confirman

y prueban el código de la `feature` rama. Cuando la función está completa, el desarrollador la promociona. Solo hay dos caminos hacia adelante desde una rama de funciones:

- Incorpórese a la `sandbox` rama
- Crea una solicitud de fusión en la `develop` sucursal

Convención de nomenclatura:	`feature/<story number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`feature/123456_MS_Implement_Feature_A`

rama sandbox

La `sandbox` rama es una rama no estándar y de corto plazo para Gitflow. Sin embargo, es útil para el desarrollo de canalizaciones de CI/CD. La `sandbox` rama se utiliza principalmente para los siguientes propósitos:

- Realice una implementación completa en el entorno sandbox mediante las canalizaciones de CI/CD en lugar de una implementación manual.
- Desarrolle y pruebe una canalización antes de enviar solicitudes de fusión para realizar pruebas completas en un entorno inferior, como el de desarrollo o las pruebas.

`Sandbox`las sucursales son de naturaleza temporal y no están destinadas a ser duraderas. Deben borrarse una vez finalizadas las pruebas específicas.

Convención de nomenclatura:	`sandbox/<story number>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`sandbox/123456_MS_Test_Pipeline_Deploy`

desarrollar una rama

La `develop` sucursal es una rama de larga duración en la que las funciones se integran, crean, validan e implementan en el entorno de desarrollo. Todas las `feature` sucursales se fusionan en la `develop` sucursal. Las fusiones en la `develop` sucursal se realizan mediante una solicitud de fusión que requiere una compilación correcta y la aprobación de dos desarrolladores. Para evitar que se eliminen, habilita la protección de sucursales en la `develop` rama.

Convención de nomenclatura:	`develop`

rama de lanzamiento

En Gitflow, las `release` ramas son ramas a corto plazo. Estas ramas son especiales porque puedes desplegarlas en múltiples entornos, adoptando la metodología de construir una vez y desplegar muchas veces. `Release`las sucursales pueden centrarse en los entornos de prueba, puesta en escena o producción. Una vez que un equipo de desarrollo ha decidido promover funciones en entornos superiores, crea una nueva `release` rama y utiliza un aumento del número de versión con respecto a la versión anterior. En las puertas de cada entorno, las implementaciones requieren aprobaciones manuales para poder llevarse a cabo. `Release`las sucursales deberían requerir que se modifique una solicitud de fusión.

Una vez que la `release` rama se haya implementado en producción, se debe volver a fusionar con las `main` ramas `develop` y para garantizar que las correcciones de errores o revisiones se fusionen de nuevo en futuras iniciativas de desarrollo.

Convención de nomenclatura:	`release/v{major}.{minor}`
Ejemplo de convención de nomenclatura:	`release/v1.0`

rama principal

La `main` rama es una rama de larga duración que siempre representa el código que se está ejecutando en producción. El código se fusiona automáticamente en la `main` rama desde una rama de lanzamiento tras una implementación correcta desde el proceso de publicación. Para evitar la eliminación, habilita la protección de la rama en la `main` rama.

Convención de nomenclatura:	`main`

rama de corrección de errores

La `bugfix` rama es una rama a corto plazo que se utiliza para solucionar problemas en las ramas de lanzamiento que no se han lanzado al mercado de producción. Una `bugfix` rama solo debe usarse para promover correcciones en las `release` sucursales para los entornos de prueba, preparación o producción. Una `bugfix` sucursal siempre se ramifica a partir de una `release` sucursal.

Una vez que se haya probado la corrección del error, se puede ascender a la `release` rama mediante una solicitud de fusión. A continuación, puedes impulsar la `release` rama siguiendo el proceso de publicación estándar.

Convención de nomenclatura:	`bugfix/<ticket>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`bugfix/123456_MS_Fix_Problem_A`

rama de hotfix

La `hotfix` sucursal es una rama a corto plazo que se utiliza para solucionar problemas en la producción. Solo se utilizará para promover soluciones que deben acelerarse para que lleguen al entorno de producción. Siempre se `hotfix` ramifica una sucursal desde. `main`

Una vez que se haya probado la revisión, puedes pasarla a producción mediante una solicitud de fusión en la `release` rama desde la que se creó. `main` Para realizar las pruebas, puedes impulsar la `release` rama siguiendo el proceso de publicación estándar.

