

PET-TRACKING

CASO DE USO

OBJETIVO

Diseño e implementación de un sistema inteligente que permita **interpretar en tiempo real las emociones y necesidades básicas de mascotas**(perros y gatos), en base a un análisis de su comportamiento físico, vocalizaciones, ritmo cardíaco y movimientos realizados.

Para lograrlo se utilizarán herramientas digitales para:

- **Analizar grandes volúmenes de datos** estructurados y no estructurados.
- **Identificar patrones de conducta** relacionados con los estados emocionales.
- **Generar traducciones** de esas emociones (ej. "quiere jugar", "está estresado").
- **Facilitar visualizaciones** en PowerBI para el cuidador o el profesional veterinario.

RECOLECCIÓN DE DATOS

FUENTE	TIPO DE DATOS
Wearables (collares/sensores)	Ritmo cardíaco, temperatura corporal, actividad física, ubicación GPS
Micrófonos ambientales	Vocalizaciones, tono emocional, frecuencia de sonidos
Cámaras de seguimiento	Postura Corporal, gestos, patrones de movimiento
Reportes de cuidadores	Etiquetado emocional (alegría, dolor, ansiedad, hambre,etc)

PROCESO DE CARGA DE DATOS

1. Batch (por lotes)

1.1. **Periodicidad:** Cada día.

1.2. **Origen:** Reportes de comportamiento e historial de sensores.

1.3. **Herramientas AWS:**

a) Amazon S3

b) AWS Glue (Glue Data Catalog y Glue Studio, Glue Data Quality)

2. Streaming (en tiempo real)

2.1. **Periodicidad:** Continua (segundos).

2.2. **Origen:** Dispositivos wearables transmitiendo señales vitales y vocalizaciones.

2.3. **Herramientas AWS:**

a) Amazon Kinesis Data Streams

b) AWS Lambda

c) Amazon Firehose

ANÁLISIS DE DATOS

1. Procesamiento de señales y vocalizaciones

- Conversión de sonidos en espectrogramas o vectores de características.
- Detección de patrones emocionales en vocalizaciones.

2. Análisis de comportamiento

- Actividad física comparada con hábitos normales.
- Cambios súbitos de ritmo cardíaco o patrones de movimiento.

3. Modelos de predicción

- Modelos personalizados por especie, raza e individuo.
- Entrenamiento con datasets etiquetados.
- Clasificación en categorías emocionales.

4. Generación de traducción emocional

- Transformación de la predicción en un mensaje interpretable por humanos.

VISUALIZACIÓN DE DATOS

SALIDA	DESCRIPCIÓN	USO
Traducción emocional	Texto generado a partir del análisis de datos ("quiere salir", "tiene sed")	Interacción humano-mascota
Score de bienestar por mascota	Índice numérico promedio del estado emocional	Seguimiento veterinario en el hogar
Alertas de riesgo	Notificaciones de ansiedad, dolor u otros estados críticos	Intervenciones rápidas
Dashboard emocional en PowerBI	Visualización de patrones por día, semana o condición ambiental	Decisiones basadas en datos
Informes exportables	Archivos periódicos con métricas de salud y emoción	Control de la salud a largo plazo

TECNOLOGÍAS AWS UTILIZADAS POR ETAPA

ETAPA	SERVICIOS AWS IMPLICADOS
Ingesta (Batch)	AWS Glue Studio, Glue Data Catalog, S3
Ingesta (Streaming)	Kinesis Data Streams, Lambda, Firehose
Almacenamiento	S3, Redshift, Athena
Procesamiento	Amazon EMR con Apache Spark
Visualización	PowerBI

BENEFICIOS E IMPACTO

BENEFICIO	IMPACTO
Comunicación emocional animal-humano	Mejora la relación y cuidado responsable de las mascotas
Prevención de estrés o enfermedad	Actuación temprana con datos objetivos
Herramienta para veterinarios	Diagnóstico complementario y basado en patrones
Personalización por raza y comportamiento	Modelos de IA adaptativos según características del animal
Aplicación educativa y social	Potencial de uso en hogares, refugios, escuelas y clínicas veterinarias

NGESTA DE DATOS

Antes de iniciar esta parte vamos a crear la estructura desarrollada en el apartado de Almacenamiento en Amazon S3 para el proyecto (**Bucket**). Aquí se presenta la parte principal del Bucket.

The screenshot shows the AWS S3 console interface. The top navigation bar includes tabs for 'UAX Campus', 'Lanzamiento del Laboratorio', 'Página de inicio de la Consola', 'Primeros pasos: consola de...', 'pet-tracking-data-bucket', and 'TPS4_1-24: Reto Proyecto'. The main title is 'pet-tracking-data-bucket' with a 'Información' link. Below the title, there are tabs for 'Objetos', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Administration', and 'Access Points'. The 'Objetos' tab is selected, showing a list of 12 objects. The table columns are 'Nombre' (Name), 'Tipo' (Type), 'Última modificación' (Last modified), 'Tamaño' (Size), and 'Clase de almacenamiento' (Storage class). The objects listed are: archive/, dashboards/, docs/, emr/, firehose-output/, glue/, kinesis/, logs/, processed/, raw/, s3-management/, and warehouse/. Each object is a folder represented by a folder icon. At the bottom of the table, there are buttons for 'Copiar URI de S3', 'Copiar URL', 'Descargar', 'Abrir', 'Eliminar', 'Acciones', 'Crear carpeta', and 'Cargar'.

BATCH(AWS GLUE)

1. Subir el archivo batch a la ruta correspondiente del bucket S3

The screenshot shows the AWS S3 console with a success message: "Se ha realizado la carga correctamente". It displays a summary of the upload: "Realizado correctamente" (1 archivo, 294.2 KB (100.00%)) and a table showing the uploaded file "pet-behavior-data.csv". The table includes columns for Nombre, Carpeta, Tipo, Tamaño, Estado, and Error. The file is listed as "Realizado correctamente". Below the table, there are tabs for "Archivos y carpetas" and "Configuración".

Al ser una estructura basada en particiones(**year/**, **month/**, **day/**) permite usar Glue Crawlers, Athena y Redshift Spectrum de forma más eficiente y escalable.

2. Crear base de datos Glue en el Data Catalog

- Ir a AWS Glue → Data Catalog → Databases
- Click en “Add database”
- Nombre: **pet_batch_db**
- Especificar ruta de ubicación: **s3://pet-tracking-data-bucket/glue/catalog/**
- Crear

Las bases de datos en Glue organizan las tablas descubiertas por los crawlers o creadas manualmente. Esto es necesario para que servicios como Athena las puedan consultar.

The screenshot shows the AWS Glue Data Catalog "Databases" page. It lists a single database entry: "pet_batch_db". The table includes columns for Name, Description, Location URI, and Created on (UTC). The Location URI is set to "s3://pet-tracking-data-bucket/raw/batch/". The page also features a sidebar with navigation links like "AWS Glue", "Data Catalog", "Data Integration and ETL", and "Legacy pages".

3. Crear un Glue Crawler para catalogar el CSV

- Ir a AWS Glue → Crawlers
- Crear un nuevo crawler:
 - Nombre: **pet-behavior-crawler**
 - Fuente: **S3**
 - Ruta: **s3://pet-tracking-data-bucket/raw/batch/year=2025/month=6/day=23/**
 - Rol: **Labrole**
 - Formato: lo detecta automáticamente
 - Bbdd de destino: **pet_batch_db**
 - Marcar la opción “**Create a single schema for each S3 path**” ya que es útil para particionamiento por fechas.
 - Seleccionar “**Add new columns only**” útil para dar estabilidad de columnas para futuros Redshift ML.

The screenshot shows the AWS Glue Crawler configuration page. The crawler is named "pet-behavior-crawler". It has an IAM role assigned ("LabRole") and is connected to a database named "pet_batch_db". The status is "READY". Under "Advanced settings", the option "Create single schema for each S3 path" is set to "True". The "Object deletion in the data store" section indicates that the table is deprecated. The "Crawler runs" section shows one run completed on June 20, 2025, at 03:16:14. The browser address bar shows the URL: us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/crawlers/view/pet-behavior-crawler.

Este paso detecta el esquema del CSV y lo guarda como tabla en Gue Catalog.

The screenshot shows the AWS Glue Table Detail page for the table "pet_behavior_data_csv". The table has a name of "pet_behavior_data_csv", is classified as CSV, and is located in the "pet_batch_db" database. The last update was on June 20, 2025, at 03:16:14. The "Schema" tab is selected, showing 10 columns: idmascota, timestamp, raza, edad, hr (bpm), actividad (steps), gps_lat, gps_lon, temperatura, and emoción. The browser address bar shows the URL: us-east-1.console.aws.amazon.com/glue/home?region=us-east-1#/v2/data-catalog/tables/view/pet_behavior_data_csv?database=pet_batch_db&catalogId=590184033761&ve...

