

Big Data & Analytics, un enfoque práctico

Bertha Eugenia Mazón Olivo
Wilmer Braulio Rivas Asanza
Joffre Jeorwin Cartuche Calva
Ashley Paulina Aguilar Serrano
Jean Carlos Ávila Villaprado

Big Data & Analytics, un enfoque práctico

Bertha Eugenia Mazón Olivo
Wilmer Braulio Rivas Asanza
Joffre Jeorwin Cartuche Calva
Ashley Paulina Aguilar Serrano
Jean Carlos Ávila Villaprado



© **Bertha Eugenia Mazón Olivo**
Wilmer Braulio Rivas Asanza
Joffre Jeorwin Cartuche Calva
Ashley Paulina Aguilar Serrano
Jean Carlos Ávila Villaprado

© Editorial Grupo Compás, 2025
Guayaquí, Ecuador
www.grupocompas.com
<http://repositorio.grupocompas.com>

Primera edición, 2025-10-22

ISBN: 978-9942-33-954-6

DOI: <http://doi.org/10.48190/9789942339546>

Distribución online

 Acceso abierto

Cita

Mazón, B., Rivas, W., Cartuche, J., Aguilar, A., Ávila, J. (2025) Big Data & Analytics, un enfoque práctico. Editorial Grupo Compás

Este libro es parte de la colección de la Univesidad Técnica de Machala y ha sido debidamente examinado y valorado en la modalidad doble par ciego con fin de garantizar la calidad de la publicación. El copyright estimula la creatividad, defiende la diversidad en el ámbito de las ideas y el conocimiento, promueve la libre expresión y favorece una cultura viva. Quedan rigurosamente prohibidas, bajo las sanciones en las leyes, la producción o almacenamiento total o parcial de la presente publicación, incluyendo el diseño de la portada, así como la transmisión de la misma por cualquiera de sus medios, tanto si es electrónico, como químico, mecánico, óptico, de grabación o bien de fotocopia, sin la autorización de los titulares del copyright.

Prefacio

La revolución digital ha traído consigo una explosión en la generación de datos. Desde dispositivos móviles hasta sensores industriales, redes sociales y sistemas de información empresariales, hoy en día se produce una gran cantidad de datos que supera con creces la capacidad de procesamiento de los métodos tradicionales. Este nuevo escenario ha dado lugar al concepto de Big Data, que no solo se refiere al volumen de datos, sino también a la velocidad con la que se generan, su variedad, veracidad y valor. Para enfrentar este desafío, han emergido tecnologías como Hadoop, MapReduce, Apache Spark y lenguajes como Python, capaces de procesar y analizar información masiva de forma distribuida, eficiente y en tiempo real.

Este libro, titulado "**Big Data & Analytics, un enfoque práctico**", surge como una guía para quienes desean comprender y dominar los fundamentos y herramientas esenciales del ecosistema Big Data. Está diseñado para estudiantes, docentes, investigadores y profesionales que buscan una formación sólida, tanto teórica como práctica, en tecnologías de procesamiento distribuido de datos.

El objetivo de esta obra es capacitar al lector en la comprensión, instalación, configuración y uso práctico de tecnologías clave de Big Data, como Apache Hadoop, MapReduce y Apache Spark, utilizando Python como lenguaje de programación principal. A través de una secuencia didáctica, el libro permite al lector avanzar desde los fundamentos conceptuales hasta la implementación de soluciones reales en entornos pseudo-distribuidos y distribuidos.

El contenido está organizado en cinco capítulos, estructurados de manera lógica y progresiva para facilitar el aprendizaje:

Capítulo I - Fundamentos de Big Data

Este capítulo introduce los conceptos esenciales de Big Data, definiciones, evolución, las características de Big Data. Se exploran las arquitecturas Big Data, así como las principales plataformas y herramientas, el ecosistema de Hadoop y Spark para el almacenamiento y procesamiento distribuido. También se analizan las fuentes de datos y sus formatos. Además, se analizan aplicaciones en diversos sectores, beneficios, inconvenientes y el papel de Big Data Analytics, la inteligencia de negocios, minería de datos, machine learning y deep learning para extraer conocimiento útil de los datos.

Capítulo II - Instalación en Modo Pseudo-distribuido de Apache Hadoop HDFS, YARN y Spark con Ubuntu Server

Este capítulo describe el paso a paso en la creación de un entorno de pruebas con Hadoop y Spark sobre Ubuntu Server, usando Oracle VM VirtualBox. Se detallan las configuraciones del sistema operativo, instalación del JDK, creación de usuarios con acceso SSH, y la configuración e instalación de Apache Hadoop (HDFS y YARN), Apache Spark y Jupyter Notebook, para la ejecución de código Python orientado al análisis de datos.

Capítulo III - Instalación en Modo Distribuido de Apache Hadoop HDFS, YARN y Spark con CentOS

Este capítulo da un paso adelante al enseñar cómo construir un clúster distribuido de procesamiento de datos. Se explican los requisitos de hardware, la arquitectura del clúster, instalación y configuración de CentOS 8 en VMware Workstation, preparación del nodo maestro y nodos esclavos, instalación de los servicios necesarios (Hadoop, Spark, Anaconda), configuración de SSH, variables de entorno, servicios de YARN, y archivos de configuración distribuidos. Esta sección brinda una visión práctica del funcionamiento real de un clúster de Big Data.

Capítulo IV - Procesamiento de datos distribuidos con MapReduce

Se introducen los fundamentos de MapReduce como modelo de programación para el procesamiento distribuido. Se explica su funcionamiento y cómo se integra dentro de ecosistemas como Hadoop y sistemas de almacenamiento distribuidos. El capítulo incluye ejercicios prácticos en Python que ilustran cómo construir y ejecutar tareas MapReduce para procesamiento masivo de datos.

Capítulo V - Procesamiento y análisis de Big Data con Apache Spark

El capítulo final se enfoca en Apache Spark, su historia, evolución y ventajas frente a MapReduce. Se detalla su funcionamiento interno, modelo de ejecución distribuida y casos de uso en ciencia de datos y análisis en tiempo real. Se explican los componentes de Spark, su uso de RDDs y DataFrames, operaciones sobre los datos, transformaciones, acciones, explicado en el lenguaje Python. La sección culmina con ejercicios prácticos orientados al análisis y procesamiento de datos con Spark.

Para facilitar un aprendizaje efectivo, este libro cuenta con diversas herramientas pedagógicas:

- **Enfoque didáctico progresivo**, que inicia desde los conceptos fundamentales hasta la implementación de entornos distribuidos complejos.

- **Explicaciones paso a paso**, tanto para instalaciones como para configuraciones técnicas, con comandos comentados y explicaciones detalladas.
- **Capturas de pantalla y diagramas**, que acompañan los procedimientos técnicos para facilitar su comprensión.
- **Ejercicios prácticos**, diseñados para aplicar los conceptos adquiridos y resolver problemas concretos.
- **Cobertura multiplataforma (Ubuntu y CentOS)**, lo que permite al lector adaptarse a diferentes entornos según sus necesidades.
- **Aplicación de Python** como lenguaje central para tareas de procesamiento, lo que conecta al lector con el lenguaje más utilizado en ciencia de datos.
- **Elementos gráficos** para enfatizar aspectos de interés para el lector:

Icono	Descripción
	Ícono utilizado para destacar una nota que se debe tener en cuenta.
	Ícono utilizado para destacar datos relevantes e importantes.

Este libro es una guía para el desarrollo de competencias necesarias en proyectos de Big Data en el mundo académico, empresarial o investigativo.

ÍNDICE DE CONTENIDOS

CAPÍTULO I	10
FUNDAMENTOS DE BIG DATA.....	10
INTRODUCCIÓN A BIG DATA.....	11
DEFINICIONES DE BIG DATA	11
EVOLUCIÓN DE BIG DATA	12
CARACTERÍSTICAS DEL BIG DATA (LAS 5 Vs)	13
PROCESOS FUNDAMENTALES DE BIG DATA.....	15
ARQUITECTURAS DE BIG DATA	17
ARQUITECTURA LAMBDA	17
ARQUITECTURA KAPPA	19
ARQUITECTURA DE BIG DATA PROPUESTA POR NIST	20
PLATAFORMAS Y HERRAMIENTAS DE BIG DATA.....	22
PLATAFORMA DE APACHE HADOOP.....	22
LOS DATA LAKES (LAGO DE DATOS) DE BIG DATA.....	23
HERRAMIENTAS DE BIG DATA	24
FACTORES DE CLASIFICACIÓN DE BIG DATA	28
FUENTES DE DATOS EN BIG DATA	29
FORMATO DE LOS DATOS	29
ALMACENAMIENTO Y PROCESAMIENTO DE BIG DATA	32
TIPOS DE PROCESAMIENTOS EN BIG DATA	33
APLICACIONES DEL BIG DATA.....	35
IMPORTANCIA, VENTAJAS Y DESAFÍOS DE BIG DATA	36
IMPORTANCIA DE BIG DATA.....	36
BENEFICIOS / VENTAJAS DE BIG DATA / ANALYTICS	36
DESAFÍOS DE BIG DATA	37
BIG DATA ANALYTICS	39
CICLO DE VIDA DE BIG DATA ANALYTICS Y RELACIÓN CON OTRAS DISCIPLINAS	39
INTELIGENCIA DE NEGOCIOS (BI).....	40
MINERÍA DE DATOS (DM: DATA MINING).	42
APRENDIZAJE AUTOMÁTICO O APRENDIZAJE DE MÁQUINAS (ML: MACHINE LEARNING).	43
APRENDIZAJE PROFUNDO (DL: DEEP LEARNING).	43
AUTOEVALUACIÓN	45
REFERENCIAS BIBLIOGRÁFICAS	46

CAPÍTULO II	51
INSTALACIÓN EN MODO PSEUDO-DISTRIBUIDO DE APACHE HADOOP HDFS, YARN Y SPARK CON UBUNTU SERVER	51
MODOS DE INSTALACIÓN DE HADOOP	52
CONFIGURACIÓN DE UBUNTU SERVER EN ORACLE VM VIRTUALBOX .	53
INSTALACIÓN DE SERVICIOS DE HADOOP EN UN ÚNICO NODO (MÁSTER/ESCLAVO).....	55
SISTEMA OPERATIVO UBUNTU SERVER	55
INSTALACIÓN DE VBOXLINUXADDITIONS.....	64
INGRESAR A UBUNTU DESDE EL SISTEMA OPERATIVO ANFITRIÓN (WINDOWS) UTILIZANDO PUTTY.....	65
INSTALACIÓN DE JAVA JDK JAVA 8.....	68
CONFIGURAR UN USUARIO HADOOP CON ACCESO SSH.....	68
INSTALACIÓN DE APACHE HADOOP (HDFS, MAPREDUCE Y YARN).....	69
INSTALACIÓN DE APACHE SPARK	76
INSTALACIÓN DE JUPYTER-NOTEBOOK.....	78
AUTOEVALUACIÓN	80
REFERENCIAS BIBLIOGRÁFICAS	82
 CAPÍTULO III	 83
INSTALACIÓN EN MODO DISTRIBUIDO DE APACHE HADOOP HDFS, YARN Y SPARK CON CENTOS	83
REQUISITOS DE INFRAESTRUCTURA PARA UN CLÚSTER	84
REQUISITOS PARA CENTOS 8	85
ARQUITECTURA DE UN CLÚSTER DE BIG DATA	85
DEFINICIÓN DE CLÚSTER	85
DISEÑO DE UN CLÚSTER HADOOP	86
SERVICIOS QUE SE IMPLEMENTAN EN EL CLÚSTER	87
INSTALACIÓN Y CONFIGURACIÓN DE VMWARE WORKSTATION.....	89
INSTALACIÓN Y CONFIGURACIÓN DEL NODO MÁSTER	96
INSTALACIÓN DEL SISTEMA OPERATIVO CENTOS 8.....	96
INSTALACIÓN Y CONFIGURACIÓN DE HADOOP	103
INSTALAR JDK DE JAVA 8.....	105
VARIABLES DE ENTORNO DE HADOOP	106
SERVICIO SSH	108
MONTAJE DE CLÚSTER PSEUDO DISTRIBUIDO.....	109
SERVICIO YARN	112

CONFIGURACIÓN DE NODOS DE DATOS	118
CREACIÓN DE NODOS DE DATOS	118
CONFIGURACIÓN INICIAL DE NODOS.....	120
CONTINUAR CON LA CONFIGURACIÓN EN NODO MÁSTER.....	123
SERVICIO SSH.....	123
ARCHIVOS DE CONFIGURACIÓN	126
INSTALACIÓN APACHE SPARK	131
INSTALACIÓN ANACONDA.....	134
AUTOEVALUACIÓN	142
REFERENCIAS BIBLIOGRÁFICAS	144
 CAPÍTULO IV.....	 146
PROCESAMIENTO DE DATOS DISTRIBUIDOS CON MAPREDUCE.....	146
FUNDAMENTOS DE MAPREDUCE.....	147
¿QUÉ ES MAPREDUCE?	147
ARQUITECTURA FUNCIONAL DE MAPREDUCE	148
FUNCIONAMIENTO DE MAPREDUCE.....	149
INTEGRACIÓN DE MAPREDUCE EN HDFS (SISTEMAS DE ALMACENAMIENTO DISTRIBUIDO DE HADOOP).	150
ROLES DE LOS NODOS EN HADOOP HDFS Y MAPREDUCE	151
LEVANTAMIENTO DE SERVICIOS.....	152
EJERCICIOS PRÁCTICOS UTILIZANDO MAPREDUCE	152
EJERCICIOS PROPUESTOS	173
REFERENCIAS BIBLIOGRÁFICAS	174
 CAPÍTULO V	 175
PROCESAMIENTO Y ANÁLISIS DE BIG DATA CON APACHE SPARK	175
FUNDAMENTOS DE APACHE SPARK	176
¿QUÉ ES APACHE SPARK?	176
HISTORIA Y EVOLUCIÓN DE SPARK.....	177
MAPREDUCE VS SPARK	177
¿CÓMO FUNCIONA APACHE SPARK?	178
TAREAS EN CIENCIAS DE DATOS	178
APLICACIONES DE SPARK EN PROCESAMIENTO DE DATOS	179
COMPONENTES DE APACHE SPARK.....	179
LENGUAJES DE PROGRAMACIÓN COMPATIBLES CON SPARK.....	181
MODELO DE EJECUCIÓN DISTRIBUIDA	182

MODELOS DE DATOS EN SPARK	183
CONJUNTOS DE DATOS RESILIENTES (RDDs)	184
DATAFRAMES	184
DATASET.....	185
OPERACIONES EN RDDS.....	185
TRANSFORMACIONES	185
ACCIONES	188
EJERCICIOS PRÁCTICOS	191
REFERENCIAS BIBLIOGRÁFICAS	204

Capítulo I

Fundamentos de Big Data

- Introducción a Big Data
- Características de Big Data
- Arquitectura y Tecnologías de Big Data
- Fuentes de datos de Big Data
- Importancia y aplicaciones de Big Data y Big Data Analytics

Objetivos:

- Definir Big Data y su importancia en la actualidad.
- Explicar las características y arquitecturas del Big Data.
- Identificar las principales fuentes de datos en el contexto del Big Data.
- Destacar la importancia y las aplicaciones de grandes volúmenes de datos y el proceso de análisis para obtener conocimiento útil

En un mundo donde la información crece en cantidades inimaginables cada segundo, el término de Big Data pasó a ser un concepto técnico a una herramienta fundamental para la transformación digital. El impacto que tiene se extiende a varios sectores como medicina comercio educación e incluso entretenimiento. Pero ¿Qué es Big Data en sí? No es solo almacenamiento de información, si no el poder convertir datos en un conocimiento valioso para la toma de decisiones estratégicas. En este capítulo vamos a explorar los fundamentos de Big Data, su evolución, sus características esenciales y su funcionamiento en la vida cotidiana, mostrando cómo la tecnología con sus avances ha cambiado la forma en la que se interactúa con el mundo.

