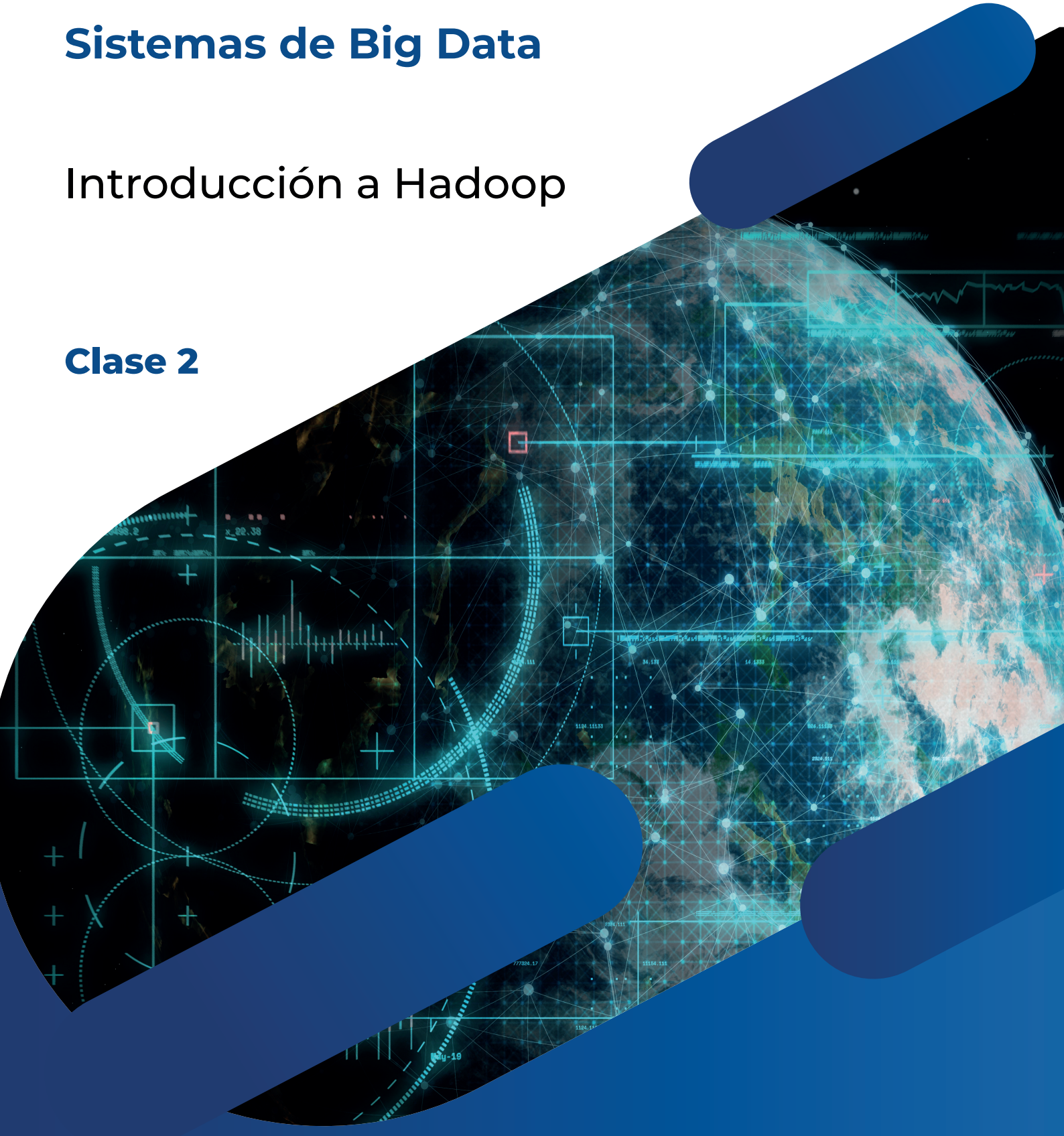
PUCE Virtual

# Sistemas de Big Data

## Introducción a Hadoop

## Clase 2



MAESTRÍA EN
SISTEMAS DE INFORMACIÓN
Mención Data Science

La excelencia no se improvisa

# 1. INTRODUCCIÓN DE LA CLASE

**RDA 1:** Utilizar tecnologías y herramientas de Big Data para grandes volúmenes de datos.

En el contexto del **Big Data**, el **procesamiento eficiente** de grandes volúmenes de información es un **desafío clave** para las **organizaciones**. **Hadoop** es un marco de trabajo de **código abierto,** desarrollado por la ***Apache Software Foundation,*** que permite el **almacenamiento** y **procesamiento distribuido** de datos masivos mediante un modelo **escalable** y **tolerante** a **fallos** (White, 2012). Su arquitectura se basa en el uso de hardware común, lo que reduce costos y facilita la gestión de datos estructurados y no estructurados en entornos empresariales y científicos (Borthakur, 2008).

El ecosistema Hadoop está compuesto por varios **componentes esenciales**, siendo los principales el sistema de archivos distribuido de Hadoop (HDFS) y el modelo de programación **MapReduce. HDFS** permite almacenar datos de forma distribuida en múltiples **nodos**, garantizando **redundancia** y **disponibilidad**, mientras que **MapReduce** facilita el **procesamiento paralelo** de grandes volúmenes de información (Dean & Ghemawat, 2004). Además, herramientas como **Apache Hive** y **Apache Pig** han sido desarrolladas para simplificar la consulta y manipulación de datos dentro del ecosistema **Hadoop**, mejorando su accesibilidad para analistas y científicos de datos (Lam, 2010).

## Clase 2. Introducción a Hadoop

## 2. Introducción a Hadoop

Antes de hablar sobre Hadoop es importante tener claro qué significa y qué implica un **sistema distribuido de archivos.** En la mayoría de las oficinas se cuenta con archiveros, en donde se almacenan físicamente carpetas y documentos en función de diferentes criterios, como puede ser en orden alfabético o por fecha.

Cuando las primeras computadoras hicieron su aparición, los archivos y programas estaban almacenados en tarjetas perforadas o ***punch cards,*** las cuales eran archivos en archiveros, tal y como se almacenan documentos físicos en la actualidad, es ahí de donde el término **'sistema de archivos'** tiene su origen.

La necesidad de almacenar información en periodos largos de tiempo surge para dar solución a problemáticas tales como:

- Acceder a los resultados de un proceso después de un tiempo.
- Almacenar grandes volúmenes de información.
- Habilitar el acceso de los archivos a múltiples procesos.

Por esos motivos almacenamos información en discos duros. Estos archivos los manejamos a través de sistemas operativos, tales como Windows o Linux; el **sistema de archivos** es la forma en que esos archivos son manejados. Eso aplica a computadoras de escritorio con una capacidad de almacenamiento limitada. Pero, ¿qué pasa si se necesita más poder de **almacenamiento**? Una opción puede ser comprar otro disco duro con más capacidad de almacenamiento. Otra opción puede ser copiar los datos a un disco duro externo.

