Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz

Documento de Arquitectura

Repositorio: https://github.com/topsy-13/LogisticaSA-CloudSolution

Nota: Todos los documentos y código para el trabajo se encuentran en el repo.

Introducción

Logística Inteligente S.A. es una empresa especializada en la gestión y optimización de rutas de entrega para empresas de transporte. A través del uso de algoritmos avanzados de optimización y predicción, la compañía busca mejorar la eficiencia de las rutas de transporte de sus clientes, reduciendo costos y tiempos operativos.

La necesidad de empresas como Logística Inteligente S.A. radica en la capacidad de mejorar las operaciones de transporte en un mercado cada vez más competitivo. Para las empresas de transporte, la optimización de rutas no solo reduce los costos de combustible y mantenimiento, sino que también mejora el tiempo de entrega, lo cual es clave para la satisfacción del cliente.

El Cloud Computing, como el que ofrece AWS, trae consigo una serie de ventajas significativas para empresas como Logística Inteligente S.A. Al migrar las operaciones a la nube, la empresa puede aprovechar la escalabilidad casi ilimitada, ajustando sus recursos de procesamiento y almacenamiento en función de la demanda. Esto permite optimizar costos al pagar solo por lo que se utiliza, en lugar de mantener una infraestructura física costosa. Además, la nube proporciona alta disponibilidad y recuperación ante desastres, asegurando que los sistemas permanezcan operativos sin interrupciones críticas. Con soluciones como **Amazon EC2** para procesamiento de datos y **Amazon RDS** para la gestión de bases de datos, se garantiza una infraestructura altamente escalable y de alto rendimiento. Además, la integración de **AWS Lambda** permite ejecutar algoritmos de optimización sin necesidad de servidores dedicados, mejorando la eficiencia y reduciendo costos operativos. El uso de **Amazon S3** para realizar respaldos de datos y almacenar modelos asegura que los datos críticos de clientes y rutas estén protegidos y disponibles en caso de fallos, lo que fortalece la seguridad y la resiliencia del sistema.

Objetivos Generales

- Definir una infraestructura escalable y eficiente en la nube para soportar el procesamiento y análisis de datos de rutas, utilizando tecnologías de AWS.

Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz

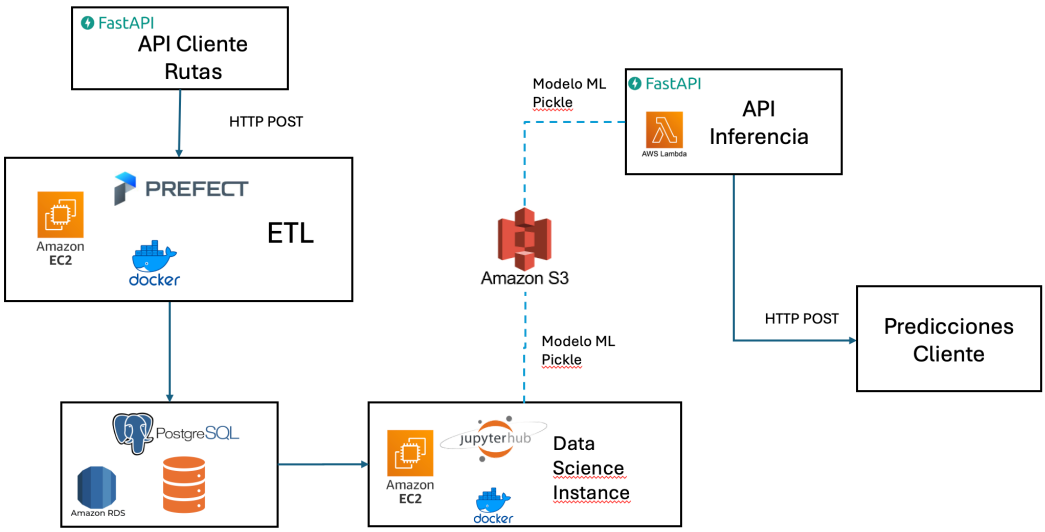
Optimizar las operaciones de entrega de transporte, reduciendo costos y tiempos mediante la implementación de algoritmos avanzados de optimización de rutas.