Convención de nomenclatura:	`hotfix/<ticket>_<developer initials>_<descriptor>`
Ejemplo de convención de nomenclatura:	`hotfix/123456_MS_Fix_Problem_A`

Ventajas y desventajas de la estrategia de Gitflow

La estrategia de ramificación de Gitflow se adapta bien a equipos más grandes y distribuidos que tienen requisitos estrictos de publicación y cumplimiento. Gitflow contribuye a un ciclo de lanzamiento predecible para la organización, algo que suelen preferir las organizaciones más grandes. Gitflow también es ideal para equipos que necesitan barreras para completar su ciclo de vida de desarrollo de software de forma adecuada. Esto se debe a que la estrategia incorpora múltiples oportunidades de revisión y control de calidad. Gitflow también es ideal para equipos que deben mantener varias versiones de versiones de producción simultáneamente. Algunas desventajas de GitFlow son que es más complejo que otros modelos de ramificación y requiere un estricto cumplimiento del patrón para completarse con éxito. Gitflow no funciona bien para las organizaciones que buscan una entrega continua debido a la naturaleza rígida de la gestión de las ramas de lanzamiento. Las ramas de publicación de Gitflow pueden ser ramas duraderas que pueden acumular deudas técnicas si no se gestionan adecuadamente y de manera oportuna.

Ventajas

El desarrollo basado en Gitflow ofrece varias ventajas que pueden mejorar el proceso de desarrollo, agilizar la colaboración y mejorar la calidad general del software. Los siguientes son algunos de los beneficios clave:

- Proceso de publicación predecible: Gitflow sigue un proceso de publicación regular y predecible. Es ideal para equipos con cadencias de desarrollo y lanzamiento regulares.
- Colaboración mejorada: Gitflow fomenta el uso de sucursales `feature` y `release` sucursales. Estas dos ramas ayudan a los equipos a trabajar en paralelo con una dependencia mínima entre sí.
- Ideal para múltiples entornos: Gitflow usa `release` ramas, que pueden ser ramas de mayor duración. Estas sucursales permiten a los equipos centrarse en los lanzamientos individuales durante un período de tiempo más largo.
- Varias versiones en producción: si tu equipo admite varias versiones del software en producción, las `release` sucursales de Gitflow admiten este requisito.
- Revisiones de calidad del código integradas: Gitflow exige y fomenta el uso de revisiones y aprobaciones del código antes de promocionarlo a otro entorno. Este proceso elimina las fricciones entre los desarrolladores al requerir este paso para todas las promociones de código.
- Beneficios organizativos: Gitflow también tiene ventajas a nivel organizativo. Gitflow fomenta el uso de un ciclo de lanzamiento estándar, lo que ayuda a la organización a entender y anticipar

el calendario de lanzamientos. Como la empresa ahora sabe cuándo se pueden ofrecer nuevas funciones, se reducen las fricciones con respecto a los plazos, ya que hay fechas de entrega establecidas.

Desventajas

El desarrollo basado en Gitflow tiene algunas desventajas que pueden afectar al proceso de desarrollo y a la dinámica del equipo. Los siguientes son algunos inconvenientes notables:

- Complejidad: Gitflow es un patrón complejo que deben aprender los nuevos equipos, y debes seguir las reglas de Gitflow para usarlo con éxito.
- Despliegue continuo: Gitflow no se ajusta a un modelo en el que muchas implementaciones se lanzan a producción de forma rápida. Esto se debe a que Gitflow requiere el uso de varias sucursales y un flujo de trabajo estricto que regule la sucursal. `release`
- Administración de sucursales: Gitflow utiliza muchas sucursales, lo que puede resultar engorroso de mantener. Puede resultar difícil rastrear las distintas ramas y fusionar el código publicado para mantener las ramas correctamente alineadas entre sí.
- Deuda técnica: dado que las versiones de Gitflow suelen ser más lentas que los otros modelos de ramificación, es posible que se acumulen más funciones antes de publicarlas, lo que puede provocar que se acumule deuda técnica.

Los equipos deberían tener en cuenta estos inconvenientes a la hora de decidir si el desarrollo basado en Gitflow es el enfoque adecuado para su proyecto.

Siguientes pasos

Esta guía explica las diferencias entre tres estrategias comunes de ramificación de Git: GitHub Flow, Gitflow y Trunk. Describe sus flujos de trabajo en detalle y también proporciona las ventajas y desventajas de cada uno de ellos. Los siguientes pasos son elegir uno de estos flujos de trabajo estándar para su organización. Para implementar una de estas estrategias de ramificación, consulta lo siguiente:

- Implemente una estrategia de ramificación troncal para entornos con varias cuentas DevOps
- Implemente una estrategia de ramificación GitHub de Flow para entornos con varias cuentas DevOps
- Implementa una estrategia de ramificación de Gitflow para entornos con varias cuentas DevOps

Si no estás seguro de por dónde empezar el viaje de tu equipo hacia el uso de Git y DevOps los procesos, te recomendamos que elijas una solución estándar y la pruebes. El uso de una convención de ramificación estándar ayuda al equipo a aprovechar la documentación existente y a aprender qué es lo que funciona mejor para ellos.

