PROPUESTA DE ARQUITECTURA EN LA NUBE Y ORQUESTACIÓN

Arquitectura General del Sistema

Usuario

S3 (Entrada de Archivos)

Versionamiento

AWS Glue (Preprocess)

S3 (Clean)

Aws Glue o SageMaker (Training)

S3 (Model)

Inferencia

S3 (Salidas)

Reentrenamiento

ORQUESTACIÓN DEL FLUJO

El orquestador principal será WorkFlow adicionalmente el EventBridge, El EventBridge te ayuda detectando cuando hay un nuevo archivo en tu s3 y desplegar reentrenamientos automáticos.

* Las ETL las realiza AWS Glue
* Pipeline de Entrenamiento con puede ser AWD Glue o Amazon SageMaker
* Evaluación del Modelo
* Correr con Pyspark
* Las predicciones las almacena en S3/Salidas
* Registra las métricas en logs en CloudWatch

COMPONENTES DETALLADO

1. Ingesta de Datos

* Amazon S3: Almacenamiento para la entrada de datos.
* Amazon EventBridge: Detecta Eventos (Entre ellos Nuevos Archivos) y realiza lambda automáticos.

1. Preprocesamiento

* AWS Glue (con PySpark): Limpieza de texto, normalización y escritura en S3.

1. Entrenamiento del Modelo

* AWS Glue (con PySpark): Utilizamiento de hash

1. Monitoreo y Trazabilidad

* CloudWatch: Logs de cada paso del pipeline y alerta de fallos.
* Amazon Glue Pipelines: Trazabilidad completa de datasets, modelos y ejecuciones.

ESCALABILIDAD Y RENDIMIENTO

* Glue: Autoescalable para ETL pesados, entrenamiento con instancias distribuibles con PySpark.
* S3: Almacenamiento sin límite y alta disponibilidad.

REPRODUCIBILIDAD Y BUENAS PRÁCTICAS MLOPS

* Pipelines definidos como código
* Control de Versiones: Datos (S3 Version), código git y Modelos (Model Registry)
* Reprocesamiento reproducible desde raw + parámetros registrados.
* Modularidad y separación clara entre etapas (ETL, Training, Inference).

JUSTIFICACIÓN DE DECISIONES

* S3: Almacenamiento económico, escalable y soporta triggers.
* Glue: ETL Serveless, escala según volumen. Ideal para preprocesamiento textual en batch y servicio especializado para ML, incluye pipelines.
* WorkFlow: Orquestación robusta con control de errores, reintentos y monitoreo.

CONTEXTO DE NEGOCIO

Se eligió como datos de entrenamiento el Twitter Customer Support, la idea es el entrenamiento de un modelo de clasificación de la variable Inbound determina si el mensaje fue saliente o recibido.

DICCIONARIO DE DATOS

Text: Texto del Mensaje

Created: Fecha de Creación del Mensaje

Inbound: Si fue un mensaje recibido o enviado

Response\_tweet\_id: Tweet al que responde, sí aplica

Los scripts se pueden ejecutar en un entorno de Amazon Glue Job (PySpark) con Glue +3.

ESQUEMA PIPELINE DE ENTRENAMIENTO DEL MODELO E INFERENCIA

Leer CSV

Limpieza y Preprocesamiento

Tokenización

Train 70%

Train ML Logistic

Store S3 Model

Inferencia 30%

Predicción

Store S3 Predicción

OBJETIVO

Construir un pipeline reducible y mantenible que cubra:

* Limpieza de los datos y Versionamiento
* Entrenamiento del Modelo
* Inferencia
* Reentrenamiento

FLUJO DEL PIPELINE

Cada una de las etapas se le realizó un notebook y un script, en el notebook se explica paso a paso las decisiones tomadas y el script es el archivo limpio para que se pueda replicar la prueba, se adjuntaran evidencias de ejecución.

