

Procesamiento batch en cloud

Ejercicio 1:

Aquí podemos ver los 3 buckets

aws

Buscar

[Alt+S]

Estados Unidos (Norte de Virginia)

voclabs/user3768009=Xavier\_Casta\_o @ 1317-9754-3006

Amazon S3

Buckets

Instantánea de la cuenta: actualizada cada 24 horas

Todas las regiones de AWS

Ver panel de Storage Lens

Storage Lens permite visualizar el uso del almacenamiento y las tendencias de la actividad. Las métricas no incluyen los buckets de directorio. Más información

Buckets de uso general

Buckets de directorio

Buckets de uso general (3)

Información

Todas las regiones de AWS

Copiar ARN

Vaciar

Eliminar

Crear bucket

Los buckets son contenedores de datos almacenados en S3.

Buscar buckets por nombre

< 1 >

Nombre	Región de AWS	Analizador de acceso de IAM	Fecha de creación
<input type="radio"/> <a href="#">accidentes-peatones-bcn-xcastanoa</a>	EE.UU. Este (Norte de Virginia) us-east-1	<a href="#">Ver analizador para us-east-1</a>	18 Jan 2025 10:36:13 AM CET
<input type="radio"/> <a href="#">equipamientos-bcn-xcastanoa</a>	EE.UU. Este (Norte de Virginia) us-east-1	<a href="#">Ver analizador para us-east-1</a>	18 Jan 2025 10:39:16 AM CET
<input type="radio"/> <a href="#">seguridad-vial-datalake-xcastanoa</a>	EE.UU. Este (Norte de Virginia) us-east-1	<a href="#">Ver analizador para us-east-1</a>	18 Jan 2025 10:43:11 AM CET

Objetos

Los objetos son las entidades fundamentales almacenadas en Amazon S3. Debe conceder explícitamente permisos a otros para acceder a los objetos. Cada objeto tiene datos, una clave y metadatos. La clave de objeto (o el nombre de clave) identifica de forma única al objeto en un bucket. Amazon S3 mantiene un conjunto de metadatos del sistema y de usuario para cada objeto y procesa los metadatos del sistema según sea necesario para la administración del almacenamiento.

Utilice esta página para ver todos los objetos de un bucket o carpeta, crear una carpeta o cargar un objeto. Puede abrir, descargar, eliminar y copiar la URL de los objetos seleccionados. También puede realizar las acciones admitidas que se muestran para cada tipo de bucket en el menú Acciones.

CloudShell

Comentarios

© 2025, Amazon Web Services, Inc. o sus filiales.

Privacidad

Términos

Preferencias de cookies

Y a continuación las carpetas dentro de estos creadas:

aws

Buscar

[Alt+S]

Estados Unidos (Norte de Virginia)

voclabs/user3768009=Xavier\_Casta\_o @ 1317-9754-3006

Amazon S3

Buckets

accidentes-peatones-bcn-xcastanoa

Objetos

Metadatos

Vista previa

Propiedades

Permisos

Métricas

Administración

Puntos de acceso

Objetos (2)

Información

Copiar URI de S3

Copiar URL

Descargar

Abrir

Eliminar

Acciones

Crear carpeta

Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

Nombre

Tipo

Última modificación

Tamaño

raw\_data/

Carpeta

-

-

staging/

Carpeta

-

-

Objetos

Los objetos son las entidades fundamentales almacenadas en Amazon S3. Debe conceder explícitamente permisos a otros para acceder a los objetos. Cada objeto tiene datos, una clave y metadatos. La clave de objeto (o el nombre de clave) identifica de forma única al objeto en un bucket. Amazon S3 mantiene un conjunto de metadatos del sistema y de usuario para cada objeto y procesa los metadatos del sistema según sea necesario para la administración del almacenamiento.

Utilice esta página para ver todos los objetos de un bucket o carpeta, crear una carpeta o cargar un objeto. Puede abrir, descargar, eliminar y copiar la URL de los objetos seleccionados. También

aws

Buscar

[Alt+S]

Estados Unidos (Norte de Virginia)

voclabs/user3768009=Xavier\_Casta\_o @ 1317-9754-3006

Amazon S3

Buckets

equipamientos-bcn-xcastanoa

Objetos

Metadatos

Vista previa

Propiedades

Permisos

Métricas

Administración

Puntos de acceso

Objetos (2)

Información

Copiar URI de S3

Copiar URL

Descargar

Abrir

Eliminar

Acciones

Crear carpeta

Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

Nombre

Tipo

Última modificación

Tamaño

raw\_data/

Carpeta

-

-

staging/

Carpeta

-

-

Objetos

Los objetos son las entidades fundamentales almacenadas en Amazon S3. Debe conceder explícitamente permisos a otros para acceder a los objetos. Cada objeto tiene datos, una clave y metadatos. La clave de objeto (o el nombre de clave) identifica de forma única al objeto en un bucket. Amazon S3 mantiene un conjunto de metadatos del sistema y de usuario para cada objeto y procesa los metadatos del sistema según sea necesario para la administración del almacenamiento.

Utilice esta página para ver todos los objetos de un bucket o carpeta, crear una carpeta o cargar un objeto. Puede abrir, descargar, eliminar y copiar la URL de los objetos seleccionados. También puede realizar las acciones admitidas que se muestran para cada tipo de bucket en el menú Acciones.

**Objetos (4)** Información

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [Inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

Nombre	Tipo	Última modificación	Tamaño
Bronce/	Carpeta	-	-
Oro/	Carpeta	-	-
Plata/	Carpeta	-	-
temp_processing/	Carpeta	-	-

**Objetos**

Los objetos son las entidades fundamentales almacenadas en Amazon S3. Debe conceder explícitamente permisos a otros para acceder a los objetos. Cada objeto tiene datos, una clave y metadatos. La clave de objeto (o el nombre de clave) identifica de forma única al objeto en un bucket. Amazon S3 mantiene un conjunto de metadatos del sistema y de usuario para cada objeto y procesa los metadatos del sistema según sea necesario para la administración del almacenamiento.

Utilice esta página para ver todos los objetos de un bucket o carpeta, crear una carpeta o cargar un objeto. Puede abrir, descargar, eliminar y copiar la URL de los objetos seleccionados. También puede realizar las acciones admitidas que se muestran para cada tipo de bucket en el menú Acciones.

## Funcion **procesado de datos\_accidentes\_vehículos\_xcastanoa**:

**Configuración del desencadenador**

S3  
aws asynchronous storage

**Bucket**  
Seleccione o escriba el ARN de un bucket de S3 que actúa como origen de eventos. El bucket debe estar en la misma región que la función.  
arn:aws:s3::accidentes-peatones-bcn-xcastanoa  
El bucket debe estar en la región us-east-1

**Tipos de eventos**  
Seleccione los eventos que desea que activen la función de Lambda. Si lo desea, también puede configurar un prefijo o un sufijo para un evento. Sin embargo, en cada bucket, no puede haber eventos individuales con configuraciones múltiples que tengan prefijos o sufijos superpuestos que puedan coincidir con la misma clave de objeto.