4. Crear Glue Studio-Job con transformaciones

- Ir a Glue Studio → Workflows o Jobs
- Crear Job visual:
 - Nombre: **transform_pet_behavior**
 - Fuente: “**From Data Catalog**” → tabla: **pet_behavior_data**
- Agregar nodos de transformaciones:

TRANSFORMACIONES	TIPO DE NODO
<p>Renombrar columnas y cambiar tipos de datos: HR(bpm)→ heart_rate_bpm Actividad(steps) → activity_steps temperatura → temperature emoción → emotion raza → breed edad → age. timestamp → timestamp edad → int(long) HR(bpm) → int(long)</p>	Change Schema
<p>Normalizar campo temperature, eliminando ‘°C’ Clasificación de riesgo por heart_rate_bpm y emotion → alert_level Clasifica la actividad física del animal en high, medium, low → activity_level</p>	SQL Query
Salida en Amazon S3	Target Amazon S3

- En la pestaña de configuración:
 - IAM Role: **LabRole**.
 - Worker type: **G.1X**
 - Número de workers: **2-5**

Datos originales en el Job

Source key	Target key	Data type	Drop
idmascota	idmascotz	long	<input type="checkbox"/>
timestamp	timestampj	string	<input type="checkbox"/>
raza	raza	string	<input type="checkbox"/>
edad	edad	long	<input type="checkbox"/>
hr (bpm)	hr (bpm)	long	<input type="checkbox"/>
actividad (steps)	actividad	long	<input type="checkbox"/>
gps_lat	gps_lat	do...	<input type="checkbox"/>
gps_lon	gps_lon	do...	<input type="checkbox"/>
temperatura	temperati	string	<input type="checkbox"/>
emoción	emoción	string	<input type="checkbox"/>

Unsaved job found.
We found an unsaved job; do you wish to restore it?

Datos transformados con CHANGE SCHEMA

The screenshot shows the AWS Glue job configuration interface. The job name is 'transform_pet_behavior'. Under the 'Transform' tab, the 'Change Schema' section is active. A table titled 'Change Schema (Apply mapping)' maps source keys to target keys and data types. The table rows are:

Source key	Target key	Data type	Drop
idmascota	idmascota	long	<input type="checkbox"/>
timestamp	timestamp	timestamp	<input type="checkbox"/>
raza	breed	string	<input type="checkbox"/>
edad	age	int	<input type="checkbox"/>
hr (bpm)	heart_rate_bpm	int	<input type="checkbox"/>
actividad (steps)	activity_steps	long	<input type="checkbox"/>
gps_lat	gps_lat	double	<input type="checkbox"/>
gps_lon	gps_lon	double	<input type="checkbox"/>
temperatura	temperature	string	<input type="checkbox"/>
emoción	emotion	string	<input type="checkbox"/>

Creación de DATA BASE QUALITY

The screenshot shows the AWS Glue Data Quality console. The left sidebar navigation includes 'AWS Glue', 'Tables', 'ETL jobs', 'Data Catalog', 'Data integration and ETL', and 'Legacy pages'. The main content area is titled 'Data quality' and includes sections for 'Getting started with data quality', 'Data quality snapshot', and 'Data quality info'. The 'Data quality info' section shows a table with 17 rule(s) starting.

The screenshot shows the 'Edit dataQuality-pet-behavior' ruleset page. The left sidebar navigation includes 'AWS Glue', 'Tables', 'ETL jobs', 'Data Catalog', 'Data integration and ETL', and 'Legacy pages'. The main content area shows the 'Helper' and 'RuleSet' sections. The 'RuleSet' section contains the following JSON code:

```

    [
      {
        "rule": 1,
        "rule_type": "Incomplete",
        "columns": ["idmascota"],
        "value": "Incomplete"
      },
      {
        "rule": 2,
        "rule_type": "Incomplete",
        "columns": ["timestamp"],
        "value": "Incomplete"
      },
      {
        "rule": 3,
        "rule_type": "ColumnLength",
        "columns": ["timestamp"],
        "value": "ColumnLength > timestamp > 19"
      },
      {
        "rule": 4,
        "rule_type": "Incomplete",
        "columns": ["breed"],
        "value": "Incomplete"
      },
      {
        "rule": 5,
        "rule_type": "ColumnType",
        "columns": ["breed"],
        "value": "ColumnType < breed > int"
      },
      {
        "rule": 6,
        "rule_type": "Incomplete",
        "columns": ["age"],
        "value": "Incomplete"
      },
      {
        "rule": 7,
        "rule_type": "ColumnType",
        "columns": ["age"],
        "value": "ColumnType < age > int"
      },
      {
        "rule": 8,
        "rule_type": "ColumnType",
        "columns": ["age"],
        "value": "ColumnType < age > between 1 and 30"
      },
      {
        "rule": 9,
        "rule_type": "ColumnType",
        "columns": ["heart_rate_bpm"],
        "value": "ColumnType < heart_rate_bpm > between 0 and 350"
      },
      {
        "rule": 10,
        "rule_type": "Incomplete",
        "columns": ["activity_steps"],
        "value": "Incomplete"
      },
      {
        "rule": 11,
        "rule_type": "ColumnType",
        "columns": ["activity_steps"],
        "value": "ColumnType < activity_steps > int"
      },
      {
        "rule": 12,
        "rule_type": "Incomplete",
        "columns": ["gps_lat"],
        "value": "Incomplete"
      },
      {
        "rule": 13,
        "rule_type": "ColumnType",
        "columns": ["gps_lat"],
        "value": "ColumnType < gps_lat > double"
      },
      {
        "rule": 14,
        "rule_type": "ColumnType",
        "columns": ["gps_lat"],
        "value": "ColumnType < gps_lat > <= 89.83"
      },
      {
        "rule": 15,
        "rule_type": "ColumnType",
        "columns": ["temperature"],
        "value": "ColumnType < temperature > float"
      },
      {
        "rule": 16,
        "rule_type": "ColumnType",
        "columns": ["temperature"],
        "value": "ColumnType < temperature > <= 179.95"
      },
      {
        "rule": 17,
        "rule_type": "Incomplete",
        "columns": ["emotion"],
        "value": "Incomplete"
      },
      {
        "rule": 18,
        "rule_type": "ColumnType",
        "columns": ["emotion"],
        "value": "ColumnType < emotion > string"
      },
      {
        "rule": 19,
        "rule_type": "ColumnType",
        "columns": ["emotion"],
        "value": "ColumnType < emotion > in [Ansiedad, Tristeza, Confusión, Tranquilo, Enfadado, Miedo, Relajación, Alerta]"
      }
    ]
  
```

Running DATA QUALITY

The screenshot shows the 'Ruleset details' page for the 'dataQuality-pet-behavior' ruleset. The left sidebar includes sections for AWS Glue, Data Catalog, Data Integration and ETL, and Legacy pages. The main content area displays the ruleset's creation and modification times, a list of tags, and a table of evaluation runs. The table has columns for Run ID, Run status, Ruleset evaluated, Data quality result, Run start time (UTC), and Run end time (UTC). It shows three runs: one completed successfully, one failed (15/17 rules passed), and another completed successfully.

Run ID	Run status	Ruleset evaluated	Data quality result	Run start time (UTC)	Run end time (UTC)
dqrun-77026abd1b3480acce9ed9c6	Completed	dataQuality-pet-behavior	DQ passed (20/20 rules passed)	June 22, 2025 at 02:12:36	June 22, 2025 at 02:12:50
dqrun-2b3481892b915fdafdc4ccb1	Completed	dataQuality-pet-behavior	DQ failed (15/17 rules passed)	June 22, 2025 at 02:01:37	June 22, 2025 at 02:01:52
dqrun-885921ccf35b9bd3a3fec7d8	Completed	dataQuality-pet-behavior	DQ failed (7/17 rules passed)	June 22, 2025 at 01:46:24	June 22, 2025 at 01:46:34

Flujo de ETL

The screenshot shows the 'transform_pet_behavior' ETL job editor. The top navigation bar includes options for Actions, Save, and Run. The left sidebar lists Visual, Script, Job details, Runs, Data quality, Schedules, and Version Control. The main workspace displays a visual flow diagram with four stages: 'Data source - Data Catalog AWS Glue Data Catalog', 'Transform - Change Schema', 'Transform - SQL Query SQL Query', and 'Data target - S3 bucket Amazon S3'. Arrows indicate the flow from source to target. A message at the bottom right indicates an unsaved job found.

```
graph LR; A[Data source - Data Catalog AWS Glue Data Catalog] --> B[Transform - Change Schema]; B --> C[Transform - SQL Query SQL Query]; C --> D[Data target - S3 bucket Amazon S3]
```