No dudes en cambiar tu estrategia si no funciona para tu organización o tus equipos de desarrollo. Las necesidades y los requisitos de los equipos de desarrollo pueden cambiar con el tiempo y no existe una solución única y perfecta.

Recursos

Esta guía no incluye formación sobre Git; sin embargo, hay muchos recursos de alta calidad disponibles en Internet si necesitas esta formación. Te recomendamos que comiences por el sitio de documentación de Git.

Los siguientes recursos pueden ayudarte con tu viaje de ramificación de Git en el Nube de AWS.

AWS Guía prescriptiva

- Implemente una estrategia de ramificación troncal para entornos de cuentas múltiples DevOps
- Implemente una estrategia de ramificación GitHub de Flow para entornos con varias cuentas DevOps
- Implementa una estrategia de ramificación de Gitflow para entornos con varias cuentas DevOps

Otra orientación AWS

- AWS DevOps Orientación
- AWS Arquitectura de referencia para la canalización de despliegue
- ¿Qué es DevOps?
- DevOpsrecursos

Otros recursos

- Metodología de aplicaciones de doce factores (12factor.net)
- Secretos de Git () GitHub
- Gitflow
  - El blog original de Gitflow (entrada del blog de Vincent Driessen)
  - Flujo de trabajo de Gitflow (Atlassian)
  - Gitflow en GitHub: Cómo usar los flujos de trabajo de Git Flow con repositorios GitHub basados (vídeo) YouTube
  - Ejemplo de Git Flow Init (YouTube vídeo)

  - La rama de lanzamiento de Gitflow de principio a fin (vídeo) YouTube
- GitHub Flujo
  - GitHub Flow Quickstart (GitHub documentación)
  - ¿Por qué GitHub Flow? (Sitio web GitHub de Flow)
- Tronco
  - Introducción al desarrollo basado en troncos (sitio web de desarrollo basado en troncos)

Colaboradores

Creación

- Mike Stephens, arquitecto sénior de aplicaciones en la nube, AWS
- Rayjan Wilson, arquitecto sénior de aplicaciones en la nube, AWS
- Abhilash Vinod, jefe de equipo, arquitecto sénior de aplicaciones en la nube, AWS

Revisando

- Stephen DiCato, consultor sénior de seguridad, AWS
- Gaurav Samudra, arquitecto de aplicaciones en la nube, AWS
- Steven Guggenheimer, jefe de equipo, arquitecto sénior de aplicaciones en la nube, AWS

Redacción técnica

- Lilly AbouHarb, redactora técnica sénior, AWS

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 febrero de 2024

AWS Glosario de las Recomendaciones de

Los siguientes son términos de uso común en las estrategias, guías y patrones que se ofrecen en las AWS Recomendaciones de. 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:

- Refactorizar/rediseñar: traslade una aplicación y modifique su arquitectura mediante el máximo aprovechamiento de las características nativas en 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 la base de datos de Oracle en las instalaciones a Amazon Aurora Postgre Edition. SQL
- 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: Migre la base de datos de Oracle en las instalaciones a Amazon Relational Service (RDSAmazon) para Oracle en. 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 el sistema de administración de las relaciones con los clientes (CRM) a Salesforce.com.
- Volver a alojar (migrar mediante lift-and-shift): traslade una aplicación a la nube sin realizar cambios para aprovechar las capacidades de la nube. Ejemplo: Migre la base de datos de Oracle en las instalaciones a Oracle en una EC2 instancia del 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 local a un servicio en la nube para la misma plataforma. Ejemplo: migrar un 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

ABAC

Consulte control de acceso basado en atributos.

servicios abstractos

Consulte servicios gestionados.

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 la migración activa y 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 base de datos de origen gestiona las transacciones de las aplicaciones conectadas, mientras los datos se replican en la base de datos de destino. La base de datos de destino no acepta ninguna transacción durante la migración.

función de agregación ación

SQLFunción que opera en un grupo de filas y calcula un único valor de retorno para el grupo. Entre los ejemplos de funciones agregadas se incluyen `SUM` y`MAX`.

IA

Véase inteligencia artificial.

AIOps

Consulte las 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

Un enfoque de seguridad que permite el uso únicamente de 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 AIOps se utiliza en la estrategia de AWS migración de, 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 ABACla AWS documentación de 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

Ubicación diferenciada de una Región de AWS que está aislada de los errores que se producen en otras zonas de disponibilidad y que brinda 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

Marco de directrices y prácticas recomendadas de AWS para ayudar a las empresas a desarrollar un plan eficiente y eficaz a fin de migrar con éxito a la nube de. AWS CAForganiza la orientación en seis áreas de enfoque llamadas perspectivas: empresarial, humana, gobernanza, 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 brinda orientación para el desarrollo, la capacitación y la comunicación de las personas, con el fin de ayudar a preparar la organización para una adopción exitosa de la nube. Para obtener más información, consulte el AWS CAFsitio web y el AWS CAFdocumento técnico.

