efRouting

Prueba Técnica: Desarrollador(a) Fullstack

A continuación, se describe una prueba técnica que involucra el uso de **Amazon DynamoDB**, **AWS Lambda** (con Python), la **API pública de SpaceX** y el despliegue de una aplicación web mediante **Amazon ECS Fargate**. La meta es obtener información de lanzamientos en tiempo real, almacenarla en DynamoDB y mostrarla a través de un sitio web responsivo.

1. Creación de la base de datos en Amazon DynamoDB

Se requiere la creación de una tabla en Amazon DynamoDB que almacene la información relevante de los lanzamientos de SpaceX. Deberá contener, al menos, una clave primaria que identifique de forma única cada lanzamiento (por ejemplo, el Launch ID) y campos como el nombre de la misión, el nombre del cohete, la fecha del lanzamiento y el estado del mismo (success, failed, upcoming). Además, es recomendable incluir cualquier otro atributo que consideres de interés, asegurándote de describir en la documentación los pasos necesarios para configurar la tabla.

2. Función Lambda en Python para consumir la API de SpaceX

Deberás implementar una función Lambda en Python que, **cada seis horas**, consulte directamente los endpoints disponibles de la API de SpaceX para obtener los datos de lanzamientos (por ejemplo, https://api.spacexdata.com/v4/launches o https://api.spacexdata.com/v4/launches/upcoming). La API de SpaceX provee información detallada sobre cada lanzamiento, incluyendo nombre de la misión, cohete utilizado, fecha, estado y más, y su documentación puede consultarse en https://github.com/r-spacex/SpaceX-API. Tu Lambda deberá parsear la información necesaria (misión, fecha, rocket, estado, etc.) y guardarla o actualizarla en la tabla de DynamoDB. Además, la función Lambda deberá, en una invocación manual, devolver un JSON con la información insertada o un recuento de los registros para propósitos de prueba. Se espera que exista al menos una prueba unitaria que valide la lógica principal de parsing e inserción/actualización de los datos.

3. Aplicación web con despliegue en Amazon ECS Fargate

Finalmente, se debe construir y desplegar una aplicación web dentro de un contenedor Docker, utilizando Amazon ECS con Fargate para su ejecución. Esta aplicación debe leer los datos almacenados en DynamoDB y mostrarlos en un formato claro y atractivo (por ejemplo, un gráfico que ilustre la cantidad de lanzamientos por año, una tabla con filtros o una línea de tiempo). La interfaz debe ser responsiva y agradable, valorándose positivamente el uso de bibliotecas de visualización (como Chart.js, D3.js o Plotly) y, especialmente para los postulantes con foco en el front-end, se apreciará el uso de React. En la documentación, deberás detallar cómo se construye la imagen Docker (incluyendo el Dockerfile), cómo se registra en Amazon ECR (u otro repositorio de contenedores) y cómo se configura el servicio de ECS Fargate para que la aplicación sea accesible mediante una URL pública.

Entregables

• **Documentación detallada** con los pasos para crear la tabla en DynamoDB, configurar la Lambda (incluyendo cómo programarla cada seis horas) y desplegar la aplicación web en un contenedor Docker usando ECS Fargate.

• **Código fuente** de la Lambda en Python (con pruebas unitarias) y de la aplicación web (front-end y/o back-end, según corresponda).

• **Dockerfile** con las instrucciones para construir la imagen de la aplicación web.

• **Instrucciones de despliegue** (cómo subir la imagen a ECR o a otro repositorio, cómo configurar la tarea y el servicio de ECS Fargate, etc.).

Criterios de evaluación

• Uso correcto de Amazon DynamoDB (configuración de la tabla, definición de claves, inserción/actualización de datos).

• Diseño y calidad del código Python para la Lambda, incluyendo manejo de errores y pruebas unitarias.

• Solidez y claridad de la solución de Docker + ECS Fargate (Dockerfile bien definido, despliegue sin errores, configuración adecuada).

• Capacidad de documentación y explicación detallada de cada paso.

• Calidad y diseño de la interfaz de la aplicación web (responsividad, facilidad de uso, buena presentación de datos).