- **Paso 7.** Detener o terminar el clúster:

Una vez que hayas terminado de trabajar con tu clúster EMR, es importante detenerlo para evitar cargos adicionales.

- Detener el clúster. En la consola de EMR, selecciona tu clúster y haz clic en «Terminate». Esto apagará todas las instancias EC2 asociadas al clúster.

Procesamiento de datos con AWS EMR

En este apartado, vamos a explorar cómo se realiza el procesamiento de datos en un clúster EMR con distintas herramientas dentro de AWS EMR, describiendo los tipos de procesamiento, las herramientas principales que utiliza EMR y el modo en el que ejecutar tareas dentro de este entorno.

Herramientas de procesamiento de datos en EMR

AWS EMR soporta varias herramientas para el procesamiento de datos a gran escala. Las principales herramientas que puedes utilizar son:

- **Apache Hadoop:**

Apache Hadoop es un marco de procesamiento distribuido que permite procesar grandes cantidades de datos en paralelo a través de un clúster de máquinas. Hadoop se compone principalmente de dos partes:

- HDFS (Hadoop Distributed File System). Un sistema de archivos distribuido que permite almacenar grandes volúmenes de datos en múltiples nodos.
- MapReduce. Un modelo de programación que distribuye el procesamiento de datos en paralelo en varios nodos.

- **Uso de Hadoop en EMR:** Hadoop es ideal para el procesamiento por lotes y se utiliza para ejecutar trabajos de análisis de grandes volúmenes de datos que no requieren respuestas inmediatas.

- **Apache Spark:**

Apache Spark es una de las herramientas más populares para el procesamiento de datos en tiempo real y en lotes. Spark es más rápido que Hadoop, gracias a su capacidad de almacenar datos en memoria y realizar operaciones en paralelo.

- **Ventajas** de Spark:

  - Procesamiento más rápido.
  - Ideal tanto para procesamiento en tiempo real como por lotes.
  - Permite operaciones complejas como *machine learning* y análisis de gráficos.

- **Uso de Spark en EMR.** Puedes usar Spark en EMR para ejecutar trabajos en tiempo real o por lotes, hacer análisis complejos y construir aplicaciones de *machine learning.*

- **Apache Hive:**

Apache Hive es un sistema de almacenamiento de datos que facilita el uso de Hadoop al permitir escribir consultas en un lenguaje similar a SQL. Esto lo convierte en una opción popular para quienes están familiarizados con bases de datos tradicionales, pero necesitan procesar datos a gran escala.

- **Uso de Hive en EMR.** Hive se utiliza en EMR para ejecutar consultas SQL sobre grandes volúmenes de datos almacenados en HDFS, facilitando el análisis de datos sin tener que escribir código complejo.

Preparación de los datos para el procesamiento

Antes de comenzar a procesar datos en EMR, es esencial preparar y cargar esos datos correctamente. A continuación, te mostramos cómo hacerlo.

- **Paso 1.Subir datos a S3.** AWS S3 (Simple Storage Service) es un servicio de almacenamiento que se utiliza para almacenar los datos que procesarás con EMR. Los pasos son simples:

- Crear un *bucket* en S3. En la consola de AWS, accede a S3 y crea un *bucket* para almacenar tus datos.

- Subir los datos. Puedes cargar tus archivos desde tu computadora o desde otros servicios en la nube. S3 soporta diferentes tipos de datos: CSV, JSON, Parquet, etc.

- **Paso 2.Conectar EMR con S3.** Una vez que los datos están almacenados en S3, EMR puede acceder a ellos para su procesamiento. No necesitas realizar configuraciones complicadas, ya que EMR puede conectarse a los archivos de S3 directamente durante la ejecución de trabajos.

Ejecución de trabajos en EMR

Con los datos listos en S3 y tu clúster EMR en funcionamiento, el siguiente paso es ejecutar trabajos de procesamiento. Para ello, se pueden utilizar varias herramientas, como Hadoop, Spark o Hive. Aquí te mostramos cómo hacerlo.

