- KsqlDB: es un motor de procesamiento de *streaming* SQL que permite a los desarrolladores realizar consultas en tiempo real sobre los datos de Kafka.

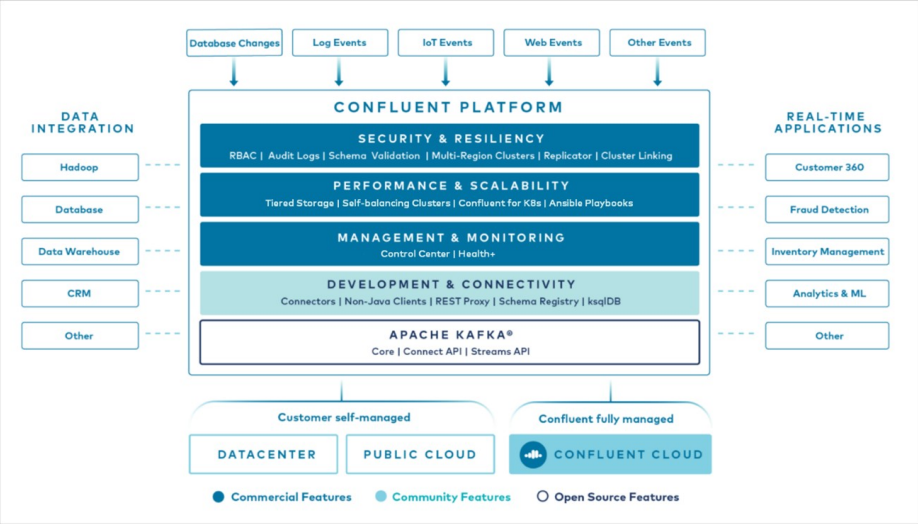

Figura 11. Diagrama de arquitectura de la plataforma Confluent. Fuente: Analytics, s. f.

Puedes encontrar la documentación de desarrollo de Confluent en el siguiente enlace: https://hevodata.com/learn/install-kafka-on-windows/

Casos de uso de Confluent

- **Análisis de datos en tiempo real:** confluent se utiliza para analizar datos de IoT, redes sociales, *logs* de aplicaciones y otros sistemas en tiempo real.
- **Sistemas de recomendación:** se utiliza para construir sistemas de recomendación personalizados en tiempo real.
- **Microservicios:** se utiliza para desacoplar microservicios y permitir una comunicación asíncrona entre ellos.
- **Integración de datos:** se utiliza para integrar datos de diferentes fuentes en una plataforma unificada.

Ventajas de usar Confluent sobre Apache Kafka puro

- **Facilidad de uso:** Confluent simplifica la gestión y la administración de Kafka.
- **Características adicionales:** ofrece características como el registro de esquemas, KsqlDB y herramientas de monitoreo.
- **Soporte empresarial:** proporciona soporte empresarial y actualizaciones regulares.

AWS MSK: Kafka en la nube de Amazon

Introducción

En las secciones anteriores, hemos explorado los fundamentos de Apache Kafka y cómo Confluent ofrece una plataforma empresarial para gestionar clústeres de Kafka. Ahora, nos centraremos en AWS MSK, un servicio totalmente gestionado de Amazon Web Services que simplifica aún más el despliegue y la gestión de clústeres de Kafka en la nube.

Sitio oficial de Amazon MSK: https://aws.amazon.com/es/msk/

¿Qué es AWS MSK?

AWS MSK es un servicio de *streaming* de datos totalmente gestionado, que facilita a los desarrolladores y administradores de DevOps la ejecución de aplicaciones de Apache Kafka en AWS, sin necesidad de administrar la infraestructura subyacente. AWS MSK se encarga de todas las tareas operativas, como la instalación, la configuración, el escalado y el mantenimiento de los clústeres de Kafka.

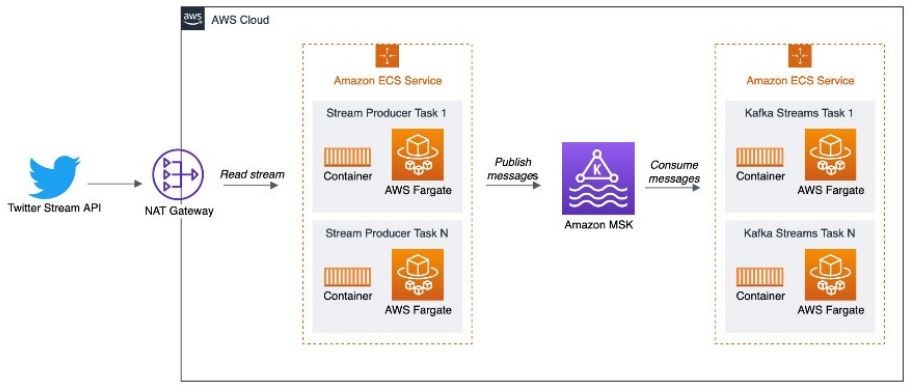

Figura 12. Diagrama de funcionamiento AWS MSK. Fuente: Grygoryan y Peyer, 2021.

¿Por qué usar AWS MSK?