STREAM (KINESIS)

1. Crear Función Lambda

- Ir a AWS Lambda → Create function
- Configurar función
 - Nombre: **lambda-pet-sounds-data-firehose**
 - Runtime: **Node.js 22.x**
 - Execution role: **LabRole**
- Escribir el código
- Click en Deploy

The screenshot shows the AWS Lambda console. In the top navigation bar, 'Lambda' is selected under 'Funciones'. The URL in the address bar is <https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>. The main area displays the code editor for the function 'lambda-pet-sounds-data-firehose'. The code is as follows:

```
index.js
1  export const handler = async (event) => {
2    const output = event.records.map((record) => {
3      try {
4        const jsonStr = Buffer.from(record.data, 'base64').toString('utf-8');
5        const input = JSON.parse(jsonStr);
6
7        // Extract timestamp and partition keys
8        const timestamp = input.timestamp;
9        const partitionKey = input.partitionKey;
10       const year = date.getUTCFullYear().toString();
11       const month = String(date.getUTCMonth() + 1).padStart(2, '0');
12       const day = String(date.getUTCDate()).padStart(2, '0');
13
14       // Cleaned object with English field names
15       const cleaned = {
16         pet_id: input.idmascota ?? null,
17         emoción: input.emocion ?? null,
18         decibel_level: parseFloat(input.nivel_decibel ?? 0),
19         timestamp,
20         year,
21         month,
22         day
23       };
24
25       return {
26         recordId: record.recordId,
27         result: 'Ok',
28         data: Buffer.from(JSON.stringify(cleaned) + '\n').toString('base64'),
29         metadata: {
30           partitionKeys: { year, month, day }
31         }
32       };
33     }
34   });
35
36   const records = output.map((record) => {
37     return {
38       partitionKey: partitionKey,
39       data: record.data
40     };
41   });
42
43   return {
44     records
45   };
46 }
```

The sidebar on the left shows the function structure: 'LAMBDA-PET-SOUNDS-DATA-FIREHOSE' > 'index.js'. Below the code editor, there are buttons for 'Deploy' (highlighted) and 'Test'. The status bar at the bottom indicates 'Ln 46, Col 1' and 'UTF-8'.

2. Crear Table manualmente

- Ve a AWS Glue → Data Catalog → Tables
- Add Table
- Propiedades de la Table
 - Database: **pet_streaming_db**
 - Nombre: **pet-sounds-data-cleaned**
 - Tipo: **Amazon S3**
 - Include path: **s3://pet-tracking-data-bucket/firehose-output/**
 - Data format: **Parquet**
 - Compressed: **Sí** (con Snappy en Firehose)
- Definir Columnas
 - **idmascota** → int
 - **emocion** → string
 - **nivel_decibel** → float
 - **timestamp** → string
 - **year** → string → **Add partition key**
 - **month** → string → **Add partition key**
 - **day** → string → **Add partition key**

Esto permite que las consultas de Athena y Redshift filtren y escaneen menos datos.

3. Crear Database y Glue Crawler

- Ve a AWS Glue → Data Catalog → Databases
- Create database
- Nombre: **pet_streaming_db**
- Ve a AWS Glue → Crawlers → Create Crawlers
- Nombre: **pet-sounds-data-crawler**

Antes de nada debemos convertir el archivo CSV en formato JSON para el Firehose.

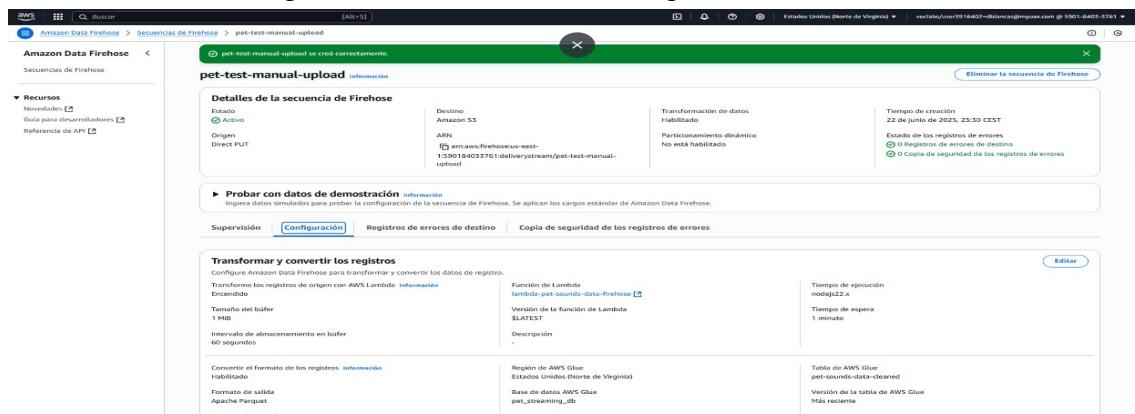
4. Crear un Firehose con origen manual (DIRECT PUT)

- Ve a Kinesis → Firehose → Create Delivery Stream
- Nombre: **pet-test-manual-upload**
- Origen: **DIRECT PUT**
- Destino: **Amazon S3**
- Configuración
 - Transformar tus datos con Lambda:
 - Activar la transformación de datos
 - Seleccionar el Lambda creado anterior
 - Versión: **\$LATEST**
 - Tamaño del búfer: **1 MB**
 - Intervalo búfer: **60 seg**

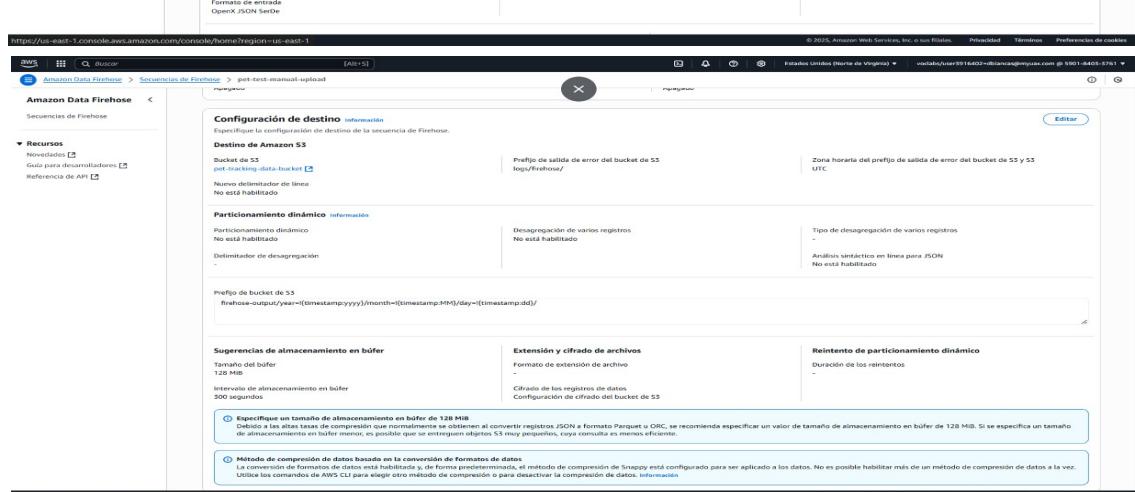
Esto se aplica a tu función Lambda para limpiar y estandarizar cada registro antes de que se almacene.

- Convertir formato de registros:
 - Habilitar la conversión de formatos
 - Formato de salida: **Apache Parquet**
 - Esquema registros de origen:
 - Región Glue: **us-east-1**
 - Bbdd Glue: **pet_streaming_db**
 - Tabla Glue: **pet-sounds-data-cleaned**
 - Versión: **LATEST**
 - Configuración de destino:
 - Bucket S3: **s3://pet-tracking-data-bucket**
 - Nuevo delimitador de línea: **No**
 - Particionamiento dinámico: **No**
 - Prefijo de Bucket:
S3: firehose-output/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day!=!{timestamp:dd}/
 - Prefijo de error de Bucket S3: **logs/firehose/**

Para saber si alguna transformación ha fallado o algún JSON era inválido



The screenshot shows the AWS Lambda function configuration for the 'pet-test-manual-upload' function. It includes details like the function name, runtime (Node.js 12.x), memory (128 MB), timeout (1 minute), and role (arn:aws:lambda:us-east-1:1590184051761:deliverystream/pet-test-manual-upload). The 'Handler' field is set to 'index.handler'. The 'Code' section shows the uploaded ZIP file. The 'Environment' section lists variables: AWS_REGION (us-east-1), AWS_DEFAULT_REGION (us-east-1), and AWS_LAMBDA_FUNCTION_NAME (pet-test-manual-upload).