PUT X

**Prefijo - Opcional**  
Introduzca un único prefijo opcional para limitar las notificaciones a los objetos cuyas claves comiencen por los caracteres coincidentes. Todos los [caracteres especiales](#) deben estar codificados en URL.  
raw\_data

**Sufijo - Opcional**  
Introduzca un único sufijo opcional para limitar las notificaciones a los objetos cuyas claves terminen por los caracteres coincidentes. Todos los [caracteres especiales](#) deben estar codificados en URL.  
p. ej., .jpg

Lambda añadirá los permisos necesarios para AWS S3 para invocar la función Lambda desde este desencadenador. [Obtenga más información](#) sobre el modelo de permisos de Lambda.

El código completo es este:

```
import json
import boto3
import pandas as pd
import io
import logging

# Configurar el logger
logger = logging.getLogger()
```

```

logger.setLevel(logging.INFO)

# Inicializar el cliente de S3
s3_client = boto3.client('s3')

# Nombre del segundo bucket
second_bucket_name = 'seguridad-vial-datalake-xcastanoa' # Nombre del
segundo bucket

def lambda_handler(event, context):
    # Extraer la información del evento
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    file_key = event['Records'][0]['s3']['object']['key']

    # Registra el inicio de la operación
    logger.info("Leyendo archivo " + file_key + " desde el bucket " +
bucket_name)

    # Leer el archivo CSV desde S3
    try:
        # Descargar el archivo desde S3 a memoria
        csv_obj = s3_client.get_object(Bucket=bucket_name, Key=file_key)
        csv_data = csv_obj['Body'].read().decode('utf-8')

        # Usar Pandas para leer el CSV desde un string
        df = pd.read_csv(io.StringIO(csv_data))

        # Loggear las primeras filas del DataFrame para verificar
        logger.info("Contenido del archivo CSV leído exitosamente:")
        df = df.drop(['Numero_expedient', 'Codi_carrer', 'Nom_carrer',
'Num_postal', 'Nom_mes', 'Coordenada_UTM_X_ED50', 'Coordenada_UTM_Y_ED50',
'Longitud_WGS84', 'Latitud_WGS84'], axis=1)
    except Exception as e:
        logger.error("Error al leer el archivo CSV desde S3: " + str(e))
        return {
            'statusCode': 500,
            'body': json.dumps("Error al procesar el archivo CSV.")
        }

    # Definir las ubicaciones en staging y Bronze

```

```

staging_key = "staging/" + file_key.split('/')[-1]
bronze_key = "Bronce/" + file_key.split('/')[-1]

try:
    # Convertir el dataframe a CSV y cargarlo de nuevo en S3 en las
    carpetas /staging y /Bronce
    csv_buffer = io.StringIO()
    df.to_csv(csv_buffer, index=False)
    csv_buffer.seek(0)

    # Subir el archivo CSV al primer bucket en la carpeta /staging
    s3_client.put_object(Bucket=bucket_name, Key=staging_key,
Body=csv_buffer.getvalue())
    logger.info("Archivo guardado en " + staging_key + " en el bucket "
+ bucket_name)

    # Subir el archivo CSV al segundo bucket
    (seguridad-peaton-es-datalake-xcastanoaaa) en la carpeta /Bronce
    s3_client.put_object(Bucket=second_bucket_name, Key=bronze_key,
Body=csv_buffer.getvalue())
    logger.info("Archivo guardado en " + bronze_key + " en el bucket "
+ second_bucket_name)

except Exception as e:
    logger.error("Error al guardar los archivos en S3: " + str(e))
    return {
        'statusCode': 500,
        'body': json.dumps("Error al guardar los archivos en S3.")
    }

return {
    'statusCode': 200,
    'body': json.dumps("Archivo procesado correctamente y guardado en
staging y Bronce.")
}

```

AWS

BUSCAR

[Alt+] [Enter] [Esc]

Mis alertas de seguridad

voclabs/user3768009:Xavier\_Casta\_o @ S646-2490-7669

CloudWatch > Grupos de registro... > /aws/lambda/procesado\_datos\_accidentes\_vehiculos\_xcas... > 2025/01/20/\$LATESTj8b517834e8ae4d239c88bf49debc...

CloudWatch

Favoritos y recientes

Paneles

Operaciones de inteligencia artificial Vista previa

Alarmas

Registros

Grupos de registros Nuevo

Anomalías de registros

Live Tail

Logs Insights Nuevo

Contributor Insights

Métricas

Rastros de X-Ray Nuevo

Eventos

Reglas

Región de eventos

Eventos de registro

Acciones Empezar a seguir Crear un filtro de métricas

Puede utilizar la barra de filtros a continuación para buscar y hacer coincidir términos, frases o valores en sus eventos de registro. Más información sobre los patrones de filtro

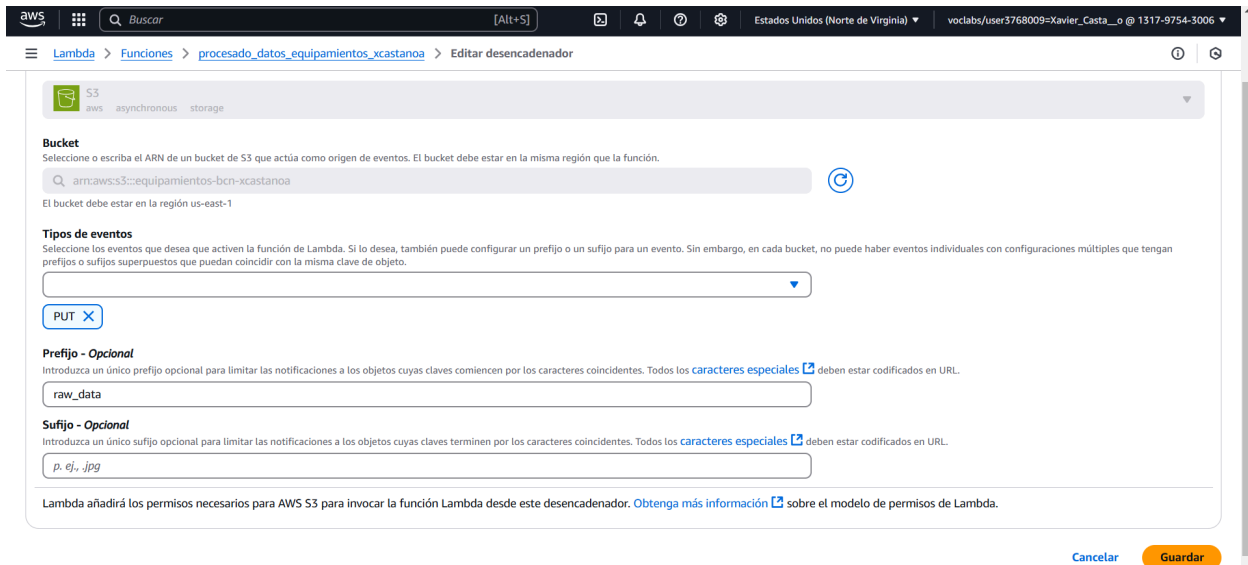
Filtrar eventos: pulse Intro para buscar