- **Simplicidad:** AWS MSK elimina la complejidad de gestionar la infraestructura de Kafka, permitiendo a los desarrolladores concentrarse en sus aplicaciones.
- **Escalabilidad:** permite escalar los clústeres de Kafka de forma automática para adaptarse a las cargas de trabajo cambiantes.
- **Alta disponibilidad:** garantiza una alta disponibilidad de los datos mediante la replicación y la tolerancia a fallos.
- **Seguridad:** ofrece una amplia gama de características de seguridad, como la autenticación, la autorización y el cifrado de datos en reposo y en tránsito.
- **Integración con otros servicios de AWS:** se integra fácilmente con otros servicios de AWS, como Amazon Kinesis, Amazon EMR y Amazon Redshift.

Características clave de AWS MSK

- **Clústeres totalmente gestionados:** AWS MSK se encarga de la creación, configuración y mantenimiento de los clústeres de Kafka.
- **Escalado automático:** permite escalar los clústeres de forma automática para adaptarse a las cargas de trabajo cambiantes.
- **Almacenamiento persistente:** los datos se almacenan de forma persistente en Amazon S3, lo que garantiza la durabilidad de los datos.
- **Seguridad:** ofrece una amplia gama de características de seguridad, como la autenticación IAM, el cifrado de datos en reposo y en tránsito y las redes privadas.
- **Integración con AWS:** se integra fácilmente con otros servicios de AWS, como Amazon Kinesis Data Analytics para el procesamiento de *streaming* de datos.

La guía de desarrollo con AWS MSK se encuentra en este enlace:
https://docs.aws.amazon.com/es_es/msk/latest/developerguide/what-is-msk.html

---

Casos de uso de AWS MSK

Podemos enumerar los siguientes:

- **Análisis de datos en tiempo real:** AWS MSK se utiliza para analizar datos de IoT, redes sociales, *logs* de aplicaciones y otros sistemas en tiempo real.
- **Sistemas de recomendación:** se utiliza para construir sistemas de recomendación personalizados en tiempo real.
- **Microservicios:** se utiliza para desacoplar microservicios y permitir una comunicación asíncrona entre ellos.
- **Integración de datos:** se utiliza para integrar datos de diferentes fuentes en una plataforma unificada.

Ventajas de usar AWS MSK sobre Confluent Cloud

Destacan las siguientes:

- **Integración profunda con AWS:** AWS MSK se integra de forma nativa con otros servicios de AWS, lo que facilita la creación de soluciones de datos en la nube.
- **Precios:** AWS MSK puede ofrecer precios más competitivos, en función de los requisitos específicos de cada cliente.
- **Soporte técnico de AWS:** los clientes de AWS MSK tienen acceso al soporte técnico de AWS.

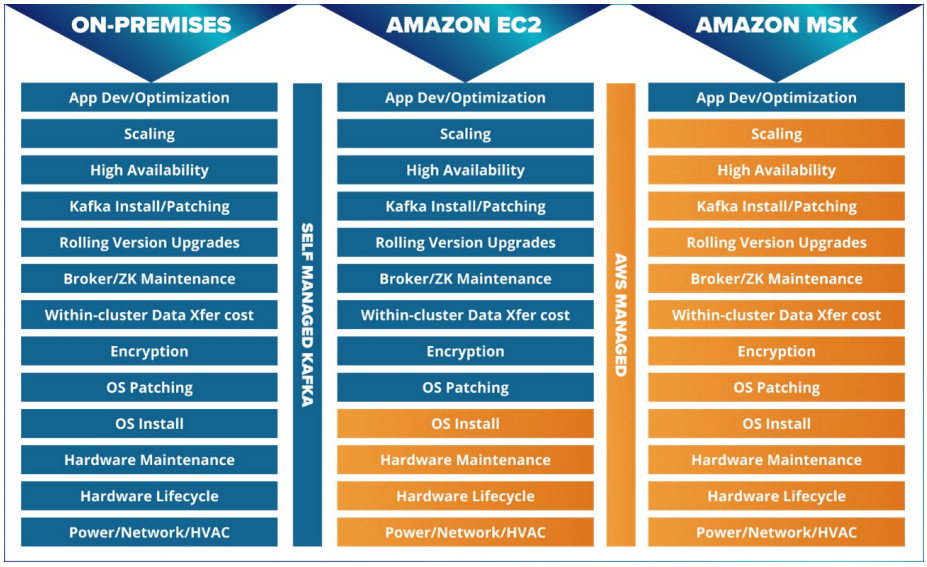

Figura 13. Características AWS MSK. Fuente: Soft serve, s. f.

Comparativa distribuciones Kafka

En la siguiente tabla resumen, puedes comprobar las diferencias que hay entre las diferentes distribuciones de Kafka que hemos visto.