The screenshot shows the AWS Firehose destination configuration for the 'pet-test-manual-upload' delivery stream. It includes details like the delivery stream name, region (us-east-1), and bucket (s3://pet-tracking-data-bucket). The 'Transformation' section shows the Lambda function used for transformation. The 'Encryption' section shows the use of AWS Lambda server-side encryption. The 'S3' section shows the prefix for the output S3 bucket (firehose-output/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day!=!{timestamp:dd}/) and the prefix for error logs (logs/firehose/).

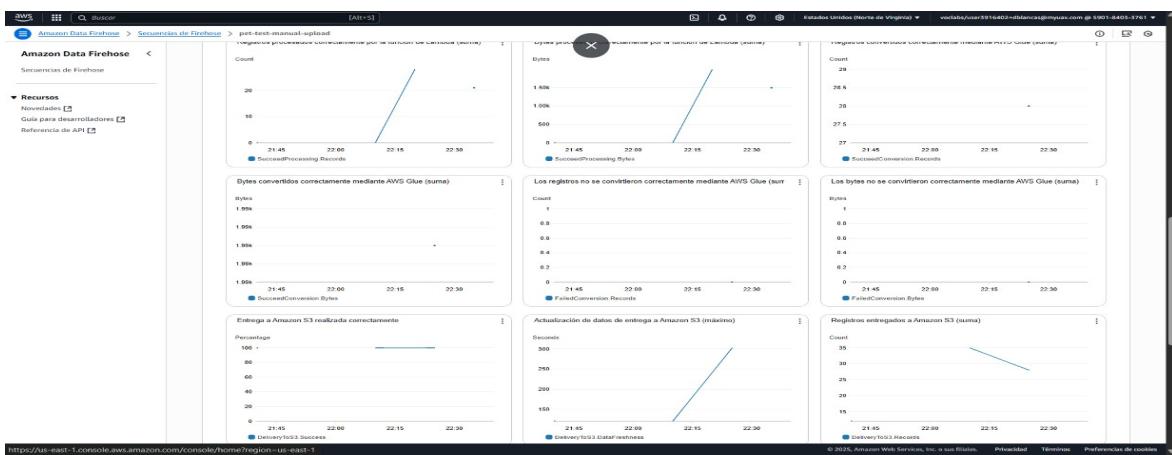
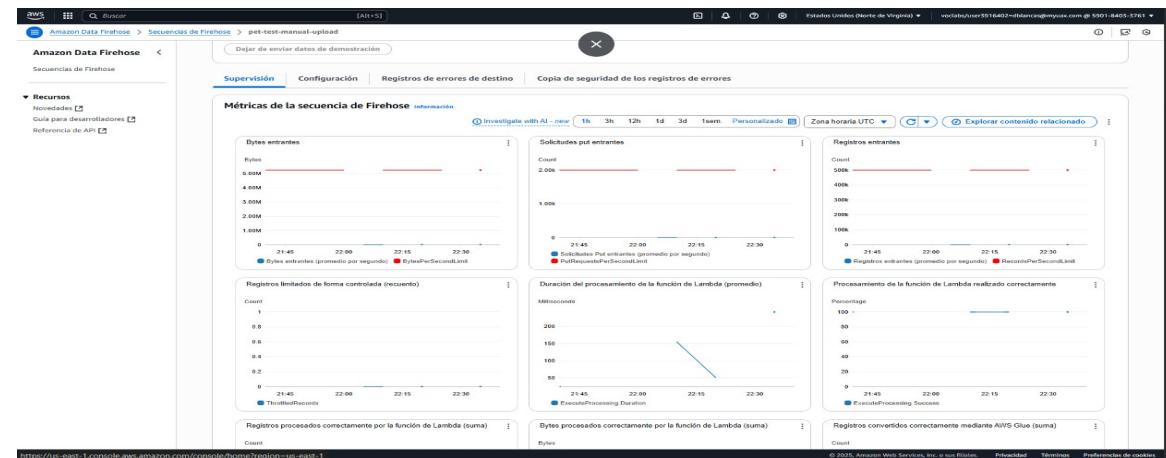
The screenshot shows the AWS Data Firehose console with the following details:

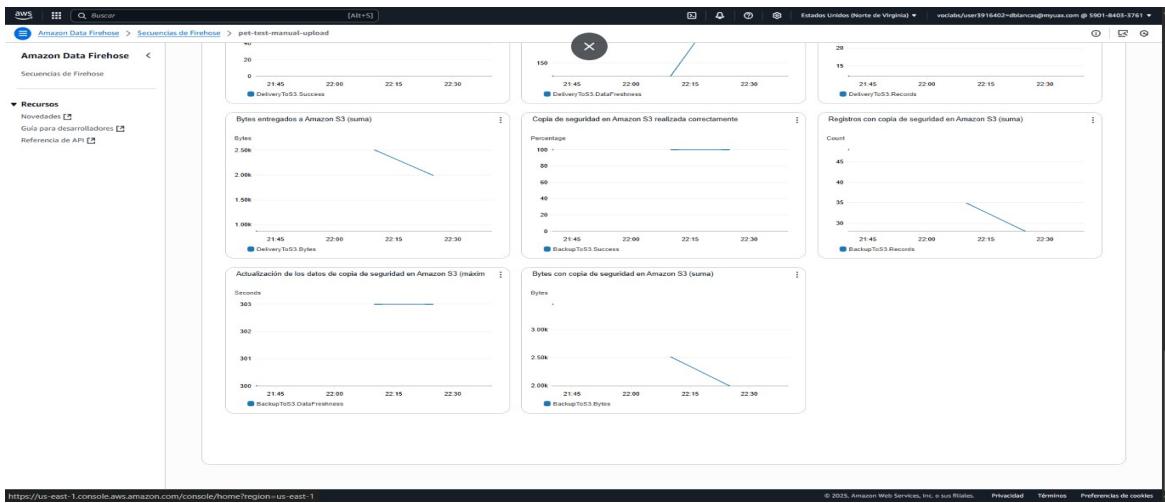
- Delivery Stream Name:** pet-test-manual-upload
- Security:**
 - Copiar la seguridad del registro de origen en Amazon S3: Bucket de copia de seguridad de S3: pet-tracking-data-bucket
 - Prefix del bucket de copia de seguridad de S3: archive/raw/
 - Cifrado del servidor (SSE): Nota: Puede utilizar AWS Key Management Service (KMS) para crear y administrar las claves y para controlar el uso del cifrado en una amplia gama de servicios de AWS en las aplicaciones.
 - Registro de errores de Amazon CloudWatch: Habilitado
- Service Access:** Amazon Data Firehose utiliza este rol de IAM para todos los permisos que la secuencia de Firehose necesita. Si desea especificar diferentes roles para los distintos permisos, utilice la API o la CLI.
- Tags:** No hay etiquetas.

KINESIS FIREHOSE

- Toma eventos crudos(por PUT o desde un Stream)
- Llama a tu Lambda para transformar
- Usa tu tabla Glue como esquema
- Convierte a Parquet
- Entrega los archivos a:
s3://pet-tracking-data-bucket/firehose-output/year=2025/month=6/day=23/

COMPROBACIÓN CON MÉTRICAS Y BUCKET S3





<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>

Amazon S3 > Buckets > pet-tracking-data-bucket > firehose-output/ > year=2025/ > month=06/ > day=22/

day=22/

Objetos Propiedades

Objetos (3)

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](#)

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
pet-test-manual-upload-1-2025-06-22-22-21-08-af54064c-4cb7-4d99-9f95-871e03232de1c9.parquet	parquet	23 Jun 2025 12:26:12 AM CEST	529.0 B	Estandar
pet-test-manual-upload-1-2025-06-22-22-37-19-6687a5b8-1494-4eb9-a9b3-976e85028ff0.parquet	parquet	23 Jun 2025 12:43:26 AM CEST	529.0 B	Estandar
pet-test-manual-upload-1-2025-06-22-22-42-28-958bf952-f01b-498f-a5c5-8a70d87fb020.parquet	parquet	23 Jun 2025 12:48:34 AM CEST	529.0 B	Estandar

Copiar URI de S3

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>

Amazon S3 > Buckets > pet-tracking-data-bucket > logs/ > firehose/ > processing-failed/ > 2025/ > 06/ > 22/ > 22/

22/

Objetos Propiedades

Objetos (1)