Borrar 1m 30m 1h 12h Personalizado Zona horaria UTC Mostrar

Marca temporal	Mensaje
	No hay eventos antiguos en este momento. Volver a intentar
▶ 2025-01-20T18:12:45.202Z	INIT_START Runtime Version: python:3.10.v46 Runtime Version ARN: arn:aws:lambda:us-east-1::runtime:732ad977322a3db48316f22772d490c...
▶ 2025-01-20T18:12:47.874Z	[INFO] 2025-01-20T18:12:47.874Z Found credentials in environment variables.
▶ 2025-01-20T18:12:48.073Z	START RequestId: 34031d88-0d73-4a0e-a7fc-25c789be5339 Version: \$LATEST
▶	Expandir fila Expandir el evento de registro Mon Jan 20 2025 19:12:48 GMT+0100 (hora estándar de Europa central) 7fc-25c789be5339 Leyendo archivo raw_data/2023_accidents_persones_gu_bcn.csv de...
▶ 2025-01-20T18:12:50.620Z	[INFO] 2025-01-20T18:12:50.620Z 34031d88-0d73-4a0e-a7fc-25c789be5339 Contenido del archivo CSV leído exitosamente:
▶ 2025-01-20T18:12:52.958Z	[INFO] 2025-01-20T18:12:52.958Z 34031d88-0d73-4a0e-a7fc-25c789be5339 Archivo guardado en staging/2023_accidents_persones_gu_bcn.csv..
▶ 2025-01-20T18:12:53.430Z	[INFO] 2025-01-20T18:12:53.430Z 34031d88-0d73-4a0e-a7fc-25c789be5339 Archivo guardado en Bronce/2023_accidents_persones_gu_bcn.csv ..
▶ 2025-01-20T18:12:53.458Z	END RequestId: 34031d88-0d73-4a0e-a7fc-25c789be5339
▶ 2025-01-20T18:12:53.458Z	REPORT RequestId: 34031d88-0d73-4a0e-a7fc-25c789be5339 Duration: 5384.77 ms Billed Duration: 5385 ms Memory Size: 128 MB Max Memory...

No hay eventos recientes en este momento. Reintentar automáticamente en pausa. Reiniciar

## Función `procesado_datos Equipamientos_xcastanoa`:



**Bucket**  
Seleccione o escriba el ARN de un bucket de S3 que actúa como origen de eventos. El bucket debe estar en la misma región que la función.  
  
El bucket debe estar en la región us-east-1

**Tipos de eventos**  
Seleccione los eventos que desea que activen la función de Lambda. Si lo desea, también puede configurar un prefijo o un sufijo para un evento. Sin embargo, en cada bucket, no puede haber eventos individuales con configuraciones múltiples que tengan prefijos o sufijos superpuestos que puedan coincidir con la misma clave de objeto.

**Prefijo - Opcional**  
Introduzca un único prefijo opcional para limitar las notificaciones a los objetos cuyas claves comiencen por los caracteres coincidentes. Todos los caracteres especiales deben estar codificados en URL.

**Sufijo - Opcional**  
Introduzca un único sufijo opcional para limitar las notificaciones a los objetos cuyas claves terminen por los caracteres coincidentes. Todos los caracteres especiales deben estar codificados en URL.

Lambda añadirá los permisos necesarios para AWS S3 para invocar la función Lambda desde este desencadenador. [Obtenga más información](#) sobre el modelo de permisos de Lambda.

[Cancelar](#) [Guardar](#)

El código es el siguiente:

```
import json
import boto3
import pandas as pd
import io
import logging

# Configurar el logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Inicializar el cliente de S3
s3_client = boto3.client('s3')

# Nombre del segundo bucket
second_bucket_name = 'seguridad-vial-datalake-xcastanoa' # Nombre del
segundo bucket

def lambda_handler(event, context):
    # Extraer la información del evento
    bucket_name = event['Records'][0]['s3']['bucket']['name']
    file_key = event['Records'][0]['s3']['object']['key']

    # Registra el inicio de la operación
```

```

logger.info("Leyendo archivo " + file_key + " desde el bucket " +
bucket_name)
# Establecer un número alto para mostrar todas las columnas
pd.set_option('display.max_columns', None)

try:
    # Descargar el archivo desde S3 a memoria
    csv_obj = s3_client.get_object(Bucket=bucket_name, Key=file_key)
    csv_data = csv_obj['Body'].read().decode('utf-8')

    # Usar Pandas para leer el CSV desde un string
    df = pd.read_csv(io.StringIO(csv_data), delimiter="\t")

    # Procesamiento del DataFrame
    df = df.drop(['register_id', 'institution_id', 'institution_name',
'modified',
                'addresses_roadtype_id', 'addresses_roadtype_name',
'addresses_road_id',
                'addresses_road_name',
'addresses_start_street_number',
                'addresses_end_street_number', 'addresses_zip_code',
'addresses_town',
                'addresses_main_address', 'addresses_type',
'values_id',
                'values_attribute_id', 'values_category',
'values_attribute_name',
                'values_value', 'values_outstanding',
'values_description',
                'secondary_filters_id', 'secondary_filters_name',
                'secondary_filters_fullpath',
'secondary_filters_tree',
                'secondary_filters_asia_id', 'geo_epgs_25831_x',
'geo_epgs_25831_y',
                'geo_epgs_4326_lat', 'geo_epgs_4326_lon',
'estimated_dates',
                'start_date', 'end_date'], axis=1)

    df = df.drop_duplicates()

    # Convertir la columna 'created' a datetime

```



```

df['created'] = pd.to_datetime(df['created'], utc=True,
format='mixed')

# Extraer el año de la columna 'created'
df['year'] = df['created'].dt.year

# Filtrar por años
df_2023 = df[df['year'] <= 2023]
df_2022 = df[df['year'] <= 2022]

# Agrupar por barrio
df_2023_grouped = df_2023.groupby(["addresses_district_id",
"addresses_district_name"])["year"].count()
df_2022_grouped = df_2022.groupby(["addresses_district_id",
"addresses_district_name"])["year"].count()

# Convertir a DataFrame para guardar en CSV
df_2023_grouped =
df_2023_grouped.reset_index(name="número_events")
df_2022_grouped =
df_2022_grouped.reset_index(name="número_events")

df_2023_grouped["addresses_district_id"] =
df_2023_grouped["addresses_district_id"].astype("string")
df_2022_grouped["addresses_district_id"] =
df_2023_grouped["addresses_district_id"].astype("string")
df_2023_grouped["addresses_district_id"] =
df_2023_grouped["addresses_district_id"].str[:-2]
df_2022_grouped["addresses_district_id"] =
df_2022_grouped["addresses_district_id"].str[:-2]
print(df_2023_grouped)
print(df_2022_grouped)

# Guardar los archivos para el año 2023
csv_buffer = io.StringIO()
df_2023_grouped.to_csv(csv_buffer, index=False)
csv_buffer.seek(0)

# Guardar el archivo en el primer bucket (staging)