Una opción más podría ser que tengas dos computadoras y guardes cierta información en una y cierta información en otra. Si eres muy organizado y no tienes muchos archivos, puede que sea una buena opción; pero qué pasaría si fuesen muchos archivos, necesitarías ayuda, ahí es donde aparece el **sistema de archivos distribuidos.**

**Los sistemas de archivos distribuidos funcionan de la siguiente manera:**

Hay ***racks*** distribuidos en diferentes **regiones geográficas** del mundo, que tienen computadores que a su vez almacenan partes constitutivas de los archivos o *datasets* que se deben almacenar.

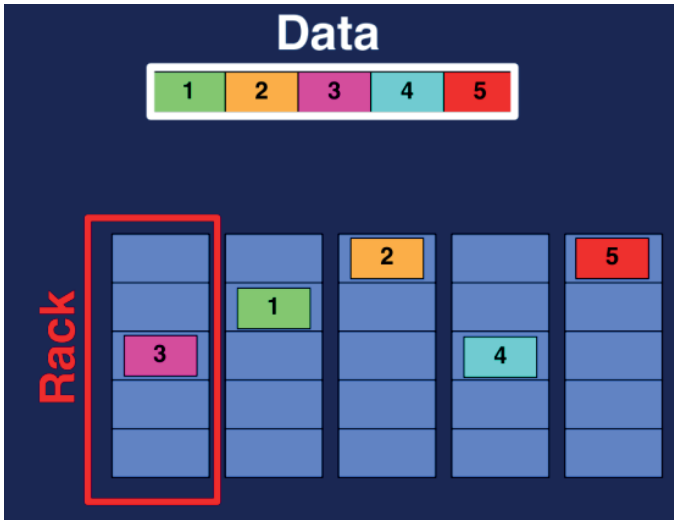



Figura N.º 1. Sistema de archivos distribuidos.

Además, replican dos datos entre los ***racks*** y entre sus computadoras; esto se conoce como **replicación** y es la pieza clave de la **tolerancia a fallas.**

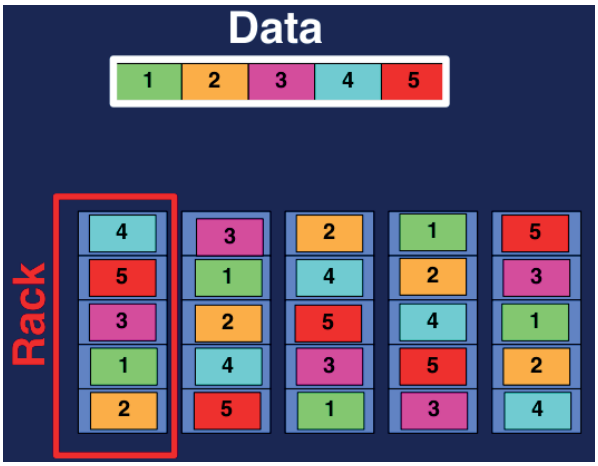



Figura N.º 2. Sistema de archivos distribuidos. Replicación.

Esto implica que, si por algún motivo uno de los **racks o nodos** (computadoras) se daña, el usuario que quiera acceder a esa información lo hará a través de otro de los nodos; garantizando –además de la tolerancia a fallas– la escalabilidad de los datos y la alta concurrencia.

**Hadoop** es un ecosistema *open-source* que brinda escalabilidad para almacenar grandes volúmenes de datos en ***commodity hardware,*** **este término está asociado al *commodity cluster*.** Además, es un ecosistema tolerante a fallas; por otro lado, una de las características principales del Big Data es la diversidad de fuentes que hay, el ecosistema de **Hadoop** acepta y administra una gran variedad de tipos de datos (textos, grafos, información de sensores, imágenes, etc.). También facilita un ambiente compartido.

Existen diferentes productos *open-source;* algunos de ellos dependen de Hadoop, mientras que otros son independientes. En el siguiente diagrama se aprecian diferentes capas. Los productos más importantes de Hadoop son **HDFS, YARN y MapReduce.** Los componentes que se encuentran en los *layers*

superiores usan las funciones o capacidades de los componentes que se encuentran en los *layers* inferiores.

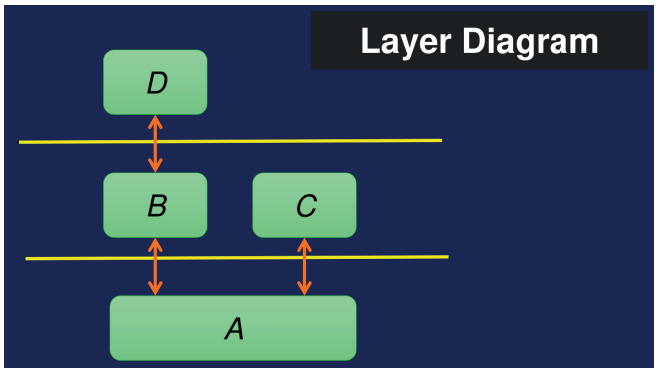



Figura N.º 3. Diagrama de capas.

Los componentes se encuentran organizados de tal forma que en la parte inferior se encuentran los de bajo nivel y permiten realizar tareas de **almacenamiento** y **calendarización**. Por otro lado, los componentes que se encuentran en la parte superior son de alto nivel y bridan mayores niveles de interactividad

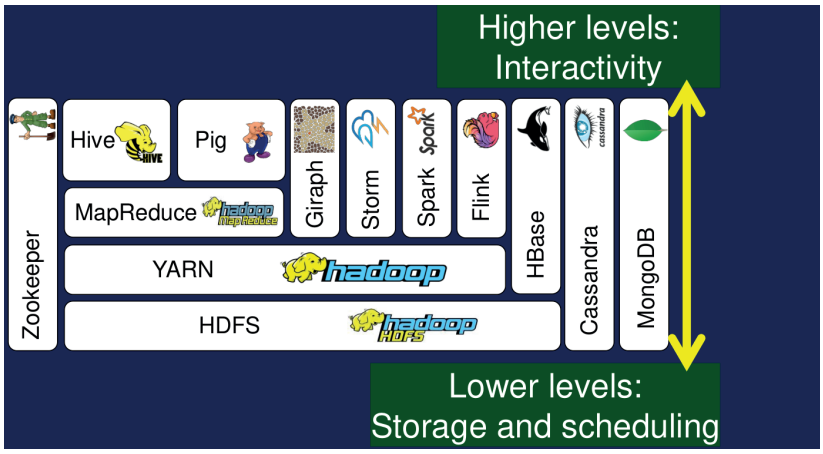



Figura N.º 4. Ecosistema Hadoop.

## 2.1. Conceptos básicos de Hadoop

### 2.1.1. Hadoop Distributed File System (HDFS)

Es la base del ecosistema de Hadoop. Brinda esencialmente **escalabilidad** para grandes volúmenes de datos, y **fiabilidad** o **confiabilidad** (***reliability***) para hacer frente a fallas de hardware.

Figura N.º 5. Hadoop Distributed File System (HDFS).