Característica	Apache Kafka	Confluent	AWS MSK
Descripción	Plataforma de transmisión de eventos de código abierto.	Distribución empresarial de Apache Kafka con herramientas adicionales.	Servicio totalmente gestionado de Apache Kafka en AWS.
Nivel de abstracción	Bajo nivel, requiere configuración y gestión manual.	Medio nivel, ofrece herramientas para simplificar la gestión.	Alto nivel, totalmente gestionado.
Características clave	*Topics, partitions, brokers,* productores, consumidores.	Schema Registry, KsqlDB, Control Center, integración con otras tecnologías.	Clústeres totalmente gestionados, escalado automático, integración con otros servicios de AWS.
Implementación	Requiere instalación y configuración en infraestructura propia.	Se puede implementar en infraestructura propia o en la nube.	Se implementa en la nube de AWS.
Gestión	Requiere conocimientos técnicos para gestionar el clúster.	Ofrece herramientas para simplificar la gestión, requiere conocimientos técnicos.	Totalmente gestionado por AWS, no requiere conocimientos técnicos.
Escalabilidad	Requiere configuración manual para escalar.	Ofrece herramientas para escalar de forma automática.	Escalado automático basado en la demanda.
Seguridad	Requiere configuración manual de seguridad.	Ofrece características de seguridad adicionales: autenticación o autorización.	Ofrece características de seguridad de nivel empresarial, como IAM y cifrado.
Soporte	Comunidad de código abierto.	Soporte comercial de Confluent.	Soporte técnico de AWS.
Costo	Gratuito (*software* de código abierto).	Costo de licencia y servicios adicionales.	Pago por uso basado en el consumo de recursos.
Casos de uso	Amplia variedad de casos de uso, desde sistemas de recomendación hasta análisis de datos en tiempo real.	Similar a Apache Kafka, con un enfoque en empresas.	Ideal para empresas que desean ejecutar aplicaciones de Kafka en la nube de AWS.

Tabla 3. Comparativa entre las distribuciones de Kafka. Fuente: elaboración propia.

1.3. Procesamiento de datos escalable con EMR

Introducción a AWS EMR (Elastic MapReduce)

AWS EMR (Elastic MapReduce) es un servicio de procesamiento de datos en la nube proporcionado por Amazon Web Services (AWS). Está diseñado para ejecutar aplicaciones de procesamiento de *big data* de forma rápida, escalable y económica. Con EMR, puedes crear clústeres de procesamiento que utilizan herramientas como Hadoop, Spark, Hive, HBase, entre otros, para analizar grandes volúmenes de datos.

Sitio oficial: https://aws.amazon.com/es/emr/

¿Por qué utilizar AWS EMR?

En el mundo de la tecnología y de los datos, las empresas y las organizaciones necesitan procesar grandes cantidades de información para obtener *insights,* realizar análisis o entrenar modelos de inteligencia artificial. Tradicionalmente, esto requería grandes infraestructuras físicas y mucha inversión en *hardware.* AWS EMR resuelve este problema al permitir usar los recursos de computación y almacenamiento de AWS para crear clústeres de procesamiento de datos, sin necesidad de tener tus propios servidores.

Beneficios de usar AWS EMR

- **Escalabilidad.** Puedes ajustar la capacidad del clúster según tus necesidades, aumentando o reduciendo nodos de forma automática. Esto te permite manejar tanto pequeños como grandes volúmenes de datos, sin preocuparte por la infraestructura.

- **Costo eficiente.** Solo pagas por los recursos que utilizas. Puedes optar por instancias *spot* (instancias EC2 no utilizadas) que ofrecen descuentos significativos, lo que puede reducir el costo de los trabajos que realizar.

- **Facilidad de gestión.** AWS EMR se encarga de tareas complejas como la instalación, configuración y mantenimiento de las aplicaciones necesarias para el procesamiento de datos. Esto te permite centrarte más en el análisis y menos en la gestión de infraestructuras.

- **Integración conotros servicios de AWS.** AWS EMR se integra fácilmente con otros servicios de AWS como Amazon S3 (para almacenamiento de datos), Amazon DynamoDB (para bases de datos NoSQL), AWS Lambda (para ejecutar funciones sin servidor), entre otros. Esta integración facilita la creación de soluciones de *big data* completas y escalables.

Arquitectura de AWS EMR (Elastic MapReduce)

Introducción a la arquitectura de EMR

La arquitectura de AWS EMR (Elastic MapReduce) está diseñada para facilitar el procesamiento de grandes volúmenes de datos a través de una serie de componentes clave que trabajan juntos para ofrecer una solución de *big data* escalable y eficiente.

Conocer la arquitectura básica de EMR es fundamental para entender cómo se gestionan los datos, se organizan los nodos y se realiza el procesamiento paralelo de grandes volúmenes de información.

Componentes básicos de la arquitectura EMR

La arquitectura de AWS EMR está compuesta por varios componentes clave que permiten la creación y gestión de clústeres de procesamiento de datos:

- Máquinas de datos *(data nodes)*:
  - Estas son instancias de Amazon EC2 que albergan los datos y realizan el procesamiento de datos. Los nodos de datos son los responsables de ejecutar trabajos de Hadoop, Spark y otras herramientas de *big data* instaladas en el clúster.
  - En los nodos de datos se almacenan los datos temporales y se ejecutan las tareas de MapReduce, lo que permite el procesamiento paralelo y distribuido de los datos.