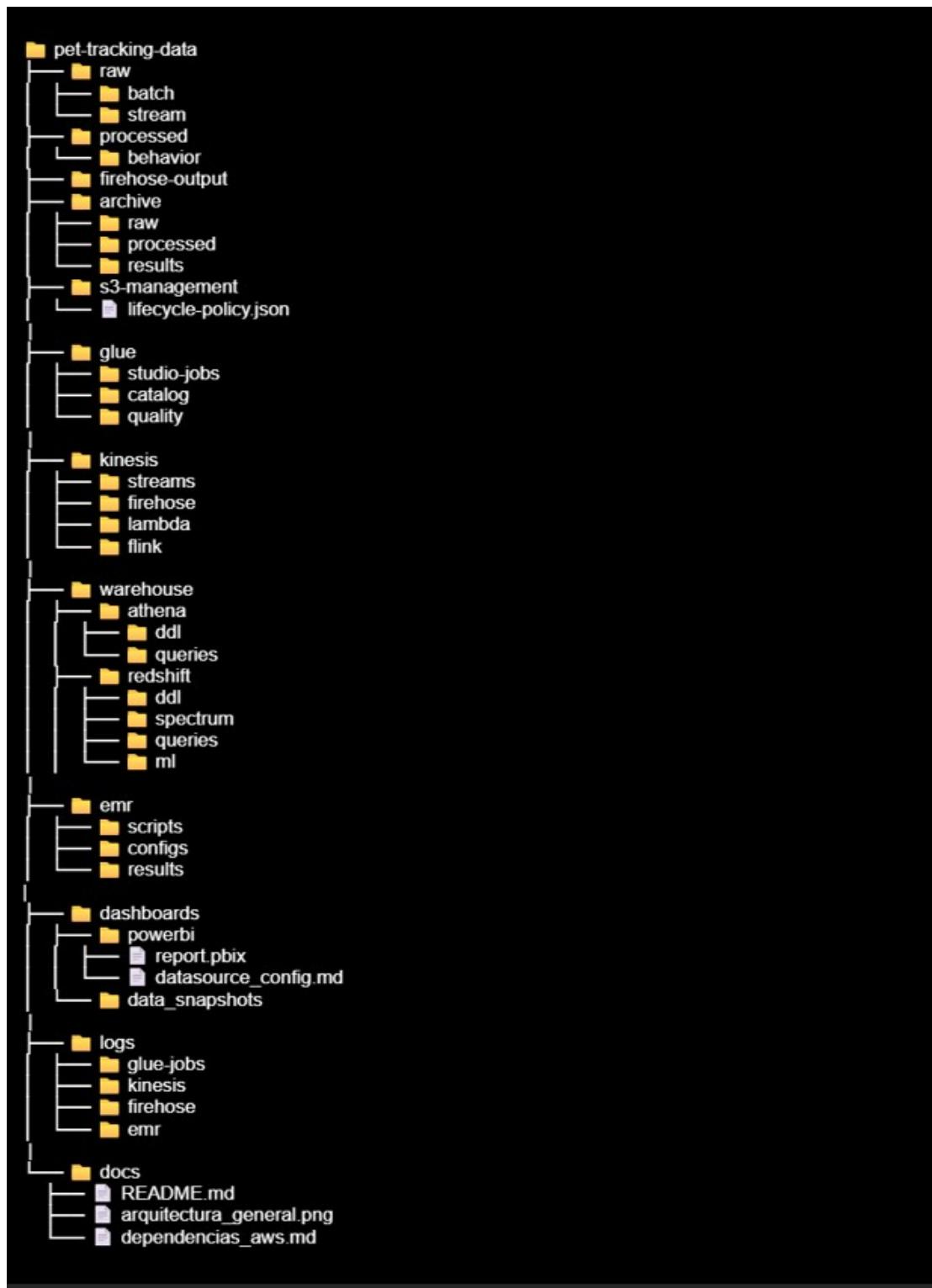
Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el [inventario de Amazon S3](#) para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
pet-test-manual-upload-1-2025-06-22-22-09-03-8fc609a4-e51-4471-b072-02d67deda11	-	23 Jun 2025 12:11:05 AM CEST	4.7 KB	Estandar

Copiar URI de S3

ALMACENAMIENTO

ESTRUCTURA DEL AMAZON S3



A continuación se explica de manera breve y clara la estructura en Amazon S3 creada para el proyecto **pet-tracking-data/**:

CARPETA DE DATOS

- **raw/**
 - Contiene la ingesta de datos en bruto.
 - **batch/**: Datos en lote (Glue).
 - Dentro se puede organizar en **year/, month/, day/**.
 - **stream/**: Datos en tiempo real (Kinesis Streams, Firehose).
 - **year/, month/, day/**.
- **processed/**
 - Datos transformados o enriquecidos, listos para analizar(Flink, Glue, EMR).
 - **year/, month/, day/**.
- **firehose-output/**
 - Archivos automáticos generados por Kinesis Firehose tras su procesamiento inicial.
 - **year/, month/, day/**.
- **archive/**
 - Backup o datos antiguos archivados mediante ciclo de vida.
 - **raw/, processed/, results/** : Copias históricas o comprimidas.
- **s3-management/**
 - Config de almacenamiento.
 - **Lifecycle-policy.json** : Define reglas automáticas de archivo y eliminación.

INGESTA: BATCH(GLUE)

- **glue/studio-jobs/**
 - Scripts visuales creados con Glue Studio para transformar datos.
- **glue/catalog/**
 - Definiciones de Glue Data Catalog(bbdd, tablas).
- **glue/quality/**
 - Reglas y reportes de Glue Data Quality (validaciones de calidad).
 - **year/, month/, day/**.

INGESTA: STREAMING(KINESIS)

- **kinesisstreams/**
 - Config de Kinesis Data Streams (fuente datos en tiempo real).
- **kinesisfirehose/**
 - Config de Kinesis Firehose (ingesta + entrega a S3/Redshift).
- **kinesislambda/**
 - Funciones Lambda utilizadas para transformar eventos en streaming.
- **kinesisflink/**
 - Aplicaciones Flink para procesamiento avanzado en streaming (enriquecimiento, agregaciones).

ALMACENAMIENTO Y CONSULTA: ATHENA/REDSHIFT

- **warehouse/athena/ddl/**
 - Definiciones de tablas externas en Athena(basadas en S3).
- **warehouse/athena/queries/**
 - Consultas SQL ejecutadas en Athena para análisis.
- **warehouse/redshift/ddl/**
 - Tablas internas en Redshift (estructuradas, persistentes).
- **warehouse/redshift/spectrum/**
 - Tablas externas en Redshift usando Spectrum(lectura directa desde S3).
- **warehouse/redshift/ml/**
 - Modelos de machine learning creados con Redshift ML.

PROCESAMIENTO ANALÍTICO (EMR/SPARK)

- **emr/scripts/**
 - Scripts PySpark o SparkSQL usados en EMR para procesamiento batch.
- **emr/configs/**
 - Configs de clústeres EMR, bootstrap actions, roles.

- **emr/results/**
Resultados procesados o exportados por EMR.

VISUALIZACIÓN

- **dashboards/powerbi/**
Reportes .pbix de Power BI y archivo de config de origen de datos.
- **dashboards/data_snapshots/**
Extractos estáticos(CSV, JSON) para usar como fuente en Power BI u otras herramientas.

LOGS

- **logs/**
Registros técnicos de ejecución del servicio.
 - **glue-jobs/, kinesis/, firehose/, emr/**

DOCUMENTACIÓN DEL PROYECTO

- **docs/README.md**
Guía general del proyecto como flujo de datos, servicios usados, instrucciones.
- **docs/arquitectura_general.png**
Diagrama visual de arquitectura(servicios y flujos).
- **docs/dependencias_aws.md**
Listado de servicios AWS utilizados, con enlaces o configs claves.

BENEFICIOS DE ESTRUCTURA

BENEFICIO	CÓMO SE CONSIGUE
Trazabilidad	Cada etapa tiene su propia carpeta y se conserva el dato crudo
Escalabilidad	Subcarpetas por year/, month/, day/ permiten crecimiento sin caos
Separación de roles	Técnicos usan logsl/ , analistas usan dashboards/
Calidad asegurada	glue-quality/ valida antes de pasar a análisis
Recuperabilidad	archive/ conserva versiones antiguas o problemáticas

BUENAS PRÁCTICAS

- Separar claramente las etapas del pipeline(ingesta, almacenamiento, procesamiento, resultados, monitoreo, visualización).
- Escalar bien para nuevos datos, análisis o fechas.
- Facilitar auditoría, trazabilidad, calidad y mantenimiento.
- Integrarse fácilmente con Glue, Kinesis, Athena, EMR, Power BI, etc.