El auge del Big Data no constituye una simple moda tecnológica, sino una auténtica revolución en la forma en que las organizaciones gestionan y aprovechan la información. Gracias a su capacidad para identificar patrones ocultos en grandes volúmenes de datos, hoy es posible predecir tendencias, optimizar la eficiencia operativa y personalizar experiencias en diversos sectores. Sin embargo, este crecimiento exponencial en la generación de datos también plantea importantes desafíos, como la garantía de la seguridad y privacidad de la información, la dependencia de herramientas especializadas y la necesidad de contar con profesionales altamente capacitados. Comprender estos fundamentos es esencial para aprovechar todo el potencial transformador de Big Data en la toma de decisiones inteligentes en esta nueva era digital.

La Figura 1 presenta un mapa conceptual con los temas principales desarrollados en este capítulo, facilitando una comprensión clara y

estructurada de los conceptos fundamentales de *Big Data* y su impacto en la sociedad actual.

«La información es el aceite del Siglo XXI, y la analítica es el motor de combustión.» - Peter Sondergaard



Figura 1. Resumen del Capítulo I

Introducción a Big Data

Definiciones de Big Data

El término de Big Data comenzó a adquirir relevancia social a partir de la década de 1990, coincidiendo con la aparición del primer navegador web y la expansión del acceso a Internet a nivel global. Este contexto marcó el inicio de una era en la que cualquier individuo, organización o dispositivo podía generar, compartir y almacenar grandes cantidades de datos en una red interconectada a escala mundial (Dynamic, 2020). El crecimiento de los datos se ha incrementado de manera exponencial gracias al desarrollo de tecnologías como el Internet de las Cosas y la Inteligencia Artificial.

Definiciones de varios autores se presentan a continuación:

- Big Data hace referencia a grandes colecciones de datos que pueden crecer en gran volumen y a un ritmo muy alto, que es difícil manejar con sistemas de bases de datos estándares o herramientas de análisis tradicionales, dentro de un tiempo tolerable (Chen et al., 2014).

- Para las empresas, Big data es un activo importante; debido a que permite extraer conocimiento mediante el cual se obtiene ventajas competitivas, en el mundo actual, digitalizado y globalizado (Chen et al., 2014).
- Big Data, o macrodatos, se refiere al conjunto de datos caracterizados por su gran volumen, alta velocidad de generación y amplia variedad de formatos. Estos datos, provenientes de diversas fuentes como redes sociales, sensores, dispositivos móviles o transacciones digitales, requieren infraestructuras de almacenamiento escalables y métodos de procesamiento avanzados que superen las capacidades de las herramientas tradicionales. El objetivo del uso de Big Data es optimizar el análisis, mejorar la comprensión de fenómenos complejos y fortalecer la toma de decisiones estratégicas en tiempo real. Debido a estas propiedades, su gestión y análisis representan un desafío técnico considerable para los enfoques convencionales (Mazón-Olivo, 2023; Moreno-Carriles, 2018).
- Chen (2024) manifiesta que Big Data funciona proporcionando información que arroja luz sobre nuevas oportunidades y modelos de negocio.

Evolución de Big Data

La evolución del Big Data ha sido un proceso progresivo marcado por hitos clave en el manejo de grandes volúmenes de información. Ya en el siglo XIX, el censo de EE.UU. de 1880 evidenció los primeros desafíos al tardar ocho años en procesarse (Wu et al., 2016). En 1997, investigadores de la NASA identificaron formalmente el "problema de Big Data", anticipando el crecimiento desbordado de los datos (Wu et al., 2016). En 2001, Laney de la empresa Garner propuso el modelo de las 3Vs (Volumen, Velocidad y Variedad), que aún define su esencia. En el 2003, el avance tecnológico se consolidó con la publicación del Google File System y MapReduce, seguidos por la creación de BigTable en el 2004 y Hadoop en el 2006, que permitieron el procesamiento distribuido de datos masivos (Wu et al., 2016). Posteriormente, se desarrollaron sistemas de bases de datos NoSQL como *MongoDB*, *HBase* y *Cassandra*, diseñados para gestionar datos no estructurados y escalar horizontalmente. En la siguiente etapa, aparecieron herramientas más avanzadas como *Apache Spark* en el 2010, que ofreció procesamiento en memoria y una mejora significativa en velocidad y eficiencia frente a Hadoop MapReduce. Otras soluciones como *Apache Flink*, *Drill* y *Kafka* ampliaron aún más el ecosistema, proporcionando capacidades para procesamiento de flujos en tiempo real, consultas interactivas y transmisión de datos (Reinsel et al., 2018). Actualmente, vivimos en la era de los zettabytes, con proyecciones que superan los 175 ZB para 2025 (Reinsel et al., 2018).

La Figura 2, presenta una línea de tiempo con algunos hitos importantes sobre la evolución de Big Data.

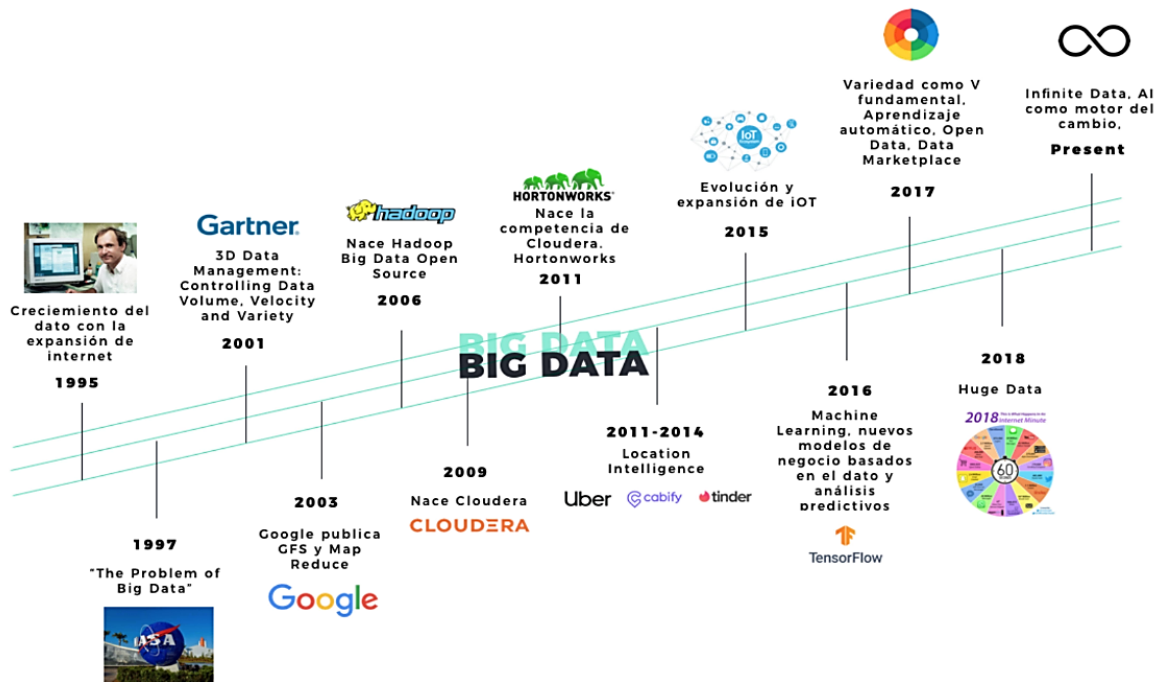


Figura 2. Evolución de Big Data

Fuente: (Raya, 2019)

Características del Big Data (Las 5 Vs)

En (Wu et al., 2016), se menciona que Doug Laney de Garner (2001), destacó el desafío de la gestión de los datos de gran volumen y presentó por primera vez el llamado modelo de 3 V's: *volumen*, *velocidad* y *variedad* (Figura 3), siendo estas características las principales para comprender y sacar el máximo provecho de los datos.



Figura 3. Principales 3V de Big Data

Fuente: Una adaptación de (Wu et al., 2016)

Posteriormente se proponen las 5Vs del Big Data (Tabla 1), que incluye las características de *valor* y *veracidad* de los datos (Wu et al., 2016).

Tabla 1. Descripción de las 5V o características de Big Data

Característica	Descripción
Volumen	Gran cantidad de datos se producen en muy poco tiempo, generados por: redes sociales, objetos inteligentes, aplicaciones móviles, navegación cotidiana por Internet, transacciones empresariales, etc.
Velocidad	Velocidad o rapidez con lo que se generan los datos (frecuencia). Puede ser por lotes, en tiempo real (RT) o cercano a RT.
Variedad	Los datos Estructurados, No Estructurados y semiestructurados. De múltiples fuentes.
Valor	Qué parte de los datos son valiosos o útiles para ayudar en las decisiones de la empresa.
Veracidad	Tiene que ver con la calidad de los datos. No todos los datos son confiables, es necesario determinar errores y anomalías en los datos y darles el respectivo tratamiento.

Fuente: (Bourany, 2018)

La enorme cantidad de los datos que se generan a diario y su variabilidad, requiere un procesamiento a alta velocidad, para optimizar tiempos de respuesta y garantizar la calidad del servicio. Además, asegurar la veracidad y confiabilidad de la información es esencial para minimizar riesgos en las transacciones y, en última instancia, convertir esos datos en valor real para la toma de decisiones y la mejora continua (Araque González et al., 2021).

En el trabajo de Raza (2021), se destaca que para aplicar correctamente las 5V de Big Data debemos tener en cuenta los siguientes factores:

- Disponer de una **planificación cuidadosa** y la selección de las tecnologías adecuadas.
- Tener un **entendimiento claro de los objetivos de negocio** y cómo el Big Data puede ayudar a alcanzar esos objetivos.
- También es recomendable **contar con un equipo de profesionales con habilidades** en manejo de datos, análisis y tecnologías relacionadas.
- Mantener una **comunicación constante entre los equipos técnico y de negocio**, y asegurar la conformidad con las regulaciones y estándares de privacidad y seguridad de datos.

Además, en el trabajo de (Wu et al., 2016), se menciona que Microsoft añade una sexta V: *Visibilidad*. Luego se presentan las siete V's: *Volumen, Velocidad, Variedad, Veracidad, Valor, Validez y Volatilidad*. En Wu, et al. (2016), también se plantean las 9 V's organizados en tres dominios, consideradas también

como las **3²V**: **Dominio de datos** (*Volumen, Velocidad y Variedad*), **Dominio estadístico** (*Validez, Veracidad y Variabilidad*), y **Dominio de Inteligencia de Negocios** (*Valor, Veredicto y Volatilidad*).

Procesos fundamentales de Big Data

El procesamiento de Big Data implica una serie de etapas sistemáticas orientadas a extraer valor de grandes volúmenes de datos generados de forma continua, variada y rápida. Estos procesos constituyen el núcleo de cualquier arquitectura de análisis de datos a gran escala y están diseñados para transformar datos brutos (crudos) en información útil y conocimiento accionable.



Figura 4. Procesos de Big Data

Fuente: Una adaptación de (Alwidian et al., 2020; Mazón-Olivo, 2023)

Según (Alwidian et al., 2020; Mazón-Olivo, 2023), los procesos son (Figura 4):

- 1. Adquisición de Datos (Data Ingestion).** Es el proceso inicial en el ecosistema de Big Data. Consiste en la recolección de datos desde diversas fuentes heterogéneas, tanto estructuradas como no estructuradas. Estas fuentes pueden incluir sensores IoT, logs de servidores, redes sociales, transacciones empresariales, dispositivos móviles, entre otros. La ingesta puede realizarse en modo batch (por

lotes) o en tiempo real (streaming), empleando herramientas como Apache Kafka, Flume, NiFi, entre otras.

2. Almacenamiento de Datos (Data Storage). Una vez adquiridos, los datos deben ser almacenados de forma eficiente para su posterior procesamiento. Dado el volumen y la variedad de los datos, las soluciones tradicionales de bases de datos relacionales resultan insuficientes. En su lugar, se utilizan tecnologías distribuidas como Hadoop Distributed File System (HDFS), Amazon S3 o sistemas de bases de datos NoSQL (por ejemplo, Cassandra, MongoDB o HBase), que permiten escalabilidad horizontal, tolerancia a fallos y acceso paralelo.

3. Procesamiento y Transformación (Data Processing and Transformation). Este proceso tiene como objetivo limpiar, integrar, transformar y preparar los datos para análisis. El procesamiento puede clasificarse en dos tipos:

- **Batch Processing:** Utiliza marcos como Apache Hadoop MapReduce o Apache Spark para procesar grandes volúmenes de datos acumulados.
- **Stream Processing:** Procesa flujos de datos en tiempo real usando herramientas como Apache Storm, Apache Flink o Spark Streaming. Durante esta fase se aplican técnicas como normalización, detección de valores atípicos, integración de fuentes y transformación de formatos.

4. Análisis de Datos (Data Analysis). El análisis de datos busca descubrir patrones, correlaciones y tendencias mediante técnicas estadísticas, minería de datos y algoritmos de aprendizaje automático. Esta fase se apoya en frameworks como Apache Spark MLlib, TensorFlow o scikit-learn, y puede incluir desde análisis descriptivos hasta modelos predictivos y prescriptivos. La analítica avanzada permite tomar decisiones fundamentadas y automatizar procesos críticos en distintos dominios como finanzas, salud, agricultura, manufactura y más.