Aquí puedes encontrar una guía de ejecución Jobs en EMR:

https://docs.aws.amazon.com/es_es/emr/latest/EMR-Serverless-UserGuide/jobs-studio.html

---

Ejemplo de procesamiento con Apache Spark en EMR

Supongamos que tienes un archivo CSV con información de ventas y deseas analizarlo usando Spark en EMR. Aquí tienes un ejemplo de código en Python utilizando PySpark, la interfaz de Python para Spark:

- **Conexión a Spark.** Primero, necesitas crear una sesión de Spark en el nodo maestro del clúster EMR:

```
python

from pyspark.sql import SparkSession

# Crear una sesión de Spark

spark = SparkSession.builder.appName("AnálisisVentas").getOrCreate()
```

- **Leer los datos desde S3.** Luego, lee el archivo CSV almacenado en S3:

```
python

# Leer datos desde S3

df = spark.read.csv("s3://mi-bucket-de-datos/ventas.csv", header=True,
inferSchema=True)
```

- **Realizar operaciones de análisis.** Ahora puedes realizar operaciones de análisis sobre los datos, como calcular la venta total por producto:

```
python

# Calcular la venta total por producto

df.groupBy("producto").sum("venta").show()
```

- **Guardar los resultados.** Después de procesar los datos, puedes guardar los resultados de vuelta en S3:

```
python

# Guardar los resultados en S3

df.write.csv("s3://mi-bucket-de-datos/resultados/")
```

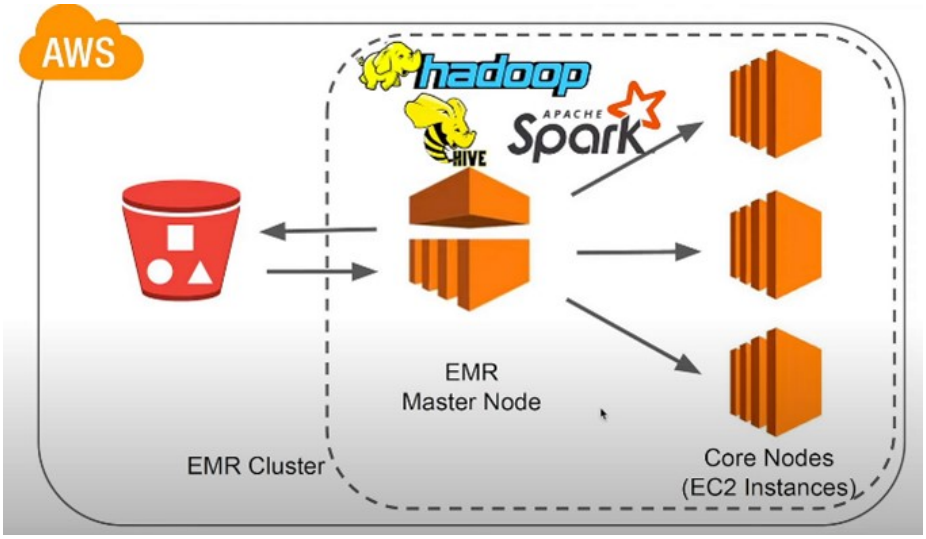

Figura 18. Flujo de trabajo AWS EMR. Fuente: Kumar, 2023.

Procesamiento de datos con Hadoop en AWS EMR

A continuación, te presento una guía detallada y paso a paso para realizar procesamiento de datos con Apache Hadoop en AWS EMR. Este ejemplo se centrará en contar la frecuencia de palabras en un archivo de texto almacenado en Amazon S3, un caso clásico para comprender el funcionamiento de Hadoop MapReduce.