```

```

        staging_key_2023 = "staging/2023_" + file_key.split('/')[-1]
        s3_client.put_object(Bucket=bucket_name, Key=staging_key_2023,
Body=csv_buffer.getvalue())
        logger.info("Archivo guardado en " + staging_key_2023 + " en el
bucket " + bucket_name)

        # Guardar el archivo en el segundo bucket (Bronce)
        bronze_key_2023 = "Bronce/2023_" + file_key.split('/')[-1]
        s3_client.put_object(Bucket=second_bucket_name,
Key=bronze_key_2023, Body=csv_buffer.getvalue())
        logger.info("Archivo guardado en " + bronze_key_2023 + " en el
bucket " + second_bucket_name)

        # Guardar los archivos para el año 2022
        csv_buffer = io.StringIO()
        df_2022_grouped.to_csv(csv_buffer, index=False)
        csv_buffer.seek(0)

        # Guardar el archivo en el primer bucket (staging)
        staging_key_2022 = "staging/2022_" + file_key.split('/')[-1]
        s3_client.put_object(Bucket=bucket_name, Key=staging_key_2022,
Body=csv_buffer.getvalue())
        logger.info("Archivo guardado en " + staging_key_2022 + " en el
bucket " + bucket_name)

        # Guardar el archivo en el segundo bucket (Bronce)
        bronze_key_2022 = "Bronce/2022_" + file_key.split('/')[-1]
        s3_client.put_object(Bucket=second_bucket_name,
Key=bronze_key_2022, Body=csv_buffer.getvalue())
        logger.info("Archivo guardado en " + bronze_key_2022 + " en el
bucket " + second_bucket_name)

    except Exception as e:
        logger.error("Error al procesar el archivo CSV: " + str(e))
        return {
            'statusCode': 500,
            'body': json.dumps("Error al procesar el archivo CSV.")
        }

    return {

```

```

    'statusCode': 200,
    'body': json.dumps("Archivos procesados y guardados
correctamente.")
}

```

Aquí tenemos los logs del proceso, en este caso:

The screenshot shows the AWS CloudWatch console interface. The breadcrumb navigation indicates the path: CloudWatch > Grupos de registros > /aws/lambda/procesado\_datos\_equipamientos\_xcastanoa > 2025/01/20/[LATEST]d35d32c2033e43a2965acfa35f1473e. The left sidebar contains various navigation options like Operaciones de inteligencia artificial, Alarmas, Registros, Métricas, and Eventos. The main panel displays a list of log entries. The first entry is a 'START' event, followed by an '[INFO]' event that logs the start of file processing. Subsequent entries show the processing of individual files, with details like 'addresses\_district\_id', 'addresses\_district\_name', and 'numero\_events'. The final entry is an 'END' event, indicating the successful completion of the Lambda function execution.

This screenshot shows another view of the AWS CloudWatch console for the same Lambda function. The breadcrumb navigation is identical. The log entries in this view include the same '[INFO]' event about file processing, but also show several 'INFO' events related to file storage. These events indicate that files were successfully saved to the 'staging/2022\_opendatabcn\_llista-equipame...' directory and the 'Bronce/2022\_opendatabcn\_llista-equipame...' directory. The final entry is an 'END' event, confirming the successful completion of the process.

Aquí obtenemos lo siguiente (tanto en la carpeta staging como en Bronce en el otro bucket):

Amazon S3 > Buckets > equipamientos-bcn-xcastanoa > staging/

staging/ Copiar URI de S3

Objetos Propiedades

Objetos (2) Información Copiar URI de S3 Copiar URL Descargar Abrir Eliminar Acciones Crear carpeta Cargar

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Buscar objetos por prefijo

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	<a href="#">2022_opendatabcn_llista-equipaments_cultura.csv</a>	csv	19 Jan 2025 10:11:19 AM CET	1.9 KB	Estándar
<input type="checkbox"/>	<a href="#">2023_opendatabcn_llista-equipaments_cultura.csv</a>	csv	19 Jan 2025 10:11:18 AM CET	1.9 KB	Estándar

No he añadido el año 2024 porque no hay datos de accidentes de 2024 en la página web por lo que descargarlos sería redundante ya que no hay nada con lo que integrarlos.

Estos son los resultados y configuraciones del job de Glue:

etl-accident-data-bcn

Last modified on 25/1/2025, 10:03:38 Actions Save Run

Visual Script Job details Runs Data quality Schedules Version Control Upgrade anal

+

Data source - S3 bucket Amazon S3 Accidentes

Data preview Output schema

Data preview (200) Info READY End session Previewing 19 of 19 fields

Filter sample dataset

codi_districte	nom_districte	codi_barri	nom_barri	descripcio_dia_na
3	Sants-Montjuic	18	Sants	Diumenge

Data source properties - S3

Name  
Amazon S3 Accidentes

S3 source type Info  
☒ S3 location  
Choose a file or folder in an S3 bucket.  
☐ Data Catalog table

S3 URL  
 View Browse S3

☒ Recursive  
Read files in all subdirectories.

Data format  
CSV

Delimiter  
Comma (,)

Escape character - optional  
Enter a character to use for escaping

aws

Buscar

[Alt+S]

Estados Unidos (Norte de Virginia)

voclabs/user3768009=Xavier\_Casta\_o @ 5646-2490-7669

etl-accident-data-bcn

Last modified on 25/1/2025, 10:03:38

Actions

Save

Run

Visual

Script

Job details

Runs

Data quality

Schedules

Version Control

Upgrade ana

+ Data source - S3 bucket Amazon S3 Accidentes

Data source - S3 bucket Amazon S3 Eventos

Data preview

Output schema

Data preview (10)

Info

READY

End session

Previewing 3 of 3 fields

Filter sample dataset

addresses_district_id	addresses_district_name	número_events
1	Ciutat Vella	308
2	Eixample	371
3	Sants-Montjuïc	213

Data source properties - S3

NameAmazon S3 Eventos

S3 source typeS3 location

S3 URLs3://seguridad-vial-datalai

Recursive

Data formatCSV

DelimiterComma (,)

Escape character - optional

CloudShell

Comentarios

© 2025, Amazon Web Services, Inc. o sus filiales.

