Curso de IA y Big Data

CPR Badajoz

Francisco Pajuelo Holguera

Sesión 4 y 5: Contenidos

- Introducción al Big Data
- AWS y Python
- Big Data aplicado: Hadoop, HDFS, Sqoop/Flume, Hive y Kafka
- Analítica de datos: Spark

Introducción al Big Data

Podemos decir que son conjuntos de datos de gran tamaño, complejidad y velocidad de crecimiento, que hacen difícil su captura, gestión y procesamiento a través de herramientas convencionales, como pueden ser las bases de datos relacionales.

Las cinco Vs del Big Data

- **Volumen**: podemos considerar que el volumen de datos que maneja esta técnica sería de muchos Terabytes.
- **Velocidad**: Los datos usados en Big Data se trabajan a mayor velocidad que los gestionados en bases de datos tradicionales.
- **Variedad**: Los macrodatos trabajan con fotografías, vídeos, audio, series de datos temporales, y muchos otros tipos de datos.
- **Veracidad**: Se trata de la integridad de los datos.
- **Valor**

Tipos de Big Data

Según su procedencia:

- Páginas web y blogs, todos aquellos datos que los usuarios generan al navegar por la Red.
- Redes sociales.
- Transacciones.
- Datos generados por la interacción entre sensores inteligentes en máquinas, también llamada comunicación machine-to-machine.
- Datos generados por la tecnología de reconocimiento biométrico.
- Datos generados por personas y organizaciones públicas y privadas a través de emails, mensajes, grabaciones de llamadas, estadísticas, historiales, etc.

Tipos de Big Data

Según su estructura:

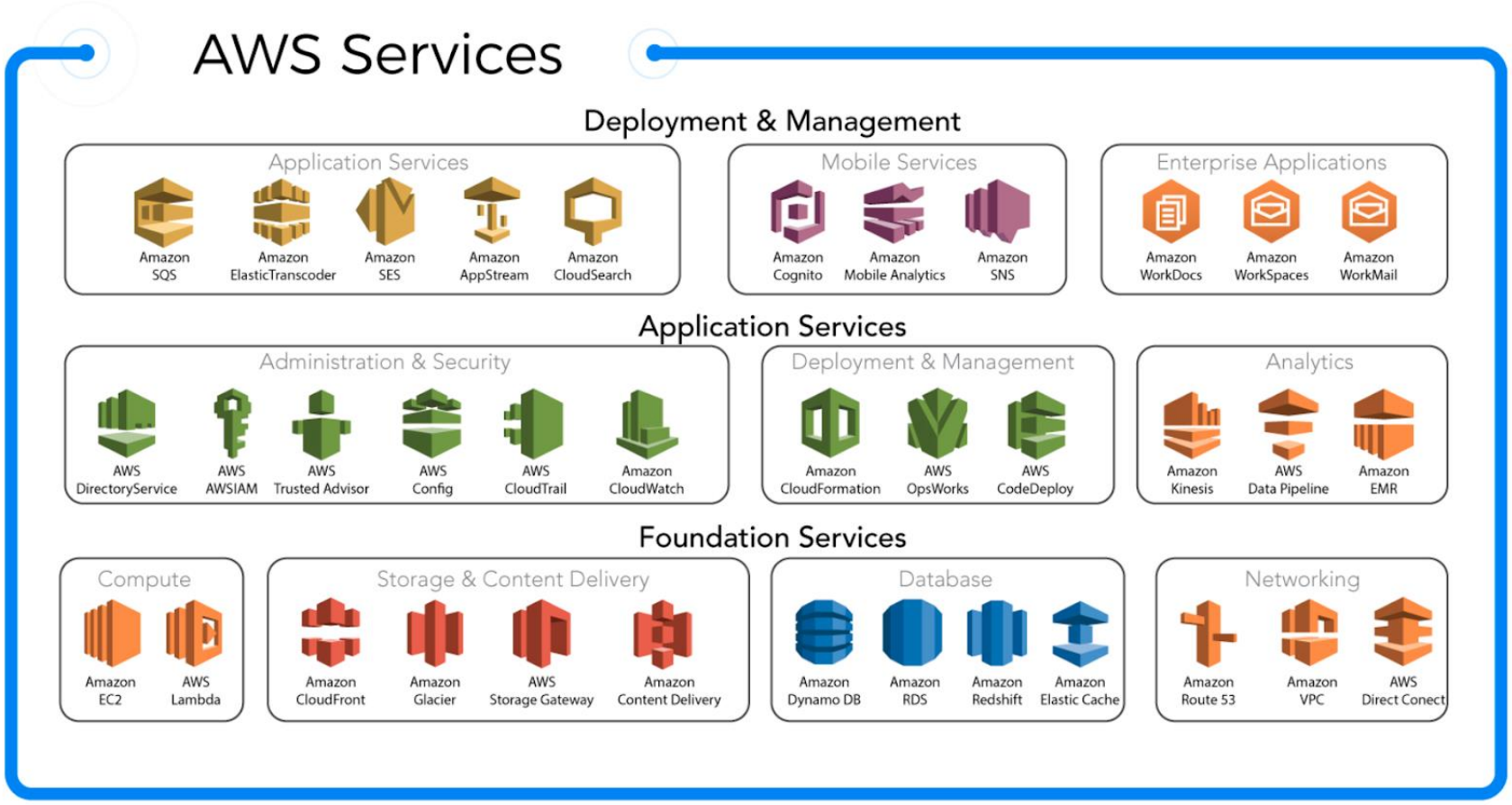
- Estructurados, datos con formato, tamaño y longitud definidas.
- Semiestructurados, son datos con una estructura flexible, como los que se usan en XML y HTML o JSON.
- No estructurados, aquellos datos que no tienen un formato específico, como los textos o los contenidos multimedia.

Elementos del Big Data

Junto con los datos, la tecnología Big Data necesita tres elementos fundamentales para garantizar que dispondrá de la capacidad suficiente para proporcionar los servicios:

- **Sistema de almacenamiento**
- **Sistema de procesamiento**
- **Sistema de comunicación**

AWS


AWS Services
Deployment & Management
Application Services
Amazon SQS
Amazon ElasticTranscoder
Amazon SES
Amazon AppStream
Amazon CloudSearch
Mobile Services
Amazon Cognito
Amazon Mobile Analytics
Amazon SNS
Enterprise Applications
Amazon WorkDocs
Amazon WorkSpaces
Amazon WorkMail
Application Services
Administration & Security
AWS DirectoryService
AWS AWSIAM
AWS Trusted Advisor
AWS Config
AWS CloudTrail
Amazon CloudWatch
Deployment & Management
Amazon CloudFormation
AWS OpsWorks
AWS CodeDeploy
Analytics
Amazon Kinesis
AWS Data Pipeline
Amazon EMR
Foundation Services
Compute
Amazon EC2
AWS Lambda
Storage & Content Delivery
Amazon CloudFront
Amazon Glacier
AWS Storage Gateway
Amazon Content Delivery
Database
Amazon Dynamo DB
Amazon RDS
Amazon Redshift
Amazon Elastic Cache
Networking
Amazon Route 53
Amazon VPC
AWS Direct Conect