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 brinda estimaciones de trabajo. AWS WQFse 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 incorrecto

Un bot destinado a interrumpir o causar daño a personas u organizaciones.

BCP

Consulte la 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 API llamadas sospechosas 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. Véase también endianismo.

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.

implementación azul/verde

Una estrategia de despliegue 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 en el otro entorno (verde). Esta estrategia le ayuda a revertirla rápidamente 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 información en Internet. Algunos otros bots, conocidos como bots malos, tienen como objetivo interrumpir o causar daños a personas u organizaciones.

botnet

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

rama

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 con cristales rotos

En circunstancias excepcionales y mediante un proceso aprobado, un usuario puede acceder rápidamente a un sitio para el Cuenta de AWS que normalmente no tiene permisos de acceso. Para obtener más información, consulte el indicador Implemente procedimientos de rotura de cristales en la guía Well-Architected AWS .

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 el marco AWS de adopción de la nube.

Implementación de valores controlados

El lanzamiento lento e incremental de una versión para los usuarios finales. Cuando está seguro, despliega la nueva versión y reemplaza la versión actual en su totalidad.

CCoE

Consulte Centro de excelencia en la nube.

CDC

Consulte la 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 utilizarse CDC para diversos fines, como auditar o replicar los cambios en un sistema de destino para mantener la sincronización.

ingeniería del caos

Introducir intencionalmente 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 y entrega continuas.

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.

cifrado del cliente

Cifrado de datos de forma local, antes de que el de destino 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 CCoEPublicaciones del 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 conectada a la tecnología de computación de punta.

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 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 landing zone, definir unCCoE, establecer un modelo de operaciones)
- Migración: migración de aplicaciones individuales

- Reinvención: optimización de productos y servicios e innovación en la nube

Stephen Orban definió estas etapas en la publicación del blog The Journey Toward Cloud-First & the Stages of Adoption (El camino hacia la nube como prioridad y las etapas de adopción) en el Blog de estrategia Nube de AWS empresarial. Para obtener información sobre cómo se relacionan con la estrategia de AWS migración de, consulte la Guía de preparación para la migración.

CMDB

Consulte la base de datos de administración de la 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. Los repositorios en la nube más comunes incluyen GitHub o AWS CodeCommit. Cada versión del código se denomina rama. En una estructura de microservicios, cada repositorio se encuentra dedicado a una única funcionalidad. Una sola canalización de CI/CD 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 aprendizaje automático para analizar y extraer información de formatos visuales, como imágenes y vídeos digitales. Por ejemplo, AWS Panorama ofrece dispositivos que añaden CV a las redes de cámaras locales, y Amazon SageMaker proporciona algoritmos de procesamiento de imágenes para CV.

desviación de configuración

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

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 se utilizan CMDB en la etapa de detección y análisis de la cartera de productos durante la migración.

paquete de conformidad

Una colección de acciones correctivas y AWS Config reglas de que puede reunir para personalizar sus controles de seguridad y conformidad. Puede implementar un paquete de conformidad como una sola entidad en una región Cuenta de AWS y, o en toda una organización, mediante una YAML plantilla. Para obtener más información, consulte Paquetes de conformidad en la AWS Config documentación de.

integración y entrega continuas (CI/CD)

El proceso de automatización de las etapas de origen, compilación, prueba, presentación y producción del proceso de lanzamiento del software. La CI/CD se describe comúnmente como una canalización. La CI/CD puede ayudarlo a automatizar los procesos, mejorar la productividad, mejorar la calidad del código y entregar con mayor rapidez. Para obtener más información, consulte Beneficios de la entrega continua. CD también puede significar 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 datos es un componente del pilar de seguridad del Marco AWS Well-Architected. Para obtener más información, consulte Clasificación de datos.

desviación 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 desviación de los 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

Un marco arquitectónico que proporciona una propiedad de datos distribuida y descentralizada con administración y gobierno centralizados.

minimización de datos

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

Perímetro de datos

Conjunto de barreras preventivas en su AWS entorno de que ayudan a garantizar que solo las identidades de confianza accedan a los recursos de confianza 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

Un sistema de administración de datos que respalde la inteligencia empresarial, como el análisis. Los almacenes de datos suelen contener grandes cantidades de datos históricos y, por lo general, se utilizan para consultas y 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 el 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.

defense-in-depth

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. Cuando se adopta esta estrategia en AWS, se suman varios controles en diferentes capas de la AWS Organizations estructura para ayudar a proteger los recursos. Por ejemplo, un defense-in-depth enfoque 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 de a fin de administrar las cuentas de la organización y los permisos para 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. DVSMamplía 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 de 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 utilizan habitualmente para restringir consultas, filtrar y etiquetar 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)

La estrategia y el proceso que utiliza para minimizar el tiempo de inactividad y la pérdida de datos causados por un desastre. Para obtener más información, consulte Recuperación de desastres de cargas de trabajo en AWS: Recuperación en la nube en AWS Well-Architected Framework.