Privacidad

Términos

Preferencias de cookies

aws

Buscar

[Alt+S]

Estados Unidos (Norte de Virginia)

voclabs/user3768009=Xavier\_Casta\_o @ 5646-2490-7669

etl-accident-data-bcn

Last modified on 25/1/2025, 10:03:38

Actions

Save

Run

Visual

Script

Job details

Runs

Data quality

Schedules

Version Control

Upgrade ana

+ Transform - Join

Data preview

Output schema

Data preview (200)

Info

READY

End session

Previewing 22 of 22 fields

Filter sample dataset

Codi_districte	Nom_districte	Codi_barri	Nom_barri	Descripcio_dia
1	Ciutat Vella	1	el Raval	Diumenge

Transform

NameJoin

Node parentsAmazon S3 AccidentesAmazon S3 Eventos

Join typeLeft join

Join conditionscodi\_districte = addresses\_district\_id

CloudShell

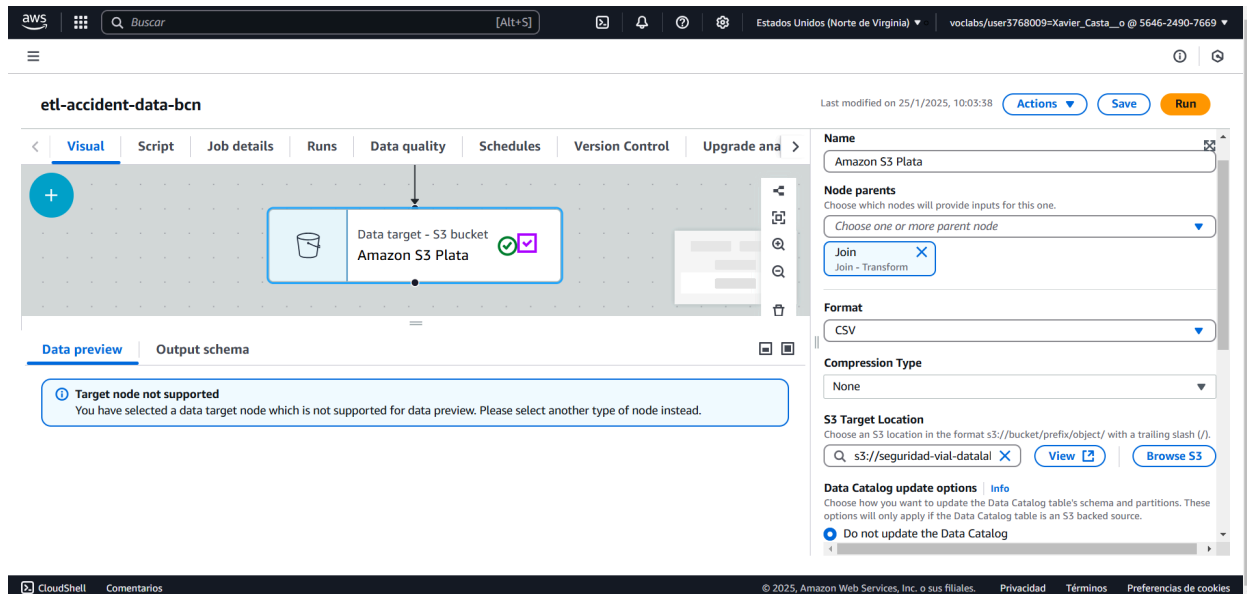
Comentarios

© 2025, Amazon Web Services, Inc. o sus filiales.

Privacidad

Términos

Preferencias de cookies



Cómo podemos ver por el esquema, la integración ha consistido en agregar el dataframe por número de eventos o puntos de ocio presentes en cada uno de los barrios segmentado por 2022 y 2023, entonces utilizamos el número identificador del barrio en cada uno de los dataframes como nexo de unión con un left join (manteniendo todas las filas de los accidentes en caso de que algún id de barrio en el dataframe de accidentes tenga algún número erróneo no perder esa fila), de manera que tenemos para cada fila de un accidente, el número de eventos en el distrito en el que ha pasado, como una variable más que podemos estudiar junto con el resto de variables de los accidentes.

Por último, veremos en qué ha consistido la función Lambda que lee los archivos de Bronze/ para un mismo año y los mueve a temp\_processing/ quitando el prefijo del año y, seguidamente, corre el job de Glue:

```
import json
import boto3
import logging

# Configurar el logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)

# Inicializar los clientes de S3 y Glue
s3_client = boto3.client('s3')
glue_client = boto3.client('glue')

# Nombre del bucket y del Glue Job
```

```

bucket_name = 'seguridad-vial-datalake-xcastanoa'
glue_job_name = 'etl-accident-data-bcn' # Nombre del Glue Job

def lambda_handler(event, context):
    try:
        # Extraer la información del evento
        record = event['Records'][0]
        file_key = record['s3']['object']['key']
        logger.info("Evento detectado para el archivo:
{}".format(file_key))

        # Verificar si el archivo está en la carpeta 'Bronze/'
        if file_key.startswith('Bronze/'):
            logger.info("Archivo detectado en carpeta 'Bronze':
{}".format(file_key))

            year = None

            # Identificar el año en el nombre del archivo
            for possible_year in ['2022', '2023']:
                if possible_year in file_key:
                    year = possible_year
                    break

            if year:
                # Construir nombres esperados de archivos
                equip_file =
'{}_opendatabcn_llista-equipaments_cultura.csv'.format(year)
                accident_file =
'{}_accidents_persones_gu_bcn.csv'.format(year)

                equip_file_key = 'Bronze/{}'.format(equip_file)
                accident_file_key = 'Bronze/{}'.format(accident_file)

                try:
                    # Verificar la existencia de ambos archivos
                    logger.info("Verificando existencia de archivos: {} y
{}".format(equip_file_key, accident_file_key))
                    s3_client.head_object(Bucket=bucket_name,
Key=equip_file_key)

```

```

        s3_client.head_object(Bucket=bucket_name,
Key=accident_file_key)

        logger.info("Ambos archivos están presentes para el
año {}".format(year))

        # Limpiar la carpeta 'temp_processing/'
        temp_processing_folder = 'temp_processing/'
        logger.info("Limpiando carpeta
'{}'.format(temp_processing_folder))

        response =
s3_client.list_objects_v2(Bucket=bucket_name,
Prefix=temp_processing_folder)
        if 'Contents' in response:
            for obj in response['Contents']:
                s3_client.delete_object(Bucket=bucket_name,
Key=obj['Key'])

                logger.info("Archivo eliminado:
{}".format(obj['Key']))

        # Mover archivo de equipamientos
        new_equip_file_key =
'{}'.format(temp_processing_folder, equip_file[5:]) # Quitar prefijo de
año

        s3_client.copy_object(
            Bucket=bucket_name,
            CopySource={'Bucket': bucket_name, 'Key':
equip_file_key},
            Key=new_equip_file_key
        )
        s3_client.delete_object(Bucket=bucket_name,
Key=equip_file_key)

        logger.info("Archivo {} movido a
{}".format(equip_file, new_equip_file_key))

        # Mover archivo de accidentes
        new_accident_file_key =
'{}'.format(temp_processing_folder, accident_file[5:]) # Quitar prefijo
de año