AWS CLI

Instalación:

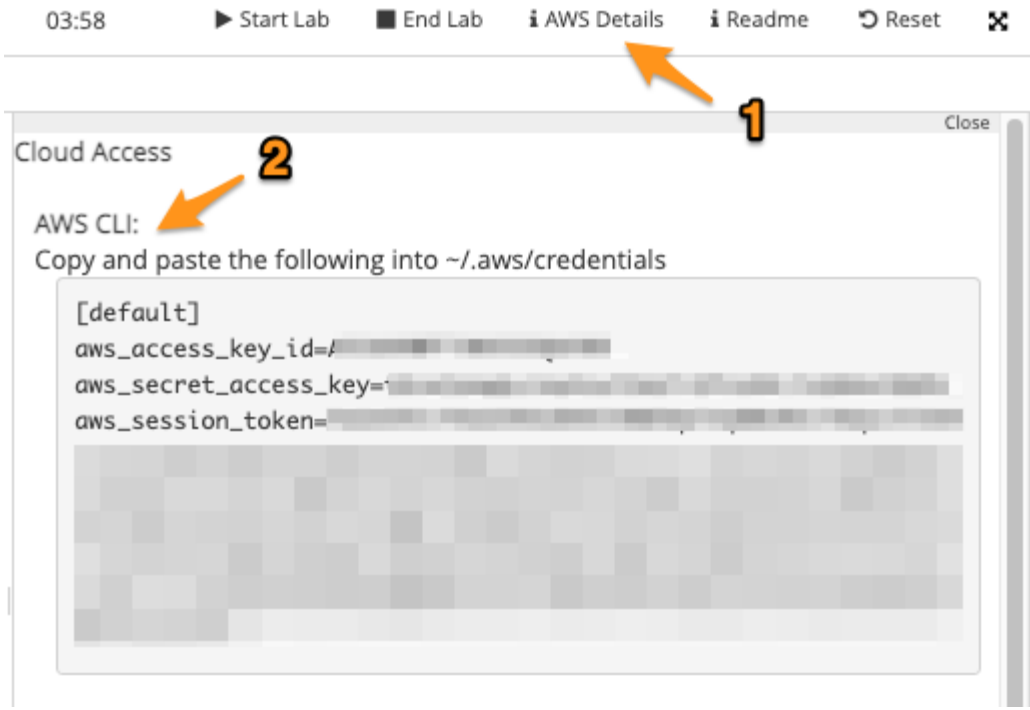
https://aws.amazon.com/es/cli/

Para comprobar si está instalada en Windows:

Abri cmd y ejecutar aws --version

AWS CLI

El siguiente paso será validarse en AWS. Para ello, desde nuestra consola del curso *Leaner Labs* , tras arrancar el laboratorio, pulsaremos (1) en la opción *AWS Details*, y posteriormente veremos los datos de acceso temporales al pulsar (2) en *Show* de la opción *AWS CLI*:



Esos datos los podemos pegar en el archivo ~/.aws/credentials

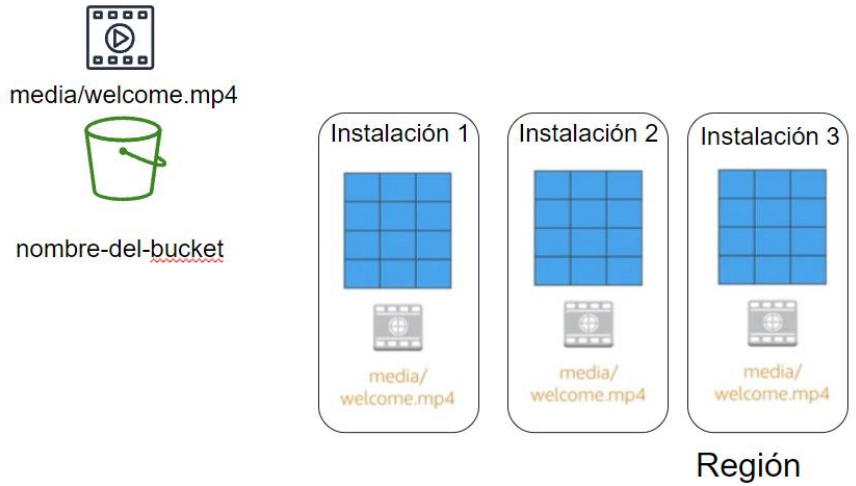
AWS S3

S3 es un servicio de almacenamiento persistente de objetos creado para almacenar y recuperar cualquier cantidad de datos desde cualquier lugar mediante una URL: sitios web y aplicaciones móviles, aplicaciones corporativas y datos de sensores o dispositivos de Internet de las cosas (IoT) y análisis de Big Data.

Los datos se almacenan como objetos dentro de recursos conocidos como **buckets**. Los objetos pueden ser prácticamente cualquier archivo de datos, como imágenes, videos o registros del servidor.

AWS S3

De forma predeterminada, en Amazon S3 los datos se almacenan de forma redundante en varias instalaciones y en diferentes dispositivos de cada instalación.



AWS S3

Los datos que almacenamos en S3 no están asociados a ningún servidor en particular (aunque los buckets se asocien a regiones, los archivos se dice que están almacenados de forma global), con lo que no necesitamos administrar ningún tipo de servidor.

Cada objeto que almacenamos en S3 tiene:

- una clave: nombre que se le asigna al objeto y que se utiliza para recuperarlo.
- un id de versión: podemos mantener un histórico de cambios mediante el versionado de los archivos, de manera que cuando actualicemos un objeto, en vez de sustituirlo, se crea una nuevo versión manteniendo un histórico. un valor: contenido real que se almacena.
- metadatos: pares clave-valor, que podemos definir nosotros como usuarios.
- subrecursos: que utiliza AWS para almacenar información adicional.

AWS RDS

AWS ofrece Amazon RDS como servicio administrado que configura y opera una base de datos relacional en la nube, de manera que como desarrolladores sólo hemos de enfocar nuestros esfuerzos en los datos y optimizar nuestras aplicaciones..

AWS RDS

Una instancia de base de datos es un entorno de base de datos aislado que puede contener varias bases de datos creadas por el usuario. Se puede acceder a él utilizando las mismas herramientas y aplicaciones que utiliza con una instancia de base de datos independiente.