5. Visualización de Datos (Data Visualization). La visualización es esencial para interpretar los resultados analíticos y facilitar la toma de decisiones. Consiste en aplicar Inteligencia de Negocios para representar gráficamente los datos procesados mediante dashboards, gráficos interactivos o mapas, con herramientas como Tableau, Power BI, Apache Superset, D3.js, etc. Una visualización eficaz permite a los usuarios finales detectar insights de manera intuitiva y comunicar hallazgos a actores no técnicos.

6. Gobierno y Seguridad de los Datos (Data Governance and Security). Este proceso transversal asegura la calidad, integridad, privacidad y seguridad de los datos en todo su ciclo de vida. Incluye políticas de gestión de metadatos, control de acceso, cumplimiento normativo, por

ejemplo: GDPR (General Data Protection Regulation), trazabilidad y auditoría. Asimismo, involucra mecanismos criptográficos, autenticación, y gestión de identidades para proteger la confidencialidad y disponibilidad de la información.

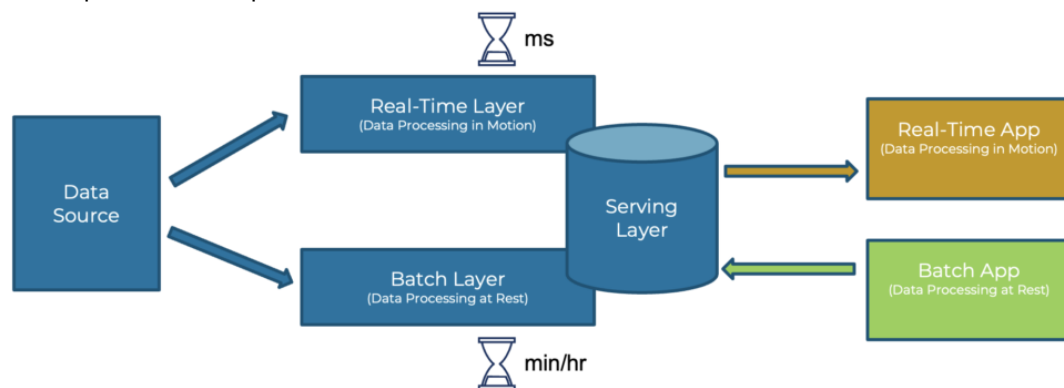
Arquitecturas de Big Data

En el contexto del procesamiento de grandes volúmenes de datos, las arquitecturas de Big Data han evolucionado para abordar desafíos relacionados con la escalabilidad, la latencia y la complejidad operativa. Entre las arquitecturas más destacadas se encuentran Lambda, Kappa, y NIST, cada una con enfoques y aplicaciones específicas.

Arquitectura Lambda

Esta arquitectura es propuesta por Nathan Marz para manejar grandes volúmenes de información, combina procesamiento por lotes y en tiempo real mediante tres capas: la capa de batch, la capa de velocidad y la capa de servicio, permitiendo a las empresas y desarrolladores tomar decisiones basadas en datos de manera eficiente (Kalipe & Behera, 2019; Waehner, 2021). Esta estructura permite manejar datos históricos y en tiempo real simultáneamente, aunque introduce complejidad al requerir mantenimiento de dos rutas de procesamiento distintas. En la Figura 5, se presenta dos opciones de esta arquitectura, a) Capa de servicio unificada y b) Capa de servicio separada.

a) Opción 1: Capa de servicio unificada



b) Opción 2: Capa de servicio separada

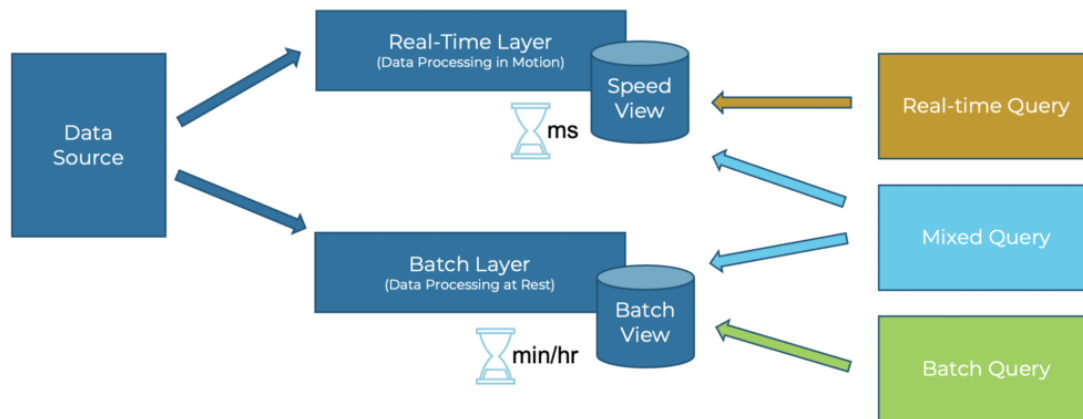


Figura 5. Arquitectura Lambda
Fuente: Tomado de (Waehner, 2021)

En el trabajo de (Waehner, 2021), se resumen los principales aspectos de la Arquitectura Lambda:

- Todos los datos se envían tanto a la capa de lotes como a la de velocidad.
- La capa Batch preprocesa las funciones de consulta desde cero, el resultado se denomina vistas de lote. La capa de lotes recalcula constantemente las vistas por lotes.
- Las vistas por lotes se indexan y almacenan en una base de datos escalable para obtener valores concretos con gran rapidez. Intercambia nuevas vistas por lotes cuando están disponibles.
- La capa de velocidad compensa la alta latencia de las actualizaciones a las vistas por lotes.
- Utiliza algoritmos incrementales rápidos y bases de datos de lectura/escritura para producir vistas en tiempo real
- Las consultas se resuelven obteniendo resultados tanto de las vistas por lotes y en tiempo real.

Los casos de aplicación de este tipo de arquitectura pueden ser varios; por ejemplo, las aplicaciones basadas en eventos requieren un procesamiento continuo y eficiente (en tiempo real o cercano), la supervisión y la notificación de alertas o de la ocurrencia del mismo evento; y, debido a las estrictas restricciones de latencia, no es posible realizar al mismo tiempo el procesamiento por lotes. Un caso puntual son las aplicaciones del Internet de las Cosas (IoT) que monitorizan grandes flujos de datos y eventos en tiempo real. Algunos datos sin procesar se capturan desde dispositivos ubicados en redes de sensores inalámbricos y se utilizan para el procesamiento dinámico de eventos generados en el contexto de uso de IoT, y luego se toma decisiones de control sobre los actuadores; por otro lado, los datos se almacenan en grandes bases de datos para un posterior procesamiento por lotes (Mazon-Olivo, Hernández-Rojas, et al., 2018).

Arquitectura Kappa

Fue propuesta por Jay Kreps como una alternativa a la arquitectura Lambda. Simplifica el procesamiento al eliminar la capa de batch, utilizando una única ruta de procesamiento en tiempo real (procesamiento en streaming). Esto es especialmente útil para aplicaciones que requieren análisis continuo de datos, aunque puede no ser adecuada para todos los casos de uso (Tejada, 2018). Kappa apuesta completamente por el procesamiento de datos en flujo, permitiendo que la información se analice y procese continuamente, sin necesidad de estructuras separadas para datos históricos y en tiempo real. Este enfoque elimina muchas de las complejidades asociadas con la Arquitectura Lambda, como la necesidad de mantener dos sistemas diferentes sincronizados o la duplicación de lógica de procesamiento. En la práctica, esto significa menos mantenimiento, menos margen de error y, en muchos casos, una implementación más eficiente (Figura 6).

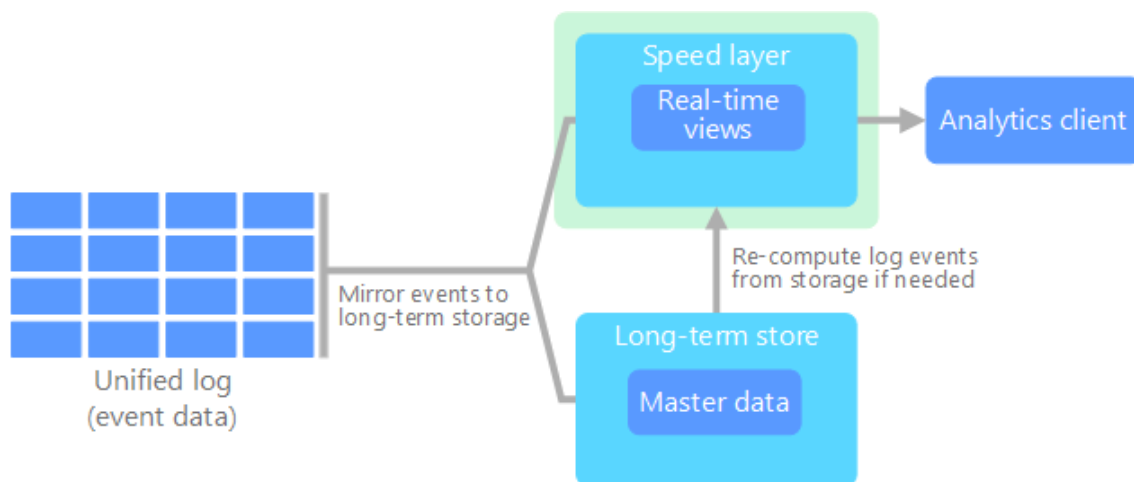


Figura 6. Arquitectura Kappa
Fuente: (Tejada, 2018)

En el trabajo de Careaga (2020), se menciona que la arquitectura Kappa tiene las siguientes ventajas:

- Es una simplificación de la arquitectura Lambda, ya que se suprime el uso de la capa de batch.
- La información es almacenada utilizando un log inmutable de sólo almacenamiento, del cual se envía a almacenamientos auxiliares para la capa de serving.
- La arquitectura Kappa permite la migración y la reorganización de la información a partir de diversas fuentes que proporcionan datos de manera rápida a través de la capa de streaming.
- Dada la ausencia de la capa de batch, sólo un código debe de ser actualizado en caso de necesitar mantenimiento.

- No utiliza un esquema de base de datos relacional o un almacenamiento basado en valores clave, como SQL o Cassandra, respectivamente.

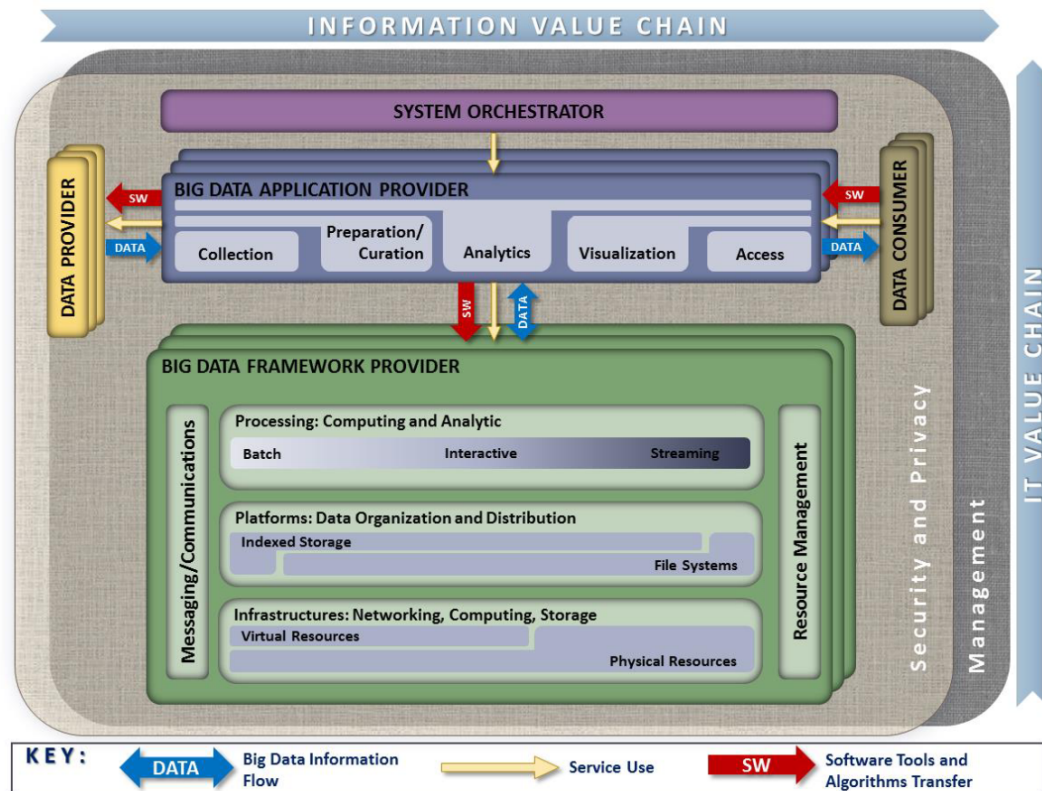
En la arquitectura Kappa, se puede desarrollar, probar, depurar y operar sistemas en un único marco de procesamiento tanto para tiempo real como por lotes. Por ejemplo, Apache Kafka puede utilizarse como un sistema de logs para recibir millones de eventos de datos a grandes velocidades. Los autores Zorrilla & García-Saiz (2017) destacan que la clave del funcionamiento de esta arquitectura radica en el uso de un sistema de mensajería, como Apache Kafka, que actúa como un registro centralizado de eventos. Esto no solo facilita el análisis en tiempo real, sino que también permite reprocesar datos históricos cuando sea necesario.