- Nodos de trabajo *(core nodes)*:
  - Estos nodos están optimizados para tareas de procesamiento de datos y ejecutan tareas de computación intensivas. Los nodos de trabajo son esenciales para las tareas de MapReduce y para el procesamiento de datos en paralelo, utilizando tecnologías como Apache Hadoop y Apache Spark.
  - Los nodos de trabajo suelen estar optimizados para un rendimiento máximo y proporcionan una capacidad de procesamiento distribuidamente eficiente.
- Nodos de núcleo *(master nodes)*:
  - Los nodos de núcleo controlan la distribución y ejecución de trabajos dentro del clúster. Son esenciales para la gestión de trabajos de Hadoop y Spark, así como para la configuración de clústeres.
  - Los nodos de núcleo incluyen el máster de Hadoop, que gestiona la distribución de datos a los nodos de datos y coordina las tareas de MapReduce. También se encuentran el máster de trabajo de Spark y el máster de YARN, que supervisan los recursos y la ejecución de tareas.
- Nodos de Zookeeper:
  - Algunos clústeres EMR utilizan nodos de Zookeeper para gestionar la coordinación entre los nodos del clúster. Zookeeper es un sistema de coordinación que permite la gestión de la configuración de datos en tiempo real y es especialmente útil para la gestión de servicios distribuidos y datos en tiempo real.
- Nodos de servicio *(service nodes)*:
  - Estos nodos se utilizan para manejar servicios adicionales, como el servidor de Apache HBase o el servidor de Presto. Los nodos de servicio están optimizados para manejar tareas específicas y permiten la ejecución de consultas en tiempo real o el almacenamiento de datos NoSQL.

Comunicaciones y red en la arquitectura EMR

- Comunicación interna:

  - Los nodos del clúster EMR se comunican a través de redes internas de AWS, utilizando una red privada virtual (VPN) o una conexión directa a Amazon Virtual Private Cloud (VPC). Esto asegura que los datos se transmitan de forma segura y eficiente entre los nodos de datos, de trabajo y de núcleo.

- Interacción con otros servicios de AWS:

  - AWS EMR se integra con otros servicios de AWS, como Amazon S3 (para almacenar datos de entrada y salida), Amazon DynamoDB (para gestionar bases de datos NoSQL) o Amazon Redshift (para análisis de datos complejos).

  - Esta integración permite que los datos sean procesados y analizados directamente en el clúster EMR mientras se almacenan o se consultan desde otros servicios de AWS.

Aquí te indicamos dónde encontrar la documentación oficial sobre la arquitectura de EMR:
https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-overview-arch.html

---

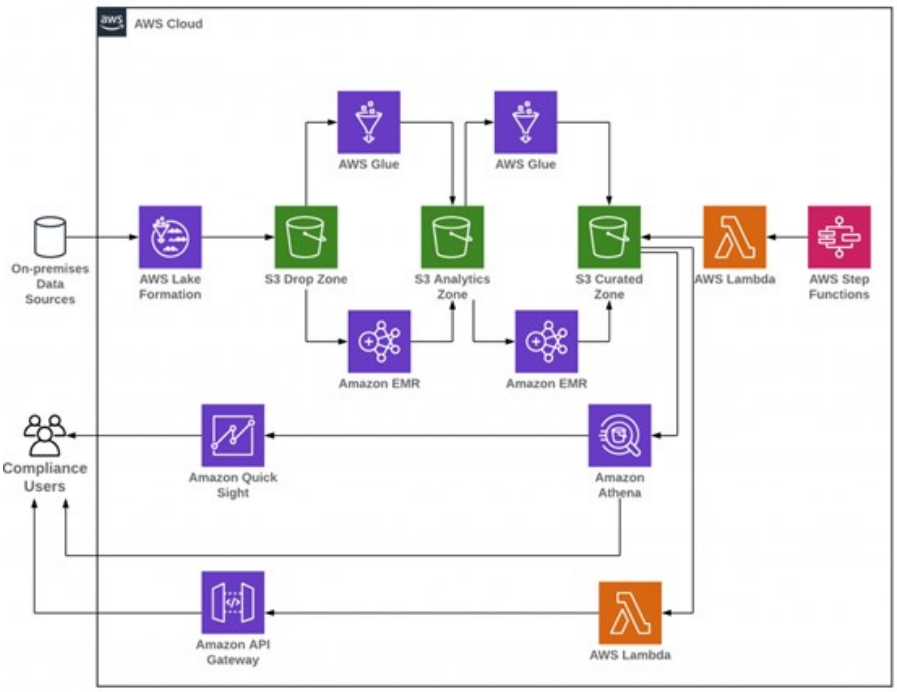

Figura 14. Diagrama de caso de uso procesamiento *streaming* usando AWS MSK. Fuente: Sodabathina, Aliabadi, Galleno y Hom, 2021.

Arquitectura de trabajo en EMR

Los principales elementos de una arquitectura de trabajo en EMR son los siguientes:

- **Manejo de tareas de computación:**

  - AWS EMR utiliza tecnologías como MapReduce, Apache Spark y Apache Hive para manejar el procesamiento paralelo de datos. Los nodos de trabajo y nodos de datos se coordinan para ejecutar estas tareas de procesamiento en paralelo, mejorando la velocidad y eficiencia del análisis de datos.

- **Uso de YARN *(yet another resource negotiator)*:**

  - YARN es un recurso central en la arquitectura de EMR que coordina los recursos de computación dentro del clúster, gestionando el almacenamiento y la distribución de los datos a través de los nodos de datos. YARN permite que diferentes *frameworks,* como Hadoop y Spark, compartan recursos y ejecuten tareas de computación de forma coordinada.

- Funcionalidades de datos en tiempo real:

  - AWS EMR permite la integración con tecnologías como Apache Flink para el procesamiento de flujos de datos en tiempo real. Esto significa que puedes ejecutar tareas de análisis de datos en tiempo real, mientras los datos se transmiten desde o hacia el almacenamiento en Amazon S3.

Flujo de trabajo en un clúster EMR

Los componentes principales en un flujo de trabajo de EMR serían los siguientes:

- Carga de datos a través de Amazon S3:

  - Los datos que se van a procesar en el clúster EMR se cargan previamente en Amazon S3. Esto permite que el clúster tenga acceso rápido a los datos necesarios para el procesamiento.