Objetivos Específicos

1. Implementar instancias EC2 para el procesamiento de datos de rutas en tiempo real, aprovechando su capacidad escalable para manejar grandes volúmenes de información.
2. Utilizar RDS para el almacenamiento de datos, asegurando la disponibilidad y seguridad de la información de clientes y rutas.
3. Desarrollar modelos de optimización en Python, aprovechando las bibliotecas especializadas para la optimización y el análisis de datos geográficos.
4. Integrar AWS Lambda para ejecutar los algoritmos de optimización de forma eficiente y a demanda, reduciendo el costo de ejecución en periodos inactivos.
5. Implementar Amazon S3 para la realización de copias de seguridad automáticas de la base de datos y archivos críticos, asegurando la disponibilidad de los datos en caso de desastres.

Arquitectura Propuesta

La arquitectura propuesta a continuación está diseñada para ser creada de manera independiente para cada cliente, lo que garantiza una gestión separada y segura de los datos de cada uno. Además, este enfoque nos facilita el cálculo de costos por cliente, lo que resulta ventajoso para la venta del servicio de manera individual.



Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz

Descripción de la Arquitectura

1. **API Cliente (FastAPI)**: Los clientes interactúan con esta API para enviar rutas y datos relevantes a través de un método HTTP POST. La API se comunica con el sistema de ETL.

2. **ETL (Extract, Transform, Load)**: Implementado con Prefect y Docker, el ETL se ejecuta en una instancia de Amazon EC2. Este sistema se encarga de procesar los datos enviados desde la API Cliente y los almacena en una base de datos PostgreSQL gestionada por Amazon RDS.

3. **Data Science Instance**: Esta instancia, configurada con JupyterHub y Docker en Amazon EC2, se utiliza para entrenar los modelos de machine learning. Los modelos entrenados son exportados y almacenados en formato Pickle en Amazon S3.

4. **Almacenamiento de Modelos (Amazon S3)**: Los modelos de machine learning entrenados y guardados en formato Pickle se almacenan aquí para ser utilizados posteriormente en el proceso de inferencia.

5. **API de Inferencia (FastAPI y AWS Lambda)**: Esta API toma los modelos almacenados en Amazon S3 y los utiliza para realizar predicciones. Está implementada usando FastAPI y AWS Lambda. El resultado de las predicciones se envía de vuelta al cliente a través de una llamada HTTP POST.

6. **Predicciones Cliente**: Los resultados de las predicciones generadas por la API de Inferencia son enviados al cliente, completando así el ciclo.

Detalles adicionales

- **PostgreSQL (Amazon RDS)**: La base de datos donde se almacenan los datos transformados luego del proceso ETL.
- **Amazon S3**: Es el repositorio central para almacenar los modelos de machine learning en formato Pickle.
- **Amazon EC2**: Utilizado tanto para la instancia de procesamiento de ETL como para la instancia de ciencia de datos.
- **Docker**: Permite la contenedorización de las instancias de ETL y ciencia de datos, garantizando un entorno controlado y replicable.

Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz

Descripción de los servicios utilizados

Instancias EC2 para almacenamiento de datos

Amazon Elastic Compute Cloud (EC2) proporciona servidores virtuales escalables que permiten realizar tareas computacionalmente intensivas, como el procesamiento de grandes volúmenes de datos de rutas en tiempo real. Esto asegura una capacidad de procesamiento adecuada sin comprometer el rendimiento de la máquina.