Arquitectura de Big Data propuesta por NIST

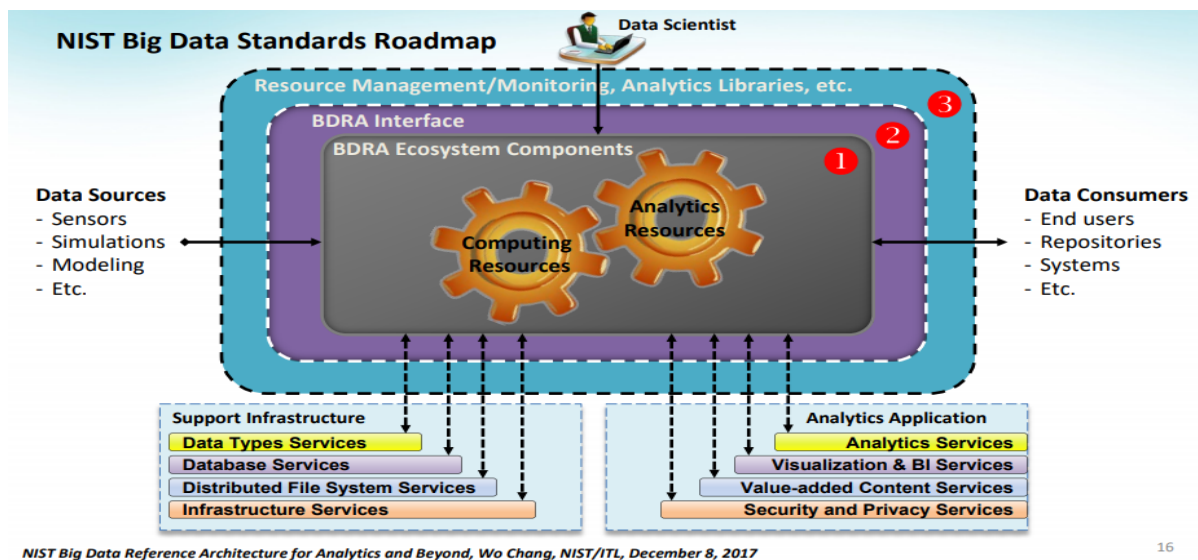
La arquitectura de Big Data propuesta por NIST (National Institute of Standards and Technology) (Chang et al., 2019), como se observa en Figura 7, a), se organiza en torno a dos cadenas de valor: **de la información** y **de las tecnologías de la información (TI)**. Su diseño modular comprende roles funcionales interrelacionados que permiten gestionar el ciclo completo de los datos masivos, desde su adquisición hasta su consumo final. En el núcleo es **Big Data Framework Provider**, proporciona la infraestructura, plataformas de almacenamiento y capacidades de procesamiento. Esta capa incluye recursos físicos y virtuales, sistemas de almacenamiento distribuido, así como motores para procesamiento *batch*, interactivo o *streaming*. En la parte superior opera **Big Data Application Provider**, que gestiona el ciclo de vida de los datos, desde la recolección, preparación y análisis, hasta la visualización y el acceso. Estas aplicaciones interactúan con el framework, mediante algoritmos, datos y servicios según las necesidades del usuario final. El **Data Provider** suministra los datos crudos al sistema, mientras que el **Data Consumer** recibe los resultados procesados o visualizados. Por su parte, el **System Orchestrator** coordina todos estos componentes para asegurar un funcionamiento eficiente y coherente. Finalmente, el modelo incluye una capa transversal de **gestión de seguridad y privacidad**, crucial para proteger los datos y asegurar su uso ético y legal. Este enfoque modular y escalable facilita la interoperabilidad, reutilización de componentes y gobernanza eficiente del ecosistema Big Data, alineándose con principios de transparencia, rendimiento y protección de datos.

En la Figura 7 b), NIST establece un marco para el diseño, operación y gestión de sistemas de análisis de datos masivos, integrando componentes clave que garantizan interoperabilidad, escalabilidad y eficiencia. En el núcleo del ecosistema BDRA (Big Data Reference Architecture) se encuentran los

recursos de cómputo y los **recursos analíticos**, responsables de ejecutar algoritmos complejos y procesar grandes volúmenes de datos. Estos elementos interactúan a través de una interfaz estandarizada que habilita la comunicación con fuentes de datos diversas, tales como sensores, simulaciones y modelos, así como con los consumidores de datos finales, como usuarios, sistemas o repositorios (Chang et al., 2019).



a) Arquitectura de Referencia de Big Data propuesta por NIST



b) Arquitectura de Referencia para Big Data Analytics propuesta por NIST

Figura 7. Arquitectura Big Data propuesta por NIST

Fuente: (Chang et al., 2019)

La arquitectura se apoya en una **infraestructura de soporte** que proporciona servicios esenciales como gestión de bases de datos, almacenamiento distribuido, tipos de datos e infraestructura computacional. Además, incluye una capa de **aplicaciones analíticas** que incorpora servicios de análisis, visualización, seguridad y generación de contenido de valor agregado. En conjunto, este modelo también contempla una capa superior orientada a la **gestión de recursos, monitoreo y bibliotecas analíticas**, permitiendo a los científicos de datos operar el sistema de manera eficiente. Esta arquitectura ofrece una guía robusta para desarrollar ecosistemas de Big Data que respondan a los retos actuales de volumen, velocidad, variedad y veracidad de la información en contextos organizacionales y científicos.

Plataformas y Herramientas de Big Data

Existen una variedad de plataformas y herramientas para el manejo de Big Data que se emplean para distinto propósito, ya sea como soluciones individuales o como parte de suites ofrecidas por proveedores de software, tanto propietario como de código abierto. Debido a la complejidad de estas herramientas, han surgido diversas plataformas diseñadas para facilitar su implementación; algunas de estas plataformas están vinculadas a infraestructuras de hardware y pueden utilizarse en todo el ciclo de vida del Big Data o en etapas específicas según las necesidades del análisis y procesamiento de datos.

Una plataforma de Big data incluye hardware, middleware y aplicaciones de ciencia de datos (estadística, inteligencia de negocios, minería de datos, aprendizaje automático y aprendizaje profundo), para el análisis eficiente de los datos de forma escalable (Wilcox et al., 2019). Las plataformas de Big Data en función de sus prestaciones pueden ser (Țăranu, 2015):

- **De alto rendimiento**, tal es el caso de herramientas como: Teradata, Vertica, IBM Netezza, Oracle Exadata, etc.
- **De bajo coste**, están las basadas en Apache Hadoop. Uno de los proveedores es Cloudera Data Platform (CDP) que incluye HDFS, YARN, Hive, Impala, Spark, HBase y más.
- **Como servicio en la nube**, por ejemplo: AWS - Amazon Web Services (S3, Glue, Elastic MapReduce, etc), Microsoft Azure (HDInsight, etc), Google Cloud Platform (BigQuery, Cloud Storage, etc.), Cloudera, DataBricks, MapR, IBM Analytics Platform, etc.

Plataforma de Apache Hadoop

Apache Hadoop de Apache Software Foundation, es un marco de trabajo distribuido, de código abierto y de bajo coste, que permite el procesamiento por lotes de grandes volúmenes de datos a través del sistema HDFS (Hadoop

Distributed File System) y el motor MapReduce. Hadoop está diseñado para el almacenamiento y computación distribuida, escalable, tolerante a fallos y por tanto es fiable para el manejo de grandes conjuntos de datos a través de clústeres de computadoras (Apache Software Foundation, 2025). Hadoop maneja un entorno de tipo High Performance Super Computer que es escalable de manera horizontal con hardware “barato”. Su implementación es a través del procesamiento en paralelo a través de varios equipos en un sistema de archivos distribuidos. Hadoop consta de una serie de librerías de código abierto para los sistemas distribuidos usando varios módulos, las cuales son (Apache Software Foundation, 2025):

- **Hadoop Common:** Son las utilidades y librerías comunes que dan soporte a otros componentes de Hadoop.
- **Hadoop Distributed File System HDFS:** Es el sistema de ficheros distribuido de Hadoop que proporciona un acceso de alto rendimiento a los datos de las aplicaciones.
- **Hadoop YARN:** Es el gestor de recursos y trabajos del clúster de Hadoop.
- **Hadoop MapReduce:** Es un sistema basado en YARN para el procesamiento en paralelo de grandes volúmenes de datos que están distribuidos en HDFS.

Los Data Lakes (Lago de datos) de Big data

Un lago de datos es un repositorio centralizado que permite almacenar grandes volúmenes de datos en su formato original, sin necesidad de estructurarlos previamente. A diferencia de los data warehouses, que requieren una estructura tabular y un esquema definido (modelo schema-on-write), los lagos de datos operan bajo el modelo schema-on-read, lo que significa que los datos se estructuran solo cuando se necesitan para análisis específicos. Para Lo Giudice et al. (2019), un Data Lake es “un conjunto de repositorios centralizados de cantidades masivas de datos sin procesar, en variedad de formatos y con descripciones de metadatos, que proporcionan funcionalidad para la integración bajo demanda”. Para Quix et al. (2016), un lago de datos proporciona un almacenamiento escalable e incorpora aplicaciones para realizar tareas como: reportes, visualizaciones, análisis de datos, etc. Un lago de datos en local se construye mediante un clúster de Hadoop (HDFS + YARN); mientras que en la nube tenemos (Mazón-Olivo, 2023): Microsoft Azure Data Lake, AWS S3 Data Lake, Databricks Lakehouse (Delta Lake), Intelligent Data Lake, Infor Data Lake, IBM Watsonx.data, etc.

Herramientas de Big Data

Existen una variedad de herramientas para la gestión de Big Data y Big Data Analytics, como se explica en (Alwidian et al., 2020; Mazón-Olivo, 2023; Oussous et al., 2017; Saggi & Jain, 2018).

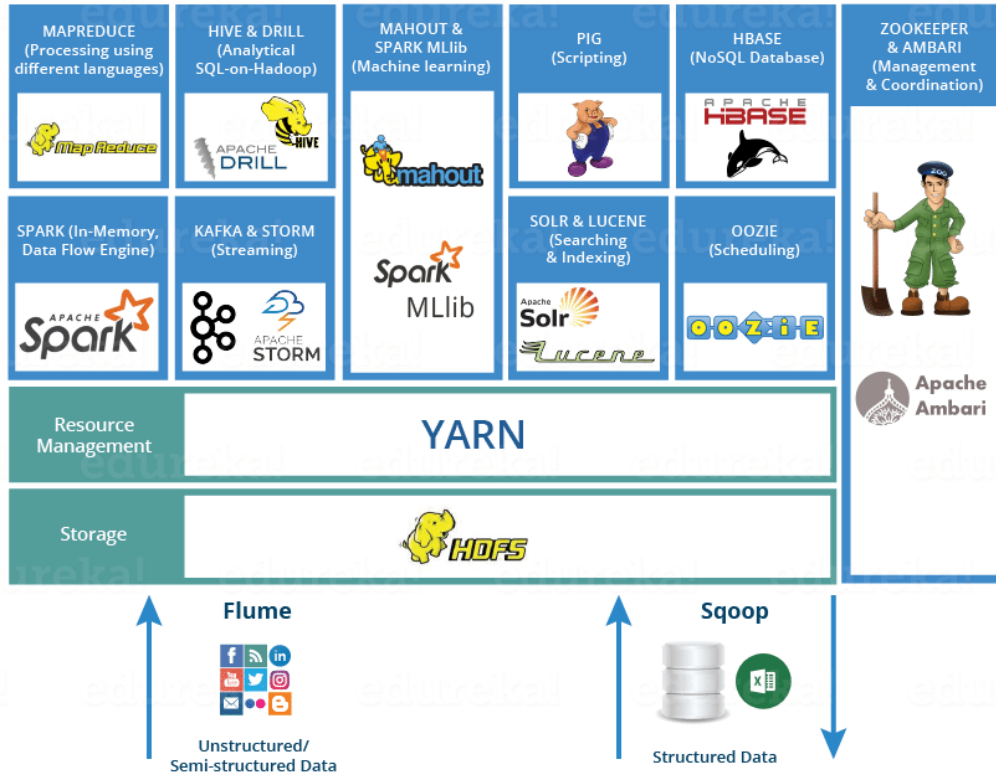


Figura 8. Ecosistema de herramientas de Hadoop

Fuente: (Mazón-Olivo, 2023; Sinha, 2020)

Tabla 2. Herramientas de Big Data

Herramientas	Descripción
 HDFS https://hadoop.apache.org/	- Hadoop Distributed File System (HDFS) es un Framework de código abierto, escalable y tolerante a fallos, escrito en Java. - Almacena y procesa grandes conjuntos de datos. - Diseñado para funcionar en hardware de bajo costo. Con una arquitectura maestro/esclavo que consta de dos componentes: NameNode y varios DataNodes. - NameNode es el servidor maestro que regula el acceso y gestiona los archivos por parte de los clientes. - DataNodes son responsables de administrar el almacenamiento de datos. Internamente, un archivo se divide en uno o más bloques y se distribuyen en los DataNodes.

Herramientas	Descripción
 https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html	- Yet Another Resource Negotiator (YARN): Es el cerebro de Hadoop, administra recursos de clústers. - Es considerado un sistema operativo distribuido para aplicaciones de big data. YARN divide un JobTracker en procesos: Resource Manager para la gestión propia de recursos en múltiples nodos; y Application Master, el cual es el responsable de negociar los recursos con el Resource Manager.
 https://hadoop.apache.org/docs/current/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html	- Framework de procesamiento distribuido. - Utiliza diferentes lenguajes de programación para los algoritmos de procesamiento como Python o Java. - Tiene las funciones: Map, Combine (opcional) y Reduce. - Map, es para realizar acciones de filtrado, agrupación y clasificación y genera un resultado por clave-valor. - Reduce, agrega y resume el resultado producido por la función Map.
 https://spark.apache.org/	- Sistema de computación distribuida, mediante clústeres de ordenadores para analizar y procesar datos masivos en memoria. - Proporciona APIs en Java, Scala, Python y R - Funcionalidades: DAG (Directed Acyclic Graph) y RDD (Resilient Distributed Dataset) - Componentes: - Spark SQL para procesamiento de datos estructurados, - MLlib para aprendizaje automático. - GraphX para procesamiento de gráficos. - Spark Streaming para procesamiento en real-time
 https://pig.apache.org/	- Apache Pig es un servicio de análisis y procesamiento de datos en paralelo y por lotes. - Dispone de un modelo de datos llamado Map Data - Utiliza un lenguaje de script llamado Pig Latin, de alto nivel que se usa con Hadoop para escribir transformaciones de datos sin conocer Java.
 https://hive.apache.org/	- Hive es un componente de almacén de datos (data warehouse) distribuido con interfaz SQL denominada HiveQL.

Herramientas	Descripción
	- Realiza la gestión de grandes conjuntos de datos en un entorno distribuido mediante una interfaz similar a SQL. - Permite la ejecución de consultas a través de Apache Tez, Apache Spark o MapReduce
 https://hbase.apache.org/	- Hbase es una base de datos NoSQL y un almacén de datos en columnas. - Es de código abierto, distribuida, no relacional, se ejecuta sobre HDFS y es tolerante a fallos.
 https://mahout.apache.org/	- Mahout es una herramienta de aprendizaje automático similar a Spark MLlib. - Permite implementar algoritmos de aprendizaje automático distribuidos o escalables enfocados en áreas de filtrado colaborativo, agrupación y clasificación.
 https://drill.apache.org/	- Drill es una herramienta para consultar almacenes de datos no relacionales utilizando SQL estándar. - Es un motor de consulta libre de esquemas, es de baja latencia - Utiliza un formato JSON para consultar datos de cualquier estructura. - Es compatible con una variedad de bases de datos y sistemas de archivos NoSQL, como HBase, MapR-DB, HDFS, MongoDB, Amazon S3, MapR-FS, etc.
 https://oozie.apache.org/	- Es un sistema de planificador de flujos de trabajo para administrar trabajos de Apache Hadoop. - Está integrado con el resto de la pila Hadoop que admite varios tipos de trabajos, como Java map-reduce, Pig, Hive, Sqoop, etc., así como trabajos específicos del sistema (como programas Java y scripts de shell).
 https://flume.apache.org/	- Es un sistema distribuido y confiable para capturar datos no estructurados y semiestructurados y cargarlos en HDFS. - Es una herramienta utilizada para recoger, agregar y mover grandes cantidades de datos en logs desde diferentes orígenes a un almacén central. - Transportar gran cantidad de logs/eventos/datos procedentes de varias fuentes como tráfico de red, redes de sensores, redes sociales, mensajes de correo electrónico, archivos de registro, etc. en HDFS.