- **Paso 1.** Preparación del entorno:
- Crear un *bucket* en S3:
  - Inicia sesión en la consola de AWS.
  - Ve a S3 y selecciona «Crear Bucket».
  - Introduce un nombre único para tu *bucket,* por ejemplo: mi-bucket-de-datos.
  - Selecciona una región cercana a tu ubicación.
  - Configura las opciones restantes según las necesidades (puedes dejar la configuración predeterminada).
  - Haz clic en «Crear Bucket».
- Subir el archivo de datos:
  - Accede al *bucket* creado.
  - Haz clic en «Subir» y selecciona el archivo de texto que deseas procesar, por ejemplo: texto_prueba.txt. Este archivo debe contener contenido de texto plano con palabras separadas por espacios.
  - Una vez subido, copia la URL del archivo, que será algo como: s3://mi-bucket-de-datos/texto_prueba.txt.

- **Paso 2.** Crear un clúster EMR:
- Ve a la consola de AWS EMR y selecciona «Crear Clúster».
- Configura el clúster de la siguiente manera:
  - Nombre del clúster: introduce un nombre como «ProcesamientoHadoopEMR».
  - Lanzar en una red VPC: selecciona una VPC existente o predeterminada.
  - Aplicaciones y *frameworks:* asegúrate de seleccionar Hadoop.
  - Tipo de instancia. Nodo maestro: m5.xlarge (puedes usar un tipo más pequeño para pruebas). Nodos núcleo: dos instancias de m5.xlarge.
  - Configuración de almacenamiento: usa el predeterminado o personaliza según tus necesidades.
  - Permisos de IAM: asegúrate de que tu rol tenga permisos para S3 y EMR.
  - Haz clic en «Crear Clúster».
- **Paso 3.** Escribir el código MapReduce:

En Hadoop, el procesamiento de datos se divide en dos fases principales: Map y Reduce. Crearás dos *scripts* para estas fases y un controlador.

- Crear el *script* del *mapper:* el *mapper* lee las líneas del archivo de entrada y emite pares clave-valor, donde la clave es la palabra y el valor es 1. Guarda este código en un archivo llamado «mapper.py» (A).
- Crear el *script* del *reducer:* el *reducer* suma los valores asociados a cada clave. Guarda este código en un archivo llamado «reducer.py» (B).

A

```
python

#!/usr/bin/env python3

import sys

# Leer líneas de entrada

for line in sys.stdin:

    # Dividir en palabras

    words = line.strip().split()

    for word in words:

        # Emitir clave-valor

        print(f"{word}\t1")
```

B

```
python

#!/usr/bin/env python3

import sys

current_word = None

current_count = 0
```

```
# Leer línea por línea

for line in sys.stdin:

    word, count = line.strip().split('\t')

    count = int(count)

    if word == current_word:

        current_count += count

    else:

        if current_word:

            # Emitir resultado para la palabra anterior

            print(f"{current_word}\t{current_count}")

        current_word = word

        current_count = count

# Emitir el último resultado

if current_word:

    print(f"{current_word}\t{current_count}")
```

- **Paso 4.** Subir los *scripts* a S3:
- Ve a tu *bucket* en S3.
- Sube los archivos mapper.py y reducer.py al *bucket.*
- **Paso 5.** Ejecutar el trabajo en el clúster EMR:
- Accede al clúster EMR creado y selecciona «Pasos».
- Haz clic en «Añadir Paso».
- Configura el paso:
  - Tipo de paso: Custom Jar.
  - Nombre: Contar Palabras.
  - Comando de *streaming:* proporciona los siguientes detalles: **Hadoop Streaming JAR:** usa la ruta predeterminada: command-runner.jar. Argumentos (ver después de paso 4).

- Haz clic en Añadir y el trabajo comenzará automáticamente.

```
bash

hadoop-streaming \

-files s3://mi-bucket-de-datos/mapper.py,s3://mi-bucket-de-datos/reducer.py
\

-mapper "python3 mapper.py" \

-reducer "python3 reducer.py" \

-input s3://mi-bucket-de-datos/texto_prueba.txt \

-output s3://mi-bucket-de-datos/salida_hadoop/
```

- **Paso 6.** Ver los resultados:
- Una vez que el trabajo haya terminado, accede a tu *bucket* en S3.
- Navega a la carpeta «salida_hadoop/».
- Descarga el archivo «part-00000».
- Abre el archivo y verás las palabras y su frecuencia:

```
palabra1   10

palabra2   5

...
```

Procesamiento de consulta SQL con Apache Hive en AWS EMR

Apache Hive es una herramienta que facilita la consulta y el análisis de grandes conjuntos de datos almacenados en Hadoop, utilizando un lenguaje similar a SQL. En esta guía, aprenderás cómo usar Hive en AWS EMR para realizar una consulta SQL básica sobre un conjunto de datos almacenado en Amazon S3.