**HDFS** permite almacenar y acceder a grandes volúmenes de información. Según **Hortonworks,** uno de los proveedores líderes a nivel mundial de los servicios de Hadoop, HDFS ha mostrado escalabilidad en producción de hasta **200 petabytes** en un clúster de 4 500 **servidores,** con cerca de un billón de archivos y bloques. Si por algún motivo nos quedamos sin espacio, simplemente se deben añadir más nodos para incrementarlo.

HDFS logra **escalabilidad** particionando o dividiendo los archivos a través de múltiples **computadoras**. Esto permite acceso en **paralelo** a grandes archivos, dado que las computadoras corren en paralelo en cada nodo donde los datos están almacenados. Por defecto, en HDFS el *chunk size* o tamaño de cada pieza de archivo es de **64 MB.** HDFS está diseñado ***fault tolerance***, HDFS replica o crea una copia de los bloques de archivos en diferentes nodos para prevenir pérdida de datos.

En el siguiente ejemplo se muestra qué es lo que pasaría si el nodo que almacena el bloque C se daña. Por más de que ese nodo no funcione más, el **bloque** C se encuentra replicado en **dos nodos más**. Por defecto, HDFS mantiene **3 copias** de cada bloque; este valor se lo conoce como ***default replication factor*.**

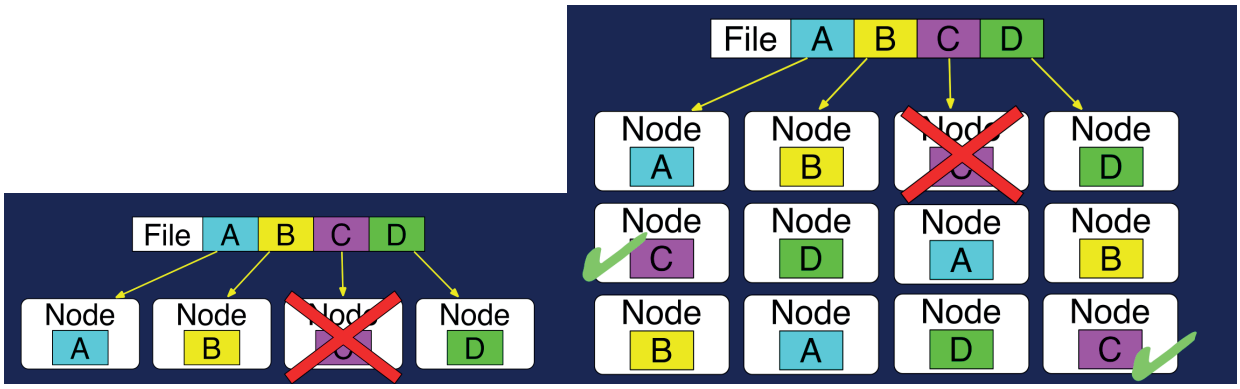



Figura N.º 6. HDFS *Fault Tolerance.*

HDFS también está diseñado para almacenar una variedad de tipos de datos en consonancia con la característica de **variedad** del *Big Data;* por lo tanto, para poder leer un archivo en HDFS es obligatorio especificar el tipo de formato del archivo de entrada, esto aplica también para los archivos de salida. HDFS brinda un conjunto de posibles formatos para archivos, pero también se pueden añadir **formatos personalizados;** por ejemplo, los archivos de texto pueden ser leídos línea por línea o palabra por palabra.

HDFS tiene dos componentes principales:

- ***Namenode***: responsable de la metadata, usualmente uno por clúster. Se lo puede considerar también como el **coordinador** del clúster de HDFS. Cuando un archivo es creado, el *namenode* guarda el **nombre**, la **ubicación** del **archivo** en el **directorio** y más metadata. El *namenode* también **decide qué *datanode* guarda** los contenidos del archivo y **guarda** este **mapeo** en su memoria.
- ***DataNode***: almacena los bloques de datos, usualmente uno por máquina (nodo).
  El *datanode* 'escucha' los comandos que manda el *namenode* para la creación de **bloques**, **eliminación** y **replicación**. La replicación permite tener **tolerancia a fallos** y **ubicación de los datos**. La replicación también significa que un mismo bloque será almacenado en diferentes nodos que se encuentran en distintas ubicaciones geográficas **(pueden ser un *rack* o un data center).**

### 2.1.2. YARN (el administrador de recursos de Hadoop)

YARN (Yet Another Resource Negotiator) es el **administrador** de **recursos** de Hadoop. Interactúa con aplicaciones y calendariza recursos para su uso. YARN habilita la ejecución de **múltiples aplicaciones**

de **HDFS** y maneja la **asignación** de **recursos**. YARN aparece desde la versión **2.0** de Hadoop; es decir, antes de esa versión no existía un administrador de recursos.

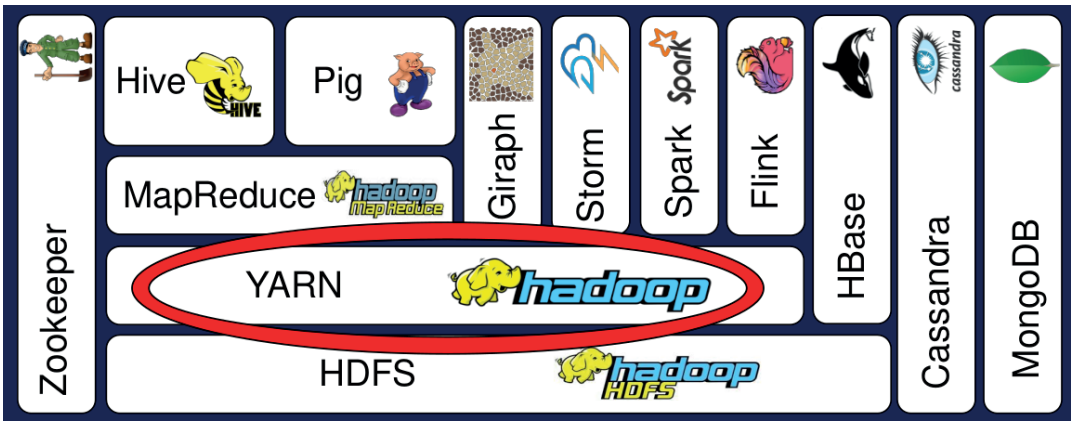



Figura N.º 7. YARN.

Algunas de las grandes limitaciones de **Hadoop 1.0** eran su incapacidad de soportar aplicaciones que no eran del tipo ***MapReduce,*** el pobre uso de los recursos y ser compatible con pocas aplicaciones. Con la aparición de **YARN** aumentó la compatibilidad con más aplicaciones, basadas por supuesto en ***MapReduce*** pero también en ***Spark, Giraph,*** entre otras, permitiendo el desarrollo de aplicaciones ***custom*** en el ecosistema de Hadoop.