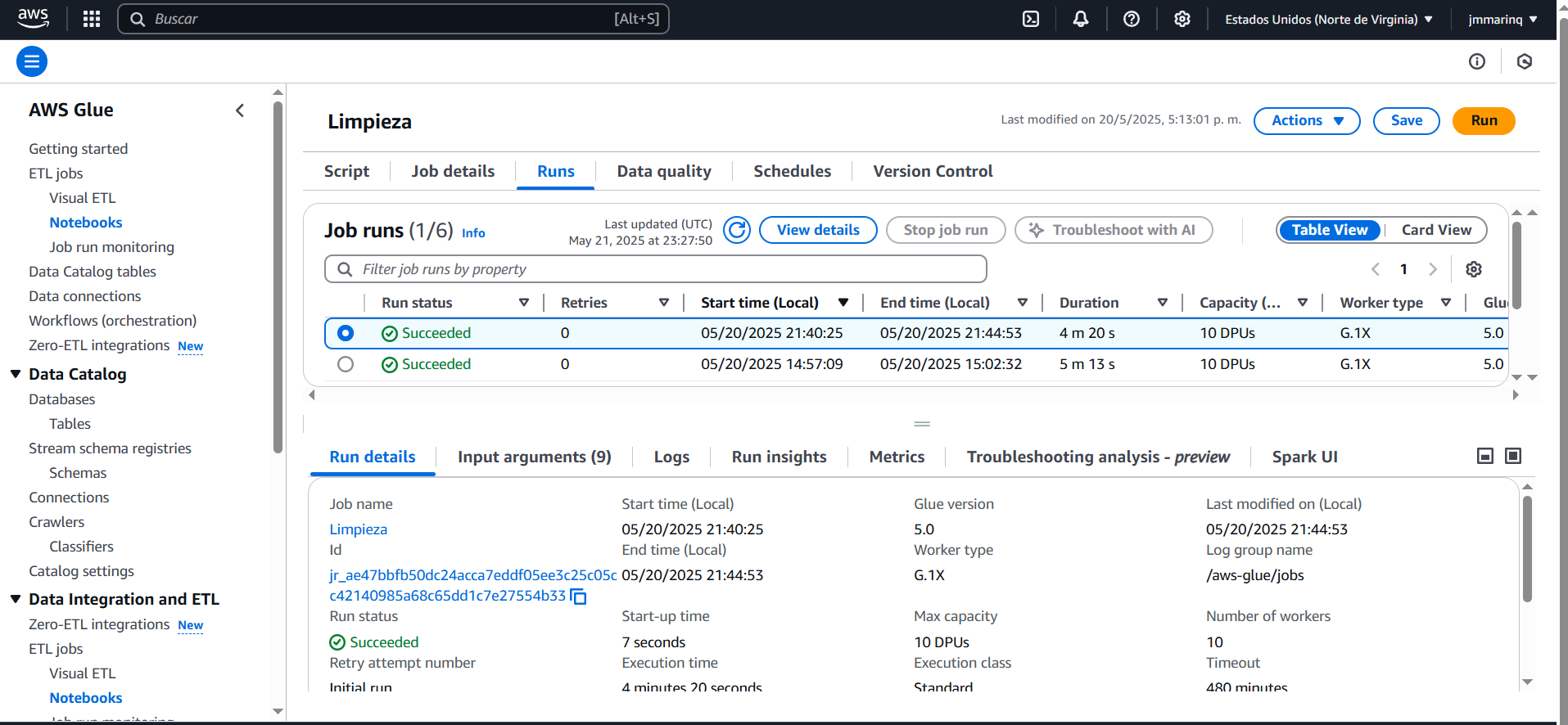
Versionamiento

El script Versionamiento propone una forma de utilizar el versionamiento automático, la idea es utilizar este script en el flujo de workflow para definir la variable de entorno llamada versión, lo que hace el script es ir a la ruta de s3 en donde se colocan los archivos para test, que es donde se ponen los archivos que se deciden testear, cada vez que se corra el modelo, mostrará las veces que se ha utilizado. Se pueden hacer distintas formas de versionamiento tanto de corridas como de entrenamiento del modelo, en este caso lo enfocamos a la corrida como tal, si lo quisiéramos afrontar al entrenamiento del modelo, debería ponersen las instrucciones en el momento en el que se utiliza el Python de reentrenamiento.