- Creación del clúster EMR:

  - Se configura el clúster con instancias de cómputo (nodos de trabajo) y servicios seleccionados (por ejemplo, Hadoop, Spark o Hive), según las necesidades del procesamiento.

- Procesamiento de datos:
  - Las tareas de procesamiento se envían a los nodos de trabajo para ejecutar trabajos de MapReduce en Hadoop, transformaciones en Spark o consultas SQL en Hive.
- Supervisión y control de la ejecución:
  - AWS CloudWatch supervisa continuamente el estado del clúster, permitiendo ajustar recursos y gestionar la ejecución de trabajos de manera eficiente.
- Almacenamiento de resultados:
  - Una vez completado el procesamiento, los resultados de los datos procesados se almacenan nuevamente en Amazon S3 para su acceso y uso posterior.

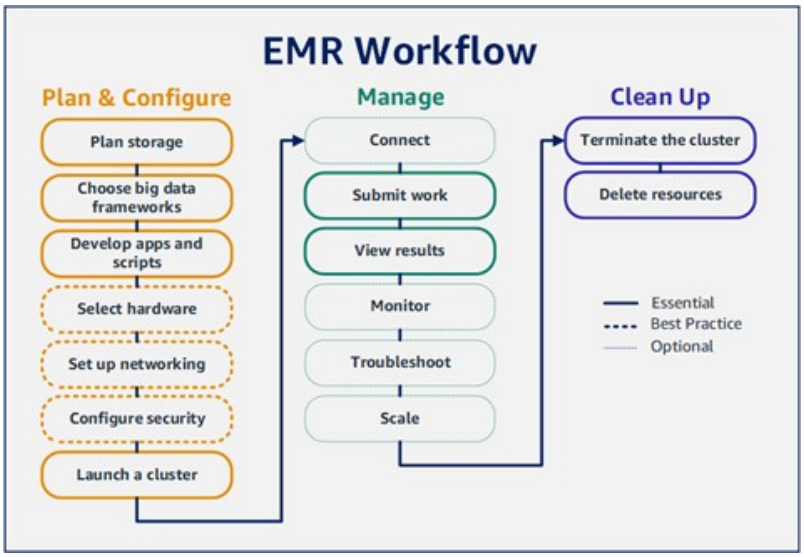

Figura 15. Flujo de trabajo AWS EMR. Fuente: Amazon, s. f.

En el siguiente enlace, puedes profundizar más sobre componentes de arquitectura y flujo de uso de EMR:
https://docs.aws.amazon.com/es_es/emr/latest/ReleaseGuide/emr-release-components.html

Componentes clave de AWS EMR

AWS EMR es una solución de procesamiento de *big data* que se ejecuta en la nube. Para comprender cómo funciona y cómo aprovechar todo su potencial, es esencial conocer los componentes clave que forman parte de su infraestructura. Estos componentes son herramientas que permiten procesar, almacenar y gestionar grandes volúmenes de datos de forma eficiente. En esta sección, exploraremos los componentes esenciales de EMR, su función y cómo trabajan juntos para procesar datos.

Guía de administración AWS EMR en el siguiente enlace:
https://docs.aws.amazon.com/es_es/emr/latest/ManagementGuide/emr-what-is-emr.html

Nodos del clúster EMR

Un clúster EMR está compuesto por varios nodos, que son instancias de máquinas virtuales en la nube (llamadas instancias EC2). Los nodos pueden ser de diferentes tipos según la tarea que se necesite realizar. Los nodos en un clúster EMR se dividen en los siguientes roles:

- **Nodos de máster.** Son responsables de la administración y supervisión del clúster. El nodo máster coordina las tareas y gestiona la distribución de trabajo entre los nodos de trabajo.
  - Función principal: gestionar el clúster, coordinar las tareas y supervisar su ejecución.
- **Nodos de trabajo.** Son los encargados de realizar el procesamiento real de los datos. Los nodos de trabajo ejecutan las tareas enviadas por el nodo máster, como cálculos, transformaciones o análisis.
  - Función principal: ejecutar las tareas de procesamiento de datos.
- **Nodos de comando** (opcional). Estos nodos proporcionan una interfaz para que el usuario pueda interactuar con el clúster. Puedes usar un nodo de comando para enviar comandos o consultar el estado del clúster.
  - Función principal: proporcionar acceso remoto y control al clúster EMR.

Herramientas y marcos de trabajo *(frameworks)*

AWS EMR está integrado con varias herramientas de procesamiento de datos que permiten ejecutar diferentes tipos de trabajos. A continuación, describimos las principales herramientas que puedes utilizar en EMR:

- **Apache Hadoop.** Hadoop es uno de los componentes fundamentales para el procesamiento de grandes volúmenes de datos. Utiliza un modelo de programación distribuido llamado MapReduce, para dividir las tareas en pequeñas unidades de trabajo, las cuales se distribuyen y procesan en paralelo en los nodos del clúster.

  - Función principal: procesar grandes volúmenes de datos utilizando el modelo MapReduce.

- **Apache Spark.** Spark es otro marco de trabajo que se utiliza para el procesamiento rápido de datos en memoria. A diferencia de Hadoop, Spark guarda los datos en la memoria RAM, lo que acelera enormemente el procesamiento, especialmente en tareas interactivas y en tiempo real.