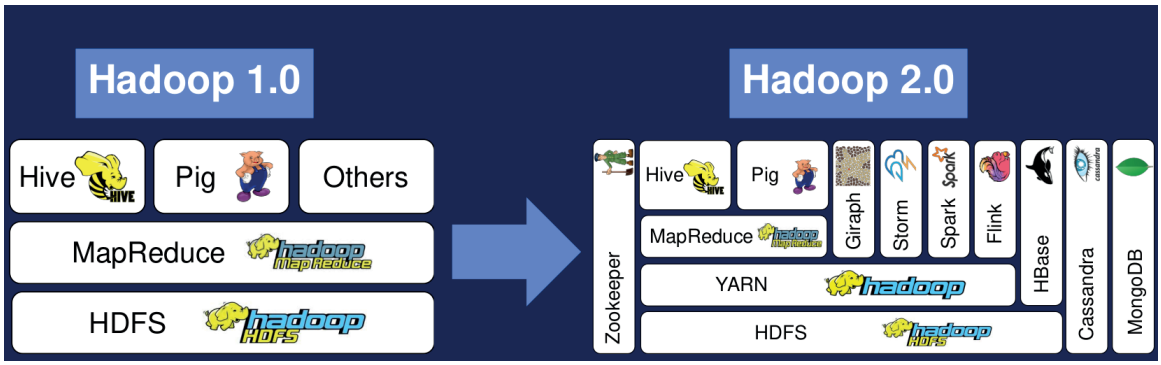



Figura N.º 8. Evolución de Hadoop 1.0 a Hadoop 2.0.

### Arquitectura de YARN

Hay dos componentes principales el ***Resource Manager*** y los ***NodeManager***. El primero controla todos los recursos y decide quién toma qué cosa. Por otro lado, el *node manager* trabaja a nivel del nodo y está a cargo de esa única máquina. El trabajo conjunto de los dos se conoce como ***Data Computation Framework***.

Adicionalmente, existe el ***Aplication Master,*** que se encarga de negociar con el *resource manager* los recursos necesarios para ejecutar una determinada **aplicación;** también se encarga de interactuar con el ***node manager,*** para garantizar que la tarea termine. Por otra parte existe el ***container,*** que es una abstracción de los **recursos** del nodo (CPU, disco, red, memoria, etc.).

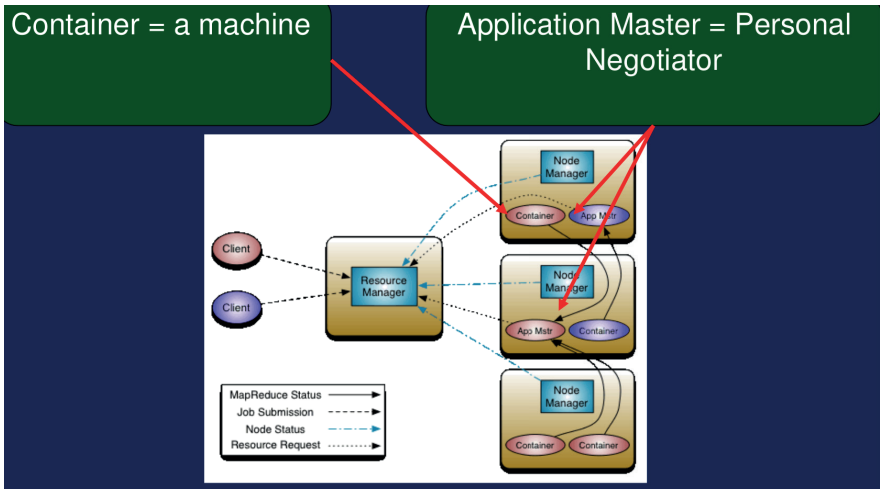



Figura N.º 9. Arquitectura de YARN.

Gracias a la aparición de **YARN** se hizo posible utilizar directamente aplicaciones en el ecosistema de **Hadoop 2.0,** como ***Spark, Apache Hama, Map Reduce, Giraph, Hbase,*** entre otras; en contraste con la versión 1.0 de Hadoop, en donde solo se podían usar aplicaciones basadas en *MapReduce*.

**Flujo de trabajo de YARN**

El proceso de trabajo en YARN comienza cuando un usuario o una aplicación solicita recursos para ejecutar un trabajo. Este trabajo se presenta al ***ResourceManager***, que evalúa qué recursos están disponibles y asigna un contenedor en un nodo del clúster. Una vez que los recursos están disponibles, el ***ApplicationMaster*** se lanza en el nodo correspondiente para gestionar la ejecución de las tareas. A medida que el trabajo avanza, el *ApplicationMaster* puede solicitar más recursos, y el ***NodeManager*** ejecutará las tareas dentro de los contenedores asignados (Shvachko et al., 2010).

Cuando la aplicación termina, el ***ResourceManager*** libera los recursos y los contenedores se cierran. En caso de que ocurra un fallo en alguno de los contenedores o nodos, YARN es capaz de redistribuir las tareas o reiniciar los contenedores, lo que garantiza la tolerancia a fallos y la continuidad del procesamiento de los datos (Vavilapalli et al., 2013).

**Ventajas de YARN**

**Escalabilidad y flexibilidad**: YARN permite una escalabilidad mucho mayor en comparación con la versión anterior de Hadoop. Al separar la gestión de recursos de la ejecución de las aplicaciones, YARN facilita la ejecución de varios tipos de aplicaciones en el mismo clúster, sin interferencias. Esto resulta en un uso más eficiente de los recursos disponibles y una mayor capacidad de procesamiento (Shvachko et al., 2010).

**Tolerancia a fallos**: la capacidad de YARN para gestionar fallos es una de sus características más importantes. Si un nodo o contenedor falla, YARN puede redistribuir las tareas a otros nodos, asegurando que el procesamiento continúe sin interrupciones significativas (Vavilapalli et al., 2013).

**Soporte para múltiples *frameworks***: a diferencia de las versiones anteriores de Hadoop, que solo soportaban ***MapReduce***, YARN permite ejecutar otros marcos de trabajo, como *Apache Spark*, *Apache Tez* y *Flink*, lo que proporciona mayor flexibilidad y optimización para distintos tipos de aplicaciones (YARN, 2014).

### 2.1.3. MapReduce

***MapReduce*** es un modelo de programación para el ecosistema de Hadoop, depende de YARN para calendarizar y ejecutar procesos en paralelo sobre los bloques distribuidos en HDFS, existen algunas herramientas que usan *MapReduce* para brindar una interfaz de más alto nivel; como por ejemplo **Hive,** que tiene una interfaz de tipo ***SQL*** o ***Pig*** que sirve para armar flujos de datos.