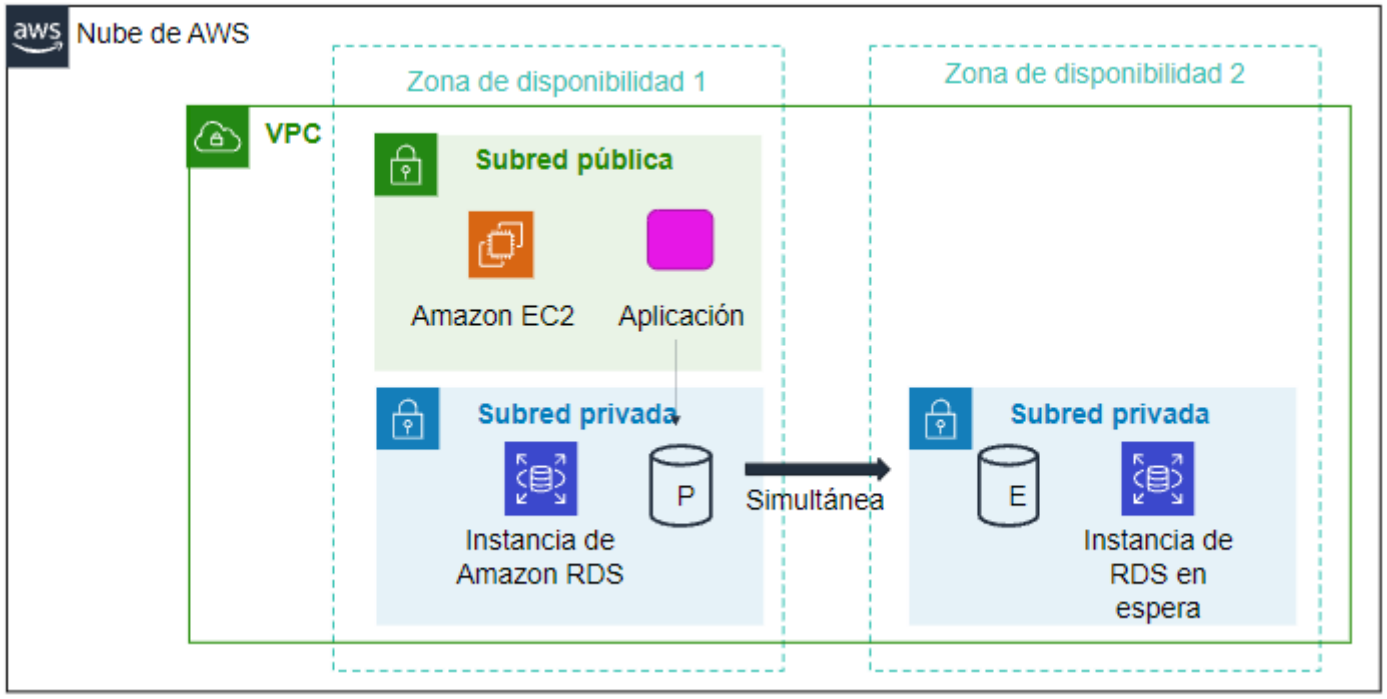
Cuando vamos a crear una instancia de base de datos, primero hemos de indicar qué motor de base de datos ejecutar. Actualmente, RDS admite seis motores de bases de datos:

- **MySQL**, compatible con las versiones 5.6, 5.7 y 8.0.
- **Amazon Aurora**
- **Microsoft SQL Server**, que permite implementar varias versiones de SQL Server (2012, 2014, 2016, 2017 y 2019), incluidas las Express, Web, Standard y Enterprise.
- **PostgreSQL**, compatible con las versiones 9.6, 10, 11 y 12.
- **MariaDB**, compatible con las versiones 10.2, 10.3, 10.4 y 10.5
- y **Oracle**, compatible con Oracle 12 y Oracle 19, con dos modelos de licencia diferentes: Licencia incluida y Bring-Your-Own-License (BYOL).

AWS RDS

Amazon RDS

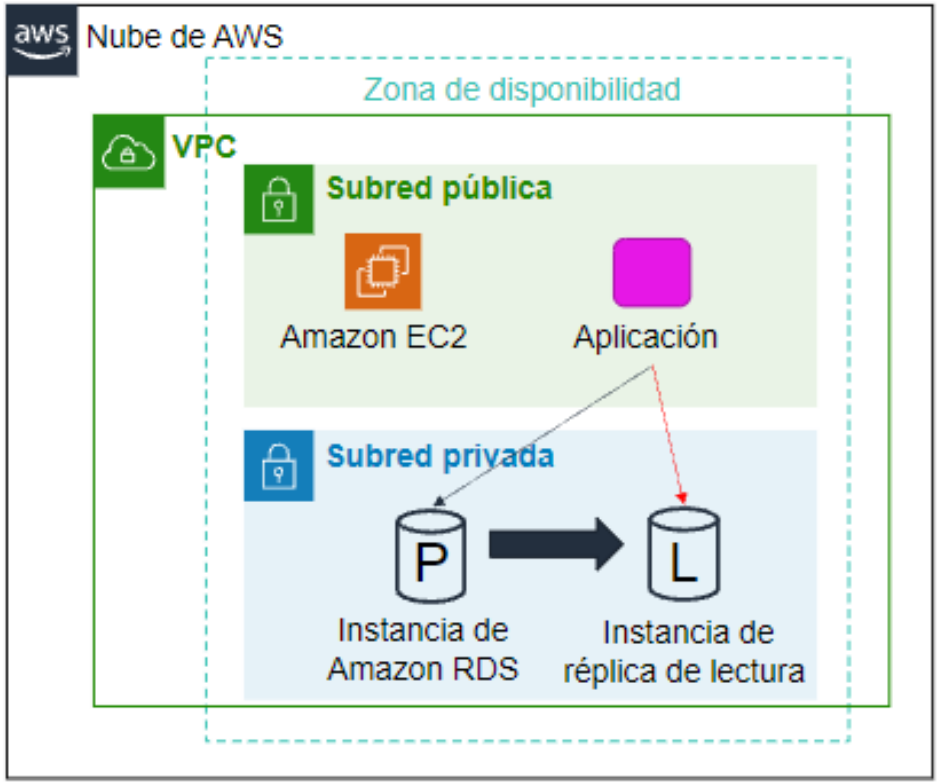
Una de las características más importantes de RDS es la capacidad de configurar la instancia de base de datos para una alta disponibilidad con una implementación Multi-AZ. Al hacerlo, se genera de manera automática una copia en espera de la instancia de base de datos en otra zona de disponibilidad dentro de la misma VPC. Después de propagar la copia de la base de datos, las transacciones se replican de forma **síncrona** a la copia en espera.



AWS RDS

RDS también admite la creación de réplicas de lectura para MySQL, MariaDB, PostgreSQLy Amazon Aurora.

Las actualizaciones que se realizan en la instancia principal se copian de manera **asíncrona** en la instancia de réplica de lectura, de manera que direccionando las consultas a esta nueva réplica reduciremos la carga de la instancia principal.



AWS DynamoDB

DynamoDB es un servicio administrado de base de datos NoSQL clave-valor y documental, rápido y flexible para todas las aplicaciones que requieren una latencia uniforme de un solo dígito de milisegundos a cualquier escala y una capacidad de almacenamiento prácticamente ilimitado.