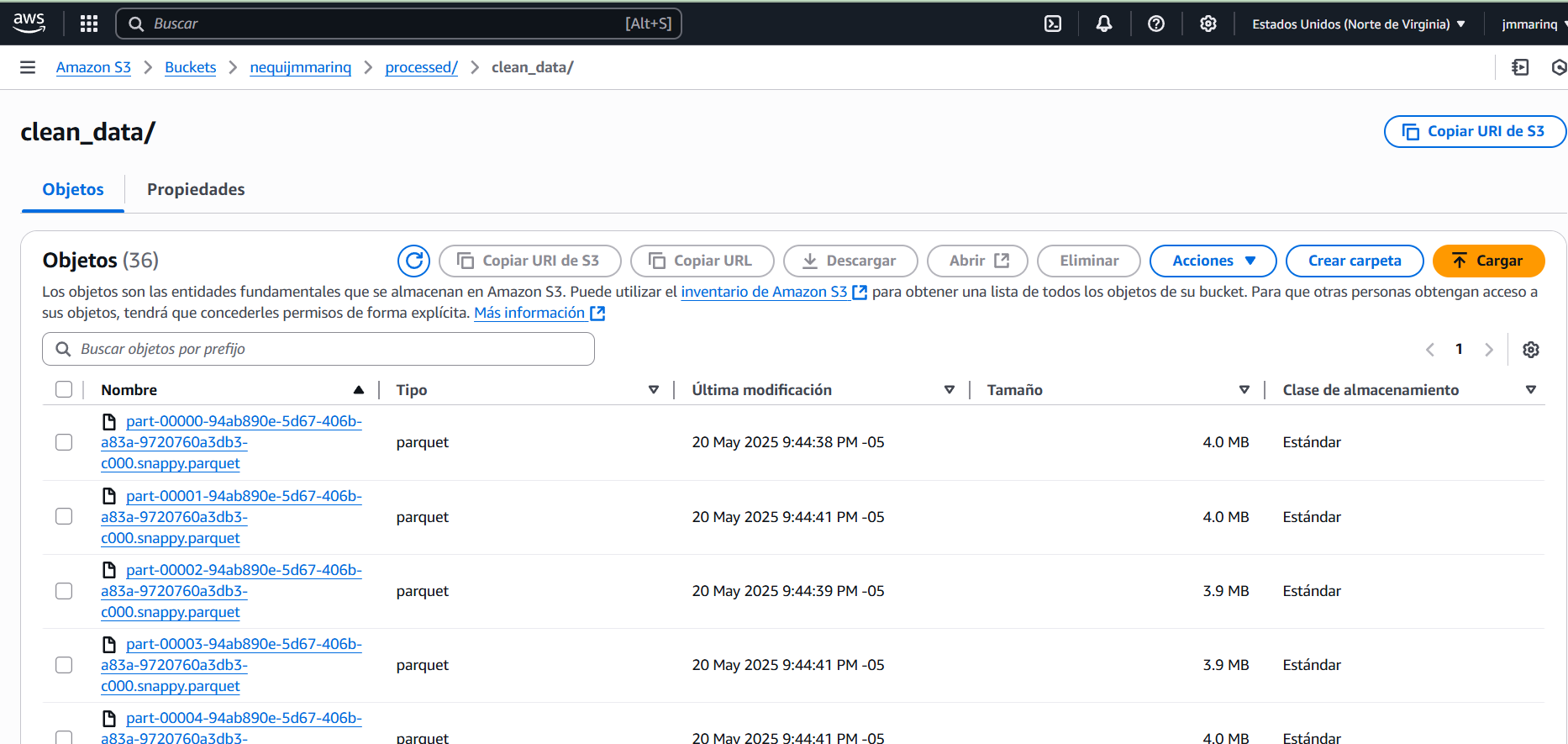
Limpieza de Datos

El notebook de este pipeline se llama Notebook\_Limpieza y el script limpio Limpieza, se carga un entorno de glue y de pyspark con las demás librerías necesarias. Se definen funciones de limpieza que se aplicarán a los datos ingresados, se les retiraran signos de puntuación, palabras “stopwords”, palabras más frecuentes y las anómalas, los emojis, emoticones, urls, tags de html y se aplica una transformación de jerga cultural. Se lee el archivo csv y se carga en pyspark desde el bucket utilizado para la realización de la prueba. Se castean las variables del archivo y se eliminan las variables que se considera no aportan información valiosa al modelo. La base cuenta con 2.812.943 registros y cuenta con 1.167 nulos en la variable Inbound que es el target, se toma la decisión de sacarlas de la base, quedando como registros 2.811.776.