  - Función principal: procesar datos en memoria y realizar análisis más rápidos que Hadoop, especialmente para tareas de *machine learning* y procesamiento en tiempo real.

- **Apache Hive.** Hive proporciona una interfaz SQL para consultar y analizar grandes conjuntos de datos almacenados en Hadoop. Permite a los usuarios escribir consultas de tipo SQL para acceder a los datos de forma más sencilla.

  - Función principal: realizar consultas SQL sobre grandes volúmenes de datos en Hadoop.

- **Apache Hbase.** HBase es una base de datos distribuida NoSQL que se utiliza para almacenar grandes cantidades de datos no estructurados y proporcionar acceso en tiempo real.
  - Función principal: almacenar y acceder rápidamente a grandes volúmenes de datos no estructurados.
- **Presto.** Presto es un motor de consultas distribuido que permite realizar consultas SQL a través de múltiples fuentes de datos, como Amazon S3, Hadoop o bases de datos relacionales. Se utiliza principalmente para el análisis interactivo de grandes volúmenes de datos.
  - Función principal: realizar consultas rápidas sobre grandes volúmenes de datos distribuidos.
- **Apache Flink.** Flink es un sistema de procesamiento de flujos *(streaming)* en tiempo real. Permite realizar análisis en tiempo real de datos en movimiento, como datos de sensores, clics de usuarios o registros de eventos.
  - Función principal: procesar y analizar flujos de datos en tiempo real.

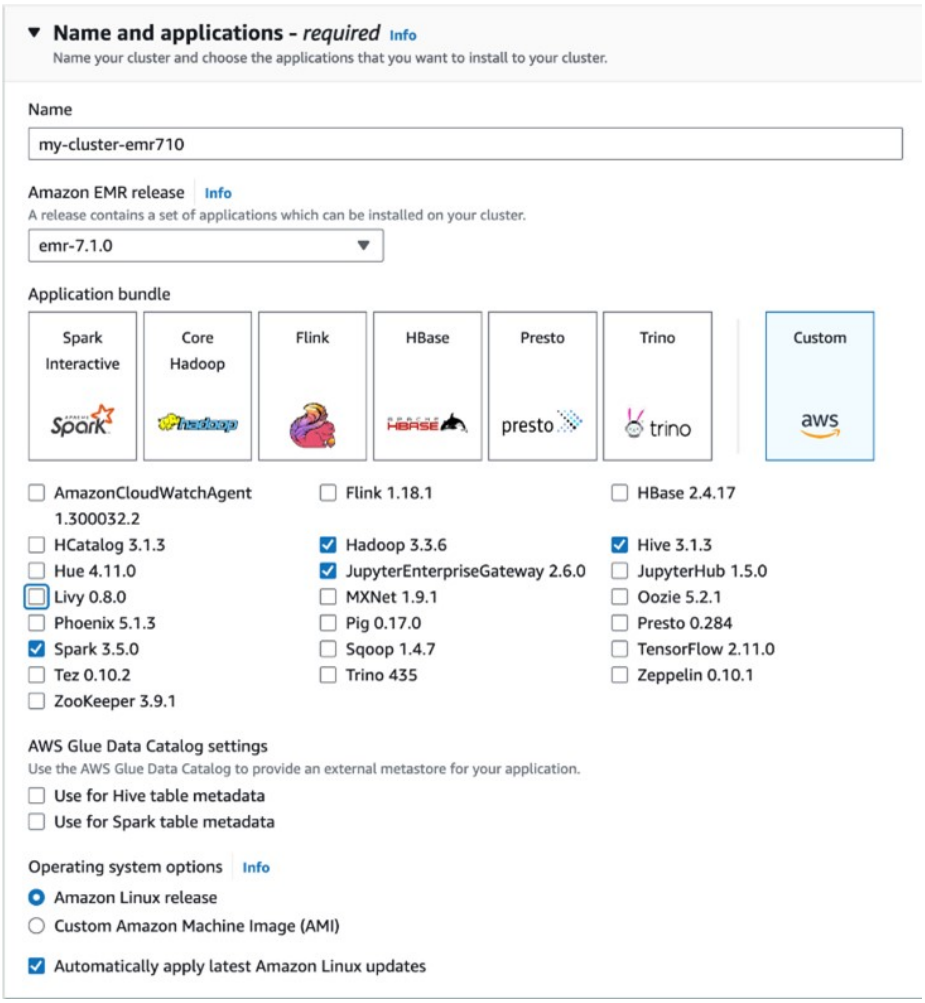

Figura 16. Configuración de un clúster en EMR. Fuente: Sodabathina, Aliabadi, Galleno y Hom, 2021.

Almacenamiento de datos en EMR

El almacenamiento de datos en AWS EMR es un aspecto clave, ya que las aplicaciones de *big data* requieren grandes cantidades de almacenamiento para trabajar con los datos. AWS ofrece varias soluciones de almacenamiento que se integran con EMR:

- **Amazon S3***(simple storage service)*. S3 es un servicio de almacenamiento en la nube que se utiliza para guardar los datos de entrada y salida del clúster EMR. Puedes cargar los datos en S3 antes de procesarlos y, una vez procesados, guardar los resultados allí. Es una solución escalable y económica, ideal para almacenar grandes volúmenes de datos no estructurados.