```



```

        s3_client.copy_object(
            Bucket=bucket_name,
            CopySource={'Bucket': bucket_name, 'Key':
accident_file_key},
            Key=new_accident_file_key
        )
        s3_client.delete_object(Bucket=bucket_name,
Key=accident_file_key)
        logger.info("Archivo {} movido a
{}".format(accident_file, new_accident_file_key))

        # Iniciar el Glue Job
        logger.info("Iniciando el Glue Job:
{}".format(glue_job_name))
        response =
glue_client.start_job_run(JobName=glue_job_name)
        logger.info("Glue Job iniciado con éxito. JobRunId:
{}".format(response['JobRunId']))

    except s3_client.exceptions.ClientError as e:
        # Si algún archivo no existe, registrar el error
        logger.error("Error al verificar o mover los archivos:
{}".format(e))

        return {
            'statusCode': 500,
            'body': json.dumps("Error: Uno o ambos archivos
requeridos no están disponibles.")
        }

    except glue_client.exceptions.ClientError as e:
        # Error al iniciar el Glue Job
        logger.error("Error al iniciar el Glue Job:
{}".format(e))

        return {
            'statusCode': 500,
            'body': json.dumps("Error al iniciar el Glue
Job.")
        }

    except Exception as e:
        # Captura cualquier error inesperado
        logger.error("Error inesperado: {}".format(e))

```

```

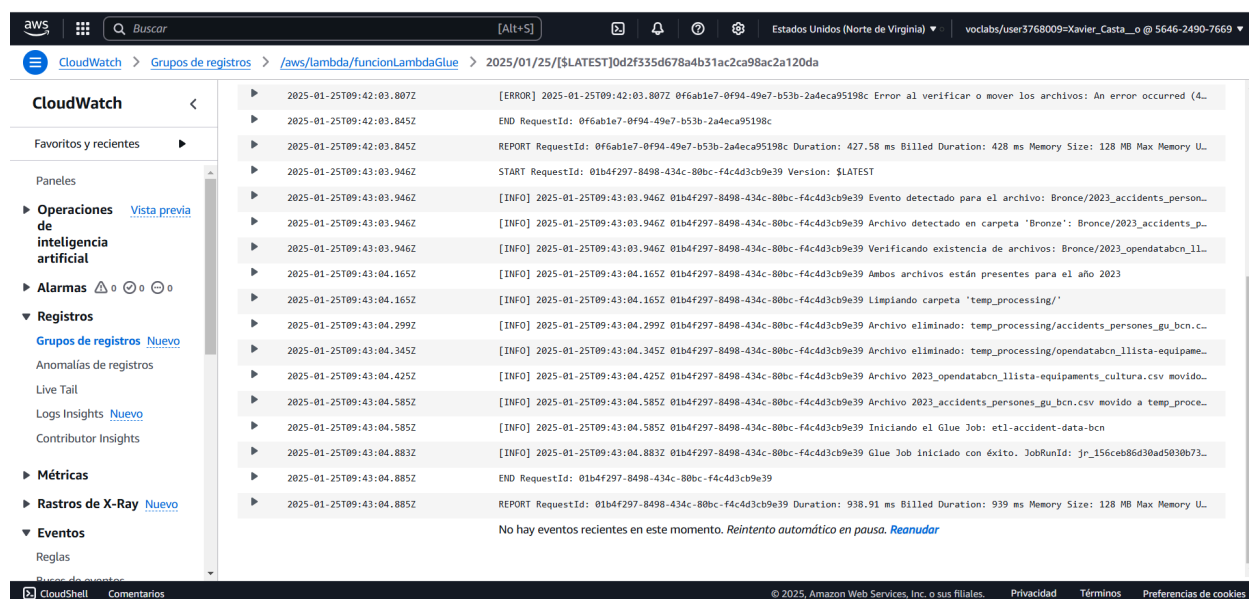
return {
    'statusCode': 500,
    'body': json.dumps("Error inesperado al procesar el evento.")
}

return {
    'statusCode': 200,
    'body': json.dumps("Proceso completado exitosamente.")
}

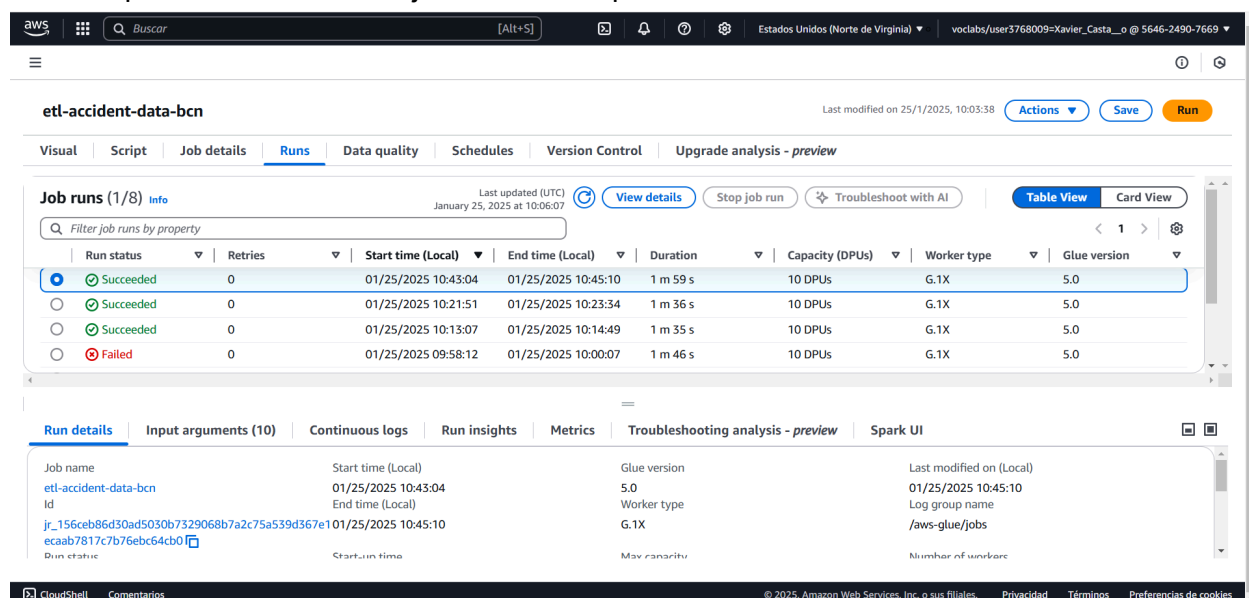
```

Estos son los dos logs que nos arroja en CloudWatch, uno por cada año ya que hay un bucle que hace un intento por cada año posible que, con los datos que tenemos son 2022 y 2023. Por lo tanto, como solo he hecho la prueba con 2023, uno de los logs da error y el otro (de 2023) hace todo el proceso:

The screenshot shows the AWS CloudWatch console interface. The top navigation bar includes the AWS logo, a search bar, and user information. The left sidebar contains navigation links for various services. The main content area is titled 'Eventos de registro' (Log Events) and shows a list of log entries for a specific Lambda function. The log entries are displayed in a table with columns for 'Marca temporal' (Timestamp) and 'Mensaje' (Message). The messages include details about the function's runtime, environment variables, and file operations. One entry shows an error message: 'Error al verificar o mover los archivos: An error occurred (4...'.