Herramientas	Descripción
 https://sqoop.apache.org/	- Sqoop es una herramienta para transferir datos masivos entre almacenes de datos estructurados y Apache Hadoop o viceversa.
 http://lucene.apache.org/	- Apache Lucene, es una API basado en Java para búsqueda de texto. - Solr es un servidor de búsqueda de texto completo, creado con Lucene Core, con API XML / HTTP y JSON / Python / Ruby - Además, permite el resaltado de resultados, clustering dinámico, manejo de documentos, realizar búsquedas distribuidas y replicación de índices.
 https://ambari.apache.org/	- Gestiona y supervisa los clústeres de Hadoop. - Incluye software para aprovisionamiento, administración y monitoreo de clústeres de Hadoop.
 https://zookeeper.apache.org/	- Zookeeper permite la coordinación y gestión de clústeres. - Servicio de coordinación de procesos distribuidos y altamente confiables para mantener la información de configuración, asignar nombres, proporcionar sincronización distribuida y brindar servicios grupales.
 https://kafka.apache.org/	- Plataforma de gestión de mensajes (cola de mensajes) desarrollado en Scala. - Se utiliza para flujos de datos en tiempo real (data streaming). - Es una con arquitectura de productores/consumidores. - Tiene varios conectores para la integración con sistema de archivos, Hadoop (HDFS), HBase, JDBC, etc.
 http://storm.apache.org/	- Storm es un sistema de cómputo distribuido en tiempo real. - Se utiliza para recuperar flujos de datos desde múltiples fuentes de manera distribuida. - Es tolerante a fallos y de alta disponibilidad.

Fuente: (Mazón-Olivo, 2023; Sinha, 2020)

El **ecosistema de Hadoop** (Figura 8), comprende un vasto conjunto de herramientas populares para trabajar con distintos procesos de Big Data y BDA. Una explicación breve de cada herramienta se presenta en la Tabla 2.

Factores de clasificación de Big Data

Según IBM, elegir una arquitectura y construir una solución de big data adecuada es un desafío porque se deben considerar muchos aspectos; es por eso que ha realizado una clasificación amplia, según varios factores relacionados y teniendo en cuenta que los datos se almacenan, adquieren, procesan y analizan de muchas maneras. Cada fuente de datos provee datos con características diferentes como: frecuencia, volumen, velocidad, tipo y veracidad. Además, cuando se procesan y almacena los datos en Big data, entran en juego dimensiones adicionales, como gobernanza, seguridad y privacidad. En la Figura 9, se representan algunos de los principales factores de Big data (Mazón-Olivo, 2023; Oussous et al., 2017), como son: fuentes de datos, formato del contenido, tipo de dato según el proceso, frecuencia de los datos, tipos de procesamiento, tipos de análisis y consumidores de datos.

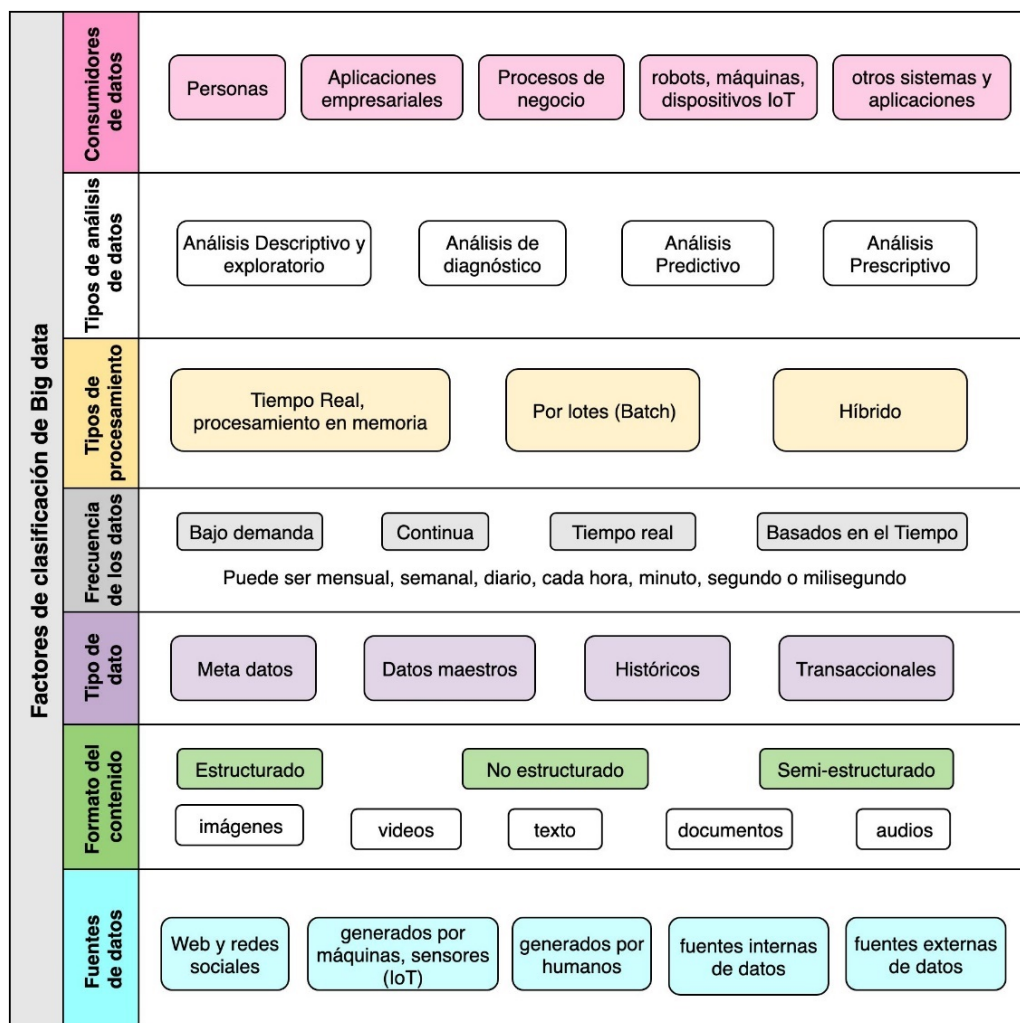


Figura 9. Clasificación de los datos en Big Data
Fuente: (Mazón-Olivo, 2023)

Fuentes de datos en Big Data

Las fuentes de Big Data son diversas y abarcan múltiples dominios, caracterizándose por generar datos en grandes volúmenes, de forma continua y con distintos niveles de estructura. Para (Jabbar et al., 2019; Mazón-Olivo, 2023), las principales fuentes que generan big data son las **organizaciones**, **personas** y **máquinas** (Figura 10). Entre fuentes las más relevantes se encuentran las máquinas como los dispositivos IoT (Internet of Things), sensores y actuadores, cámaras, objetos inteligentes, dispositivos móviles y aplicaciones de Inteligencia Artificial, ya que producen flujos de datos en tiempo real. También destacan las interacciones que realizan las personas en redes sociales y plataformas digitales, donde generan contenidos multimedia, textos, reacciones y comportamientos de navegación que son valiosos para el análisis de tendencias y preferencias. Otra fuente clave de datos son las organizaciones mediante los sistemas transaccionales empresariales, como registros de ventas, operaciones bancarias, logística o e-commerce, los cuales generan datos estructurados críticos para la toma de decisiones; asimismo, los registros de salud electrónicos, las bases de datos científicas, las imágenes satelitales y los datos gubernamentales abiertos contribuyen significativamente al ecosistema de Big Data (Jabbar et al., 2019; Mazón-Olivo, 2023). Estas fuentes alimentan entornos analíticos complejos donde los datos, estructurados o no, requieren ser integrados, procesados y analizados para extraer conocimiento útil.



Figura 10. Principales fuentes de Big Data
Fuente: (Mazón-Olivo, 2023)

Formato de los datos

En Big Data, los datos pueden presentarse en diversos formatos, los cuales se clasifican en: estructurados, semiestructurados, no estructurados, binarios y

formatos optimizados para Big Data. Una comparación de estos tipos de datos se presenta en la Tabla 3.

Datos Estructurados

Los datos estructurados representan información organizada de forma tabular, compuesta por filas y columnas con tipos de datos definidos. Son gestionados mediante sistemas de bases de datos relacionales (RDBMS) y se accede a ellos a través del lenguaje SQL; también se clasifican en esta categoría los archivos CSV y hojas de cálculos como Excel. Este tipo de formato es común en aplicaciones transaccionales y operativas, como registros financieros, sistemas ERP o catálogos de productos. Su principal ventaja radica en su alta eficiencia de consulta y análisis cuando los datos siguen un esquema fijo. Sin embargo, en contextos de Big Data, su rigidez limita la capacidad de capturar información heterogénea proveniente de múltiples fuentes dinámicas (Chang et al., 2019).

Tabla 3. Comparativa de tipos de formatos de datos en Big Data

Tipo de dato	Estructura	Ejemplos	Ventajas	Aplicaciones en Big Data
Estructurado	Tabular (filas y columnas)	CSV, SQL, Excel, MySQL, Postgres, etc.	Alta eficiencia en consultas, integración con RDBMS	Análisis transaccional, dashboards BI, reportes empresariales
Semi-estructurado	Jerárquica / flexible	JSON, XML, Avro, YAML	Flexibilidad de esquema, ideal para transmisión de datos	APIs, logs de aplicaciones, bases NoSQL, flujos Kafka, intercambio o transferencia de datos
No estructurado	Libre / sin esquema fijo	Texto, imágenes, videos, audio	Información rica y contextual, apto para IA y análisis profundo	PLN, Deep Learning, análisis multimedia, minería de texto, análisis de redes sociales
Binario	Bajo nivel (bytes)	.jpg, .mp4, .zip, .pkl	Almacenamiento compacto de contenidos complejos	Modelos de ML, contenido multimedia, backups, análisis offline
Columnar / Serializado	Columnar o binario optimizado	Parquet, ORC, Avro, Protobuf, Delta	Alto rendimiento, compresión eficiente, soporte para Big Data frameworks	Procesamiento distribuido (Spark, Hive), almacenamiento en data lakes

Fuente: Una adaptación de (Chang et al., 2019; Kumar, 2021a; Mazon-Olivo, Pan, et al., 2018)

Datos Semiestructurados

Los datos semiestructurados no siguen un esquema tabular rígido, pero conservan una estructura interna legible por máquina que permite su interpretación. Formatos como JSON, XML, Avro o YAML se utilizan ampliamente para representar objetos jerárquicos o anidados, lo cual es fundamental en el intercambio de información entre sistemas distribuidos, APIs RESTful y bases de datos NoSQL. Su flexibilidad facilita la ingestión de datos desde fuentes variadas, como registros web o flujos de sensores, adaptándose bien al modelo schema-on-read de los data lakes. Además, su compatibilidad con entornos de procesamiento como Apache Spark y Kafka los hace esenciales para pipelines de datos modernos (Chang et al., 2019).

Datos No Estructurados

Los datos no estructurados constituyen la mayor parte del volumen generados en la actualidad y abarcan contenidos como texto libre, imágenes, audios, videos, correos electrónicos y documentos sin una estructura definida. Se requiere técnicas avanzadas para su procesamiento, como procesamiento de lenguaje natural (PLN), visión por computador o análisis de contenido multimedia. Los datos no estructurados representan un reto técnico debido a su diversidad y volumen, pero también una oportunidad, ya que contienen información contextual valiosa para la toma de decisiones, personalización de servicios y desarrollo de modelos predictivos en ámbitos como la salud, el comercio electrónico y la seguridad (Chang et al., 2019).

Datos binarios

Los datos binarios son representaciones en formato de bajo nivel de cualquier contenido digital, almacenado como secuencias de bytes. Este tipo de datos incluye archivos de imagen (.jpg, .png), videos (.mp4, .avi), modelos de aprendizaje automático serializados (.pkl, .onnx), archivos comprimidos y ejecutables. Aunque no son directamente interpretables por sistemas analíticos tradicionales, en entornos Big Data se almacenan en lagos de datos y se procesan mediante herramientas especializadas que permiten su interpretación, análisis o transformación. Su almacenamiento eficiente y seguro es esencial, especialmente en aplicaciones de machine learning, sistemas multimedia, realidad aumentada o videojuegos (Kumar, 2021a).

Formatos optimizados para Big Data (columnar y serializados)

Los entornos de Big Data requieren formatos de almacenamiento que maximicen la eficiencia en el acceso, consulta y transmisión de datos. Los formatos columnares como Parquet y ORC permiten una alta compresión y lectura selectiva de columnas, optimizando el rendimiento en consultas analíticas distribuidas (por ejemplo, con Apache Hive, Spark o Trino). Por otro

lado, formatos serializados como Avro y Protocol Buffers (Protobuf) son usados para transmitir datos de forma compacta entre servicios o sistemas de mensajería como Apache Kafka. Además, formatos como Delta Lake combinan almacenamiento columnar con transacciones ACID, facilitando arquitecturas Lakehouse. Estos formatos especializados son fundamentales para lograr eficiencia y consistencia en infraestructuras de análisis de datos masivos (Kumar, 2021a).

Almacenamiento y Procesamiento de Big Data

En la Tabla 4, se propone algunas prácticas para el almacenamiento y procesamiento en Big data (DbExperts, 2023).