  - Función principal: almacenar datos de entrada y salida de los trabajos de procesamiento en EMR.

- **HDFS** (Hadoop Distributed File System). HDFS es el sistema de archivos distribuido de Hadoop. En un clúster EMR, HDFS distribuye los datos entre los diferentes nodos, permitiendo que se procesen de forma paralela y eficiente.

  - Función principal: almacenar grandes volúmenes de datos en clústeres distribuidos para su procesamiento.

Herramientas de monitoreo y gestión

Para gestionar y supervisar el rendimiento del clúster EMR, AWS proporciona varias herramientas que permiten controlar la salud y el progreso de los trabajos de procesamiento de datos:

- **Amazon CloudWatch.** CloudWatch es una herramienta que proporciona métricas en tiempo real sobre el uso de recursos, la salud de las instancias y el progreso de los trabajos de procesamiento. Con CloudWatch, puedes configurar alarmas para recibir notificaciones si algo no va como se esperaba.

  - Función principal: supervisar el rendimiento y el estado del clúster en EMR.

- **AWS CloudTrail.** CloudTrail es un servicio que permite rastrear y registrar todas las actividades realizadas en tu cuenta de AWS, incluyendo las interacciones con EMR. Es útil para auditorías y para resolver problemas.

  - Función principal: realizar un seguimiento de las actividades y cambios en los recursos de AWS.

Seguridad en AWS EMR

La seguridad es un aspecto crítico en el procesamiento de datos y AWS EMR proporciona varias capas de protección:

- **IAM (Identity and Access Management).** IAM permite gestionar los permisos y el acceso a los recursos dentro de AWS, como los clústeres EMR. Puedes crear roles y políticas para asegurar que solo los usuarios autorizados puedan acceder a los datos y ejecutar tareas.
  - Función principal: Controlar el acceso y la seguridad de los recursos en EMR.
- **Cifrado de datos.** AWS EMR permite cifrar los datos en tránsito y en reposo para garantizar que la información esté protegida. Puedes usar el cifrado de S3, HDFS o Amazon RDS, según tus necesidades de seguridad.
  - Función principal: proteger los datos con cifrado para mantener la confidencialidad y seguridad.

Configuración de un clúster EMR: guía paso a paso

Como hemos visto, AWS EMR es una poderosa plataforma que permite procesar grandes volúmenes de datos en la nube, utilizando herramientas como Apache Hadoop, Apache Spark y otras. Configurar un clúster EMR es el primer paso para empezar a trabajar con procesamiento de datos a gran escala.

A continuación, se ofrece una guía paso a paso para crear y configurar tu propio clúster EMR, de forma clara, sencilla y accesible para quienes no tienen experiencia previa.

Pasos para su configuración

- **Paso 1.** Crear una cuenta de AWS:

Antes de empezar a trabajar con EMR, necesitas una cuenta en AWS. Si aún no tienes una, sigue estos pasos:

- Visita el sitio web de AWS. Ve a AWS (https://aws.amazon.com/es/) y haz clic en «Create an AWS Account» (crear una cuenta de AWS).
- Proporciona tus datos. Completa los campos requeridos (nombre, dirección de correo, datos de facturación, etc.).
- Configura la autenticación de dos factores. Este paso es esencial para asegurar tu cuenta.
- Finaliza la creación. Una vez que hayas completado el proceso, tu cuenta estará lista para usar.
- **Paso 2.** Iniciar sesión en la consola de administración de AWS:

Una vez que tienes tu cuenta de AWS, inicia sesión en la consola de administración de AWS:

- Accede a la consola de AWS. Ve a AWS Management Console (https://aws.amazon.com/es/console/).
- Inicia sesión. Usa las credenciales de tu cuenta (nombre de usuario y contraseña).

- **Paso 3.** Crear un clúster EMR:

Con tu cuenta activa, ahora vamos a crear el clúster EMR. Para hacerlo, sigue estos pasos:

- Accede a la consola de EMR. En la barra de búsqueda de la consola de AWS, escribe «EMR» y selecciona «Amazon EMR» en los resultados.
- Crear un nuevo clúster. Haz clic en el botón «Create cluster» (crear clúster) que aparece en la pantalla de inicio de EMR.
- Configurar el clúster:
  - *Cluster Name* (nombre del clúster). Elige un nombre para tu clúster, por ejemplo: «MiPrimerClusterEMR».
  - *Software configuration* (configuración de *software*). Elige el paquete de *software* que deseas instalar. AWS EMR soporta diferentes marcos, tales como Apache Hadoop, Apache Spark, Hive, etc. Para este ejemplo, seleccionaremos Apache Hadoop y Apache Spark.
  - *Release* (versión). Deja la opción predeterminada (usualmente la más reciente).
  - *Log Files* (archivos de registro). Selecciona un *bucket* de Amazon S3 para almacenar los registros del clúster.

- Configuración de *hardware:*

  - *Master node* (nodo maestro). Este nodo gestiona el clúster y las tareas que se distribuyen entre los nodos de trabajo. AWS EMR te ofrece varias opciones, como instancias con más o menos recursos, dependiendo del volumen de datos que planeas procesar. Puedes elegir una instancia m5.xlarge para comenzar.

  - *Core nodes* (nodos principales). Selecciona las instancias que se utilizarán para procesar los datos. Si eres principiante, puedes empezar con solo dos nodos, cada uno con instancias de tipo m5.xlarge.