- **Paso 1.** Preparación del entorno:
- Crear un *bucket* en S3:
  - Inicia sesión en la consola de AWS.
  - Ve a S3 y selecciona «Crear Bucket».
  - Introduce un nombre único para tu *bucket,* por ejemplo: mi-bucket-hive-datos.
  - Selecciona una región cercana a tu ubicación.
  - Configura las opciones restantes según las necesidades (puedes dejar la configuración predeterminada).
  - Haz clic en «Crear Bucket».
- Subir el conjunto de datos: prepara un archivo CSV con datos tabulares. Por ejemplo, crea un archivo «ventas.csv» con el siguiente contenido (ver tras paso 5).
- Accede al *bucket* creado.
- Haz clic en «Subir» y selecciona el archivo «ventas.csv».
- Copia la URL del archivo, que será algo como: s3://mi-bucket-hive-datos/ventas.csv.

```
csv

id,producto,categoria,precio,fecha

1,Camiseta,Ropa,20,2024-01-01

2,Pantalón,Ropa,30,2024-01-02

3,Zapatos,Calzado,50,2024-01-03

4,Sombrero,Ropa,15,2024-01-04

5,Calcetines,Calzado,5,2024-01-05
```

- **Paso 2.** Crear un clúster EMR con Hive:
- Ve a la consola de AWS EMR y selecciona «Crear Clúster».
- Configura el clúster de la siguiente manera:
  - Nombre del clúster: introduce un nombre como «ProcesamientoHiveEMR».
  - Lanzar en una red VPC: selecciona una VPC existente o la predeterminada.
  - Aplicaciones y *frameworks:* asegúrate de seleccionar Hive y Hadoop.
  - Tipo de instancia. Nodo maestro: m5.xlarge (puedes usar un tipo más pequeño para pruebas). Nodos núcleo: dos instancias de m5.xlarge.
  - Configuración de almacenamiento: usa el predeterminado o personaliza según tus necesidades.
  - Permisos de IAM: asegúrate de que tu rol tenga permisos para S3, EMR y Hive.

- Haz clic en Crear Clúster.
- **Paso 3.** Acceder al clúster:
- Una vez que el clúster esté en estado «En ejecución», selecciona el clúster y haz clic en «Conectar».
- Elige la opción de conexión SSH y copia el comando para conectarte al nodo maestro desde tu terminal.
- **Paso 4.** Configurar el entorno de Hive:
- Una vez conectado al nodo maestro, inicia la consola de Hive ejecutando (A).
- Configura el acceso a los datos en S3 ejecutando (B).
- (Sustituye TU_ACCESS_KEY y TU_SECRET_KEY por tus credenciales de acceso de AWS).

A

```
bash
```

```
hive
```

B

```
sql
```

```
SET fs.s3a.access.key=TU_ACCESS_KEY;
```

```
SET fs.s3a.secret.key=TU_SECRET_KEY;
```

- **Paso 5.** Crear una tabla y cargar los datos:
- Crear una tabla externa. Crea una tabla externa que apunte al archivo en S3 (A).
- Verificar los datos. Ejecuta una consulta simple para asegurarte de que los datos se cargaron correctamente (B).
- Deberías ver el contenido del archivo ventas.csv.