DML

Consulte el 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. Este concepto lo introdujo Eric Evans en su libro, Diseño impulsado por el dominio: abordando la complejidad en el corazón del 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 de la versión antigua de Microsoft. ASP NET(ASMX) servicios web de forma incremental mediante contenedores y Amazon API Gateway.

DR

Consulte recuperación ante desastres.

detección de desviaciones

Seguimiento de las desviaciones con respecto a una configuración de referencia. 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 el mapeo de flujos de valor para el desarrollo.

E

EDA

Consulte el análisis exploratorio 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 de, la computación en la periferia puede reducir la latencia de la comunicación y mejorar el tiempo de respuesta.

cifrado

Proceso informático que transforma datos de texto plano, 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. Los sistemas big-endianos almacenan primero el byte más significativo. Los sistemas Little-Endian almacenan primero el byte menos significativo.

punto de conexión

Consulte el punto final del 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 otros Cuentas de AWS o AWS Identity and Access Management (IAM) principales AWS PrivateLink y conceder permisos a ellos. Estas cuentas o entidades principales de pueden conectarse a su servicio de punto de conexión de forma privada mediante la creación de puntos de conexión de interfazVPC. Para obtener más información, consulte Creación de un servicio de punto de conexión en la documentación de Amazon Virtual Cloud (AmazonVPC).

planificación de recursos empresariales (ERP)

Un sistema que automatiza y gestiona los procesos empresariales clave (como la contabilidad y la gestión de proyectos) de una empresa. MES

cifrado de sobre

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

environment

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 una canalización de CI/CD, el entorno de producción es el último entorno de implementación.
- entornos superiores: todos los entornos a los que pueden acceder usuarios 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 de AWS CAF seguridad incluyen la administración 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 la 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. EDAse realiza mediante el cálculo de estadísticas resumidas y la creación de visualizaciones de datos.

F

tabla de datos de datos

La tabla central de un esquema en forma de estrella. Almacena datos cuantitativos sobre las operaciones comerciales. Normalmente, 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.

fallar rápido

Una filosofía que utiliza pruebas frecuentes e incrementales para reducir el ciclo de vida del desarrollo. Es una parte fundamental de un enfoque ágil.

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 obtener más información, consulte Límites de AWS aislamiento de errores.

rama de característica

Consulte la sucursal.

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 machine learning 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.

FGAC

Consulte control de acceso detallado.

control de acceso detallado () 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 cambios 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.

G

bloqueo geográfico

Consulta las 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, consulte Restricción de la distribución geográfica de su 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, y el flujo de trabajo basado en troncos es el enfoque moderno preferido.

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 (OUs). 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 IAM permisos. 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 GuardDuty AWS Trusted Advisor, Amazon Inspector y AWS Lambda cheques personalizados.

H

JA

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.

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 for 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, el hotfix suele realizarse fuera del flujo de trabajo típico 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

Vea la infraestructura como código.

políticas basadas en identidad

Una política asociada a una o más entidades IAM principales que define sus permisos en el Nube de AWS entorno de la.

aplicación inactiva

Aplicación que utiliza un promedio CPU de 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

Un modelo que implementa una nueva infraestructura para las cargas de trabajo de producción en lugar de actualizar, aplicar parches o modificar la infraestructura existente. Las infraestructuras inmutables son intrínsecamente más consistentes, fiables y predecibles que las infraestructuras mutables. Para obtener más información, consulte las Mejores prácticas de implementación con una infraestructura inmutable en Well-Architected Framework AWS .

entrante (entrada) VPC

En una arquitectura de AWS varias cuentas, VPC que acepta, inspecciona y enruta las conexiones de red desde fuera de una aplicación. La Arquitectura de referencia de AWS seguridad de recomienda configurar su cuenta de red con entradas, salientes y de inspección VPCs 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 la inteligencia artificial/aprendizaje automático.

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).

inspección VPC

En una arquitectura de AWS varias cuentas de, una arquitectura centralizada VPC que administra las inspecciones del tráfico de red entre VPCs (en la misma o en diferentes Regiones de AWS), Internet y las redes en las instalaciones. La Arquitectura de referencia de AWS seguridad de recomienda configurar su cuenta de red con entradas, salientes y de inspección VPCs 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 machine learning 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. ITILproporciona la base paraITSM.