Tabla 4. Estrategias de Almacenamiento y Procesamiento de Big Data

Almacenamiento de Big Data	
Almacenamiento distribuido	Distribución de datos en múltiples nodos de un clúster, utilizando sistemas como Hadoop Distributed File System (HDFS) o Apache Cassandra, permite una mayor escalabilidad, tolerancia a fallos y acceso rápido a los datos.
Almacenamiento en la nube:	Los servicios en la nube, como Amazon Web Services (AWS), Microsoft Azure, GCP y otros, ofrecen soluciones escalables y flexibles que permiten almacenar grandes volúmenes de datos de manera segura y rentable.
Almacenamiento estructurado, semi-estructurado y no estructurado:	El Big Data puede incluir datos estructurados (RDBMS: MySQL, Postgres, etc.), semiestructurados como las BD NoSQL (Mongo DB, Cassandra, Redis, CouchDB, Neo4j, etc.) y no estructurados (texto libre, imágenes o videos).
Procesamiento de Big Data	
Computación distribuida:	Dividir tareas de procesamiento en múltiples nodos de un clúster acelera el procesamiento y mejora la escalabilidad. Frameworks como Apache Spark o Hadoop MapReduce son ampliamente utilizados para el procesamiento distribuido de Big Data.
Paralelismo y almacenamiento en memoria:	La aplicación de técnicas de paralelismo y almacenamiento en memoria, como la computación en memoria o el uso de GPU, acelera el procesamiento al reducir los cuellos de botella de la CPU y permitir un acceso más rápido a los datos.
Herramientas especializadas de procesamiento:	Dependiendo de los requisitos del proyecto, existen diversas herramientas de procesamiento de Big Data disponibles. Por ejemplo, Apache Hive o Apache Impala son ideales para consultas ad hoc, mientras que Apache Kafka o Apache Flink son útiles para análisis en tiempo real.

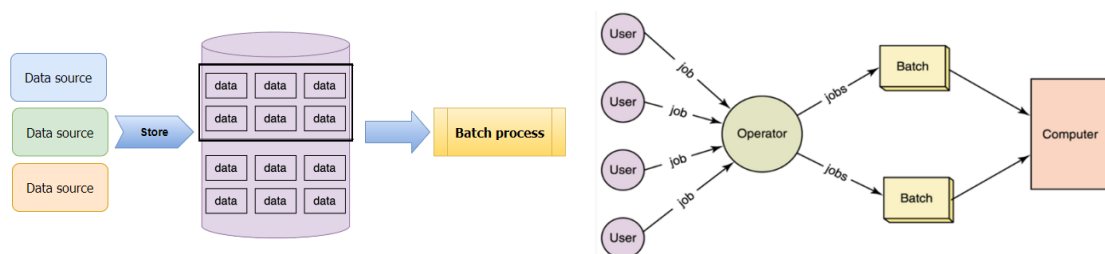
Fuente: (Chang et al., 2019; DbExperts, 2023).

Tipos de Procesamientos en Big Data

El procesamiento de datos se refiere a la forma en que los sistemas gestionan y transforman grandes volúmenes de información para obtener valor analítico, cada uno adaptado a diferentes necesidades de latencia, volumen y complejidad. Los principales tipos de procesamiento son por lotes (batch), tiempo real (Stream Processing) y Micro-batch. Además, están el procesamiento interactivo (Interactive Query Processing) y Procesamiento híbrido (Chang et al., 2019).

Procesamiento por lotes (Batch)

Este tipo de procesamiento se caracteriza por la ejecución de trabajos de gran escala y baja frecuencia, donde no se requiere de un resultado inmediato debido a su alta latencia. Como se observa en la Figura 11 a), este es un método que opera de manera periódica entre intervalos programados y está diseñado para analizar grandes volúmenes de datos históricos acumulados. Un ejemplo de estos pueden ser los sistemas de nómina o facturación, los cuales suelen ser procesados cada cierto tiempo. Para implementar este tipo de herramientas, se usan herramientas como Hadoop MapReduce (Figura 11 b) el cual aprovecha el planificador de recursos YARN para poder gestionar las tareas distribuidas en clúster. Apache Spark es otra herramienta para procesamiento en lotes pero en memoria (Candel, 2022a).



a) Grandes volúmenes de datos b) Tareas en lotes
procesados en lotes

Figura 11. Funcionamiento del procesamiento Batch

Fuente: (AWS, 2024; Gomar, 2018)

Procesamiento en tiempo real (streaming)

Este tipo de procesamiento maneja flujos de datos de manera continua para generar resultados inmediatos, como se ve en la Figura 12. Utilizado por ejemplo en aplicaciones de IoT, detección de fraudes o redes sociales en donde los datos se procesan, almacenan y se analizan simultáneamente. Tecnologías como Apache Kafka, Apache Flink y Apache Storm gestionan estos flujos, garantizando escalabilidad y baja latencia (Candel, 2022a).

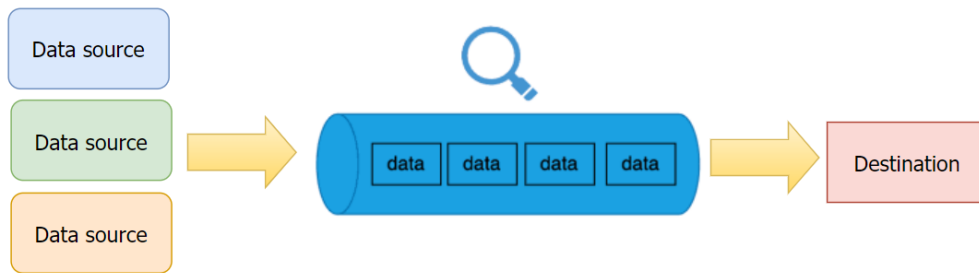


Figura 12. Funcionamiento del procesamiento streaming
Fuente: (AWS, 2024)

Procesamiento en Micro-batch

Combina las ventajas del procesamiento por lotes y streaming, dividiendo los datos en pequeños lotes o micro-batches que se procesan en intervalos cortos y regulares (segundos o minutos). Como se aprecia en la Figura 13, este es un enfoque que permite un análisis casi en tiempo real, aunque su latencia es mínima, producto de la acumulación temporal de datos antes de su procesamiento. Es adecuado para aplicaciones que no requieren respuestas inmediatas, pero si rápidas, donde una demora de segundos es aceptable. Herramientas para este tipo de procesamiento son Apache Spark Streaming, Databricks y Apache Deam. Por ejemplo, en Spark Streaming, los flujos de datos se dividen en micro-lotes para ser procesador mediante operaciones similares a las del batch tradicional, pero con una frecuencia mucho mayor y ejecución en memoria que otorga velocidad (Andreoni, 2018).

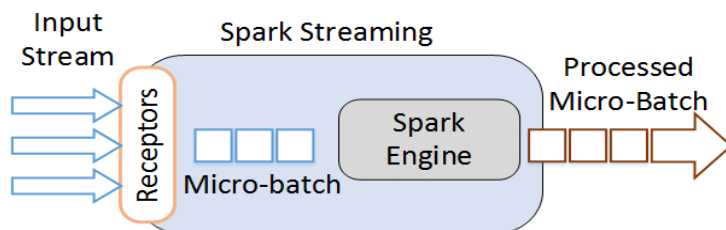


Figura 13. Funcionamiento del procesamiento Micro-batch
Fuente: (Andreoni, 2018)

Procesamiento interactivo

Permite realizar consultas dinámicas sobre grandes volúmenes de datos, con tiempos de respuesta razonables. Se utiliza en plataformas de inteligencia de negocios (BI) para análisis ad hoc. Herramientas como Presto, Apache Drill y Apache Impala permiten explorar datos en data lakes o warehouses sin necesidad de moverlos (Chang et al., 2019).

Aplicaciones del Big Data

El Big Data tiene aplicaciones transversales en múltiples sectores, transformando la forma en que las organizaciones procesan información, toman decisiones y generan valor (Badshah et al., 2024; Rahul et al., 2023).

En el **sector de la salud**, se aplica en el análisis de grandes volúmenes de datos clínicos y genómicos para apoyar el diagnóstico temprano, la medicina personalizada y el monitoreo de enfermedades en tiempo real (Badshah et al., 2024; Rahul et al., 2023). Por ejemplo, durante la pandemia de COVID-19, el Big Data fue clave para rastrear contagios y predecir necesidades hospitalarias.

En **finanzas**, se emplea para detectar fraudes, evaluar riesgos crediticios y personalizar productos financieros a través del análisis en tiempo real de transacciones y patrones de comportamiento. Las tecnologías de Big Data también permiten a los bancos automatizar decisiones de inversión y optimizar procesos regulatorios mediante análisis predictivos avanzados (Badshah et al., 2024).

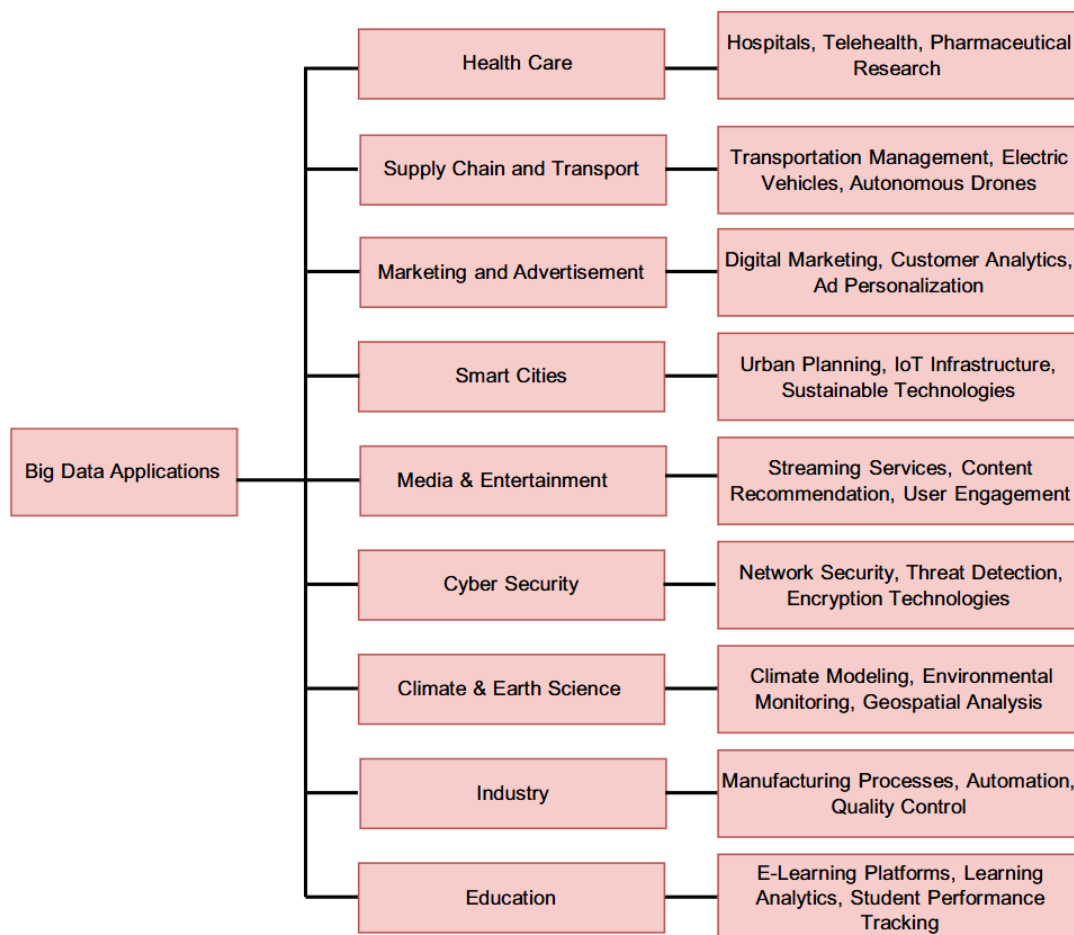


Figura 14. Aplicaciones de Big Data
Fuente: (Badshah et al., 2024)

En el ámbito del **marketing y comercio digital**, el Big Data permite segmentar audiencias, analizar comportamiento de usuarios y personalizar campañas en función de patrones de consumo, preferencias y datos sociales (Badshah et al., 2024). Las plataformas de e-commerce integran Big Data para recomendar productos, predecir abandono de carritos y maximizar la conversión mediante análisis en tiempo real.

En aplicaciones de **Internet de las cosas (IoT)** como gobiernos y **ciudades inteligentes**, se utiliza para mejorar la planificación urbana, optimizar servicios públicos, gestionar tráfico vehicular y promover la transparencia mediante el análisis de datos en tiempo real provenientes de sensores, dispositivos móviles y redes sociales (Badshah et al., 2024; Mazón-Olivo, 2023; Rahul et al., 2023). En **agricultura**, el Big Data habilita la agricultura de precisión al integrar datos climáticos, de suelos y satelitales para mejorar la productividad y sostenibilidad (Kamilaris et al., 2017; Mazón-Olivo, 2023).

También se aplica en sectores donde se manejan datos en gran volumen, como educación, logística (cadena de suministros y transporte) y energía, para predecir deserción estudiantil, optimizar rutas de distribución y gestionar redes eléctricas inteligentes, respectivamente. Estas y otras aplicaciones como entretenimiento y multimedia, Ciberseguridad, Clima y Ciencias de la Tierra, Industria, etc. (Figura 14), evidencian el impacto estratégico del Big Data como motor de innovación y eficiencia organizacional (Badshah et al., 2024; Mazón-Olivo, 2023; Rahul et al., 2023).

Importancia, ventajas y desafíos de Big Data

Importancia de Big Data

El verdadero valor del Big Data no se encuentra en la cantidad de información disponible sino en cómo utilizamos esta información para generar conocimiento y para la toma de decisiones estratégicas. Su impacto radica en la reducción de costos, optimización del tiempo, en el desarrollo de nuevos productos o mejoras en las ofertas de servicios y la posibilidad de tomar decisiones más acertadas (Badshah et al., 2024; Rahul et al., 2023). La mayor parte del esfuerzo de los proyectos que adaptan Big Data, entre el 70 y 80% es invertido en recuperación y la preparación de los datos dejando una pequeña parte de tiempo para el análisis y aprovechamiento (Badshah et al., 2024; Rahul et al., 2023).

Beneficios / Ventajas de Big Data / Analytics

En (GCP, 2025; Reis Digital, 2022) se presentan algunas ventajas de Big Data en las empresas (Tabla 5):

Tabla 5. Ventajas de aplicar el Big Data en las empresas

1. Mejora de la toma de decisiones	El analizar la información ayuda a reducir riesgos y a tomar decisiones más inteligentes rápidamente.
2. Mejora en la eficiencia y optimización	Big Data es capaz de reducir el tiempo y optimizar recursos en el desarrollo de un producto.
3. Segmentación de los clientes	Permite usar los datos sobre los clientes para planificar un marketing acorde a su consumo; para personalizar la experiencia y satisfacer sus necesidades y expectativas.
4. Seguridad en los datos	Mantiene un control rápido y eficaz de los datos para poder identificar amenazas internas, evaluar mejor los riesgos y planificar estrategias de control y mitigación más sólidas.
5. Mejora de la accesibilidad de la información dentro de la empresa	Esto es gracias a las herramientas que hacen fácil la búsqueda de información.
6. Nuevas fuentes de ingresos	La información obtenida del análisis del Big Data puede ser comercializada a otras empresas como tendencias.
7. Ventajas competitivas	El análisis de datos ayuda a comprender el comportamiento de los clientes y poder fijar precios en base a ese análisis.