Se corre el archivo y se adjunta evidencias de corrida exitosa



Se crea una ruta en s3 en la corrida y se ponen los archivos spark procesados.



Entrenamiento del Modelo

El notebook de este pipeline se llama Notebook\_Modelo y el script Modelo, se carga un entorno de glue y pyspark con las demás librerías necesarias. Se declaran variables de entorno como el input y el output del modelo, con su correspondiente versión. Se lee los archivos del pipeline anterior que ya se encuentran limpios, y se aplica el proceso de tokenización que crea un vector de palabras por cada mensaje de texto, luego se aplica un proceso de hashing en donde se establece un vector de longitud variable a una longitud constante para cada uno de los mensajes de texto, luego se aplica un proceso llamado tfidf que básicamente saca las frecuencias del vector, luego se aplica un modelo de regresión logística, así que el flujo de entrenamiento del modelo queda de la siguiente forma:

Modelo Logístico

Tfidf

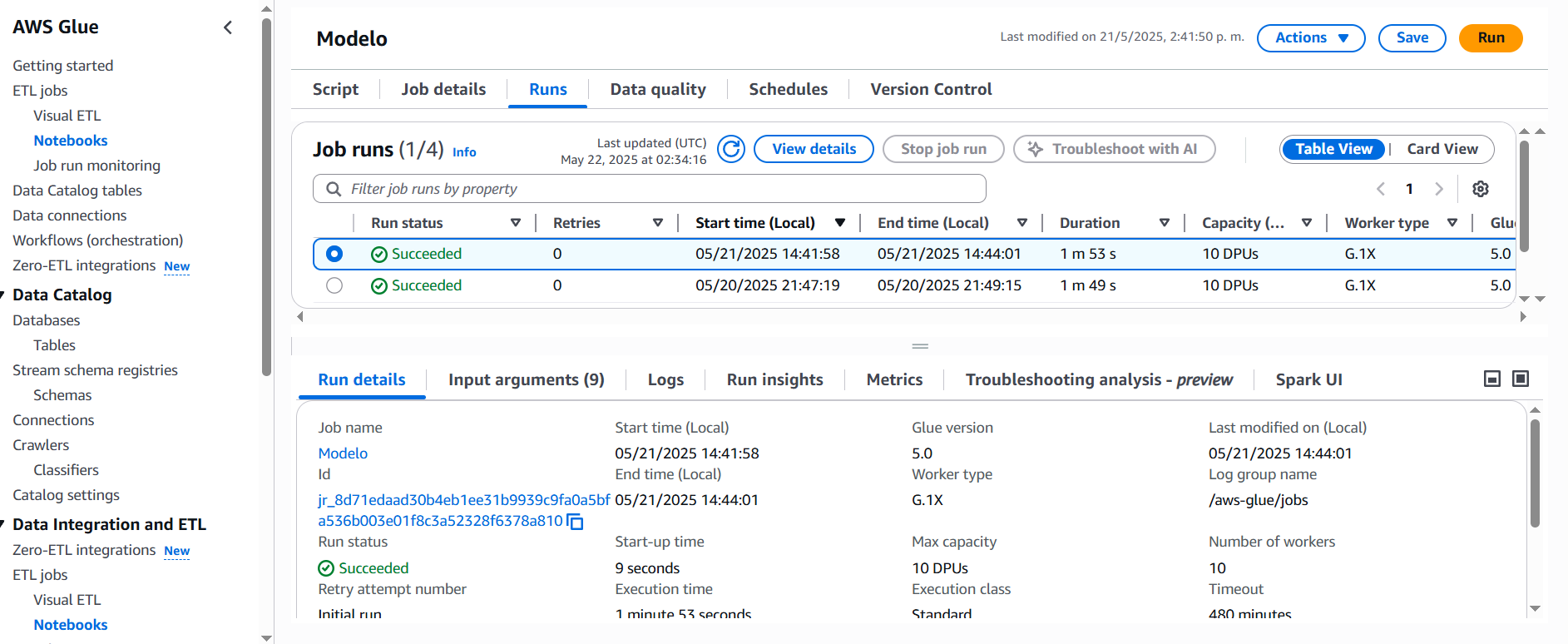
Hashing

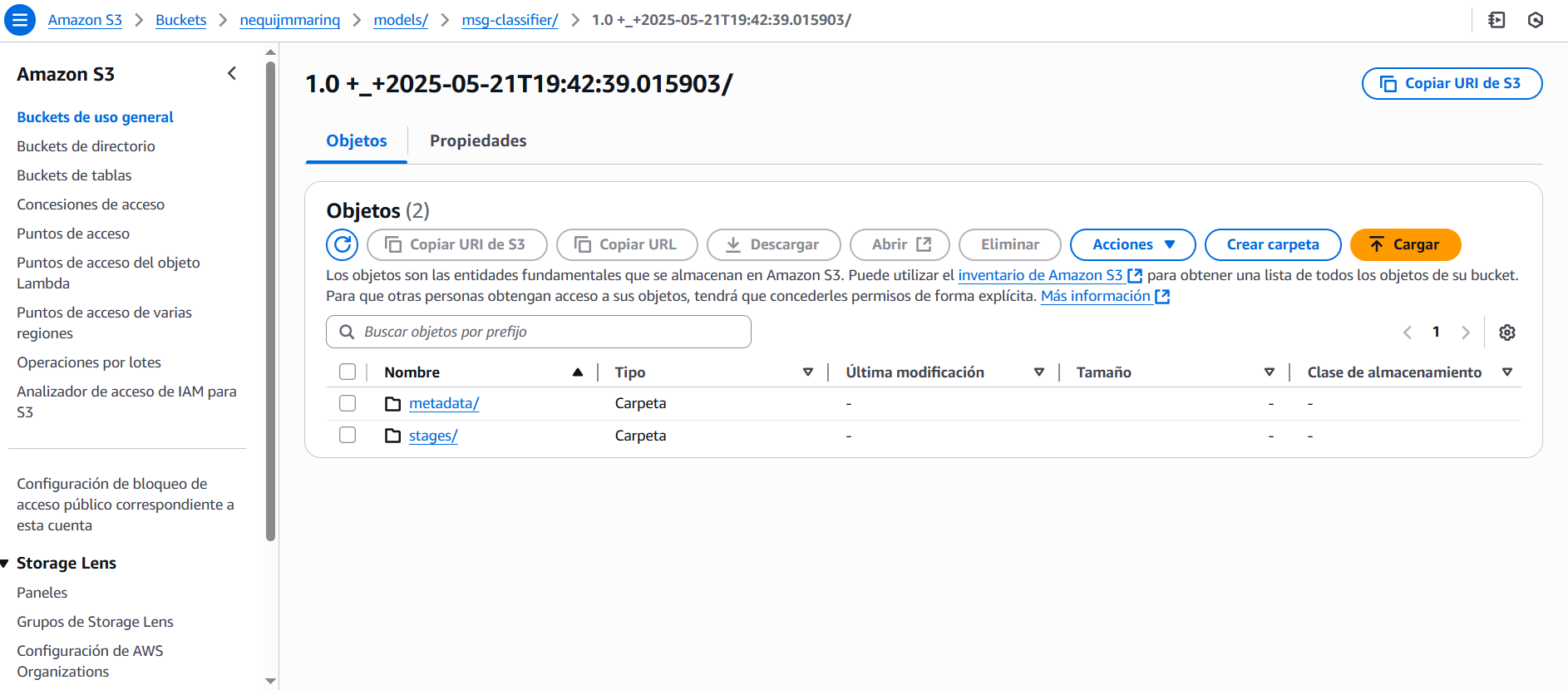
Tokenizar

Después de que se ejecuta el pipeline anterior se utiliza la división del dataset en un 80% para el train y 20% para el test. Se utiliza un modelo logístico y el auc fue algo sobrescoreado eso se puede dar a que los mensajes en twitter por lo general están tagueados, a pesar de que se les quita la arroba, puede quedar el numero o el tag de usuario al que el mensaje va dirigido y se escribe el test en s3.