  - *Task nodes* (nodos de tarea, opcional). Si planeas hacer procesamiento intensivo de datos, puedes agregar nodos adicionales de tarea.

- Red y seguridad:

  - VPC *(virtual private cloud)*. AWS te ofrece una VPC predeterminada que se usa para configurar el aislamiento de redes. Puedes dejar esta opción como está o crear una VPC personalizada si necesitas configuraciones más avanzadas.

  - IAM *roles* (roles de IAM). Asigna un rol de IAM adecuado para que el clúster pueda interactuar con otros servicios de AWS (como S3, DynamoDB, etc.). Usa el rol predeterminado llamado «EMR_EC2_DefaultRole».

  - S3 *bucket* para *logs.* Asegúrate de tener un *bucket* de S3 donde se almacenarán los registros de tus trabajos en EMR.

- Lanzar el clúster. Después de configurar todo, haz clic en «Create cluster», para lanzar tu clúster EMR.

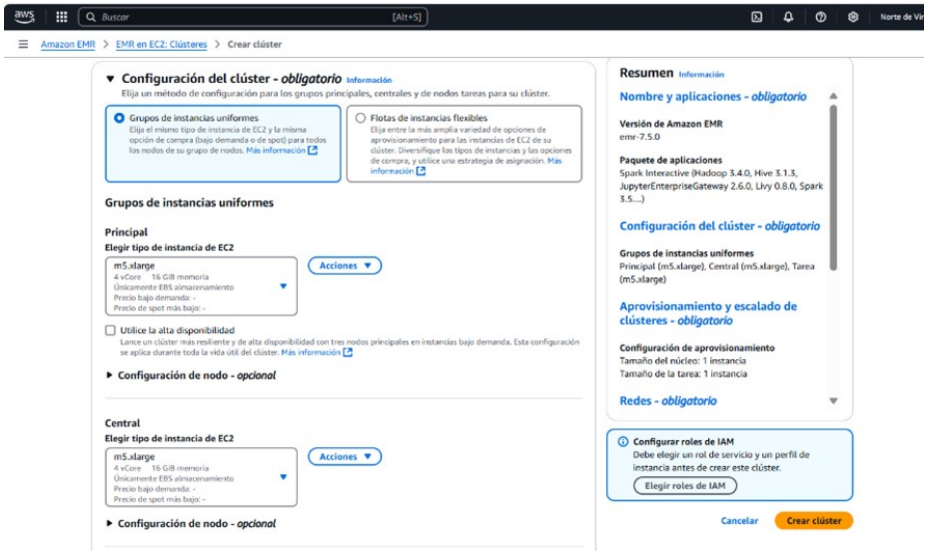

Figura 17. Configuración de un clúster EMR. Fuente: elaboración propia.

- **Paso 4.** Acceder al clúster EMR:

Una vez que tu clúster EMR esté en ejecución, podrás acceder a él y gestionar tus trabajos de procesamiento de datos.

- Verificar el estado del clúster. En la consola de EMR, puedes ver el estado de tu clúster. El clúster debe tener el estado *«waiting»* (esperando), lo que indica que está listo para procesar tareas.

- Acceder al nodo maestro:

  - Para interactuar con tu clúster, necesitarás acceder al nodo maestro, donde se ejecutan las tareas de administración y distribución.

  - Para ello, selecciona el nodo maestro desde la consola de EMR y haz clic en «Connect». Esto te proporcionará instrucciones detalladas sobre cómo conectarte al nodo a través de SSH *(secure shell)*.

- **Paso 5.** Cargar datos en el Clúster EMR:

Para empezar a procesar datos en tu clúster, necesitas cargarlos en S3. A continuación, explicamos cómo hacerlo:

- Subir datos a S3. Abre la consola de S3 en AWS, crea un *bucket* y sube los archivos de datos que deseas procesar en tu clúster EMR.

- Acceder a los datos en EMR. Puedes configurar tu clúster para acceder a esos datos en S3. Al usar herramientas como Hadoop o Spark, especificas la ruta del archivo en S3 para que el clúster pueda leerlo y procesarlo.

- **Paso 6.** Ejecutar tareas en el clúster EMR:

Una vez que tu clúster está en funcionamiento y tienes datos cargados, puedes ejecutar trabajos de procesamiento en él.

- Ejecutar una tarea de ejemplo:
  - Puedes ejecutar trabajos de ejemplo utilizando Spark o Hive, para procesar tus datos. Para ello, puedes usar el nodo maestro y ejecutar un *script* simple que invoque a las herramientas que instalaste (como Spark).
  - Por ejemplo, para ejecutar un trabajo con Spark, puedes escribir un *script* de Python que lea datos desde S3, realice un procesamiento y guarde los resultados de vuelta en S3.
- Ejemplo de código de Spark en Python (tras el paso 3).
- Monitorear el trabajo. Usa la consola de EMR para verificar el estado de tu trabajo en tiempo real. Puedes ver las métricas de los trabajos y revisar los registros para identificar cualquier posible problema.

```
python

from pyspark.sql import SparkSession

# Crear una sesión de Spark

spark = SparkSession.builder.appName("EjemploSparkEMR").getOrCreate()

# Leer datos desde un archivo CSV en S3

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

# Realizar una operación simple, como mostrar los primeros 5 registros

df.show(5)

# Guardar los resultados en S3

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