Fuente: (GCP, 2025; Reis Digital, 2022)

Desafíos de Big Data

Big Data conlleva una serie de desafíos organizacionales, técnicos y éticos (Figura 15) que impactan directamente en su implementación efectiva. Entre los más significativos se encuentran la gestión del volumen, variedad y velocidad de los datos, que requieren infraestructuras altamente escalables y mecanismos eficientes de procesamiento en tiempo real (Lv et al., 2017). Asimismo, la calidad e integración de datos heterogéneos provenientes de múltiples fuentes representa un obstáculo para garantizar consistencia y confiabilidad en el análisis. Otro reto importante es la seguridad y privacidad, especialmente en contextos regulados como salud o finanzas, donde el cumplimiento de normativas es obligatorio. Adicionalmente, muchas organizaciones enfrentan dificultades en la gobernanza de datos, en la disponibilidad de talento especializado y en la adopción de una cultura basada

en datos. Por último, se suma la complejidad de transformar datos masivos en conocimiento útil y accionable que realmente aporte valor al negocio (GCP, 2025; Mazón-Olivo, 2023). Estos desafíos exigen enfoques holísticos que combinen tecnologías avanzadas, marcos regulatorios y estrategias organizacionales integradas.



Figura 15. Retos de Big Data
Fuente: (Mazón-Olivo, 2023)

En (GCP, 2025; Reis Digital, 2022) se menciona que, así como se tienen beneficios también existen algunas dificultades en el manejo y análisis del Big Data, por ejemplo:

- Falta de personal y de habilidades del recurso humano. Carencia de conocimiento técnico en análisis de Big data.
- Necesidades de hardware para la implementación de la arquitectura de un sistema de Big Data.
- Poca comprensión del dominio del negocio o del problema a abordar con el manejo de Big Data y presentación de resultados útiles para los usuarios finales.
- Necesidad de cambio cultural, falta de patrocinio empresarial y ausencia de un argumento empresarial convincente.
- Necesidad de abordar los problemas de calidad de los datos, antes de usarlos para el análisis de Big Data.
- Complejidad de la integración; la mayoría de las empresas trabajan con datos aislados en varios sistemas y aplicaciones.
- Problemas de seguridad; Big Data contiene información valiosa sobre las empresas y los clientes, estos son objetivos de gran valor para los atacantes.
- Problemas de escalabilidad de Big Data, entre otros

Big Data Analytics

Big Data Analytics (BDA) es el proceso de examinar, transformar y modelar grandes volúmenes de datos heterogéneos, estructurados, semiestructurados y no estructurados, con el objetivo de descubrir patrones ocultos, correlaciones, tendencias y preferencias de usuarios, que permitan tomar decisiones informadas y generar ventajas competitivas (Mazón-Olivo, 2023). En BDA se integran la ciencia de datos y la inteligencia artificial para el análisis de grandes volúmenes de datos; es decir, consiste en la aplicación de procedimientos, técnicas y herramientas de análisis de datos a gran escala, con la finalidad de obtener conocimiento útil (Berisha et al., 2022; Mazón-Olivo, 2023).

Ciclo de vida de Big Data Analytics y relación con otras disciplinas

La analítica de grandes volúmenes de datos (Big Data Analytics, BDA) se nutre de diversas disciplinas que aportan métodos, técnicas y herramientas fundamentales para extraer conocimiento a partir de datos estructurados, semiestructurados y no estructurados. Entre las más destacadas se encuentran la inteligencia de negocios, la minería de datos, el aprendizaje automático y el aprendizaje profundo.

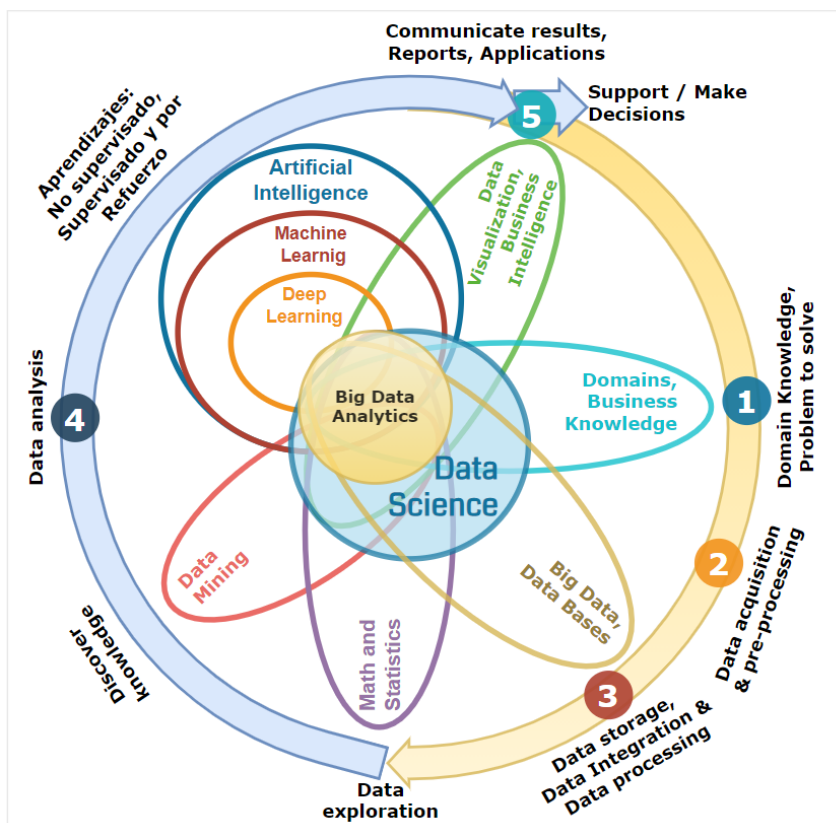


Figura 16. Ciclo de vida de Big Data/Big Data Analytics y relación con otras disciplinas

Fuente: (Mazón-Olivo, 2023).

El Big Data Analytics comienzan con el conocimiento del dominio del negocio y luego aplica procedimientos y técnicas de disciplinas como Matemática, Estadística, Ciencia de Datos e Inteligencia Artificial, con el fin de generar valor de grandes volúmenes de datos, desde una perspectiva tanto técnica como estratégica. En la Figura 16 se presentan las fases de un ciclo de vida de BDA (Mazón-Olivo, 2023):

1. Conocimiento del dominio del negocio o comprensión del problema a resolver.
2. Adquisición y preprocesamiento de datos provienen de varias fuentes y formatos.
3. Almacenamiento, integración y procesamiento de datos en bases de datos SQL y No SQL, repositorios distribuidos.
4. Exploración y análisis de datos con técnicas estadísticas, de minería de datos, aprendizaje automático, entre otras técnicas.
5. Comunicación de resultados mediante reportes y paneles de control con Inteligencia de Negocios para la toma de decisiones.

Estas etapas permiten transformar datos masivos en conocimiento útil, aplicando técnicas avanzadas como la minería de datos, el aprendizaje automático (machine learning), el aprendizaje profundo (deep learning) u otras ramas de la inteligencia artificial, potenciando el descubrimiento de patrones complejos. Además, se destaca el papel central de la visualización de datos, la inteligencia de negocios (BI) y la estadística como herramientas clave en todo el ciclo. A continuación, se describe cada una de las disciplinas relacionadas con BDA, su aplicación en el contexto de BDA y las principales herramientas utilizadas, tanto en entornos tradicionales como en la nube.

Inteligencia de Negocios (BI)

La Inteligencia de Negocios (Business Intelligence, BI) se enfoca en transformar datos brutos de diversas áreas organizativas, como ventas, finanzas o recursos humanos, en información útil para la toma de decisiones. En el ámbito de BDA, BI permite el análisis y visualización de datos a gran escala mediante procesos de integración (ETL), almacenamiento en data warehouses y explotación, utilizando herramientas de consulta, reportes y dashboards. Estas soluciones facilitan la identificación de patrones, tendencias y comportamientos relevantes, mejorando así el desempeño estratégico de las organizaciones (Mazon-Olivo, Bertha et al., 2017).

Se dispone de una variedad de herramientas para BI, tales como Tableau, Power BI, Qlik Sense, Apache Superset. También se tienen las plataformas en la nube como Looker y Data Studio en Google Cloud Platform, Amazon QuickSight en AWS, Power BI Service en Azure, dashboards SQL de Databricks,

etc, permiten integrar el análisis con procesamiento distribuido (Mazon-Olivo et al., 2019; Mazon-Olivo, Pan, et al., 2018). Los campos de aplicación de BI más comunes son: industria, comercio, banca, educación, turismo, entre otros.

Una arquitectura BI se compone: una capa de Fuentes de Datos, procesos ETL (Extraction Transformation and Loading), Almacén de datos (data warehouse físico o virtual), y aplicaciones BI (OLAP, consultas y reportes Ad-hoc, dashboard, minería de datos y otras técnicas de análisis, que son útiles para encontrar nuevos patrones de datos y realizar pronósticos que sirvan de soporte para la toma de decisiones (Mazon-Olivo, Pan, et al., 2018; Mazon-Olivo, Bertha et al., 2017).

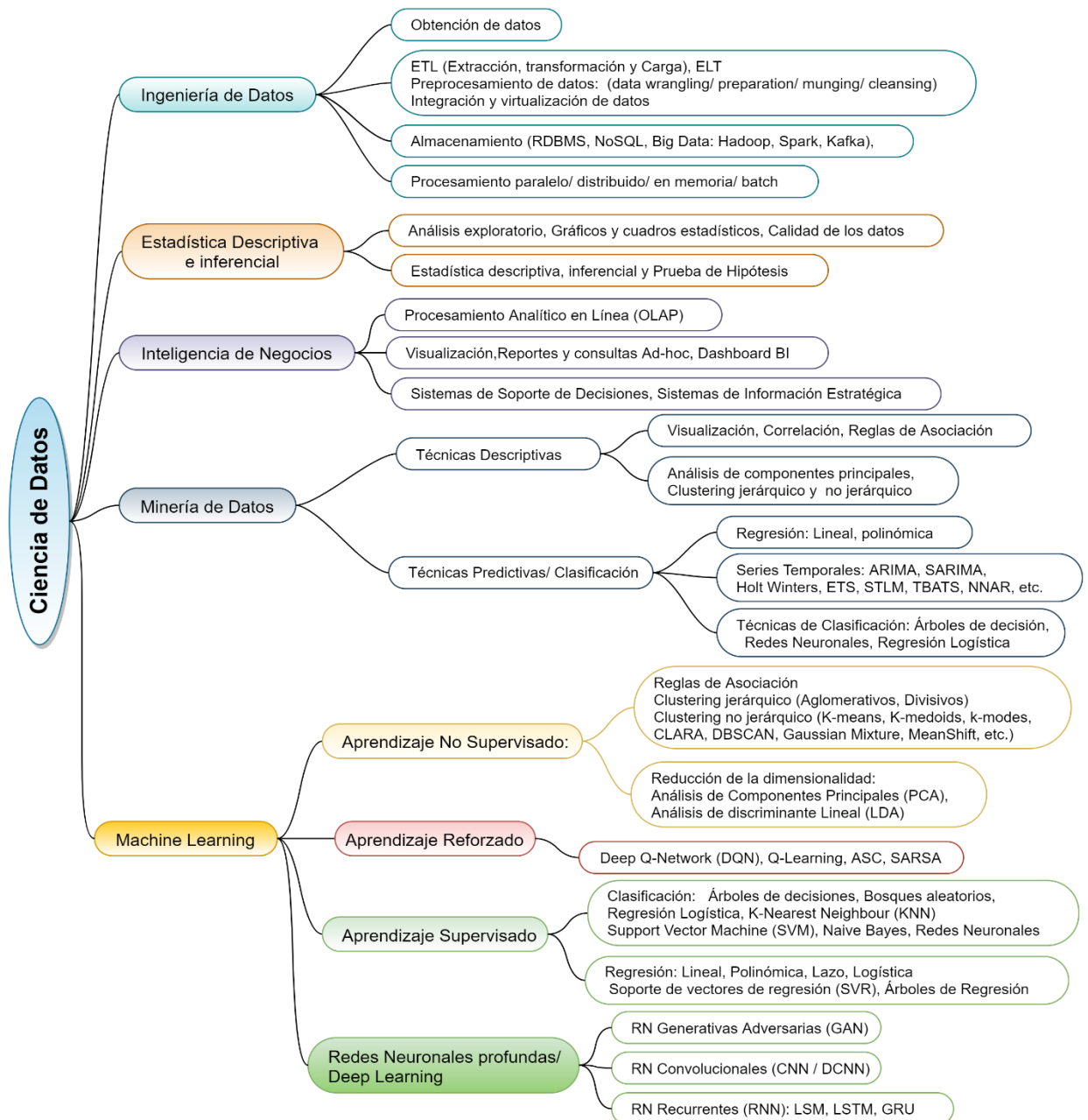


Figura 17. Disciplinas con las que se relaciona BDA
Fuente: (Mazón-Olivo, 2023).

Minería de Datos (DM: Data Mining).

Data Mining permite descubrir conocimiento oculto mediante el análisis automatizado de grandes volúmenes de datos. Se utilizan técnicas descriptivas y predictivas para detectar reglas de asociación, agrupamientos, árboles de decisión, o anomalías que serían difíciles de identificar mediante análisis tradicionales. En el contexto de BDA, la minería de datos permite el análisis de datos en entornos distribuidos y heterogéneos, aportando valor en tareas como la segmentación de clientes, la detección de fraude o el análisis de sentimiento. Herramientas clásicas como RapidMiner, KNIME han evolucionado hacia entornos compatibles con Big Data. En la nube, destacan soluciones como Google Cloud Dataprep en combinación con BigQuery ML, Azure Machine Learning junto a Data Lake Analytics, y AWS Glue integrado con SageMaker Data Wrangler, que permiten minar datos de manera escalable y automatizada. (Mazon-Olivo et al., 2020; Mazón-Olivo et al., 2018; Shadroo & Rahmani, 2018).