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 ITSM las herramientas, consulte la Guía de integración de operaciones.

ITIL

Consulte la 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 zona de aterrizaje es un AWS entorno de correctamente diseñado, con varias cuentas, que es 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.

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. IAM

migrar mediante lift-and-shift

Consulte 7 Rs.

sistema little-endian

Un sistema que almacena primero el byte menos significativo. Véase también endianness.

entornos inferiores

Véase 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

Ver sucursal.

malware

Software diseñado para comprometer la seguridad o la privacidad de la computadora. El malware puede 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 keyloggers.

servicios administrados

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

sistema de ejecución de fabricación () MES

Un sistema de software para rastrear, monitorear, documentar y controlar los procesos de producción que convierten las materias primas en productos terminados en el taller.

MAP

Consulte Migration Acceleration Program.

mecanismo

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

cuenta de miembro

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

MES

Consulte el sistema de ejecución de la fabricación.

Message Queuing Telemetry Transport () MQTT

Un protocolo de comunicación ligero machine-to-machine (M2M), basado en el patrón de publicación/suscripción, para dispositivos de IoT con recursos limitados.

microservicio

Un servicio pequeño e independiente que se comunica a través de equipos pequeños APIs 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 Integración de microservicios mediante servicios AWS sin servidor de servidor de.

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 un ligero. APIs 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.

Migration Acceleration Program (MAP)

AWS Programa de que brinda soporte de consultoría, capacitación y servicios para ayudar a las empresas a construir una base operativa sólida para migrar a la nube y ayudar a compensar el costo inicial de las migraciones. MAPincluye 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

Equipos multifuncionales que agilizan la migración de las cargas de trabajo mediante enfoques automatizados y ágiles. Los equipos de la fábrica de migración suelen incluir operaciones, analistas y propietarios de negocios, ingenieros de migración, desarrolladores y DevOps profesionales que trabajan en tiempo y forma. 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 las subredes de destino, los grupos de seguridad y las AWS cuentas de.

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: Volver a alojar la migración en Amazon EC2 con AWS Application Migration Service.

Migration Portfolio Assessment (MPA)

Herramienta en línea que brinda información a fin de validar los argumentos comerciales necesarios para migrar a. Nube de AWS MPAofrece una evaluación detallada de la cartera (adecuación del tamaño de los servidores, precios, TCO comparaciones, 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 MPAherramienta (requiere inicio de sesión) está disponible de forma gratuita para todos los AWS consultores y consultores APN asociados.

Evaluación de la preparación para la migración (MRA)

Proceso de obtener información sobre el estado de preparación de la nube de una organización, identificar los puntos fuertes y débiles y elaborar un plan de acción para cerrar las brechas identificadas, utilizando la AWS CAF. Para obtener más información, consulte la Guía de preparación para la migración. MRAes la primera fase de la estrategia de AWS migración.

estrategia de migración

Enfoque utilizado para migrar una carga de trabajo a Nube de AWS. Para obtener más información, consulte la entrada de las 7 R de este glosario y consulte Movilice a su organización para acelerar las migraciones a gran escala.

ML

Consulte aprendizaje automático.

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 obtener más información, consulte Estrategia para modernizar las aplicaciones en la 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 obtener más información, consulte Evaluación de la preparación para la modernización de las aplicaciones en 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 la evaluación de la cartera de migración.

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

Un modelo que actualiza y modifica la infraestructura existente para las cargas de trabajo de producción. Para mejorar la coherencia, la fiabilidad y la previsibilidad, el AWS Well-Architected Framework recomienda el uso de una infraestructura inmutable como práctica recomendada.

O

OAC

Consulte el control de acceso de origen.

OAI

Consulte la identidad de acceso de origen.

OCM

Consulte gestión 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 machine-to-machine (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

Una lista de preguntas y las mejores prácticas asociadas que le ayudan a comprender, evaluar, prevenir o reducir el alcance de los incidentes y posibles fallos. Para obtener más información, consulte Operational Readiness Reviews (ORR) en AWS Well-Architected Framework.

tecnología operativa (OT)

Sistemas de hardware y software que funcionan con el entorno físico para controlar las operaciones, los equipos y la infraestructura industriales. En la industria manufacturera, 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

Registro de seguimiento creado por AWS CloudTrail que registra todos los eventos para todas las Cuentas de AWS en una organización en 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 Creación de un registro de seguimiento 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. OCMayuda 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 de, este marco se denomina aceleración de personas, debido a la velocidad de cambio requerida en los proyectos de adopción de la nube. Para obtener más información, consulte la OCMguía.

control de acceso de origen (OAC)