También podemos ver cómo el job en Glue empieza a correr solo:



Y además podemos ver los efectos de los directorios de S3:

## Bronce/

[Copiar URI de S3](#)[Objetos](#)[Propiedades](#)

### Objetos (1)

[Copiar URI de S3](#)[Copiar URL](#)[Descargar](#)[Abrir](#)[Eliminar](#)[Acciones](#)[Crear carpeta](#)[Cargar](#)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

< 1 >

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	<a href="#">2022_opendatabcn_llista-equipaments_cultura.csv</a>	csv	25 Jan 2025 10:42:01 AM CET	250.0 B	Estándar

Primero, los archivos de Bronce/ correspondientes a 2023 desaparecen. Y son trasladados a temp\_processing/ sin el prefijo:

## temp\_processing/

[Copiar URI de S3](#)[Objetos](#)[Propiedades](#)

### Objetos (2)

[Copiar URI de S3](#)[Copiar URL](#)[Descargar](#)[Abrir](#)[Eliminar](#)[Acciones](#)[Crear carpeta](#)[Cargar](#)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

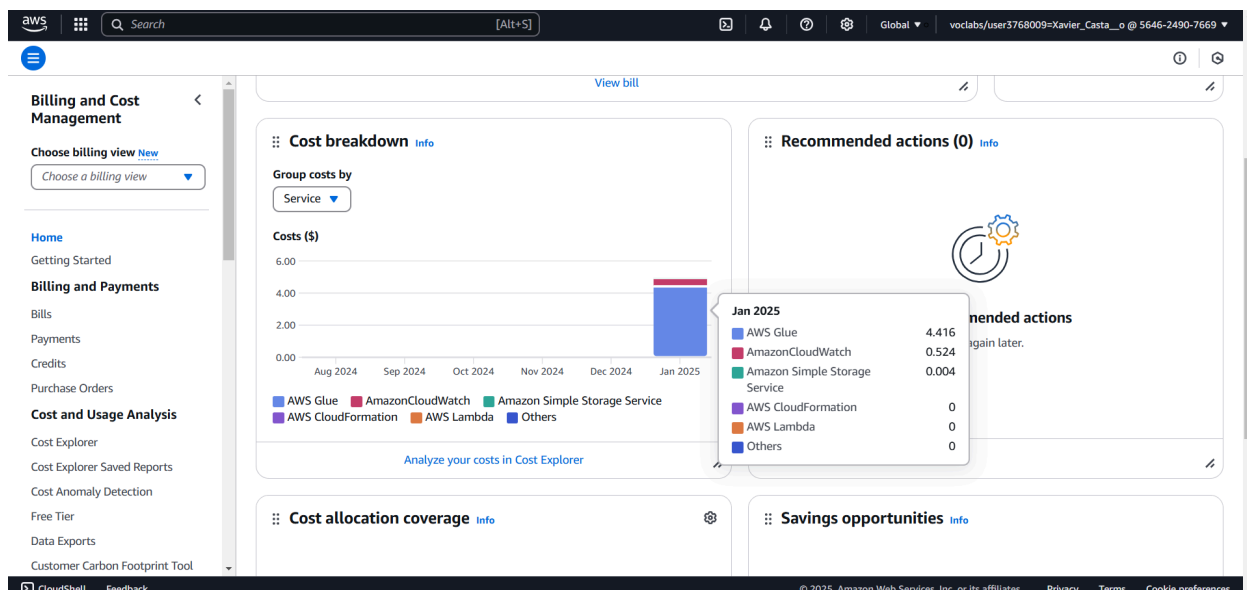
< 1 >

<input type="checkbox"/>	Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
<input type="checkbox"/>	<a href="#">accidents_persones_gu_bcn.csv</a>	csv	25 Jan 2025 10:43:05 AM CET	2.9 MB	Estándar
<input type="checkbox"/>	<a href="#">opendatabcn_llista-equipaments_cultura.csv</a>	csv	25 Jan 2025 10:43:05 AM CET	250.0 B	Estándar

Por último, podemos ver cómo en Plata/ se han cargado los datos integrados mediante el job de Glue:

The screenshot shows the Amazon S3 console interface. At the top, there's a search bar and navigation tabs. The main area displays a list of objects in the 'seguridad-vial-datalake-xcastanoa' bucket. A preview window is open for one of the objects, showing a detailed JSON record. The record includes various fields related to a traffic accident, such as location, date, time, and details about the vehicles and individuals involved.

Por último, el total gastado cómo podemos ver aquí son unos 4'9\$, de los cuales casi la totalidad han ido a financiar los trabajo de Glue, el segundo servicio más caro ha sido CloudWatch, S3 parece el servicio más barato con apenas unos céntimos y, curiosamente, Lambda no tuvo ningún coste. Muy probablemente porque lo que hace Lambda es disparar otros sistemas de manera orquestada según la función Lambda que hayamos escrito.



A modo de consejo y para optimizar el despliegue del proyecto aprovechando al máximo cada euro que invertimos yo creo que es muy importante que en lugar de estar generando unos eventos mediante borrado y pegado nuevamente de los archivos que queremos tratar con

Lambda para disparar el desencadenador todo el rato e ir mirando los logs de CloudWatch cada vez dentro de Lambda, creamos nuestros tests de eventos.