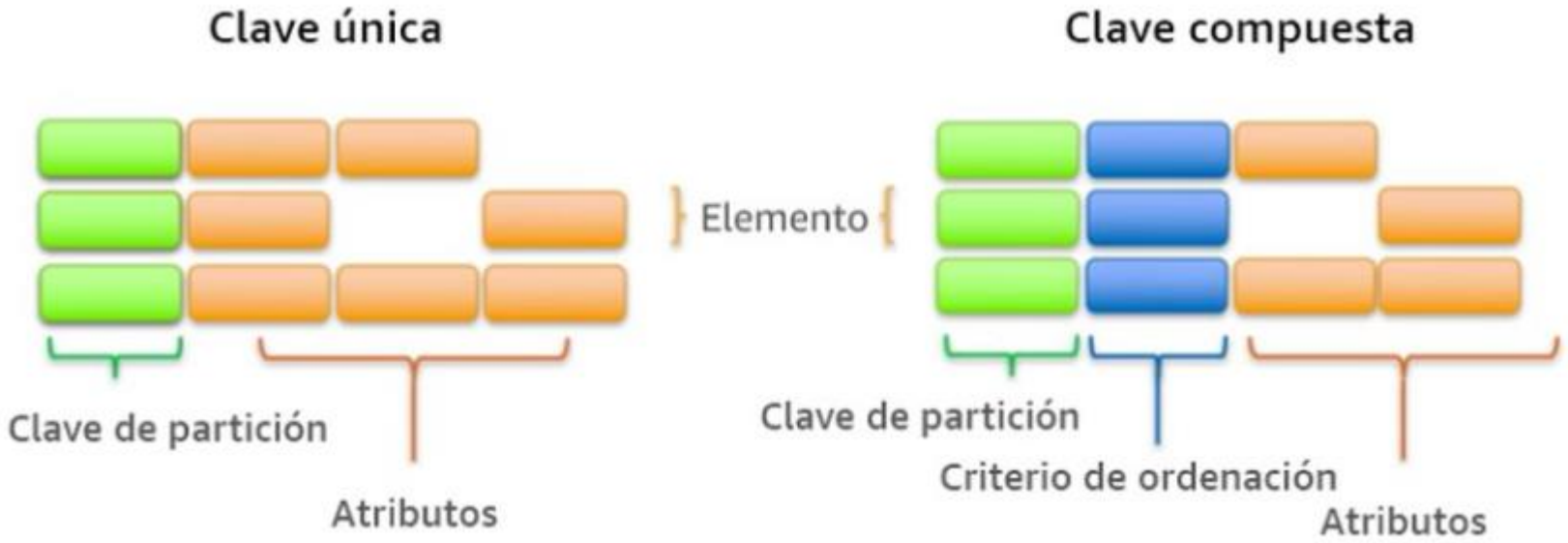
Así pues, es un almacén de claves/valor (similar a Redis y MongoDB a la vez), flexible y sin estructura fija (los elementos pueden tener atributos diferentes), diseñado para garantizar un determinado rendimiento así como una determinada disponibilidad para cada tabla (en NoSQL suele haber pocas tablas), es decir, se definen elementos por tabla y se paga según lo exigido en cada una.

AWS DynamoDB

Amazon DynamoDB

DynamoDB soporta dos tipos de claves principales:

- La clave de partición es una clave principal simple.
- La clave de partición y de ordenamiento, también conocidas como clave principal compuesta, ya que está formada por dos atributos.

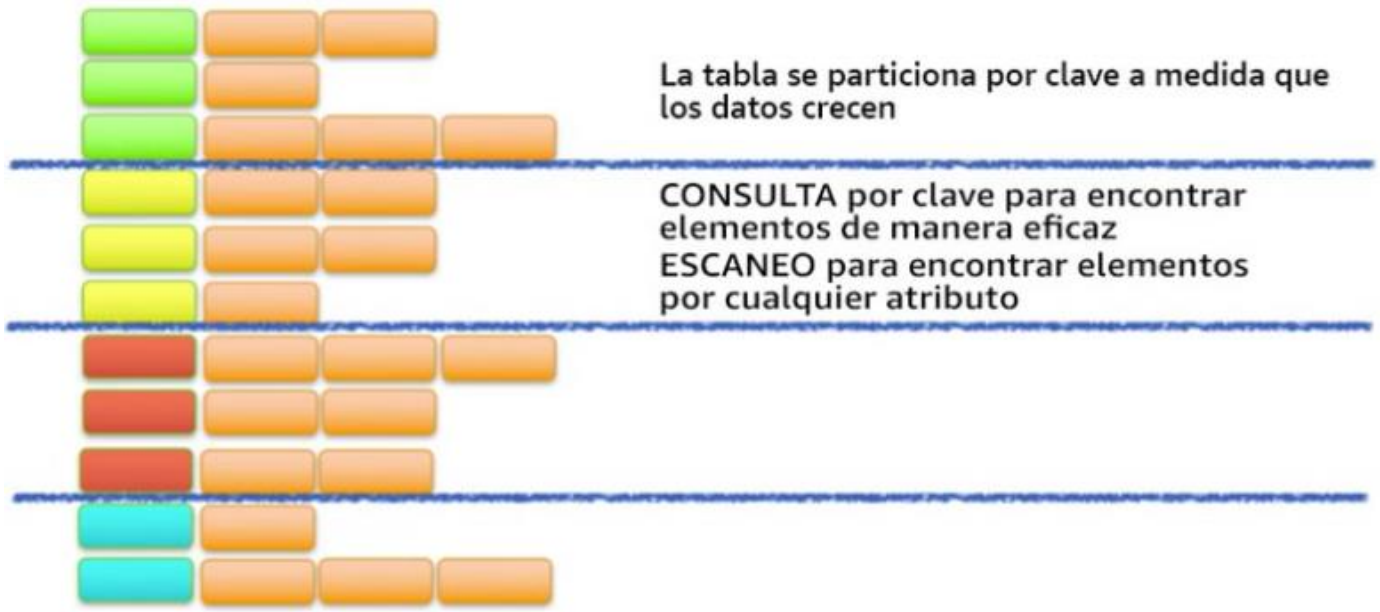


AWS DynamoDB

Amazon DynamoDB

A medida que aumenta el volumen de datos, la clave principal particiona e indexa los datos de la tabla.

Podemos recuperar los datos de una tabla de DynamoDB de dos formas distintas, bien por la clave y hacer una consulta directa, o utilizar un escaneo de todos los elementos en busca de aquello que coincida con el parámetro de búsqueda.



Big Data aplicado: Hadoop

Si Big Data es la filosofía de trabajo para grandes volúmenes de datos, Apache Hadoop es la tecnología catalizadora. Hadoop puede escalar hasta miles de ordenadores creando un clúster con un almacenamiento del orden de petabytes de información.

