Título

FASE 4.0: CI-CD Pipeline para migración/creación nuevas funciones serverless

Estado

Aceptada

Fecha

2023-10-28

Contexto

- Tenemos ya Migration Hub Refactor spaces configurado, entre otras tenemos una cuenta dedicada a nuestros microservicios donde podemos ir migrando las funciones de nuestro monolito de forma incremental. Recordemos nuestro que nuestro cliente nos solicitaba una solución que permitiera la evolución de los servicios y las funcionalidades mediante despliegues parciales por regiones o grupos y que además permita la adición de nuevas funciones en el futuro sin afectar a los servicios en operación. Teniendo esto en cuenta. ¿Como podemos realizarlo siguiendo las mejores prácticas?

Decisión

- Para conseguirlo hemos decidido utilizar un procedimiento de CI/CD, (integración continua/despliegue continuo), mediante aplicaciones administradas de AWS (CodePipeline, CodeCommit, CodeBuild, CodeDeploy Cloudformation). Desplegaremos 2 cuentas diferentes (una para desarrollo/preproducción y otra para Produccion. Tener 2 cuentas nos permitirá:
- **Ventajas**
    - **Mayor aislamiento y seguridad**: Cada cuenta de AWS es independiente, lo que significa que los recursos y las credenciales están aislados entre sí. Esto puede ayudar a reducir el riesgo de que un problema en una cuenta afecte a otras cuentas.

        - **Mayor flexibilidad**: Las cuentas de AWS pueden tener diferentes configuraciones y límites, lo que permite una mayor flexibilidad en la implementación y el uso de recursos.
        - **Mayor control**: Las cuentas de AWS permiten un mayor control sobre el acceso y la administración de los recursos.
- **Desventajas**
    - **Mayor complejidad:** La gestión de varias cuentas puede ser más compleja que la gestión de una sola cuenta.
    - **Mayor costo:** El uso de varias cuentas puede aumentar los costos debido a la necesidad de duplicar algunos recursos.
    - Para suplir estas carencias usaremos organizations que nos permite administrar múltiples cuentas, consolidar el Billing, conseguir beneficios por volumen de uso, compartir Instancias reservadas entre cuentas y Savings plans.

Consecuencias

Positivas:

- **Mayor velocidad de entrega**: Un pipeline de CI/CD automatiza el proceso de construcción, prueba y despliegue de software, lo que permite a los equipos de desarrollo lanzar nuevas funciones más rápidamente.
- **Mayor calidad del software**: La integración continua (CI) y la entrega continua (CD) aseguran que el software se pruebe y se implemente automáticamente después de cada cambio en el código fuente. Esto ayuda a detectar errores temprano en el proceso de desarrollo y garantiza que el software entregado sea de alta calidad.
- **Mayor eficiencia**: Un pipeline de CI/CD automatiza tareas repetitivas, como la compilación y las pruebas, lo que permite a los desarrolladores centrarse en tareas más importantes.
- **Mayor colaboración**: Un pipeline de CI/CD fomenta la colaboración entre los miembros del equipo al proporcionar una plataforma centralizada para la construcción, prueba y despliegue de software.

Negativas:

- **Costo inicial alto**: Configurar un pipeline de CI/CD puede ser costoso debido a la necesidad de herramientas y servicios adicionales.
- **Curva de aprendizaje elevada**: Los equipos pueden necesitar tiempo para aprender cómo configurar y utilizar un pipeline de CI/CD.
- **Mayor complejidad**: Un pipeline de CI/CD puede ser complejo debido a la necesidad de integrar múltiples herramientas y servicios.
- **Mayor riesgo**: La automatización del proceso de construcción, prueba y despliegue puede aumentar el riesgo de errores si no se configura correctamente.

Compliance

- Nuestro cliente opera en Europa y Estados Unidos Por lo que ha de cumplir con los estándares GDPR e HIPAA)

Procedimiento

Describamos en detalle el flujo de trabajo de Code Pipeline para cada paso.

1. Un usuario de IAM que pertenece al equipo de desarrollo inicia sesión en la interfaz de línea de comandos de AWS (AWS CLI) desde una máquina local utilizando un secreto de IAM y una clave de acceso.
2. A continuación, el usuario de IAM asume el rol de IAM para las actividades correspondientes: AWS Code Commit , AWS CodeBuild , AWS CodeDeploy , AWS CodePipeline Execution e implementa el código para la preproducción.
3. Un AWS CodePipeline típico consta de etapas de construcción, prueba e implementación. En la etapa de compilación, el servicio AWS CodeBuild genera la pila de plantillas de Cloudformation (*template-export.yaml*) en Amazon S3 .
4. En la etapa de implementación, AWS CodePipeline utiliza una plantilla de CloudFormation (*un archivo yaml*) para implementar el código desde un bucket S3 que contiene los puntos finales de la API de la aplicación a través de Amazon API Gateway en el entorno de preproducción.

5. Si la implementación es satisfactoria en el entorno de preproducción indicado en el punto 4, el último paso en el flujo de trabajo de la canalización es implementar los cambios en el código de la aplicación en el entorno de producción asumiendo la función de IAM de producción de STS.

Dado que AWS CodePipeline está completamente automatizado, podemos utilizar el mismo canal cambiando entre cuentas de preproducción y producción. Estas cuentas asumen el rol de IAM apropiado para el entorno de destino e implementan la compilación validada en ese entorno mediante plantillas de CloudFormation.

1. **Notas**

- Author: Fran Díaz
- Version: 0.2
- Changelog:
    - 0.1: Versión Inicial propuesta.
    - 0.2: Cambios propuestos