IMPLEMENTAR CICLO DE VIDA EN S3 (BATCH/STREAM)

CARPETA BUCKET S3	ACCIÓN CICLO DE VIDA
raw/batch/	Mover a Glacier a los 30 días
raw/stream/	Eliminar automáticamente a los 7 días
processed/	Eliminar después de 90 días
firehose-output/	Eliminar después de 60 días

1. Ir a Amazon S3 → Bucket (**pet-tracking-data-bucket**)

- Haz click en “Management”
- En “Lifecycle rules”, click en “Create lifecycle rule”
 - Nombre: **raw-stream-delete**
 - Choose rule scope: Apply to prefix → **raw/batch/**
 - Transition current version of objects: **30 days** → **Glacier Flexible Retrieval**
 - Al no marcar ni expire, permanently delete or remove no se borra sino que se archiva.

2. Mismos pasos:

- Nombre: **raw-stream-delete**
- Choose rule scope: Apply to prefix → **raw/stream/**
- Expire current version of objects: **7 days**

The screenshot shows the AWS S3 Lifecycle Configuration page. At the top, there's a green banner stating: "La regla 'firehose-output-delete' se ha agregado correctamente y la configuración del ciclo de vida se ha actualizado. Puede que la configuración tarde algún tiempo en actualizarse. Actualice la lista de reglas del ciclo de vida si no se muestran los cambios en la configuración." Below the banner, the "Lifecycle Configuration" section is visible, showing the configuration for all 128K storage classes. The "Lifecycle rules (4)" table lists the following rules:

Lifecycle rule name	State	Scope	Current version actions	Actions of non-current ...	Expired Item Removal ...	Incompl.
raw-batch-archive	Enabled	Filtered	Transition to Glacier Flexible Retrieval (formerly Glacier)	-	-	-
raw-stream-delete	Enabled	Filtered	Expires	-	-	-
processed-delete	Enabled	Filtered	Expires	-	-	-
firehose-output-delete	Enabled	Filtered	Expires	-	-	-

Guardo una copia JSON de la política de los ciclos de vida implementados en **s3-management/**

The screenshot shows the AWS S3 Objects page for the 's3-management/' folder. The table displays one object:

Name	Last modified	Size	Storage class
lifecycle-policy.json	June 23, 2025, 1:35:54 AM CEST	725.0 B	Standard

CREAR TABLA EN ATHENA Y REALIZAR UNA QUERY

1. Crear tabla particionada con Athena(formato Parquet)

The screenshot shows the AWS Management Console with the Athena service selected. In the left sidebar, under 'Tablas y vistas', there is a table named 'pet_behavior_parquet'. The table details are shown in a modal window:

idmascota	int
emotion	string
heart_rate_bpm	int
activity_steps	int
temperature	string
timestamp	timestamp
year	string (Partitionado)
month	string (Partitionado)
day	string (Partitionado)

The table is stored in the 'pet_batch_db' database and has a location of 's3://pet-tracking-data-bucket/processed/behavior/'. The TBLPROPERTIES include 'parquet.compress' set to 'SNAPPY'.

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> © 2025, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

2. Agregar particiones manualmente si no usas un Crawlers

The screenshot shows the AWS Management Console with the Athena service selected. The table 'pet_behavior_parquet' is selected. A modal window displays the SQL command:

```
HOACK REPAIR TABLE `pet_behavior_parquet`;
```

The results show the command completed successfully with a timestamp of 2025-06-11 14:00:00.000 and a duration of 5,553 sec.

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> © 2025, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

3. Realizamos la query: Vista Table_Athena

The screenshot shows the AWS Management Console with the Athena service selected. The table 'pet_behavior_parquet' is selected. A modal window displays the SQL command:

```
SELECT * FROM "pet_batch_db"."pet_behavior_parquet" limit 10;
```

The results show 10 rows of data from the table, including columns: #, idmascota, emotion, heart_rate_bpm, activity_steps, temperature, timestamp, year, month, and day. The data includes various animal IDs, emotions like Ansiedad, Relajación, Alerta, Confusión, Tristeza, Tranquilo, and their corresponding values for heart rate, activity steps, temperature, and timestamp.

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1> © 2025, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

4. Query: Emociones más frecuentes

Consulta 1 : `SELECT emotion, COUNT(*) AS count FROM pet_batch_db.pet_behavior_parquet WHERE year='2025' AND month='6' AND day='11' GROUP BY emotion ORDER BY count DESC;`

#	emotion	count
1	Ansiedad	1082
2	Tristeza	1080
3	Confusión	1028
4	Tranquilo	988
5	Enfadado	986
6	Miedo	984
7	Relajación	930
8	Alerta	922

AMAZON REDSHIFT

1. CREAR CLÚSTER

- Amazon Redshift → Crear Cluster
- Nombre: **pet_behavior_data_cluster**
- Node Type: **ra3.large**
- Nodes: **2**
- Admin user: **awsuser**
- Manually password: la que quieras
- Rol: **LabRole**

pet-behavior-data-cluster se ha creado correctamente.

Clúster	Nodo reservados	Estado	Espacio de nombre...	Average query dura...	Average number of ...	Zona de disponibilidad	Multi-AZ	Capacidad de almac...	Utilización
pet-behavior-data-cluster	ra3.large 2 nodos 16 TB	Available	20a47adf-a751-46dd...			us-east-1b	No	0 %	0 %

2. HABILITAR ACCESO DESDE REDSHIFT A S3 (SPECTRUM)

- Selecciona tu clúster
- Opción “**Query data**” click en “**Query in the query editor v2**”
- Click en los 3 puntos al lado de **pet_behavior_data_cluster**
- “Crear Conexión”**

- Introducir la **password** que pusimos en el clúster

The screenshot shows the AWS Redshift Query Editor interface. On the left, there's a sidebar with 'Services' like Editor, Queries, Notebooks, Charts, History, and Scheduled queries. The main area shows a tree view of databases: 'pet-behavior-data-cluster' has 'native databases (2)' which include 'dev' and 'public'. 'dev' contains 'Tables' like 'pet_behavior_data_redshift'. A modal window titled 'Edit connection for pet-behavior-data-cluster' is open. It has sections for 'IAM Identity Center' (disabled), 'Federated user' (disabled), 'Temporary credentials using a database user name' (disabled), 'Temporary credentials using your IAM identity' (disabled), 'Database user name and password' (selected), and 'AWS Secrets Manager' (disabled). Under 'Database' settings, 'dev' is selected. In the 'User name' field, 'awsuser' is typed. The 'Password' field is empty and has a note: 'The password must be 8-64 characters.' At the bottom right of the modal, the 'Save' button is highlighted.

- Creamos la tabla en el editor

The screenshot shows the AWS Redshift Query Editor interface. The left sidebar is identical to the previous one. The main area shows the same tree view of databases. In the editor pane, the following SQL code is written:


```

    1. DROP TABLE IF EXISTS pet_behavior_data_redshift CASCADE;
    2. CREATE TABLE pet_behavior_data_redshift (
    3.     pet_id BIGINT,
    4.     upload_date_hour TIMESTAMP,
    5.     breed VARCHAR(30),
    6.     age INT,
    7.     heart_rate_bpm INT,
    8.     activity_steps INT,
    9.     gps_long DOUBLE PRECISION,
    10.    gps_lat DOUBLE PRECISION,
    11.    temperature VARCHAR(8),
    12.    emotion VARCHAR(20)
    13. );
    
```

 The 'Run' button at the top of the editor is highlighted. Below the code, the 'Summary' section shows 'Returned rows: 0' and 'Elapsed time: 209ms'. The 'Result set query:' section shows the same SQL code again.

- Abrimos otro editor en el +
- Introducimos los siguientes parámetros:
 - **COPY**: nombre de la tabla que hemos creado en el editor
 - **FROM: URI de s3** del archivo parquet en el bucket
 - **IAM_ROLE**: el ARN de nuestro rol en este caso **LabRole**
 - **FORMAT AS PARQUET**
 - No es necesario ni la **REGION** ni el **DELIMITER** en el caso de PARQUET

The screenshot shows the AWS Redshift Query Editor v2 interface. In the left sidebar, under 'Queries', there is a tree view of databases: 'pet-behavior-data-cluster' (native databases dev, public), 'sample_data_dev', and 'external databases (1)'. Under 'public', a table named 'pet_behavior_data_redshift' is selected. The main pane shows a query being run:

```

1 COPY pet_behavior_data.redshift;
2 FROM s3://pet-behavior-data-bucket-salida/run-1751329681021-part-block-0-r-00000-snappy.parquet'
3 IAM_ROLE 'arn:aws:iam::835459758363:role/LabRole';
4 FORMAT AS PARQUET;

```

The 'Summary' section indicates that the load into the table completed successfully, returning 4000 rows.