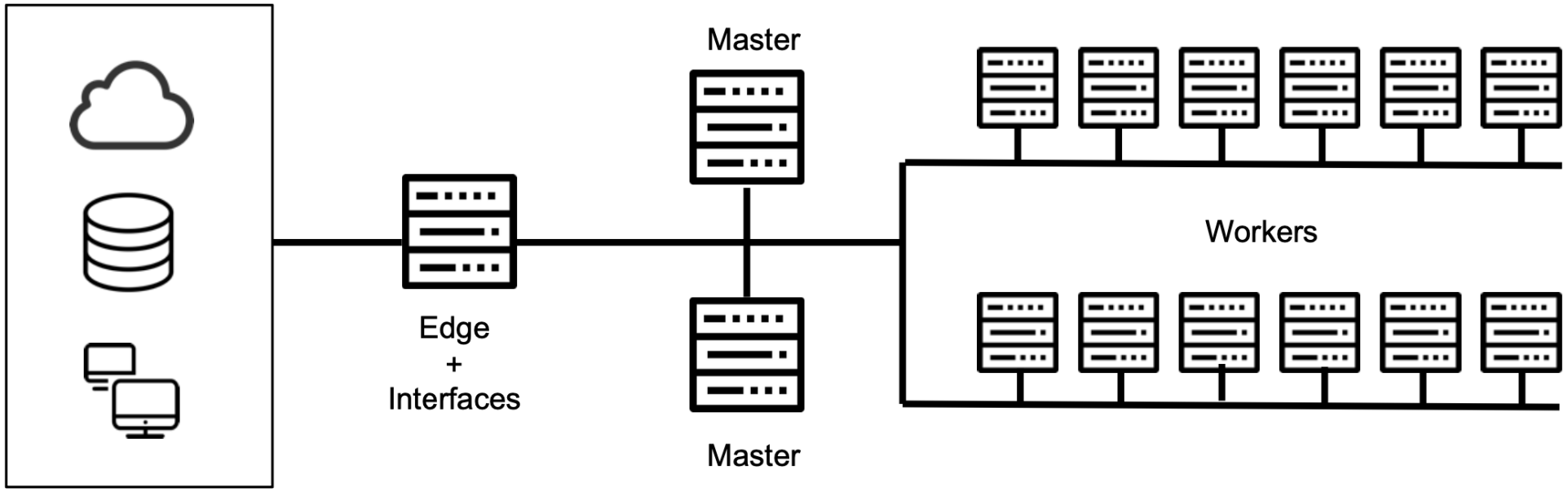
Más que un producto, es un proyecto open source que aglutina una serie de herramientas para el procesamiento distribuido de grandes conjuntos de datos a través de clústers de ordenadores utilizando modelos de programación sencillos.

Big Data aplicado: Hadoop

Sus características son:

- Confiable: crea múltiples copias de los datos de manera automática y, en caso de fallo, vuelve a desplegar la lógica de procesamiento.
- Tolerante a fallos: tras detectar un fallo aplica una recuperación automática. Cuando un componente se recupera, vuelve a formar parte del clúster. En Hadoop los fallos de hardware se tratan como una regla, no como una excepción.
- Escalable: los datos y su procesamiento se distribuyen sobre un clúster de ordenadores (escalado horizontal), desde un único servidor a miles de máquinas, cada uno ofreciendo computación y almacenamiento local.
- Portable: se puede instalar en todo tipos de hardware y sistemas operativos.

Big Data aplicado: Hadoop

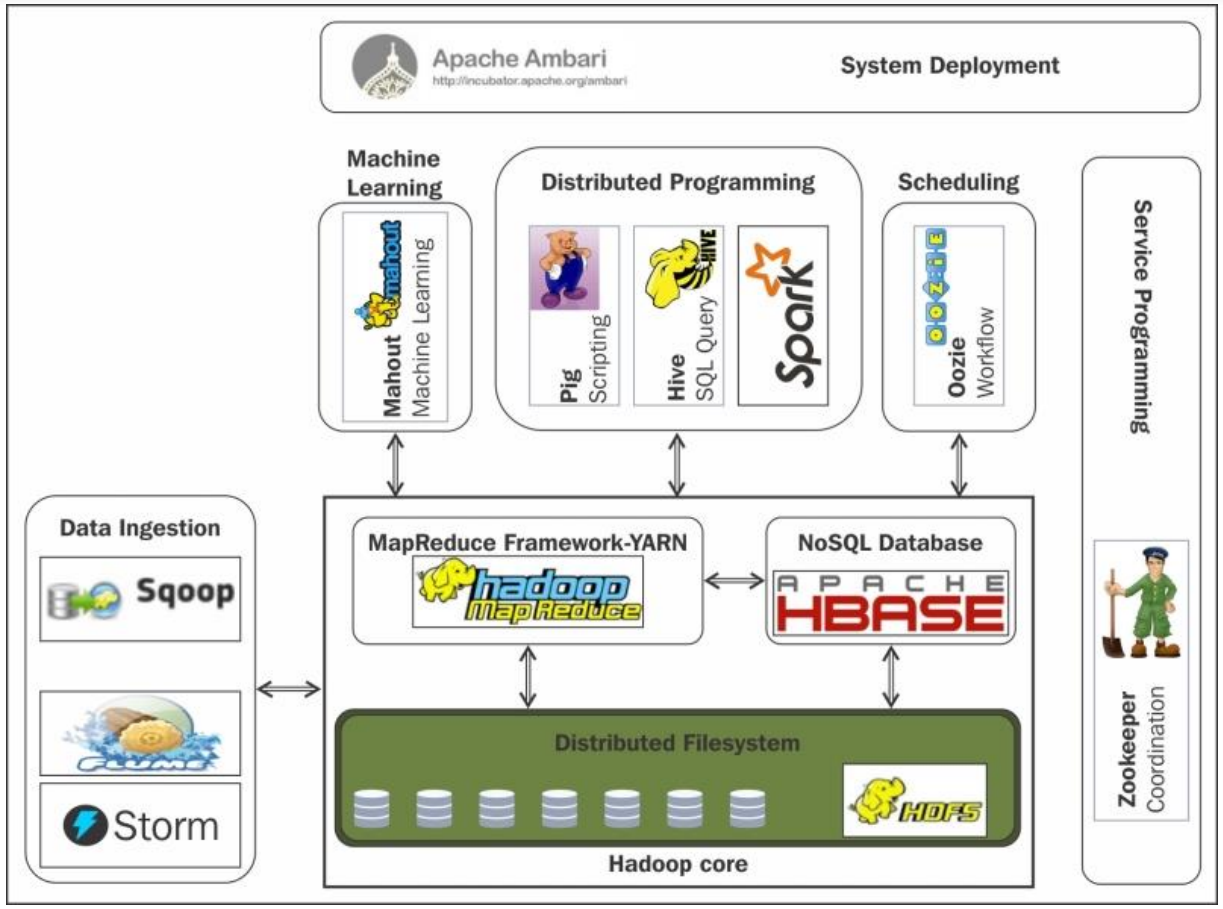


Big Data aplicado: Hadoop

El núcleo de Hadoop se compone de:

- un conjunto de utilidades comunes (Hadoop Common)
- un sistema de ficheros distribuidos (Hadoop Distributed File System ↔ HDFS).
- un gestor de recursos para el manejo del clúster y la planificación de procesos (YARN)
- un sistema para procesamiento paralelo de grandes conjuntos de datos (MapReduce)