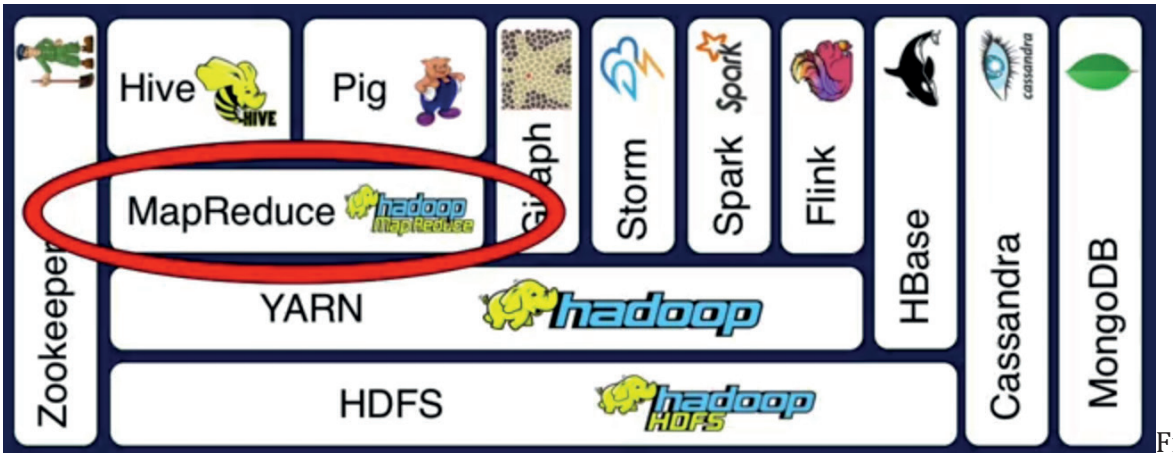


Figura N.º 10. MapReduce.

El procesamiento de código en paralelo requiere de un gran ***expertise*** en gran número de **herramientas** y **conceptos,** tales como semaforización, hilos de datos, *locks*, compartimiento de memoria, monitoreo, etc. Mediante el modelo de programación ***MapReduce,*** todo se reduce a tareas de tipo ***map*** y de tipo ***reduce***. Los conceptos de ***MapReduce*** están basados en programación funcional:

- ***Map:*** aplica una operación a cada uno de los elementos.
- ***Reduce***: resume el resultado de las operaciones de elementos de alguna manera.

El ***'hello word'*** de ***MapReduce*** es el ***WordCount.*** Este programa cuenta la aparición de cada una de las palabras en todos los archivos. El resultado es un archivo que tiene el listado de palabras con su respectiva cantidad de ocurrencias.

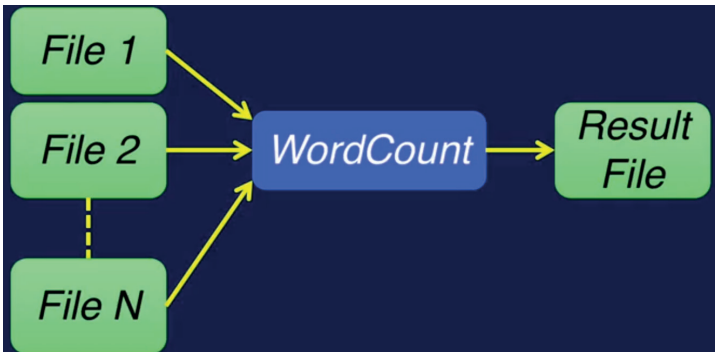



Figura N.º 11. *Word Count.*

Como primer paso, o paso 0, los archivos son almacenados en HDFS.

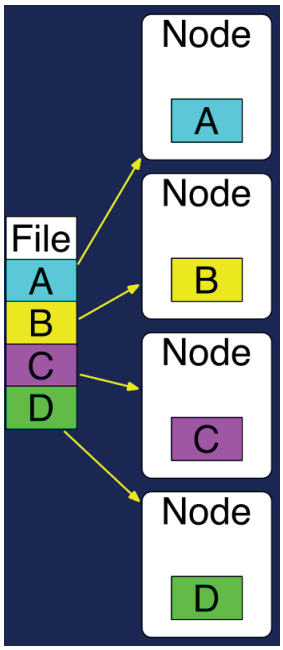



Figura N.º 12. Almacenamiento de archivos en HDFS.

Luego se hace un proceso **map** en cada uno de los nodos.

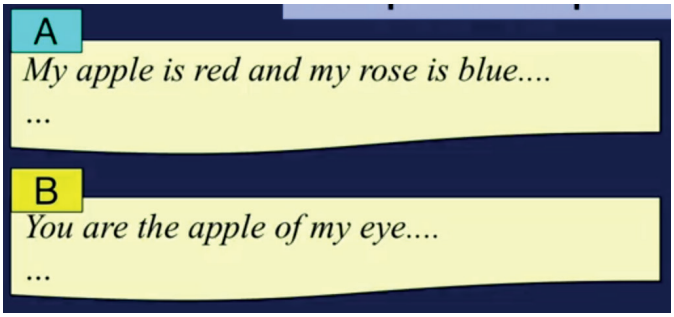



Figura N.º 13. Textos en cada partición.

El proceso **map** crea pares de tipo **clave-valor,** en donde la clave es la palabra y el valor es 1, es por eso que para el ejemplo tendríamos los siguientes pares clave-valor para la **partición A**:

- *My, my ==> (my, 1), (my, 1)*
- *Apple ==> (apple, 1)*
- *Is, is ==> (is, 1), (is, 1)*
- *Red ==> (red, 1)*
- *And == >(and, 1)*
- *Rose ==> (rose, 1)*
- *Blue ==> (blue, 1)*

Para la **partición B** tendríamos los siguientes pares clave-valor:

- *You ==> (You,1)*
- *Are ==> (are, 1)*
- *The ==>(the, 1)*
- *Apple ==> (apple, 1)*
- *Of ==>(of, 1)*
- *My ==>(my,1)*
- *Eye ==> (eye,1)*

Posteriormente los **pares** que tienen el **mismo key** son movidos al **mismo nodo** y, por supuesto, un mismo nodo puede tener **key values** distintos.

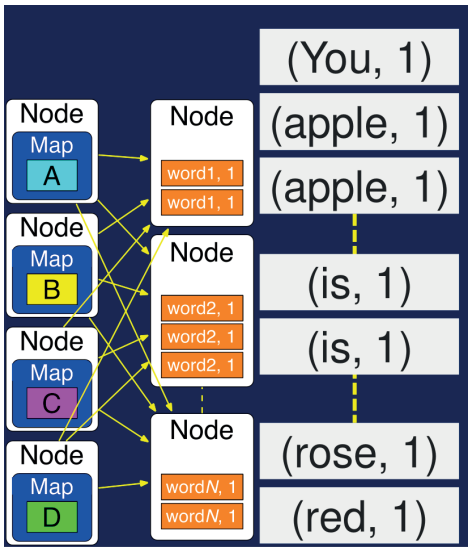



Figura N.º 14. *Map Operation.*

Finalmente, la operación **reduce** suma los valores para los pares que tienen la misma **key**:

- *(You, 1) ==> (You, 1)*
- *(apple,1), (apple, 1) ==> (apple, 2)*
- *(my,1), (my,1), (my,1) ==> (my,3)*

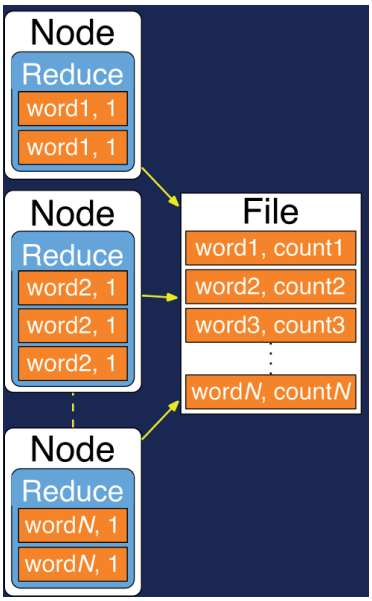



Figura N.º 15. Reduce.