Las características que seleccionamos para las instancias EC2 son las siguientes:

1. **Sistema Operativo:** Ubuntu Linux.
2. **Número de instancias:** 2
    - Una instancia se encargará de la conexión con FastAPI para la recolección de información de rutas enviadas por los clientes.
        1. **RAM**: 2 GB.
        2. **Disco (Almacenamiento EBS)**: 20 GB de almacenamiento SSD (gp3).
        3. **Justificación**: FastAPI es un framework ligero y eficiente para la creación de APIs. Dado que esta instancia manejará principalmente solicitudes HTTP y procesará datos de rutas en tiempo real, 2 GB de RAM serán suficientes para garantizar un rendimiento adecuado sin comprometer la velocidad de respuesta. El almacenamiento de 20 GB será más que suficiente para gestionar el procesamiento temporal de datos, ya que estos serán enviados a la base de datos RDS.

    - La segunda instancia proporcionará un entorno de JupyterHub en línea para que el equipo de ciencia de datos pueda trabajar de manera colaborativa y realizar ajustes al modelo y análisis de los datos.
        1. **RAM:** 16 GB.
        2. **Disco (Almacenamiento EBS):** 100 GB de almacenamiento SSD (gp3)
        3. **Justificación:** JupyterHub es una herramienta colaborativa utilizada por el equipo de ciencia de datos para desarrollar y ejecutar experimentos. Dado que puede haber varios usuarios trabajando simultáneamente, 16 GB de RAM proporcionarán el margen necesario para ejecutar notebooks complejos sin

interrupciones. El almacenamiento de 100 GB está recomendado para gestionar datasets medianos y los resultados generados por el equipo, así como para almacenar múltiples notebooks y experimentos.

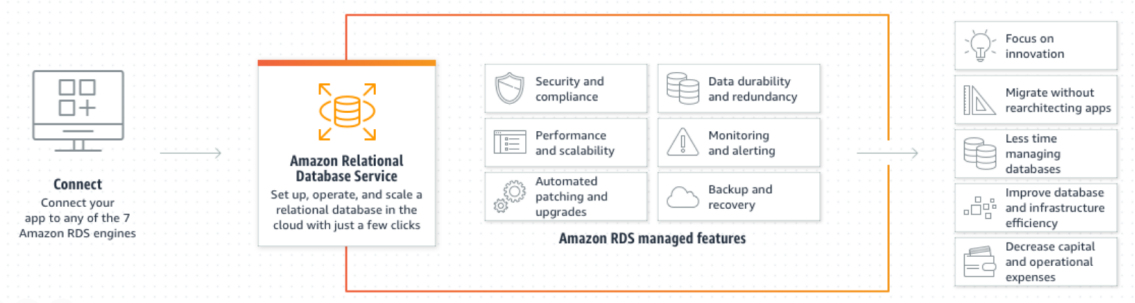
3. **Escalabilidad**:
    - **Auto Scaling**: Ajusta automáticamente la capacidad según la carga.
    - **Tipos de Instancias**: Selección flexible de instancias t3.medium, para Jupyter Hub t3.large
4. **Flexibilidad**:
    - **Configuración Personalizada**: RAM, CPU y almacenamiento ajustables.
    - **EBS**: Almacenamiento persistente y seguro.
5. **Elasticidad**:
    - **Cambio de Tamaño**: Permite cambiar instancias en tiempo real.
    - **Load Balancer**: Distribución eficiente de tráfico.
6. **Costo-Eficiencia**:
    - **Instancias Spot**: Menor costo aprovechando capacidad no utilizada.
    - **Pago por Uso**: Solo se paga por las instancias activas.

RDS para almacenar información de clientes y rutas:

Amazon Relational Database Service (RDS) es un servicio de base de datos relacional administrado que se utilizará para almacenar información de los clientes y las rutas de entrega. Este servicio automatiza las tareas de administración de la base de datos, como las copias de seguridad, el aprovisionamiento y la actualización, lo que permite un manejo eficiente y seguro de los datos.

Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz



Características seleccionadas:

1. Tipo de instancia: db.t3.micro (2vCPU, 4 GB de RAM)
2. Base de datos PostgreSQL:
   - Compatible con la información que se va a utilizar.
3. Almacenamiento:
   - SSD de uso general (gp3) con una capacidad inicial de 20GB.
   - Se deshabilita el autoscaling.
4. Copia de seguridad automática
5. Replicación en varias zonas de disponibilidad (Multi-AZ): Mejora la disponibilidad y proporciona conmutación por error automática en caso de una interrupción.

Justificación:

La elección de estas especificaciones de RDS ofrece a la empresa la capacidad de manejar las operaciones de la base de datos, automatizando tareas críticas como copias de seguridad, actualizaciones y administración de almacenamiento.

AWS Lambda para la inferencia de modelos

AWS Lambda es un servicio de computación sin servidor que permite ejecutar código en respuesta a eventos sin necesidad de aprovisionar ni gestionar servidores. En esta arquitectura, AWS Lambda se utilizará para ejecutar los algoritmos de optimización de manera escalable y bajo demanda, lo que significa que se activa únicamente cuando se requiere realizar una optimización.

- **Ejecución Bajo Demanda**: AWS Lambda permite ejecutar el algoritmo de optimización solo cuando es necesario, lo que optimiza el uso de recursos al evitar el aprovisionamiento constante de instancias dedicadas. Esto reduce costos, ya que solo se paga por el tiempo de ejecución efectivo del código.

- **Escalabilidad Automática**: Lambda escala automáticamente para gestionar múltiples solicitudes simultáneamente, lo que lo convierte en una excelente opción para ejecutar algoritmos de optimización que puedan ser invocados por varios usuarios o sistemas en paralelo.
- **Integración con Otros Servicios AWS**: Lambda se integra fácilmente con servicios como Amazon S3, donde se almacenan los modelos de machine learning, y con API Gateway o FastAPI para recibir solicitudes de optimización. Esta integración fluida permite ejecutar procesos complejos de forma sencilla y automatizada.
- **Costo-Eficiencia**: Lambda es particularmente útil para aplicaciones con cargas de trabajo variables o impredecibles, ya que su modelo de facturación basado en el tiempo de ejecución garantiza que solo se incurre en costos cuando realmente se utiliza. No hay cargos por tiempo ocioso.
- **Sin Servidor (Serverless)**: La naturaleza sin servidor de AWS Lambda elimina la necesidad de gestionar la infraestructura, como el aprovisionamiento, el escalado o el mantenimiento de servidores. Esto permite al equipo centrarse en el desarrollo y la mejora de los algoritmos de optimización, sin preocuparse por la gestión operativa.

Justificación:

El uso de AWS Lambda para ejecutar los algoritmos de optimización a demanda proporciona una solución flexible y escalable que optimiza el uso de recursos y minimiza costos. Además, su integración nativa con otros servicios de AWS simplifica la automatización de los procesos, asegurando un flujo eficiente y altamente disponible.

Amazon S3 para almacenamiento de modelos

Amazon S3 (Simple Storage Service) es un servicio de almacenamiento de objetos que proporciona escalabilidad, disponibilidad y seguridad para el almacenamiento de datos. En esta arquitectura, Amazon S3 se utiliza para almacenar los modelos de machine learning entrenados en formato Pickle, permitiendo su uso en procesos de inferencia bajo demanda.

Características seleccionadas para Amazon S3:

Almacenamiento de objetos:

- Amazon S3 es ideal para almacenar modelos de machine learning en formato Pickle, lo que permite un acceso rápido y eficiente desde diferentes instancias y servicios, como AWS Lambda o las instancias de inferencia.

Alta durabilidad y disponibilidad:

- S3 garantiza una durabilidad del 99.999999999% y una alta disponibilidad, asegurando que los modelos de machine learning almacenados estén siempre accesibles para ser utilizados en procesos de predicción.

Seguridad:

- Los datos almacenados en Amazon S3 están protegidos mediante políticas de control de acceso (IAM) y cifrado tanto en tránsito como en reposo, lo que garantiza la seguridad de los modelos y su integridad.