Big Data aplicado: Hadoop



Big Data aplicado: Hadoop

Las más utilizadas son:

- **Hive**: Permite acceder a HDFS como si fuera una Base de datos, ejecutando comandos muy parecido a SQL para recuperar valores (HiveSQL). Simplifica enormemente el desarrollo y la gestión con Hadoop.
- **Sqoop**: Permite transferir un gran volumen de datos de manera eficiente entre Hadoop y sistemas gestores de base de datos relacionales.
- **Flume**: Servicio distribuido y altamente eficiente para distribuir, agregar y recolectar grandes cantidades de información. Es útil para cargar y mover información en Hadoop, como ficheros de logs, datos de Twitter/Reddit, etc.
- **Spark**: Es un motor muy eficiente de procesamiento de datos a gran escala. Implementa procesamiento en tiempo real al contrario que MapReduce, lo que provoca que sea más rápido. Para ello, en vez de almacenar los datos en disco, trabaja de forma masiva en memoria. Puede trabajar de forma autónoma, sin necesidad de Hadoop.

Big Data aplicado: Hadoop

Si queremos empezar a utilizar Hadoop y todo su ecosistema, disponemos de diversas distribuciones con toda la arquitectura, herramientas y configuración ya preparadas. Las más reseñables son:

- **Amazon Elastic MapReduce (EMR) de AWS.**
- CDH de Cloudera
- Azure HDInsight de Microsoft.
- DataProc de Google.

Big Data aplicado: HDFS

Es la capa de almacenamiento de Hadoop, y como tal, es un sistema de ficheros distribuido y tolerante a fallos que puede almacenar gran cantidad de datos, escalar de forma incremental y sobrevivir a fallos de hardware sin perder datos. Se basa en el paper que publicó Google detallando su Google File System en 2003.

Es un sistema que reparte los datos entre todos los nodos del clúster de Hadoop, dividiendo los ficheros en bloques (cada bloque por defecto es de 128MB) y almacenando copias duplicadas a través de los nodos. Por defecto se replica en 3 nodos distintos (esto se conoce como el factor de replicación).

Big Data aplicado: HDFS

No ofrece buen rendimiento para:

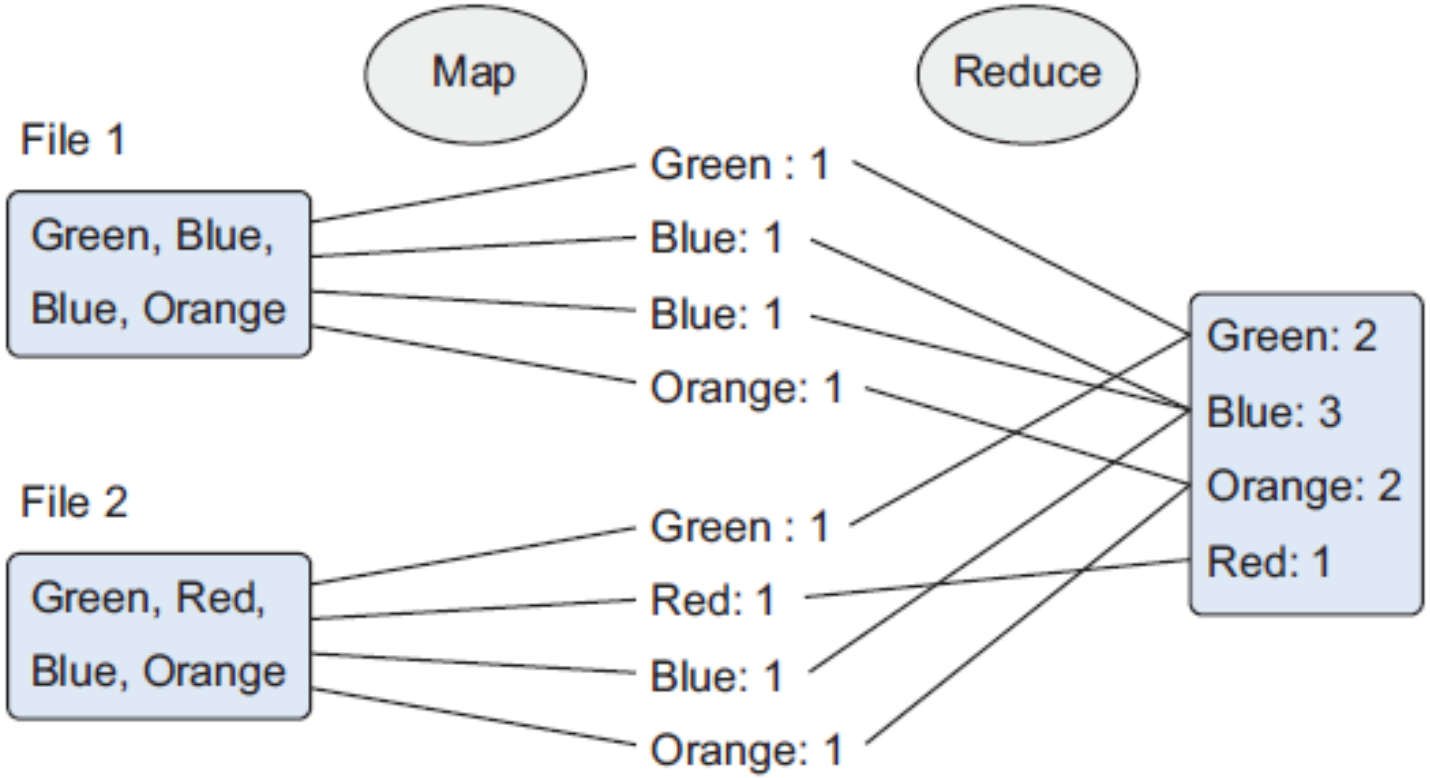
- Accesos de baja latencia. Realmente se utiliza para almacenar datos de entrada necesarios para procesos de computación.
- Ficheros pequeños (a menos que se agrupen). Funciona mejor con grandes cantidades de ficheros grandes, es decir, mejor millones de ficheros de 100MB que billones de ficheros de 1MB.
- Múltiples escritores.
- Modificaciones arbitrarias de ficheros.

Big Data aplicado: MapReduce

Se trata de un paradigma de programación funcional en dos fases, la de mapeo y la de reducción, y define el algoritmo que utiliza Hadoop para paralelizar las tareas.

Un algoritmo MapReduce divide los datos, los procesa en paralelo, los reordena, combina y agrega de vuelta los resultados mediante un formato clave/valor.

Big Data aplicado: MapReduce



AWS EMR

Amazon EMR es un servicio de Amazon Web Services que permite crear clusters Hadoop y Spark, mediante el cual podemos realizar analíticas sobre datos y cargas de BI, así como transformar y mover grandes volúmenes de datos, tanto cargando como almacenando datos en servicios de AWS como S3 y DynamoDB.