A

```
sql

CREATE EXTERNAL TABLE ventas (

  id INT,

  producto STRING,

  categoria STRING,

  precio FLOAT,

  fecha STRING

)

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

STORED AS TEXTFILE

LOCATION 's3://mi-bucket-hive-datos/';
```

B

```
sql

SELECT * FROM ventas;
```

- **Paso 6.** Realizar consultas SQL:
- Consulta: ventas totales por categoría. Calcula el total de ventas (suma de precios) por cada categoría (A).
- Consulta: productos con precio mayor a 20. Obtén los productos cuyo precio es superior a 20 (B).
- Consulta: ordenar ventas por fecha. Ordena las ventas por fecha de forma ascendente (C).

A

```
sql

SELECT categoria, SUM(precio) AS ventas_totales

FROM ventas

GROUP BY categoria;
```

B

```
sql

SELECT producto, precio

FROM ventas

WHERE precio > 20;
```

C

```
sql

SELECT *

FROM ventas

ORDER BY fecha ASC;
```

- **Paso 7.** Exportar los resultados a S3:
- Si deseas guardar los resultados de una consulta en S3, usa el siguiente comando (los resultados estarán disponibles en la carpeta «resultados/» dentro del *bucket* S3):

```
sql

INSERT OVERWRITE DIRECTORY 's3://mi-bucket-hive-datos/resultados/'

ROW FORMAT DELIMITED

FIELDS TERMINATED BY ','

SELECT categoria, SUM(precio) AS ventas_totales

FROM ventas

GROUP BY categoria;
```

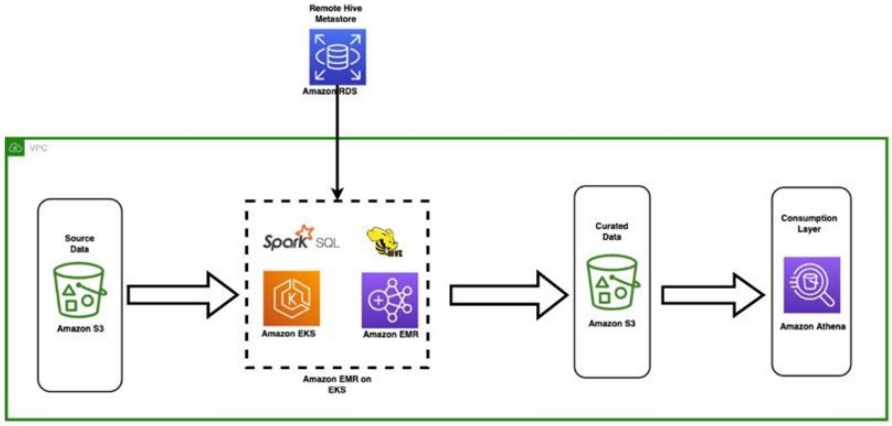

Figura 19. Ejemplo de diagrama de arquitectura de procesamiento AWS ESK y AWS EMR. Fuente: Maindola y Yang, 2023.

Analítica con Apache Flink en AWS EMR: guía paso a paso

Apache Flink es un motor de procesamiento de datos en tiempo real y por lotes altamente eficiente. Esta guía paso a paso te mostrará cómo realizar una analítica básica utilizando Flink en AWS EMR.

- **Paso 1.** Preparación del entorno:
- Crear un *bucket* en S3. Inicia sesión en la consola de AWS.
-
  - Ve a S3 y selecciona «Crear Bucket».
  - Introduce un nombre único para tu *bucket,* por ejemplo: mi-bucket-flink-datos.
  - Selecciona una región cercana a tu ubicación.
  - Haz clic en «Crear Bucket».

- Subir un conjunto de datos. Crea un archivo de ejemplo transacciones.csv con el siguiente contenido (expuesto después de los pasos):
  - Accede al *bucket* creado.
  - Haz clic en «Subir» y selecciona el archivo transacciones.csv.
  - Copia la URL del archivo: s3://mi-bucket-flink-datos/transacciones.csv.

```
csv

id,usuario,importe,categoria,timestamp

1,Alice,120,Compras,2024-01-01T10:15:30

2,Bob,80,Alimentos,2024-01-01T11:00:45

3,Charlie,200,Compras,2024-01-01T12:20:10

4,Alice,50,Transporte,2024-01-01T13:30:00

5,Bob,40,Transporte,2024-01-01T14:00:00
```