En CloudFront, una opción mejorada para restringir el acceso a su contenido del Amazon Simple Storage Service (Amazon S3). OACadmite todos los depósitos de S3 Regiones de AWS, el cifrado del lado del servidor con AWS KMS (SSE-KMS) y el cifrado dinámico `PUT` y `DELETE` las solicitudes al depósito de S3.

identidad de acceso de origen (OAI)

En CloudFront, una opción para restringir el acceso al contenido de Amazon S3. Cuando lo utilizaOAI, CloudFront crea una entidad principal con la que Amazon S3 puede autenticarse. Las entidades principales autenticadas solo pueden acceder al contenido de un bucket de S3 a través de una distribución específica CloudFront . Consulte también OAC, que proporciona un control de acceso más detallado y mejorado.

ORR

Consulte la revisión de la preparación operativa.

NO

Consulte tecnología operativa.

saliente (salida) VPC

En una arquitectura de AWS varias cuentas, VPC que gestiona las conexiones de red que se inician desde una aplicación. La Arquitectura de referencia de AWS seguridad de recomienda configurar su cuenta de red con entradas, salientes y de inspección VPCs para proteger la interfaz bidireccional entre su aplicación e Internet en general.

P

límite de permisos

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

Información personalmente identificable (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 PII son los nombres, las direcciones y la información de contacto.

PII

Consulte la 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 la gestión del ciclo de vida del producto.

política

Un objeto que puede definir los permisos (consulte la política basada en la identidad), especifique las condiciones de acceso (consulte la política basada en los recursos) o defina los permisos máximos para todas las cuentas de una organización AWS Organizations (consulte la política de control de servicios).

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. Para obtener más información, consulte Habilitación de la persistencia de datos en los microservicios.

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

Una condición de consulta que devuelve `true` o`false`, por lo general, se encuentra en una cláusula. `WHERE`

inserción de predicados

Técnica de optimización de consultas de bases de datos que filtra los datos de la consulta antes de transferirlos. Esto reduce la cantidad de datos que se deben recuperar y procesar de la base de datos relacional y 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

Entidad de AWS que puede realizar acciones y obtener acceso a los recursos. Esta entidad suele ser un usuario raíz de un Cuenta de AWS, un IAM rol o un usuario. Para obtener más información, consulte los términos y conceptos de Principal in Roles en la IAM documentación.

Privacidad desde el diseño

Un enfoque de ingeniería de sistemas que tiene en cuenta la privacidad durante todo el proceso de ingeniería.

zonas alojadas privadas

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

control proactivo

Control de seguridad diseñado para evitar el despliegue de recursos que no cumplan con las normas. Estos controles escanean los recursos antes de aprovisionarlos. Si el recurso no cumple con el control, significa que no está aprovisionado. 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.

gestión del ciclo de vida del producto (PLM)

La gestión 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 el rechazo y la retirada.

entorno de producción

Consulte el entorno.

controlador lógico programable () PLC

En la industria manufacturera, una computadora adaptable y altamente confiable que monitorea las máquinas y automatiza los procesos de fabricación.

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.

publicar/suscribirse (pub/sub)

Un patrón que permite las comunicaciones asíncronas entre microservicios para mejorar la escalabilidad y la capacidad de respuesta. Por ejemplo, en un microservicio basado en microservicios MES, un microservicio puede publicar mensajes de eventos en un canal al que se puedan suscribir otros microservicios. El sistema puede añadir 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 SQL relacional.

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

RACIMatriz

Véase responsable, confiable, confiable, confiable, consultada e informada (RACI).

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.

RASCIMatriz

Véase responsable, confiable, confiable, confiable, consultada e informada (RACI).

RCAC

Consulte control de acceso por filas y columnas.

read replica

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

Ver 7 Rs.

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

Ver 7 Rs.

Región

Conjunto de AWS recursos de que se encuentran en un área geográfica. Cada Región de AWS está aislada y es independiente de las demás para ofrecer tolerancia a errores, estabilidad y resistencia. Para obtener más información, consulte Regiones de AWS Especificar qué cuenta puede usar.

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 7 Rs.

versión

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

trasladarse

Ver 7 Rs.

redefinir la plataforma

Ver 7 Rs.

recompra

Ver 7 Rs.

resilencia

La capacidad de una aplicación para resistir las interrupciones o recuperarse de ellas. La alta disponibilidad y la recuperación ante desastres son consideraciones comunes a la hora de planificar la resiliencia en el. Nube de AWS Para obtener más información, consulte Nube de AWS Resiliencia.

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, confiable, 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 RASCImatriz y, si la excluye, se denomina RACImatriz.