Para ello, utiliza una distribución propia de AWS que permite seleccionar los componentes que van a lanzarse en el cluster (Hive, Spark, Presto, etc...)

AWS EMR

Respecto al hardware, se ejecuta sobre máquinas EC2 (IaaS), las cuales podemos configurar según necesidades. Utiliza HDFS y S3 para el almacenamiento, de manera que podemos guardar los datos de entrada y los de salida en S3, mientras que los resultados intermedios los almacenamos en HDFS.

Los cluster de EMR se componen de:

- un nodo maestro, encargado de gestionar el cluster y ejecutar los servicios de coordinación de datos.
- varios nodos principales, los cuales ejecutan las tareas y almacenan los datos en el clúster HDFS.
- nodos tareas, los cuales son opcionales, y no almacenan datos, y podemos añadir a un cluster para incrementar la capacidad de procesamiento (e eliminarlos una vez no los necesitamos para reducir costes).

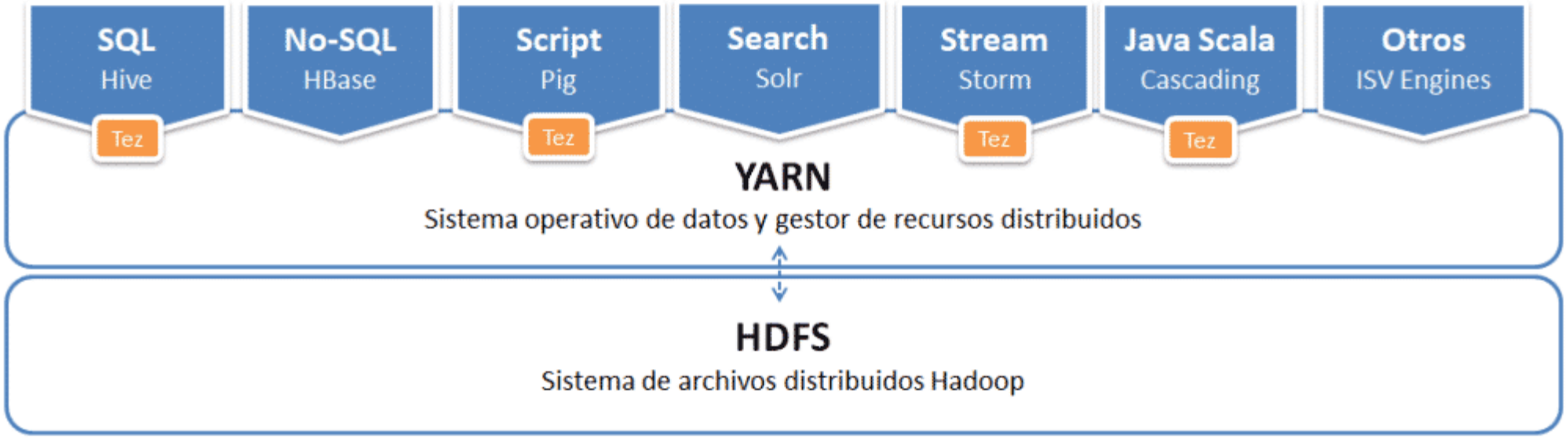
AWS EMR

A nivel de servicios, podemos definir su arquitectura en cuatro capas:

- Almacenamiento: mediante HFDS, EMR FS o el sistema de archivos local (almacenamiento de las instancias EC2).
- Gestor de recursos del cluster: YARN
- Frameworks de procesamiento de datos: Hadoop MapReduce y Apache Spark
- Aplicaciones: Apache Spark, Apache Hive, etc...

YARN

Yet Another Resource Negotiator es un distribuidor de datos y gestor de recursos distribuidos. Forma parte de Hadoop desde la versión 2, y abstrae la gestión de recursos de los procesos MapReduce lo que implica una asignación de recursos más efectiva. YARN soporta varios frameworks de procesamiento distribuido, como MapReduce v2, Tez, Impala, Spark, etc..



YARN

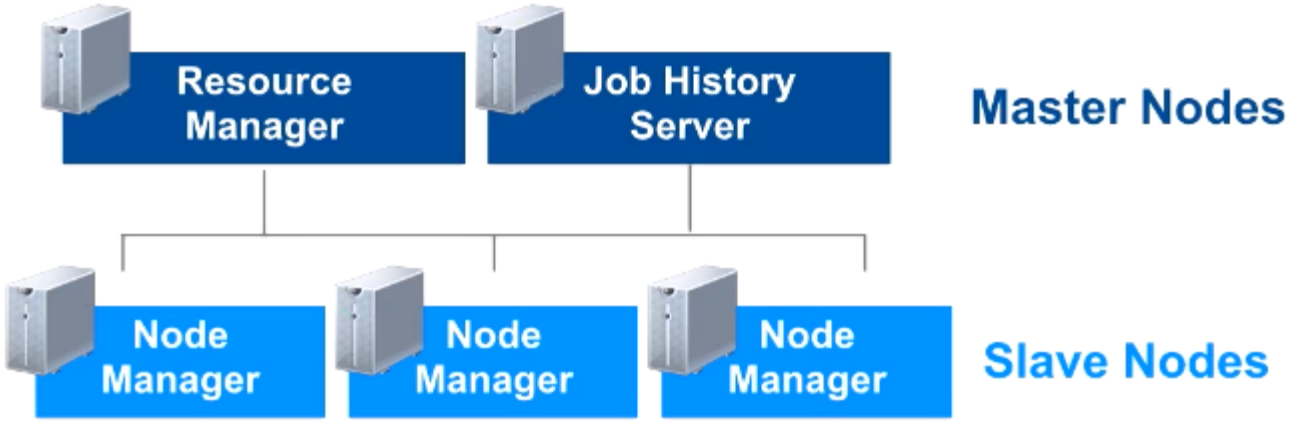
El objetivo principal de YARN es separar en dos servicios las funcionalidades de gestión de recursos de la monitorización/planificación de tareas.

Por un lado, un gestor de los procesos que se ejecutan en el clúster, que permite coordinar diferentes aplicaciones, asignar recursos y prioridades, permitir su convivencia, etc.

Y por otro lado, las aplicaciones, que pueden desarrollarse utilizando un marco de ejecución más ligero, no atado a un modelo estricto sobre cómo ejecutarse, lo que da más libertad para poder desarrollar las aplicaciones.