***Workflows for Data Science Center of Excellence [https://words.sdsc.edu/words-data-science/mapreduce]***

### 2.1.4. Principales proveedores de Hadoop

El ecosistema de Hadoop es una infraestructura distribuida que permite el procesamiento de grandes volúmenes de datos. En este ecosistema, existen varios proveedores clave que contribuyen a la optimi-

zación y expansión de sus capacidades. A continuación, se describen algunos de los principales proveedores del ecosistema Hadoop.

**Cloudera**

Cloudera es uno de los **principales proveedores** de Hadoop, ofrece una plataforma de datos empresariales que facilita la gestión y análisis de grandes volúmenes de datos. La empresa proporciona un entorno controlado, herramientas de seguridad avanzadas y soluciones de gestión que permiten a las organizaciones **implementar** Hadoop de manera eficiente y escalable. Cloudera se destaca por su distribución de Hadoop, que incluye herramientas adicionales como **Apache Impala** y **Apache Kudu**, lo que facilita el procesamiento interactivo de grandes datos (Cloudera, 2025).

**Hortonworks**

Hortonworks, ahora parte de Cloudera, fue un proveedor clave del ecosistema Hadoop antes de su adquisición. Su principal aporte fue el desarrollo de **Hortonworks Data Platform** (HDP), una **distribución** de código abierto de Hadoop. Hortonworks se especializó en ofrecer soluciones para la gestión de datos y la integración de herramientas de Big Data en entornos empresariales. Su enfoque en la comunidad y el desarrollo abierto impulsó la **adopción** de Hadoop en **diversas industrias** (Hortonworks, 2025).

**MapR**

MapR Technologies, otro importante proveedor de Hadoop, ofreció una plataforma que integraba el procesamiento de datos con capacidades de almacenamiento y análisis en tiempo real. MapR se diferenció por su sistema de archivos distribuido **MapR-FS**, que proporcionaba un rendimiento mejorado en comparación con el HDFS tradicional. Esta plataforma permitía la integración con diversas aplicaciones y tecnologías de Big Data como **Apache Kafka** y **Apache Drill**, lo que la convirtió en una opción popular en sectores como telecomunicaciones, banca y energía (MapR Technologies, 2025).

**Amazon EMR**

Amazon Elastic MapReduce (EMR) es un servicio en la nube que facilita el procesamiento de grandes volúmenes de datos utilizando Hadoop y otras herramientas de Big Data. EMR ofrece la flexibilidad de escalar clústeres de manera dinámica, lo que lo convierte en una **solución** conveniente para empresas que requieren procesar **grandes volúmenes** de datos sin la necesidad de gestionar **infraestructura** física. Además, EMR se integra de manera eficiente con otros servicios de ***Amazon Web Services*** (AWS), lo que permite una mayor interoperabilidad en el **ecosistema de datos** (Amazon Web Services, 2025).

## 2.2. Implementación de máquina virtual y ambiente de Hadoop

La **instalación** y **configuración** del ecosistema de Hadoop puede llegar a consumir mucho tiempo de trabajo y desarrollo. Sin embargo, con el uso de **imágenes preinstaladas** se puede ganar mucho tiempo y estar seguro de que la configuración es la adecuada. Es similar a comprar un mueble preinstalado, solamente hay que ajustar ciertos parámetros para que se pueda utilizar sin problemas.

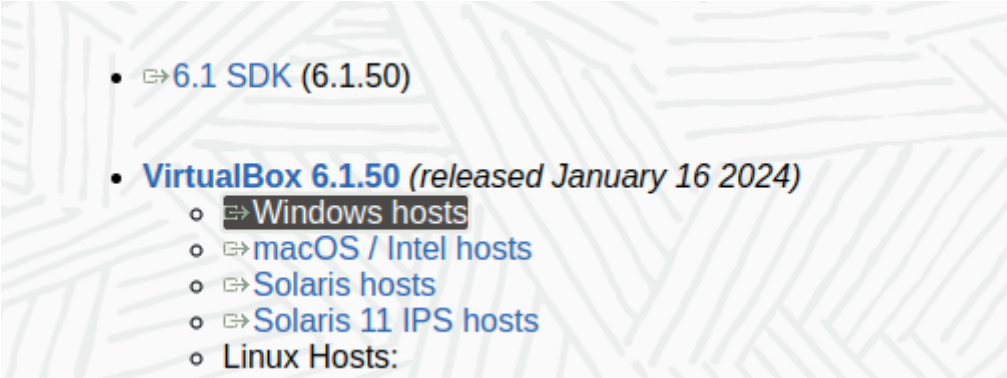
Estas imágenes se pueden usar mediante software de virtualización, tales como **VirtualBox** o **Vmware.** Hay varias empresas que proveen máquinas virtuales con el ecosistema de Hadoop instalado, entre ellas tenemos a **Hortonworks** y **Cloudera;** también tienen tutoriales para la configuración y uso de Hadoop.

***Cloudera [https://www.cloudera.com/open-source.html]***

A continuación, se describen los pasos que se deben seguir para la **instalación** y **configuración** de una máquina virtual de **Cloudera** con Hadoop instalado.

**Descargar e instalar** VirtualBox usando el siguiente link, y seleccionando la opción resaltada en la imagen:

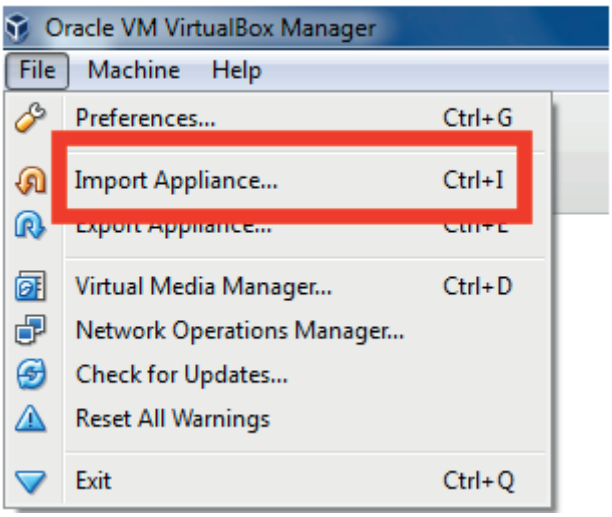
https://www.virtualbox.org/wiki/Download_Old_Builds_6_1