- **Paso 2.** Crear un clúster EMR con Flink:
- Ve a la consola de AWS EMR y selecciona «Crear Clúster».
- Configura el clúster de la siguiente manera:
  - Nombre del clúster: AnaliticaFlinkEMR.
  - Lanzar en una red VPC: selecciona una red VPC disponible.
  - Aplicaciones y *frameworks:* asegúrate de seleccionar Apache Flink.
  - Tipo de instancia. Nodo maestro: m5.xlarge. Nodos núcleo: dos instancias de m5.xlarge.
  - Almacenamiento: usa la configuración predeterminada o personalízala.
  - Permisos de IAM: asegúrate de que el rol tenga permisos para S3, EMR y Flink.
- Haz clic en «Crear Clúster».
- **Paso 3**. Configuración del entorno:
- Conectar al Clúster:
  - Una vez que el clúster esté en estado «En ejecución», selecciona el clúster y haz clic en «Conectar».
  - Usa la opción SSH para conectarte al nodo maestro desde tu terminal.

- Configurar Flink:
  - Navega a la carpeta donde se encuentra Flink (A).
  - Verifica que Flink esté instalado ejecutando (B):

A

```
bash

cd /usr/lib/flink/bin
```

B

```
bash

./flink -v
```

- **Paso 4.** Crear una aplicación de analítica con Flink:
- Preparar el código de la aplicación. Crea un archivo Java o Python (dependiendo del lenguaje que prefieras). A continuación, se muestra un ejemplo en Python usando PyFlink para procesar las transacciones:

Archivo: **analitica_flink.py**

```
python

from pyflink.datastream import StreamExecutionEnvironment

from pyflink.table import StreamTableEnvironment

from pyflink.table.descriptors import FileSystem, Schema, Csv

# Crear el entorno de ejecución

env = StreamExecutionEnvironment.get_execution_environment()

table_env = StreamTableEnvironment.create(env)

# Configurar la fuente de datos desde S3

input_path = "s3://mi-bucket-flink-datos/transacciones.csv"

output_path = "s3://mi-bucket-flink-datos/resultados/"

table_env.connect(FileSystem().path(input_path)) \

    .with_format(Csv().field_delimiter(',').derive_schema()) \

    .with_schema(Schema()

                 .field("id", "INT")

                 .field("usuario", "STRING")

                 .field("importe", "FLOAT")

                 .field("categoria", "STRING")

                 .field("timestamp", "STRING")) \

    .create_temporary_table("Transacciones")
```

```
# Definir la consulta para calcular el total por categoría

resultado = table_env.sql_query("""

    SELECT categoria, SUM(importe) AS total_importe

    FROM Transacciones

    GROUP BY categoria

""")

# Guardar los resultados en S3

table_env.connect(FileSystem().path(output_path)) \

    .with_format(Csv().field_delimiter(',').derive_schema()) \

    .with_schema(Schema()

                 .field("categoria", "STRING")

                 .field("total_importe", "FLOAT")) \

    .create_temporary_table("Resultados")

resultado.execute_insert("Resultados")
```

- **Paso 5.** Ejecutar la aplicación:
- Carga el archivo analitica_flink.py al nodo maestro del clúster (A).
- En el nodo maestro, ejecuta el script (B).

A

```
bash
scp analitica_flink.py hadoop@<master-node-ip>:/home/hadoop/
```

B

```
bash
/usr/lib/flink/bin/flink run -py /home/hadoop/analitica_flink.py
```

- **Paso 6.** Verificar los resultados:
- Ve a la consola de S3 y accede a la carpeta «resultados/» en tu *bucket.*
- Descarga el archivo generado y verifica que contiene los resultados de la consulta:

```
csv
categoria,total_importe
Compras,320.0
Alimentos,80.0
Transporte,90.0
```

Figura 20. Ejemplo de diagrama de flujo de trabajo de procesamiento con AWS Flink. Fuente: Sakthivel, 2021.

