LATAM AIRLINES

Challenge DevSecOps/SRE

En Advanced Analytics se construyen productos de datos que al ser consumidos añaden valor a diferentes áreas de nuestra aerolínea. Los servicios exhiben datos obtenidos por procesos de analítica mediante APIs, tablas y procesos recurrentes. Uno de los principales pilares de nuestra cultura es la **resiliencia y calidad** de lo que construimos. Esto nos permite preservar la correcta operación de nuestros servicios y no deteriorar el valor añadido hacia el resto de áreas.

Instrucciones de entrega:

1. Enviar un POST request tipo JSON al endpoint: https://advana-challenge-check-api-cr-k4hdbggvoq-uc.a.run.app/devops con tu nombre, mail y repositorio git:

```
{
"name": "Pedro Picapiedra",
"mail": "Pedro.picapiedra@example.com",
"github_url":
"https://github.com/ppicapiedra/tu-latam-challenge.git"
}
```

2. Se aceptarán cambios en el repositorio hasta el día especificado en el e-mail junto a este desafío

Instrucciones previas

1) Utiliza un repositorio público de git para resolver el desafío (GitHub, BitBucket, GitLab, etc).
2) Utiliza una rama master y otra develop al resolver el problema. Opcionalmente, utilizar alguna práctica de desarrollo de GitFlow.
3) En el repositorio deben estar todos los archivos utilizados para la resolución del desafío.
4) Ten en cuenta que nuestro lenguaje preferido es Python. De todas formas, usa el que más te acomode.
5) Escribe un README.md para responder cada punto del desafío. Escribe supuestos y cualquier comentario que nos ayude a entender tu solución y tu forma de resolver el problema.
6) Utiliza cualquier buena práctica que consideres en el repositorio, código o flujo de desarrollo para demostrar tu conocimiento.
7) **No necesitas resolver por completo ni a la perfección el desafío, pero te recomendamos detallar claramente en el .md cómo mejorar cada parte de tu ejercicio en caso de que tenga opción de mejora o no te haya dado el tiempo**

para resolverlo como te hubiese gustado. Lo importante es demostrar tus conocimientos.

¡ÉXITO!

Desafío Técnico DevSecOps/SRE

Contexto

Se requiere un sistema para ingestar y almacenar datos en una DB con la finalidad de hacer analítica avanzada. Posteriormente, los datos almacenados deben ser expuestos mediante una API HTTP para que puedan ser consumidos por terceros.

Objetivo

Desarrollar un sistema en la nube para ingestar, almacenar y exponer datos mediante el uso de IaC y despliegue con flujos CI/CD. Hacer pruebas de calidad, monitoreo y alertas para asegurar y monitorear la salud del sistema.

Parte 1:Infraestructura e IaC

1. Identificar la infraestructura necesaria para ingestar, almacenar y exponer datos:
    a. Utilizar el esquema Pub/Sub (no confundir con servicio Pub/Sub de Google) para ingesta de datos
    b. Base de datos para el almacenamiento enfocado en analítica de datos
    c. Endpoint HTTP para servir parte de los datos almacenados
2. (Opcional) Deployar infraestructura mediante Terraform de la manera que más te acomode. Incluir código fuente Terraform. No requiere pipeline CI/CD.

Comentarios:

- **La estructura de datos no es relevante para el problema, asume cualquiera**
- **Existen múltiples formas de resolver el problema y escoger la infraestructura, recomendamos simplicidad**

Parte 2: Aplicaciones y flujo CI/CD

1. API HTTP: Levantar un endpoint HTTP con lógica que lea datos de base de datos y los exponga al recibir una petición GET
2. Deployar API HTTP en la nube mediante CI/CD a tu elección. Flujo CI/CD y ejecuciones deben estar visibles en el repositorio git.
3. (Opcional) Ingesta: Agregar suscripción al sistema Pub/Sub con lógica para ingresar los datos recibidos a la base de datos. El objetivo es que los mensajes recibidos en un tópico se guarden en la base de datos. **No requiere CI/CD.**
4. Incluye un diagrama de arquitectura con la infraestructura del punto 1.1 y su interacción con los servicios/aplicaciones que demuestra el proceso end-to-end de ingesta hasta el consumo por la API HTTP

Comentarios:

- **Recomendamos usar un servicio serverless mediante Dockerfile para optimizar el tiempo de desarrollo y deployment para la API HTTP**
- **Es posible que lógica de ingesta se incluya nativamente por tu servicio en la nube. De ser así, solo comentar cómo funciona**
- **Al ser 1.3 opcional, el flujo de ingesta de datos quedará incompleto, está bien**
- **Para el punto 1.4 no se requiere un diagrama profesional ni que siga ningún estándar de diagramas en específico**

Parte 3: Pruebas de Integración y Puntos Críticos de Calidad

1. Implementa en el flujo CI/CD en test de integración que verifique que la API efectivamente está exponiendo los datos de la base de datos. Argumenta.
2. Proponer otras pruebas de integración que validen que el sistema está funcionando correctamente y cómo se implementarían.
3. Identificar posibles puntos críticos del sistema (a nivel de fallo o performance) diferentes al punto anterior y proponer formas de testearlos o medirlos (no implementar)
4. Proponer cómo robustecer técnicamente el sistema para compensar o solucionar dichos puntos críticos

Comentarios:

- **Los test de integración no necesariamente deben cubrir todos los casos de uso**

Parte 4: Métricas y Monitoreo

1. Proponer 3 métricas (además de las básicas CPU/RAM/DISK USAGE) críticas para entender la salud y rendimiento del sistema end-to-end
2. Proponer una herramienta de visualización y describe textualmente qué métricas mostraría, y cómo esta información nos permitiría entender la salud del sistema para tomar decisiones estratégicas
3. Describe a grandes rasgos cómo sería la implementación de esta herramienta en la nube y cómo esta recolectaría las métricas del sistema
4. Describe cómo cambiará la visualización si escalamos la solución a 50 sistemas similares y qué otras métricas o formas de visualización nos permite desbloquear este escalamiento.
5. Comenta qué dificultades o limitaciones podrían surgir a nivel de observabilidad de los sistemas de no abordarse correctamente el problema de escalabilidad

Comentarios:

- **No se requiere implementación**

Parte 5: Alertas y SRE (Opcional)

1. Define específicamente qué reglas o umbrales utilizarías para las métricas propuestas, de manera que se disparan alertas al equipo al decaer la performance del sistema. Argumenta.
2. Define métricas SLIs para los servicios del sistema y un SLO para cada uno de los SLIs. Argumenta por qué escogiste esos SLIs/SLOs y por qué desechaste otras métricas para utilizarlas dentro de la definición de SLIs.

Comentarios:

- **No se requiere implementación**