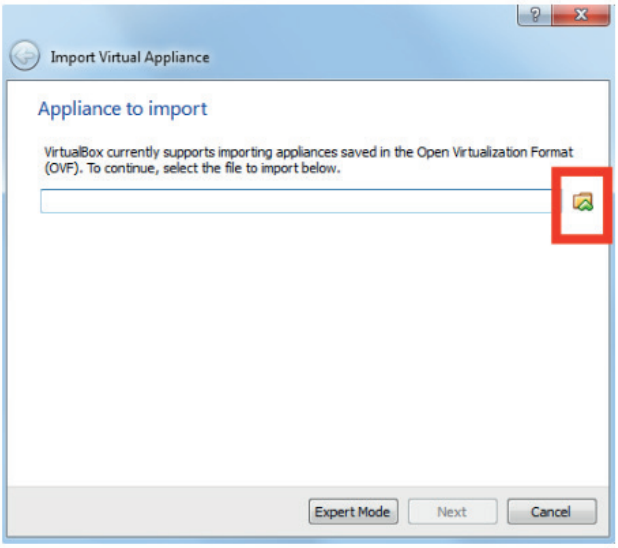

**Descargar** la máquina virtual usando este link: Máquina Virtual

**Iniciar VirtualBox**

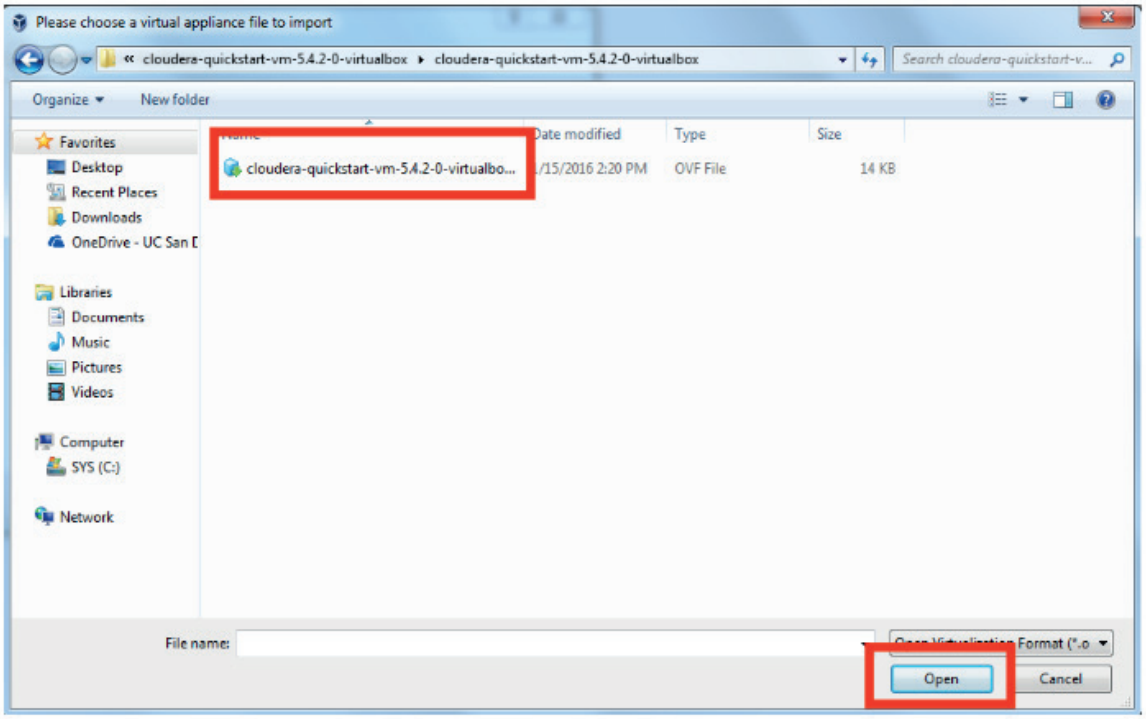
**Importar** la máquina virtual descargada seleccionando Archivo -> Importar servicio virtualizado




**Cliquear** el ícono de carpeta.



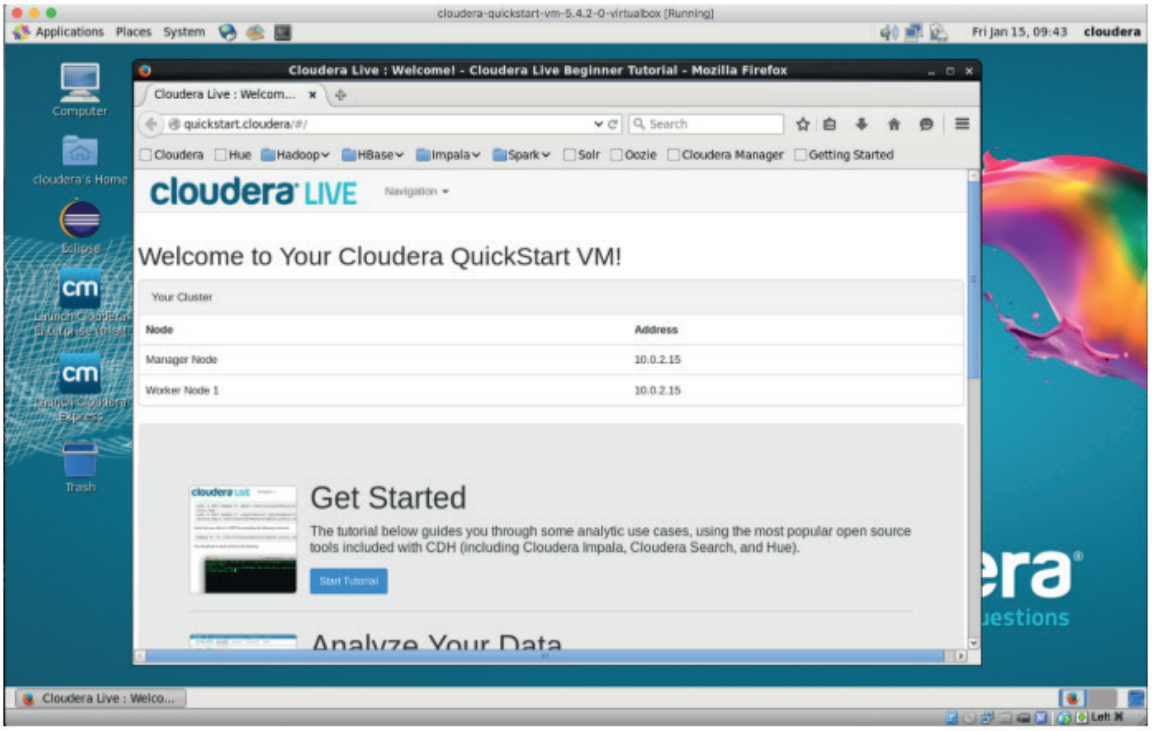
**Seleccionar** cloudera-quickstart-vm-5.4.2-0-virtualbox.ovf.