- Hacemos una query de **SELECT ***

The screenshot shows the AWS Redshift Query Editor v2 interface. The table 'pet_behavior_data_redshift' is selected in the sidebar. A new tab 'Untitled 4' is open with the following query:

```

1 SELECT *
2 *
3 FROM
4   "dev"."public"."pet_behavior_data_redshift";

```

The results pane displays the first 4000 rows of the table. The columns are: pet_id, upload_date_hour, breed, age, heart_rate_bpm, activity_steps, gps_lat, gps_lon, temperature, and emotion. The data includes various dog breeds like Dachshund, Shetland Sheepdog, and Beagle, along with their ages, heart rates, GPS coordinates, and temperatures.

MACHINE LEARNING REDSHIFT

- CREAMOS UN BUCKET S3 DE DESTINO**
- CREAR UN MODEL DE MACHINE LEARNING REDSHIFT**
 - Entrenar un modelo K-means con datos numéricos por compatibilidad:
 - age
 - heart_rate_bpm
 - activity_steps
 - gps_lat
 - gps_lon
 - Descartamos temperature porque contiene ºC y es string/varchar
 - IAM_ROLE: ARN de nuestro LabRole
 - Run y se Entrena

```

CREATE MODEL pet_behavior_cluster_model
FROM (
    SELECT
        age,
        heart_rate_bpm,
        activity_steps,
        gps_lat,
        gps_lon
    FROM pet_behavior_data_redshift
)
FUNCTION new_cluster_pet_behavior
IAM_ROLE 'arn:aws:iam::8354978363:role/LabRole'
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT
EXCEPT ('')
SETTINGS (
    S3_BUCKET 'ml-kmeans-pet-behavior-data'
);

```

Summary
Returned rows: 0
Query ID: 4327
Elapsed time: 7s
Result set query:

```

/* REVOKE EXECUTE ON */
CREATE MODEL pet_behavior_cluster_model
FROM (
    SELECT
        age,
        heart_rate_bpm,
        activity_steps,
        gps_lat,
        gps_lon
    FROM pet_behavior_data_redshift
)
FUNCTION new_cluster_pet_behavior
IAM_ROLE 'arn:aws:iam::8354978363:role/LabRole'
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT
EXCEPT ('')
SETTINGS (
    S3_BUCKET 'ml-kmeans-pet-behavior-data'
);

```

3. USAR EL MODEL K-MEANS

- Predicción de clústeres

```

SELECT
    pet_id,
    age,
    heart_rate_bpm,
    activity_steps,
    gps_lat,
    gps_lon,
    new_cluster_pet_behavior(
        age,
        heart_rate_bpm,
        activity_steps,
        gps_lat,
        gps_lon
    ) AS cluster_id
FROM pet_behavior_data_redshift;

```

Result 1 (4000)

pet_id	age	heart_rate_bpm	activity_steps	gps_lat	gps_lon	cluster_id
205	19	146	163	51.64	162.19	1
212	20	165	194	72.32	-49.73	2
223	10	72	19	2.2	-105.48	2
225	6	170	170	-82.5	-178.02	0
231	15	157	120	82.17	108.93	1
232	8	70	106	42.75	25.94	1
234	15	115	174	-36.62	-74.34	0
239	19	126	55	-58.66	-139.02	0
243	11	101	107	-42.29	-2.32	4
244	5	68	188	-86.23	72.81	3
253	14	138	0	20.7	38.85	1
255	12	131	34	25.9	116.56	1
265	4	87	125	56.12	-138.78	2
267	4	136	28	-66.94	-15.75	4
271	10	104	170	-23.01	90.42	3
274	19	152	171	20.33	-52.28	2

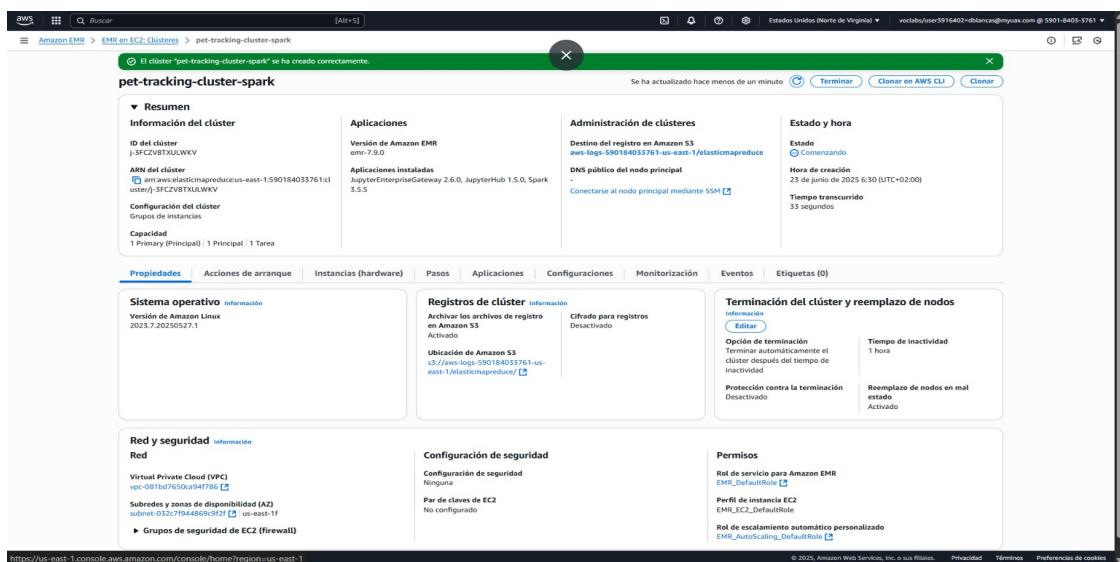
Query ID 4495 Elapsed time: 5702 ms Total rows: 4000

PROCESAMIENTO

AWS EMR

1. Crear cluster

- Ir a EMR → Clusters → Crear Clusters
- Seleccionar “**Go to advanced options**”
- Configuración
 - Amazon EMR release: versión más reciente
 - Componentes: **Spark(Jupyter)**
- Edit Steps
 - **Spark application**
 - Tipo: **Spark**
 - Action on failure: **Continue**
- Nombre: **pet-behavior-emr**
- Instance type: **m4.large**
- Roles: **EMR_DefaultRole...**



2. Crear un archivo .py con las transformaciones

- Nombre: **pet_behavior_processing.py**
- Contenido: Script creado
- Destino: **s3://pet-tracking-data-bucket/emr/scripts/pet_behavior_processing.py**

3. Crear un Studio y Workspace

- Marcar la opción “**Personalizado**”
- Nombre: **pet-behavior-emr-studio**
- Ubicación: **s3://pet-tracking-data-bucket/emr/results**
- Rol: **LabRole**
- Nombre Workspace: **pet-behavior-emr-workspace**
- Seleccionar la VPC y Subred: las primeras opciones del desplegable

The screenshot shows the AWS EMR Studio interface. On the left, there's a sidebar with navigation options like 'Amazon EMR', 'EMR sin servidor', 'EMR en EC2' (selected), 'EMR en EKS', 'EMR Studio', and 'Novedades'. The main area displays a success message: 'Ha creado correctamente el Studio pet-behavior-emr-studio y al espacio de trabajo pet-behavior-emr-workspace.' Below this, there's a table for 'Estudios (1) informes' with one row for 'pet-behavior-emr-studio'. The table includes columns for 'Nombre del Studio', 'Hora de creación (UTC+02:00)', 'Autenticado por', and 'URL de acceso a Studio'. A 'Create a Studio' button is visible at the top right.

4. Empezar Jupyter

- Seleccionamos el Workspace → Acciones → Detener
- En Inactivo asociamos el clúster
- Lanzar en Jupyter
- Desplegamos y elegimos el clúster

This screenshot shows the JupyterLab interface. On the left is a file browser showing a folder named 'pet-behavior-emr-worksp...'. The main area is the 'Launcher' panel, which contains three sections: 'Notebook' (Python 3, Notebook Examples, PySpark, Spark, SparkR), 'Console' (Python 3, PySpark, Spark, SparkR), and 'Other' (Terminal, Collaboration, Text File, Markdown File, Python File, Show Contextual Help, SQL Explorer). The 'Launcher' tab is selected at the bottom.