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 7 Rs.

jubilarse

Ver 7 Rs.

rotación

Proceso en el que periódicamente se cambia el 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 SQL expresiones básicas y flexibles que tienen reglas de acceso definidas. RCACconsta de permisos de fila y máscaras de columnas.

RPO

Consulte el objetivo del punto de recuperación.

RTO

Consulte el 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

SAML2.0

Un estándar abierto que utilizan muchos proveedores de identidades (IdPs). Esta característica permite el inicio de sesión único (SSO) federado para que los usuarios puedan iniciar sesión en la o invocar las operaciones de sin necesidad de crear un inicio de sesión único () federado para que los usuarios puedan iniciar sesión en la AWS Management Console o invocar AWS API las operaciones de sin necesidad de crear un inicio de sesión IAM para cada persona de la organización. Para obtener más información sobre la federación SAML basada en 2.0, consulte Acerca de la federación SAML basada en 2.0 en la IAM documentación.

SCADA

Consulte el control de supervisión y la adquisición de datos.

SCP

Consulte la política de control de servicios.

secreta

Información confidencial o restringida, como una contraseña o credenciales de usuario, que almacene de forma cifrada. AWS Secrets Manager Se compone del valor secreto y sus

metadatos. El valor secreto puede ser binario, una sola cadena o varias cadenas. Para obtener más información, consulte ¿Qué hay en un secreto de Secrets Manager? en la documentación de Secrets Manager.

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. Hay cuatro tipos principales de controles de seguridad: 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 SIEM sistema 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 respuesta de seguridad

Una acción predefinida y programada que está diseñada para responder automáticamente a un evento de seguridad o remediarlo. Estas automatizaciones sirven como controles de seguridad detectables o adaptables que le ayudan a implementar las mejores prácticas AWS de seguridad. Algunos ejemplos de acciones de respuesta automática incluyen la modificación de un grupo VPC de seguridad, la aplicación de parches a una EC2 instancia de Amazon o la rotación de credenciales.

cifrado del servidor

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

política de control de servicios (SCP)

Una política que proporciona un control centralizado de los permisos de todas las cuentas de una organización en AWS Organizations. SCPsdefina barreras de protección o establezca límites a las acciones que un administrador puede delegar en los usuarios o roles. Puede utilizarlas SCPs como listas de permitidos o rechazados, para especificar qué servicios o acciones se encuentra

permitidos o prohibidos. Para obtener más información, consulte Políticas de control de servicios en la AWS Organizations documentación de.

punto de enlace de servicio

El 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

Una 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 que compartes con respecto a la seguridad y AWS 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.

SIEM

Consulte Información sobre seguridad y sistema de administración de eventos.

punto único de fallo (SPOF)

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

SLA

Consulte el acuerdo de nivel de servicio.

SLI

Consulte el indicador de nivel de servicio.

SLO

Consulte el objetivo de nivel de servicio.

split-and-seed modelo

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 obtener más información, consulte Enfoque gradual para modernizar las aplicaciones en. Nube de AWS

SPOF

Consulte el punto único de fallo.

esquema de estrellas

Estructura organizativa de una base de datos que utiliza una tabla de hechos grande para almacenar datos medidos o transaccionales y una o más tablas dimensionales más pequeñas para almacenar los atributos de los datos. Esta estructura está diseñada para usarse 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 con la aplicación de este patrón, consulte Modernización gradual de MicrosoftASP. NET(ASMX) servicios web de forma incremental mediante contenedores y Amazon API Gateway.

subred

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

control de supervisión y adquisición de datos (SCADA)

En la industria manufacturera, un sistema que utiliza hardware y software para monitorear 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 de síntesis

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

T

etiquetas

Pares de clave y valor que funcionan como metadatos para organizar los recursos de. AWS Las etiquetas pueden ayudarle a administrar, identificar, organizar, buscar y filtrar recursos. 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.

puerta de enlace de tránsito

Centro de tránsito de red que puede utilizar para interconectar sus redes VPCs y redes en las instalaciones. Para obtener más información, consulte ¿Qué es una puerta de enlace de tránsito? en la AWS Transit Gateway documentación de.

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

Concesión de permisos a un servicio que especifique para realizar tareas en su nombre 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 pequeño DevOps equipo 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. Para más información, consulte la guía Cuantificación de la incertidumbre en los sistemas de aprendizaje profundo.

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

Ver 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.

VPCmirando

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

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

SQLFunción que realiza un cálculo en un grupo de filas que se relacionan de alguna manera con el registro actual. Las funciones de ventana son útiles para procesar tareas, 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

Mira, escribe una vez, lee muchas.

WQF

Consulte el marco AWS de calificación de la carga de trabajo.

escribe una vez, lee muchas (WORM)

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

Z

ataque de día cero

Un 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.

aplicación zombi

Aplicación que utiliza un promedio CPU de 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.