Se corre el archivo y se adjunta evidencia de corrida exitosa.

Se adjunta también la ubicación del s3 donde se guarda el modelo cargado con metadatos y stages.





Inferencia

El notebook de este pipeline se llama Notebook\_Inferencia y el script Inferencia, se carga un entorno glue y spark con las demás librerías necesarias. Se carga el modelo entrenado en la ruta que especificamos en s3, leemos el archivo de test que asumimos con el que vamos hacer la inferencia, se puede realizar con cualquier otro archivo que se necesite realizar dicha predicción, el proceso de inferencia crea el archivo de predicción en una ruta en s3.

Reentrenamiento

El script de Reentrenamiento propone una forma de utilizar el reentrenamiento automático, la idea es utilizar este script en el flujo de workflow para definir la variable entorno llamada reentrenamiento, lo que hace el script es ir a la ruta de s3 llamada incoming, en donde se piensan colocar los nuevos archivos de entrenamiento.

RUN EL PIPELINE

El pipeline esta diseñado para correr en AWS Glue, orquestado por WorkFlow, en el siguiente orden de script.

1. Versionamiento.py
2. Limpieza.py
3. Modelo.py
4. Inferencia.py
5. Reentrenamiento.py

En el orquestador de WorkFlow, en la etapa de reentrenamiento es en la única que debe haber un trigger de EventBridge donde se estipule que si el reentrenamiento es diferente de cero, disparé el pipeline desde el inicio, para que se pueda reentrenar el modelo con los nuevos archivos. Cuando se quiera correr el flujo en WorkFlow se debe definir cada uno de los archivos .py como funciones def () es decir el código expuesto en función, y seguir el orden del pipeline.

ESTRATEGIA DE MONITOREO OPERATIVO Y SEGURIDAD PARA EL SISTEMA ML EN AWS GLUE

Monitoreo Operativo

|  |  |  |
| --- | --- | --- |
| Componente | Métrica | Objetivo |
| S3 | Conteo de Archivos Nuevos | Detectar nuevos archivos para procesar en especial en el script de reentrenamiento. |
| Glue Job | Status de Ejecución | Verificar si fue exitosa o fallo el pipeline. Como mostramos la evidencia debe aparecer en suceeded |
| Glue Job | Tiempo de Ejecución | Identificar Cuellos de Botella |
| Glue Job | Registros Procesados | Se puede poner en cada uno de los scripts ejecutado |
| Lambda (WorkFlow) | Invocaciones y Errores | Verificar Ejecución Automática |

Alertas y Acciones Correctivas

|  |  |  |
| --- | --- | --- |
| Job Fallido | Alarma CloudWatch | Reintento automatico o alertas al usuario |
| Job Demora Excesiva | Alarma CloudWatch | Revisión Manual |
| Nuevos Datos No Procesados | Métricas S3 | Ejecutar WorkFlow con Lambda |
| Lambda error > 0 | Alarma Cloud Watch | Revisión de permisos y lógicas |

Observabilidad

CloudWacth Logs habilitado para todos los Glue Jobs y Lambdas.

Dashboards personalizados para Glue WorkFlow y Jobs Críticos

Eventos auditados con CloudTrail

Controles de Seguridad

Definir los roles para las aplicaciones de Glue lectura y escritura, Lambda StartFlowRun por función, permisos en s3 de lectura y escritura.

Encriptación

En los buquets en la ingesta e inferencia y el uso de claves dedicadas a Glue.

Auditoría

Con los logs en cada uno de los pasos con políticas de retención y archivado.

NUEVO DIAGRAMA

Usuario

S3 (Entrada de Archivos)

Versionamiento

AWS Glue (Preprocess)

S3 (Clean)

Aws Glue o SageMaker (Training)

S3 (Model)

Inferencia

S3 (Salidas)

Reentrenamiento

Logs

Alarmas

Prácticas Recomendadas

* Versionado explícito de modelos en nombres de archivo o s3 prefijos
* Uso de logs en los recursos (Glue, S3, Lambda) para trazabilidad
* Validación de esquema de datos antes del entrenamiento
* Separar entornos (dev,test,prod) mediante particiones de buckets.