**Importamos** la máquina virtual.

**Iniciamos** la máquina virtual.

Se **tendrá** una máquina virtual similar a esta:



**Resumen final de la clase**

**Introducción a Hadoop y su ecosistema**

Hadoop es un **marco** de **software** de código abierto diseñado para almacenar y procesar **grandes volúmenes** de datos de manera distribuida y eficiente. Su ecosistema incluye varios componentes claves, entre ellos **HDFS**, **YARN** y **MapReduce**. Cada uno de estos componentes tiene un papel esencial en el procesamiento de datos en paralelo, esto permite que las empresas manejen y analicen grandes cantidades de información de forma eficiente (Shvachko et al., 2010).

**Hadoop Distributed File System (HDFS)**

**HDFS** es el sistema de almacenamiento distribuido que forma la base del **ecosistema Hadoop**. Está **diseñado** para almacenar grandes volúmenes de datos, de manera que estos se distribuyan en múltiples nodos en una red de computadoras. Esto permite la tolerancia a fallos, ya que los datos se replican en diferentes nodos, asegurando que si un nodo falla los datos no se pierdan. Además, **HDFS** se optimiza para operaciones de lectura secuencial, esto lo convierte en una opción ideal para **almacenar datos** de **Big Data** y para procesar grandes volúmenes de información de manera eficiente (Dean & Ghemawat, 2004).

**HDFS** se compone de dos **componentes principales**: **NameNode**, que gestiona la metadata y estructura del sistema de archivos, y **DataNode**, que almacena los datos reales. Esta **arquitectura** permite una gestión **eficiente** de los **recursos** y una **recuperación rápida** en caso de fallos (Shvachko et al., 2010).

**MapReduce**

**MapReduce** es un modelo de programación que permite el **procesamiento** de **grandes conjuntos** de datos en un entorno distribuido. En Hadoop, este modelo se utiliza para dividir las tareas de procesamiento en dos **fases**: **Map** y **Reduce**. Durante la fase Map, los datos se dividen en fragmentos que son **procesados en paralelo** en diferentes nodos del clúster. En la fase Reduce, los resultados del procesamiento de los fragmentos de datos se combinan para obtener una salida final (Dean & Ghemawat, 2004).

El modelo **MapReduce** es altamente eficiente para tareas de procesamiento en paralelo, lo que lo hace adecuado para tareas de análisis de grandes volúmenes de datos, como la minería de datos, el **análisis** de **logs** y otros procesos que requieren manipular grandes conjuntos de información de manera distribuida. Sin embargo, uno de los desafíos del modelo es su eficiencia para operaciones que no son fácilmente paralelizables, como las que requieren un alto **grado de dependencia** entre los datos (Dean & Ghemawat, 2004).

## Referencias citadas en la Clase 2


- Amazon Web Services. (2025). *Amazon Elastic MapReduce (EMR)*. Recuperado de https://aws.amazon.com/emr/
- Borthakur, D. (2008). *The Hadoop Distributed File System: Architecture and Design*. Apache Software Foundation.
- Cloudera. (2025). *Cloudera's Enterprise Data Cloud* . Recuperado de https://www.cloudera.com/
- Dean, J., & Ghemawat, S. (2004*). MapReduce: Simplified data processing on large clusters*. *Communications of the ACM*, 51(1), pp. 107-113. Lam, C. (2010). *Hadoop in Action*. Manning Publications.

- Hortonworks. (2025). *Hortonworks Data Platform*. Recuperado de https://hortonworks.com/
- MapR Technologies. (2025). *MapR Data Platform*. Recuperado de https://mapr.com/
- Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). *The Hadoop Distributed File System*. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies* (pp. 1-10). IEEE. https://doi.org/10.1109/MSST.2010.5496972
- Vavilapalli, V. K., Murthy, A. V., & Agarwal, S. (2013). *Apache Hadoop* YARN*: Yet Another Resource Negotiator. ACM Symposium on Cloud Computing*, pp. 1–16. https://doi.org/10.1145/2523616.2523621
- White, T. (2012). Hadoop*: The Definitive Guide*. O'Reilly Media.
- YARN. (2014). *Apache Hadoop YARN ResourceManager*. Recuperado de https://Hadoop.apache.org


## Definición de los términos citados en la Clase 2

**Commodity Cluster.** Conjunto de **servidores estándar** y de **bajo costo** que trabajan juntos para proporcionar **capacidad** de **cómputo distribuida**. A diferencia de los sistemas tradicionales basados en hardware especializado y costoso, los **commodity clusters** usan nodos compuestos por equipos comerciales convencionales (commodity hardware), como servidores comunes con procesadores, memoria y almacenamiento estándar. Estos nodos están conectados por redes de alta velocidad y funcionan como una **unidad coordinada** para ejecutar tareas de procesamiento de datos a gran escala.

**Apache Pig.** Plataforma de alto nivel para procesar grandes volúmenes de datos en Hadoop. Utiliza un lenguaje llamado ***Pig Latin***, que permite escribir *scripts* para la manipulación y análisis de datos de manera más sencilla en comparación con Java en MapReduce. Apache Pig facilita la ejecución de tareas complejas como filtrado, agregación y unión de datos mediante una sintaxis declarativa similar a **SQL**. Su principal ventaja es la abstracción de los detalles de procesamiento en Hadoop, permitiendo a los analistas y desarrolladores trabajar con grandes conjuntos de datos de forma más eficiente.

## Profundización Clase 2

Ejemplos en Python de la implementación de MapReduce.

**Ejemplo 1.**
**Contador de palabras:** este *script* cuenta la frecuencia de cada palabra en un conjunto de texto.

```
from mrjob.job import MRJob

class WordCount(MRJob):
    def mapper(self, _, line):
        for word in line.split():
            yield word.lower(), 1

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == "__main__":
    WordCount.run()
```

**Explicación**

- **Mapper:** divide el texto en palabras y emite cada una con el valor 1.
- **Reducer:** suma las ocurrencias de cada palabra.

**Ejemplo de entrada:** hola mundo hola data science

**Ejemplo de salida:**

```
"hola"   2
"mundo"  1
"data"   1
"science" 1
```

**Ejemplo 2.**
**Promedio de calificaciones por estudiante:** este *script* calcula el promedio de calificaciones por estudiante.

```
from mrjob.job import MRJob

class AverageGrade(MRJob):
    def mapper(self, _, line):
        student, grade = line.split(',')
        yield student, float(grade)

    def reducer(self, student, grades):
        grades_list = list(grades)
        yield student, sum(grades_list) / len(grades_list)

if __name__ == "__main__":
    AverageGrade.run()
```

**Explicación**

- **Mapper:** divide la línea en estudiante, calificación y emite el estudiante con su calificación.
- **Reducer:** calcula el promedio de calificaciones por estudiante.

**Ejemplo de entrada:**

Ana,85 Carlos,90 Ana,95 Carlos,80

```
"Ana"   90.0
"Carlos"  85.0
```

PUCE Virtual

**La excelencia no se improvisa**

síguenos