YARN

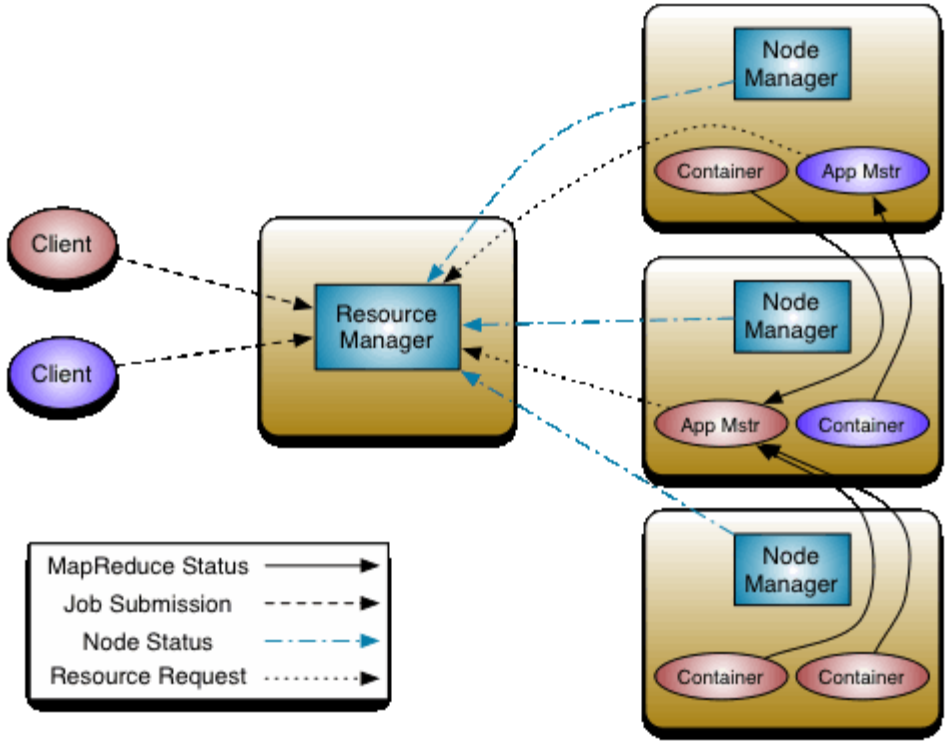
Componentes



YARN

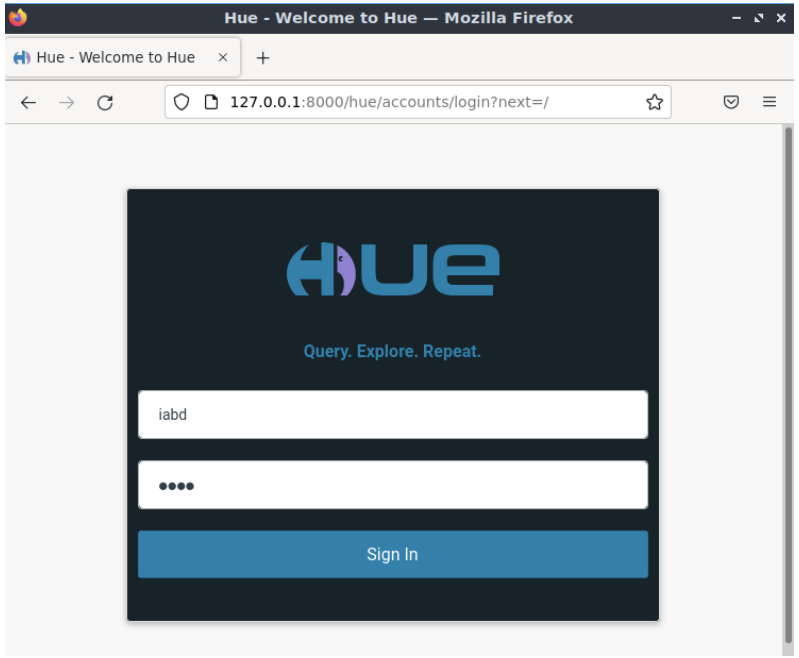
Funcionamiento

1. El cliente envía una aplicación YARN.
2. Resource Manager reserva los recursos en un contenedor para su ejecución.
3. El Application Manager se registra con el Resource Manager y pide los recursos necesarios.
4. El Application Manager notifica al Node Manager la ejecución de los contenedores. Se ejecuta la aplicación YARN en el/los contenedor/es correspondiente.
5. El Application Master monitoriza la ejecución y reporta el estado al Resource Manager y al Application Manager.
6. Al terminar la ejecución, el Application Manager lo notifica al Resource Manager.



Hue

Hue (Hadoop User Experience) es una interfaz gráfica de código abierto basada en web para su uso con Apache Hadoop. Hue actúa como front-end para las aplicaciones que se ejecutan en el clúster, lo que permite interactuar con las aplicaciones mediante una interfaz más amigable que el interfaz de comandos



Spark

Spark es un framework de computación distribuida similar a Hadoop-MapReduce (así pues, Spark no es un lenguaje de programación), pero que en vez de almacenar los datos en un sistema de ficheros distribuidos o utilizar un sistema de gestión de recursos, lo hace en memoria. El hecho de almacenar en memoria los cálculos intermedios implica que sea mucho más eficiente que MapReduce.

En el caso de tener la necesidad de almacenar los datos o gestionar los recursos, se apoya en sistemas ya existentes como HDFS, YARN o Apache Mesos. Por lo tanto, Hadoop y Spark son sistemas complementarios.

Spark

El diseño de Spark se basa principalmente en cuatro características:

- Velocidad: enfocado al uso en un clúster de commodity hardware con una gestión eficiente del multihilo y procesamiento paralelo. Spark construye sus consultas mediante un grafo dirigido acíclico (DAG) y utiliza un planificador para descomponer el grafo en tareas que se ejecutan en paralelo en los nodos de los clústers. Facilidad de uso: Spark ofrece varias capas de abstracción sobre los datos, como son los RDD, DataFrames y Dataset. Al ofrecer un conjunto de transformaciones y acciones como operaciones de su API, Spark facilita el desarrollo de aplicaciones Big data.
- Modularidad: soporte para todo tipo de cargas mediante cualquiera de los lenguajes de programación soportados: Scala, Java, Python, SQL y R, así como los módulos de Spark SQL para consultas interactivas, Spark Structured Streaming para procesamiento de datos en streaming, Spark MLlib para machine learning y GraphX para trabajar con grafos. De esta manera, mediante una única aplicación Spark se puede hacer todo sin necesidad de utilizar APIs separadas.
- Extensibilidad: Al centrarse únicamente en el procesamiento, la gestión de los datos se puede realizar a partir de Hadoop, Cassandra, HBase, MongoDB, Hive o cualquier SGBD relacional, haciendo todo en memoria. Además, se puede extender el API para utilizar otras fuentes de datos, como Apache Kafka, Amazon S3 o Azure Storage.

Spark vs Hadoop

La principal diferencia es que la computación se realiza en memoria, lo que puede implicar un mejora de hasta 100 veces mejor rendimiento. Para ello, se realiza una evaluación perezosa de las operaciones, de manera, que hasta que no se realiza una operación, los datos realmente no se cargan.

Para solucionar los problemas asociados a MapReduce, Spark crea un espacio de memoria RAM compartida entre los ordenadores del clúster. Este permite que los NodeManager/WorkerNode compartan variables (y su estado), eliminando la necesidad de escribir los resultados intermedios en disco. Esta zona de memoria compartida se traduce en el uso de RDD, DataFrames y DataSets, permitiendo realizar procesamiento en memoria a lo largo de un clúster con tolerancia a fallos.