This screenshot shows a Jupyter Notebook cell containing PySpark code. The code filters alerts based on heart rate and emotion, writes them to a JSON file, and then runs a job to classify activity levels. A progress bar indicates the job is complete. The notebook tab is titled 'pet-tracking-jupyter.ipynb'.

```

# 2. Filtrar alertas altas (dolor + ritmo alto)
alertas_altas = df.filter((col("heart_rate_bpm") > 130) & (col("emotion") == "dolor"))
alertas_altas.write.mode("overwrite").json("s3://pet-tracking-data-bucket/emr/results/json_alertas_altas/")
Last executed at 2025-06-23 11:17:36 in 5.33s

# 3. Clasificar nivel de actividad
df = df.withColumn(
    "activity_level",
    when(col("activity_steps") >= 10000, "high")
    .when(col("activity_steps") >= 5000, "medium")
    .otherwise("low")
)
Last executed at 2025-06-23 11:17:36 in 200ms

# 4. Agrupaciones por mascota
agg = df.groupby("idmascota").agg(

```

Lanzamiento del Laboratorio | Página de inicio de la Comunidad | emr/results/json_errors/ | Managed Apache Flink | EMR Studio: WorkSpaces | pet-tracking-ju - JupyterLab

File Edit View Run Kernel Git Tabs Settings Help

Launcher pet-tracking-jupyter.ipynb Cluster attached ... PySpark

Filter files by name /

Name Last Modified

pet-behavior-emr-worksp... an hour ago

pet-tracking-jupyter.ipynb seconds ago

[1]: from pyspark.sql import SparkSession
from pyspark.sql.functions import when, col, avg

Inicia sesión Spark
spark = SparkSession.builder.appName("PetBehaviorProcessing").getOrCreate()

Last executed at 2025-06-23 11:17:17 in 41.83s

Spark Job Progress

Starting Spark application

ID	YARN Application ID	Kind	State	Spark UI	Driver log	User	Current session?
5	application_17506652140.0006	pyspark	idle	Link	Link	None	✓

SparkSession available as 'spark'.

[2]: # 1. Leer desde Parquet (ruta base del procesamiento por lotes)
df = spark.read.parquet("s3://pet-tracking-data-bucket/processed/behavior/")

Last executed at 2025-06-23 11:17:31 in 13.36s

Spark Job Progress

Job [0]: parquet at NativeMethodAccessorImpl.java:0

Progress for parquet at NativeMethodAccessorImpl.java:0		Job Progress: 1/1 Tasks Complete		
Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)
Stage [0]: parquet at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	7.284	

Saving completed Mode: Command Ln 1, Col 1 pet-tracking-jupyter.ipynb

Lanzamiento del Laboratorio | Página de inicio de la Comunidad | emr/results/json_errors/ | Managed Apache Flink | EMR Studio: WorkSpaces | pet-tracking-ju - JupyterLab

File Edit View Run Kernel Git Tabs Settings Help

Launcher pet-tracking-jupyter.ipynb Cluster attached ... PySpark

Filter files by name /

Name Last Modified

pet-behavior-emr-worksp... an hour ago

pet-tracking-jupyter.ipynb a minute ago

[5]: # 4. Agregaciones por mascota
agg = df.groupby("idmascota").agg(
 avg("heart_rate_bpm").alias("avg_heart_rate"),
 avg("activity_steps").alias("avg_activity_steps")
)
agg.write.mode("overwrite").json("s3://pet-tracking-data-bucket/emr/results/json_agregados_por_mascota/")

Last executed at 2025-06-23 11:17:41 in 5.33s

Spark Job Progress

Job [2]: json at NativeMethodAccessorImpl.java:0

Progress for json at NativeMethodAccessorImpl.java:0		Job Progress: 1/1 Tasks Complete		
Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)
Stage [2]: json at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	1.985	

Saving completed Mode: Command Ln 6, Col 82 pet-tracking-jupyter.ipynb

Job [3]: json at NativeMethodAccessorImpl.java:0

Progress for json at NativeMethodAccessorImpl.java:0		Job Progress: 1/1 Tasks Complete		
Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)
Stage [3]: json at NativeMethodAccessorImpl.java:0	SKIPPED	0/1	n/a	
Stage [4]: json at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	1.115	

Lanzamiento del Laboratorio | Página de inicio de la Comunidad | emr/results/json_errors/ | Managed Apache Flink | EMR Studio: WorkSpaces | pet-tracking-ju - JupyterLab

File Edit View Run Kernel Git Tabs Settings Help

Launcher pet-tracking-jupyter.ipynb Cluster attached ... PySpark

Filter files by name /

Name Last Modified

pet-behavior-emr-worksp... an hour ago

pet-tracking-jupyter.ipynb 2 minutes ago

[6]: # 5. Detección de valores anómalos
errores = df.filter((col("heart_rate_bpm") < 0) | (col("activity_steps") > 50000))
errores.write.mode("overwrite").json("s3://pet-tracking-data-bucket/emr/results/json_errores/")

Last executed at 2025-06-23 11:17:43 in 1.29s

Spark Job Progress

Job [4]: json at NativeMethodAccessorImpl.java:0

Progress for json at NativeMethodAccessorImpl.java:0		Job Progress: 1/1 Tasks Complete		
Stage [ID]	name at [source] [line]	Status	Task Progress	Elapsed Time (seconds)
Stage [4]: json at NativeMethodAccessorImpl.java:0	COMPLETE	1/1	0.396	

Saving completed Mode: Command Ln 3, Col 86 pet-tracking-jupyter.ipynb

[7]: # 6. Filtrar alertas bajas (dolor + ritmo bajo)
alertas_bajas = df.filter((col("heart_rate_bpm") < 60) & (col("emotion") == "dolor"))

Last executed at 2025-06-23 11:17:43 in 60ms

<https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1>

© 2025, Amazon Web Services, Inc. o sus filiales. Privacidad Términos Preferencias de cookies

VISUALIZACIÓN

DASHBOARD 1: ANÁLISIS DE DATOS Y EMOCIONES

- Cantidad de audios
- Emoción más común
- Evolución de emociones en el tiempo
- Frecuencia máxima por emoción
- Tabla con audio + emoción + espectograma

Fuente: [pet-sounds-data-csv](#)

1. IMPORTAR DATOS

- Abrir PowerBI Desktop
- Click en “Obtener datos” → “Texto/CSV”
- Seleccionar [pet-sound-data.csv](#)
- Cargar

2. LIMPIAR Y TRANSFORMAR DATOS

- Inicio (Transformar datos)
- Cambiar los nombres de las columnas:
 - **Frecuencia_MAX(Hz) → Frecuencia_Max**
 - **Espectograma(path) → Espectograma**
- Cambiar el tipo de dato de columna:
 - En la columna **Timestamp** click en el icono al lado del nombre y cambiar a formato **Fecha/Hora**
- Aplicar cambios y cerrar.

3. CREAR CAMPOS CALCULADOS (DAX)

- Total de audios:

Total audios = COUNT('pet-sound-data'[IDAaudio])

- Emoción más común:

```
Emoción Más Común =
CALCULATE(
MAXX(
TOPN(1,
SUMMARIZE('pet-sounds-data', 'pet-sounds-data'[Emoción],
"Conteo",COUNT('pet-sounds-data'[Emoción])), [Conteo], DESC
),
'pet-sounds-data'[Emoción]
)
)
```

- Mes:

Mes = FORMAT('pet-sounds-data'[Timestamp], "MMM")

4. COMPILAR GRÁFICOS, SLICERS, MAPAS... Y FORMATEAR TODO EL DISEÑO Y ESTILO

Visualizaciones - Guardado por última vez: Hoy a las 2:58

Monitor de Bienestar Animal

Actividad por Raza

Promedio de Actividad

Frecuencia Cardíaca vs Tiempo

Frecuencia

Mes

Ubicación por Estado Emocional

ID Mascota Raza Emoción

ID Mascota	Raza	Emoción
1	Siamés	Ansiiedad
2	Doberman	Relajación
3	Bull Terrier	Relajación
4	Siamés	Alerta
5	Maltes	Ansiedad
6	Dachshund	Confusión
7	Collie	Tristeza
8	Scottish Fold	Confusión

Formato

Compilar

Datos

Buscar

- > Información de la página
- > Configuración del lienzo
- > Fondo del lienzo
- > Papel tapiz
- > Panel de filtros
- > Tarjetas de filtro

Datos

Agregar datos

Página 2 de 3

Visualizaciones - Guardado por última vez: Hoy a las 2:58

Analís de comportamientos por mascota

Duración Promedio

29,9

Total Sesiones

500

Comportamientos por Día

Duración por Tipo

Distribución de Comportamientos

ID Mascota Duración Minutos Acción

1000	54	caminar
1000	17	comer
1000	135	correr

ID Mascota Todas

Formato

Compilar

Datos

Buscar

- > Información de la página
- > Configuración del lienzo
- > Fondo del lienzo
- > Papel tapiz
- > Panel de filtros
- > Tarjetas de filtro

Datos

Agregar datos

Página 3 de 3