Escalabilidad ilimitada:

- Amazon S3 proporciona un almacenamiento escalable sin límites predefinidos, lo que permite que la infraestructura crezca sin la necesidad de preocuparse por la capacidad de almacenamiento. Esto es crucial para gestionar múltiples versiones de modelos de machine learning o grandes volúmenes de datos relacionados.

Integración nativa con otros servicios AWS:

- S3 se integra fácilmente con otros servicios como AWS Lambda, que utiliza los modelos almacenados para realizar predicciones, y Amazon SageMaker para entrenar y desplegar nuevos modelos.

Justificación:

Amazon S3 es la elección ideal para almacenar los modelos de machine learning debido a su durabilidad, seguridad y capacidad de escalabilidad. Su alta disponibilidad garantiza que los modelos estén siempre listos para ser utilizados en el proceso de inferencia, mientras que su integración con otros servicios de AWS permite un flujo de trabajo eficiente y automatizado.

Contenerización de Servicios con Docker

El uso de Docker permite la creación de entornos aislados y portables que aseguran la coherencia del software y los entornos de desarrollo. La contenerización con Docker garantiza que todas las dependencias necesarias para la ejecución de nuestras aplicaciones (FastAPI, Prefect, JupyterHub, etc.) estén incluidas y

configuradas correctamente dentro de cada contenedor, lo que facilita tanto el despliegue como el mantenimiento.

- **Portabilidad**: Los contenedores Docker son independientes del sistema operativo subyacente, lo que permite ejecutar las aplicaciones de manera consistente en cualquier entorno, ya sea en local, en servidores en la nube o en entornos híbridos.
- **Reproducibilidad**: Al contenerizar cada servicio, nos aseguramos de que todas las dependencias están incluidas dentro del contenedor, eliminando posibles problemas de configuración entre diferentes entornos de desarrollo, pruebas y producción.
- **Escalabilidad**: Docker se integra fácilmente con servicios de orquestación como Kubernetes, lo que facilita el escalado horizontal de los contenedores, permitiendo agregar o eliminar instancias según sea necesario.
- **Seguridad**: Cada contenedor está aislado, lo que significa que cualquier problema en un servicio no afectará a otros. Además, los contenedores pueden configurarse para tener permisos limitados, aumentando la seguridad general del sistema.
- **Optimización de Recursos**: Docker permite la ejecución de múltiples contenedores en una misma instancia EC2, optimizando el uso de recursos como CPU y RAM. Esto ayuda a reducir costos y a mejorar el rendimiento del sistema.

Costos

Los costos fueron calculados en un Jupyter Notebook construido para ello, usando como base los precios unitarios del AWS Price Calculator.

Para consultar el Jupyter Notebook ir a este: link.

Acá un resumen:

Lista Tecnica

	Tipo Instancia	Precio por Hora	Instancias	Almacenamiento	Almacenamiento en GB	Memoria	Requests Diarios	Tiempo por Request	Almacenamiento Diario
Instancia EC2 Grande	m5.large	0.096 USD	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Instancia EC2 Mediana	t3.medium	0.0464 USD	NaN	NaN	NaN	NaN	NaN	NaN	NaN
RDS Postgres	NaN	0.145 USD	db.t3.medium	100 GB	NaN	NaN	NaN	NaN	NaN
S3 Almacenamiento de Modelos	NaN	NaN	NaN	S3 Standard (1000 GB)	1000 GB	NaN	NaN	NaN	NaN
AWS Lambda	NaN	NaN	NaN	NaN	NaN	1 GB	1000	5 segundos	2 GB

Logística Inteligente S.A.

Julián Santos, Santiago Ramírez, Daniel Pombo, Nataly Díaz

Resumen de Costos Mensuales

	Recurso	Costo Mensual (USD)
0	EC2 Grande	69.120000
1	EC2 Mediana	33.408000
2	RDS Postgres	127.400000
3	Almacenamiento S3	23.000000
4	Lambda	2.512005
5	Total	255.440005