En mi caso utilizar esta función Lambda de manera inicial para extraer el evento con el que trabajamos:

```
import json
import logging

# Configurar el logger
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    # Registrar el evento completo en los logs
    logger.info("Evento recibido: " + json.dumps(event, indent=2))

    return {
        'statusCode': 200,
        'body': json.dumps('Evento procesado correctamente')
    }
```

De esta manera solo tenemos que desencadenar una vez el evento en S3 copiando el archivo correspondiente en la carpeta correspondiente y cuando esta función se ejecute obtendremos el JSON con el evento y todos sus valores, estos tienen esta forma:

Como ejemplo, este es el evento que se genera cuando pegamos los archivos de accidentes en la carpeta raw\_data/ del bucket de accidentes:

```
{
  "Records": [
    {
      "eventVersion": "2.1",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2025-01-18T15:29:10.848Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId":
"AWS:AROAR5L565BPPBMJM6AC2:user3768009=Xavier_Casta__o"
      },
      "requestParameters": {
        "sourceIPAddress": "37.29.240.15"
      },
      "responseElements": {
```

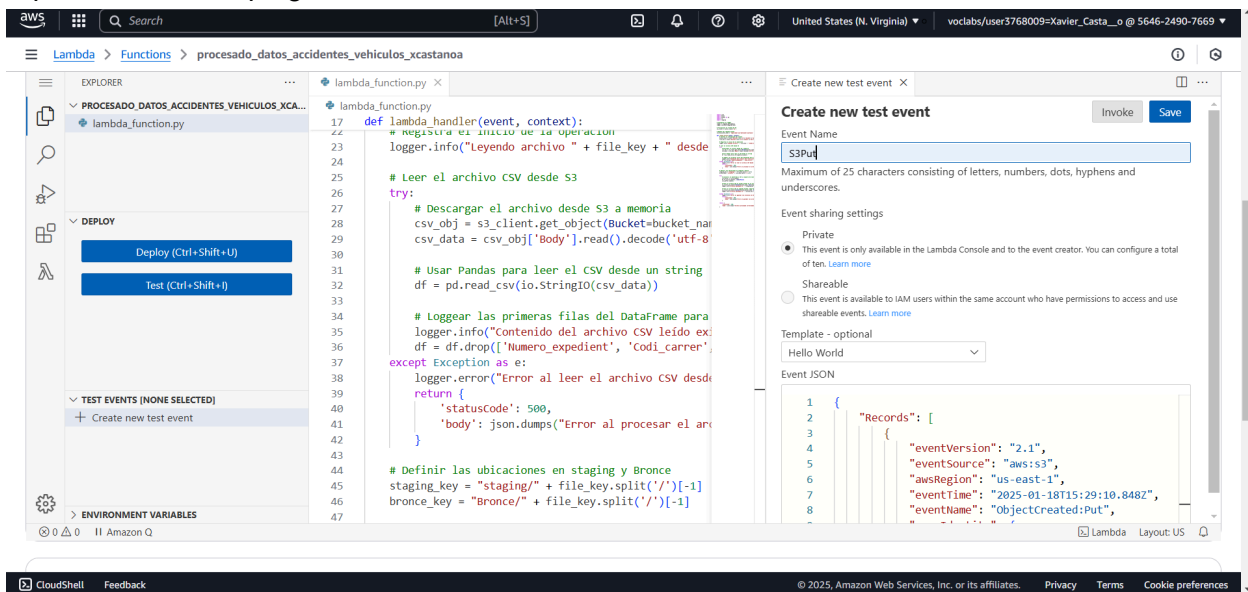
```

        "x-amz-request-id": "54V5DH4KWQGJAX15",
        "x-amz-id-2":
"tJtSC3WTjxnFb5JtqOJ6iSWaBlgoz+6Sb/GueUuSiF+mDToJZKi9C2aNn5SFVJibCo4MvwLJK
KoU0QgGcP2oXWZU0xvnFrZW"
    },
    "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "aa69af74-c432-4df0-a0ab-c07796a65142",
        "bucket": {
            "name": "accidentes-peatones-bcn-xcastanoaaa",
            "ownerIdentity": {
                "principalId": "A3KR03WFPW2DXW"
            },
            "arn":
"arn:aws:s3:::accidentes-peatones-bcn-xcastanoaaa"
        },
        "object": {
            "key": "raw_data/2023_accidents_persones_gu_bcn.csv",
            "size": 4290259,
            "eTag": "4697d0cd25f853395201d5fbd506bc7a",
            "sequencer": "00678BC8C6B2BF339F"
        }
    }
}
]
}

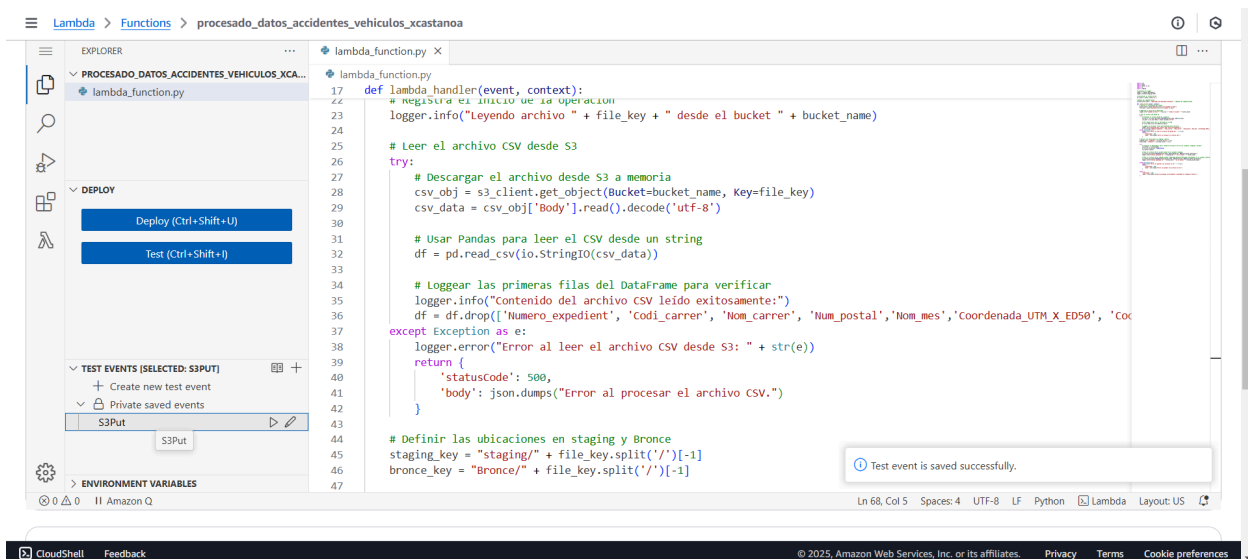
```

Si este evento es copiado de la siguiente manera en Lambda, en el apartado Create new test event dentro de la sección Event JSON y le ponemos un nombre, luego aparece abajo a la

izquierda en el desplegable TEST EVENTS:



Y queda abajo a la derecha en esa sección como S3Put, el nombre que le hemos dado:



Es importante que después disparar el desencadenador por primera vez dejando el archivo en la carpeta correspondiente de S3 lo dejemos allí, ya que el test es capaz de identificar el archivo si lo hemos dejado ahí y podemos imprimir en el output del test los resultados de las transformaciones de los dataframes pandas si queremos ir viendo como están quedando, así como poder hacer todo el proceso entero y comprobar que no hay bugs en el código, para cuando finalmente lo volvamos a correr borrando el archivo y volviéndolo a enganchar en la carpeta S3, sepamos exactamente lo que va a aparecer en los logs de CloudWatch.