Monitoreo de los trabajos

Una vez que has iniciado el procesamiento de los datos, es importante monitorear el progreso de los trabajos. AWS EMR proporciona herramientas para esto:

- EMR *console:*
  - En la consola de EMR, puedes ver los detalles del trabajo, como su estado (en progreso, completado o fallido) y el tiempo que ha tardado.
- CloudWatch:
  - AWS CloudWatch te permite monitorear métricas de rendimiento en tiempo real, como el uso de CPU y la memoria de tus nodos.
- YARN *resource manager:*
  - Si estás utilizando Hadoop, puedes acceder al *resource manager* de YARN, para obtener más detalles sobre cómo se están distribuyendo los recursos en el clúster.

Casos de uso en AWS EMR

A continuación, exploraremos algunos de los casos de uso más destacados, organizados por categorías, con explicaciones claras y ejemplos prácticos.

Análisis de datos a gran escala

- Descripción:

AWS EMR permite analizar grandes conjuntos de datos utilizando herramientas como Apache Spark, Hadoop y Hive. Esto es ideal para empresas que necesitan extraer información significativa de datos no estructurados o semiestructurados.

- Ejemplo:
- Contexto: una empresa de comercio electrónico desea analizar el comportamiento de compra de sus usuarios durante el Black Friday.
- Solución:
  - Los datos de clics, compras y visitas se almacenan en S3.
  - Se ejecutan trabajos de Spark en EMR para identificar patrones de compra, como productos más populares o tiempos pico de compras.
  - Los resultados se exportan a un panel de BI para tomar decisiones estratégicas.

Procesamiento en tiempo real

- Descripción:

Aunque EMR está diseñado principalmente para procesamiento por lotes, puede integrarse con herramientas como Apache Flink y Kafka para manejar flujos de datos en tiempo real.

- Ejemplo:
- Contexto: una aplicación de IoT genera datos en tiempo real desde sensores industriales.
- Solución:
  - Kafka se utiliza para ingerir datos en tiempo real.
  - Flink, ejecutándose en EMR, procesa los datos en tiempo real para detectar anomalías, como fallos en las máquinas.
  - Los resultados se envían a un sistema de notificaciones en tiempo real para alertar al equipo de mantenimiento.

Machine Learning **a gran escala**

- Descripción:

AWS EMR puede integrarse con bibliotecas de *machine learning* como MLlib de Spark, TensorFlow o Scikit-learn para construir y entrenar modelos predictivos.

- Ejemplo:

- Contexto: un banco quiere predecir qué clientes son más propensos a abandonar sus servicios.

- Solución:

  - Los datos históricos de clientes se procesan en EMR con Spark MLlib.
  - Se entrena un modelo de clasificación utilizando el algoritmo de regresión logística.
  - El modelo se utiliza para identificar clientes en riesgo de abandono, permitiendo campañas de retención específicas.

Transformaciones ETL *(extract, transform, load)*

- Descripción:

EMR facilita la transformación de datos brutos en datos preparados para análisis, integrándose con S3, Redshift o RDS.

- Ejemplo:

- Contexto: una empresa de *marketing* digital almacena datos de campañas en diferentes formatos en S3.

- Solución:

  - Los datos se transforman en EMR utilizando Hive para consolidar y limpiar información.

  - Los datos transformados se cargan en Hive o una BBDD para su análisis mediante SQL.

  - Esto permite generar informes detallados sobre el rendimiento de las campañas.

Genómica y análisis científico

- Descripción:

EMR es ideal para análisis intensivo de datos en el ámbito científico, como la secuenciación de ADN o el análisis de datos climáticos.

- Ejemplo:

- Contexto: un laboratorio necesita procesar datos genómicos para identificar variantes genéticas asociadas a una enfermedad.

- Solución:

  - Los datos de secuenciación se almacenan en S3.
  - Se utiliza Hadoop en EMR para alinear las secuencias genómicas y realizar cálculos masivos.
  - Los resultados ayudan a los investigadores a identificar genes relevantes para tratamientos específicos.