Tabla 6. Resumen de disciplinas y técnicas para BDA

Disciplina	Técnicas
Business Intelligence Inteligencia De Negocios (Ramírez & Mazón, 2018)	- Análisis multidimensional (OLAP) - Indicadores clave de desempeño - Tableros de control (Dashboard) - Data warehouse y Data Marts - Consultas y reportes personalizados (Ad Hoc)
Data Mining Minería de Datos (Ramírez & Mazón, 2018)	Descriptivas - Reglas de asociación - Análisis de correlación - Clustering Predictivas - Series Temporales - Árboles de decisión - Redes Neuronales - Regresión (Lineal, Polinómica, ...)
Machine Learning Aprendizaje Automático (Mazón-Olivo, 2023)	Aprendizaje No supervisado: - Clustering: - No Jerárquico: K-means, k-modes, Modelo de mezcla Gaussiana (GMM). - Jerárquico: Aglomerativo, divisivo. - Density Based Scan Clustering (DBSCAN). - Spectral clustering (SC), otros - Reducción de la dimensionalidad: - Análisis de componentes principales, - Aproximación de Kernel, - Spectral Embedded Deep Clustering (SEDC), etc. Aprendizaje Supervisado. Algoritmos de Clasificación y Regresión:

	- Regresión lineal: Lineal simple, múltiple y polinómica. Ridge / Lasso / Elastic Net - Regresión logística - Árboles de decisión - Naive Bayes - Vecinos más Cercanos (KNN) - Máquinas de Soporte Vectorial (SVC) - Random Forest - Gradient Boosting Machines (GBM): XGBoost, LightGBM, CatBoost - Redes Neuronales Artificiales
Deep Learning Aprendizaje Profundo (Soni & Kumar, 2022).	Redes neuronales profundas - Perceptrón multicapa - Redes neuronales convolucionales - Redes neuronales recurrentes - Transformes - Redes de base radial - Otras

Fuente: (Ramírez-Morales et al., 2018).

Aprendizaje automático o aprendizaje de máquinas (ML: Machine Learning).

El Aprendizaje automático es una rama de la IA que permite a los sistemas aprender a partir de datos sin programación explícita. Su integración en BDA es necesaria, ya que posibilita la automatización de tareas analíticas complejas como la predicción de comportamientos, la clasificación de registros o la recomendación de productos, en escenarios donde el volumen y la velocidad del flujo de datos requieren procesamiento distribuido. Herramientas como Scikit-learn, H2O.ai, y Spark MLlib son ampliamente utilizadas en estos entornos. En plataformas en la nube, destacan Amazon SageMaker (AWS), Vertex AI (Google Cloud), Azure Machine Learning, y Databricks MLflow, que permiten entrenar, evaluar, desplegar y gestionar modelos de forma escalable y en producción (Soni & Kumar, 2022).

Aprendizaje Profundo (DL: Deep Learning).

El Aprendizaje Profundo constituye una evolución del ML, orientado a trabajar con redes neuronales profundas capaces de modelar relaciones complejas entre datos no estructurados como imágenes, audio, texto o video. En BDA, el aprendizaje profundo es esencial para abordar tareas como el reconocimiento de imágenes satelitales, la clasificación de objetos, la detección de intrusiones, el análisis de lenguaje natural o el procesamiento de señales biométricas. Este tipo de procesamiento requiere entornos de alto rendimiento con soporte GPU o TPU, ofrecidos comúnmente en la nube. Frameworks como TensorFlow, PyTorch, Keras y DeepLearning4J se utilizan ampliamente para diseñar e implementar modelos de redes neuronales profundas. Las principales

plataformas en la nube que facilitan este tipo de procesamiento son Google Cloud AI Platform, que ofrece TPUs para acelerar el entrenamiento; AWS Deep Learning AMIs y SageMaker Studio Labs, con instancias optimizadas para DL; Azure AI con máquinas virtuales NVIDIA; y Databricks Runtime for ML, que integra Spark con bibliotecas de DL para el entrenamiento y despliegue distribuido (Dong et al., 2021; Soni & Kumar, 2022). Los tipos de redes neuronales más comunes en Deep Learning son las Redes Neuronales Convolucionales (CNN), Redes Neuronales Generativas Adversarias (GAN), Redes Neuronales Recurrentes (RNN), etc. (Dong et al., 2021; Moya et al., 2017; Rivas-Asanza, Mazon-Olivo, & Mejía-Peñafiel, 2018; Rivas-Asanza, Mazon-Olivo, & Tusa-Jumbo, 2018; Salas et al., 2019).

En la Figura 17 y la Tabla 6, se representan las técnicas y algoritmos de la disciplina asociada a BDA y Ciencia de Datos, una adaptación a partir de (Mazon-Olivo et al., 2019, 2020; Mazon-Olivo, Pan, et al., 2018; Mazón-Olivo, 2023; Sharma et al., 2020).

Resumen del Capítulo. Se presentó los fundamentos de Big Data y Big Data Analytics, abordando tanto sus aspectos conceptuales como tecnológicos. Se resaltó la importancia de gestionar grandes volúmenes de datos generados en tiempo real, provenientes de diversas fuentes, con el objetivo de transformarlos en información valiosa que sustente la toma de decisiones estratégicas en múltiples sectores. Uno de los pilares fundamentales de esta disciplina es el conjunto de características conocidas como las “5 Vs” del Big Data: volumen, velocidad, variedad, valor y veracidad. Estas dimensiones permiten comprender la complejidad y el potencial de los datos que se producen y almacenan constantemente en entornos digitales. Se analizaron diferentes arquitecturas de procesamiento masivo de datos, incluyendo los modelos Lambda, Kappa y la arquitectura de referencia propuesta por el NIST, las cuales permiten abordar flujos de datos en tiempo real y por lote. También se examinaron los principales componentes tecnológicos del ecosistema Big Data, haciendo énfasis en plataformas como Apache Hadoop para el almacenamiento distribuido y Apache Spark para el procesamiento paralelo y en memoria. También, se clasificaron las fuentes de datos según su estructura: estructurados, semiestructurados y no estructurados; y, se describieron los principales formatos de almacenamiento utilizados en estos contextos, tales como archivos columnares, binarios y serializados. Se introdujo también el concepto de data lakes, que representan una evolución flexible en la forma de almacenar datos a gran escala, permitiendo su posterior análisis bajo demanda.

En la segunda parte del capítulo se profundizó en el concepto de Big Data Analytics, entendido como el conjunto de técnicas, metodologías y herramientas que permiten descubrir patrones, correlaciones y conocimientos

útiles a partir del análisis de datos masivos. Se presentó su ciclo de vida, desde la adquisición y procesamiento de los datos hasta su modelado y visualización, y se discutió su relación con disciplinas clave como la inteligencia de negocios (BI), la minería de datos (Data Mining), el aprendizaje automático (Machine Learning) y el aprendizaje profundo (Deep Learning). Estas disciplinas amplifican el alcance de la analítica avanzada, haciendo posible la automatización de decisiones, la predicción de comportamientos y la generación de recomendaciones personalizadas. Finalmente, se abordaron los principales desafíos que enfrenta la implementación de soluciones de Big Data y Big Data Analytics, entre ellos la gestión de infraestructuras complejas, la disponibilidad de talento especializado y las preocupaciones en torno a la privacidad, seguridad y gobernanza de los datos. Se concluyó enfatizando la importancia estratégica del Big Data en la actualidad, no solo como una tendencia tecnológica emergente, sino como una herramienta transformadora para la innovación, la competitividad y el desarrollo de nuevas oportunidades en el contexto de la economía digital.

Autoevaluación

A continuación, responde las siguientes preguntas para evaluar tu comprensión del capítulo.

1. ¿Qué es Big Data y cuál es su importancia en la actualidad?
2. ¿Cómo ha evolucionado el concepto de Big Data a lo largo del tiempo?
3. ¿Cuáles son las Vs del Big Data y que representa cada una?
4. ¿Qué tecnologías se utilizan para el almacenamiento y procesamiento de Big Data?
5. ¿Cuáles son los principales desafíos en la gestión del Big Data?
6. ¿Cuál es la diferencia entre datos estructurados, semiestructurados y no estructurados en Big Data?
7. ¿Cómo influye la Veracidad en la calidad del análisis de datos?
8. ¿Qué papel juegan Hadoop y Spark en el ecosistema del Big Data?
9. ¿En qué sectores ha tenido mayor impacto el Big Data y por qué?
10. ¿Cuáles son los principales retos en la gestión y almacenamiento de Big Data?
11. ¿Qué es Big Data Analytics?
12. ¿Cuáles son las fases del ciclo de vida de Big Data Analytics?
13. ¿Con qué disciplinas se relaciona el BDA?

Tareas autónomas:

1. Realizar un cuadro comparativo con herramientas para Big Data y Big Data Analytics

2. Describir casos de aplicación de Big Data y Big Data Analytics mediante revisión bibliográfica de trabajos de investigación y académicos.

Referencias Bibliográficas

- Alwidian, J., Rahman, S., Gnaim, M., & Al-Taharwah, F. (2020). Big Data Ingestion and Preparation Tools. *Modern Applied Science*, 14(9), Article 9. <https://doi.org/10.5539/mas.v14n9p12>
- Andreoni, M. (2018). (PDF) *A Monitoring and Threat Detection System Using Stream Processing as a Virtual Function for Big Data*. ResearchGate. <https://doi.org/10.13140/RG.2.2.27570.25288>
- Apache Software Foundation. (2025). *Apache Hadoop*. <https://hadoop.apache.org/>
- Araque González, G. A., Gómez Vásquez, M., Vélez Uribe, J. P., & Suárez Hernández, A. H. (2021). Big Data y las implicaciones en la cuarta revolución industrial—Retos, oportunidades y tendencias futuras. *Revista Venezolana de Gerencia: RVG*, 26(93), 33-47.
- AWS. (2024). *Six Things to Consider When Moving From Batch to Real-time Data Stream Processing*. Community.Aws. <https://community.aws/content/2iCl0on6KVn5djWtuytFHAUhQ8/six-things-to-consider-when-moving-from-batch-to-real-time-data-stream-processing>
- Badshah, A., Daud, A., Alharbey, R., Banjar, A., Bukhari, A., & Alshemaimri, B. (2024). Big data applications: Overview, challenges and future. *Artificial Intelligence Review*, 57(11), 290. <https://doi.org/10.1007/s10462-024-10938-5>
- Berisha, B., Mëziu, E., & Shabani, I. (2022). Big data analytics in Cloud computing: An overview. *Journal of Cloud Computing*, 11(1), 24. <https://doi.org/10.1186/s13677-022-00301-w>
- Bourany, T. (2018). Les 5V du big data. *Regards croisés sur l'économie*, 23(2), 27-31. <https://doi.org/10.3917/rce.023.0027>
- Candel, J. M. O. (2022). *Big data, machine learning y data science en python*. Ra-Ma Editorial.
- Careaga, J. (2020). *Arquitectura Lambda vs Arquitectura Kappa*.
- Chang, W. L., Boyd, D., & Levin, O. (2019). NIST Big Data Interoperability Framework: Volume 6, Reference Architecture. *NIST*. <https://doi.org/10.6028/NIST.SP.1500-6r2>
- Chen, M. (2024, septiembre 23). *Big Data, grandes posibilidades: Cómo extraer el máximo valor*. Oracle. <https://www.oracle.com/es/big-data/what-is-big-data/>

- Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, 19(2), 171-209. <https://doi.org/10.1007/s11036-013-0489-0>
- DbExperts. (2023, mayo 8). Estrategias de Almacenamiento y Procesamiento de Big Data: Desbloqueando el Valor Oculto. *DbExperts*. <https://dbaexperts.tech/database/estrategias-de-almacenamiento-y-procesamiento-de-big-data-desbloqueando-el-valor-oculto/>
- Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, 40, 100379. <https://doi.org/10.1016/j.cosrev.2021.100379>
- Dynamic. (2020, junio 7). Historia del Big Data. Descubre su origen y cómo se ha ido consolidando. *DYNAMIC*. <https://www.dynamicgc.es/historia-del-big-data/>
- GCP. (2025). *Definición del Big Data: Ejemplos y ventajas*. Google Cloud. <https://cloud.google.com/learn/what-is-big-data>
- Gomar, J. (2018). *Qué es el procesamiento batch o por lotes*. Que es el procesamiento batch o por lotes. <https://www.profesionalreview.com/2018/11/25/que-es-el-procesamiento-batch/>
- Jabbar, A., Akhtar, P., & Dani, S. (2019). Real-time big data processing for instantaneous marketing decisions: A problematization approach. *Industrial Marketing Management*. <https://doi.org/10.1016/j.indmarman.2019.09.001>
- Kalipe, G., & Behera, R. (2019). Big Data Architectures: A Detailed and Application Oriented Analysis. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 8(9), 2182-2190. <https://doi.org/10.35940/ijitee.H7179.078919>
- Kamilaris, A., Kartakoullis, A., & Prenafeta-Boldú, F. X. (2017). A review on the practice of big data analysis in agriculture. *Computers and Electronics in Agriculture*, 143, 23-37. <https://doi.org/10.1016/j.compag.2017.09.037>
- Kumar, N. (2021). *Big Data Using Hadoop and Hive*. Mercury Learning and Information.
- Lo Giudice, P., Musarella, L., Sofo, G., & Ursino, D. (2019). An approach to extracting complex knowledge patterns among concepts belonging to structured, semi-structured and unstructured sources in a data lake. *Information Sciences*, 478, 606-626. <https://doi.org/10.1016/j.ins.2018.11.052>
- Lv, Z., Song, H., Basanta-Val, P., Steed, A., & Jo, M. (2017). Next-Generation Big Data Analytics: State of the Art, Challenges, and Future Research Topics. *IEEE Transactions on Industrial Informatics*, 13(4), 1891-1899. <https://doi.org/10.1109/TII.2017.2650204>

- Mazón-Olivo, B. (2023). *Una arquitectura para la recopilación, integración y análisis de información en el contexto de la Internet de las Cosas. Caso estudio: Aplicaciones en el sector agrícola* [Doctoral thesis]. <https://ruc.udc.es/dspace/handle/2183/33729>
- Mazon-Olivo, B., Hernández-Rojas, D., Maza-Salinas, J., & Pan, A. (2018). Rules engine and complex event processor in the context of internet of things for precision agriculture. *Computers and Electronics in Agriculture*, 154, 347-360. <https://doi.org/10.1016/j.compag.2018.09.013>
- Mazón-Olivo, B., Jaramillo-Paredes, M., Romero-Hidalgo, O., Borja-Herrera, A., Aguirre-Benalcazar, M., & Contento-Segarra, M. (2018). Business Intelligence and Data Mining Technologies for the analysis of cocoa production and commercialization. *Espacios*, 39(32), 6.
- Mazon-Olivo, B., Pan, A., & Tinoco-Egas, R. (2018). Inteligencia de negocios en el sector agropecuario. En I. Ramírez-Morales & B. Mazón-Olivo (Eds.), *Análisis de Datos Agropecuarios* (First Ed., pp. 246-278). Universidad Técnica de Machala. <http://repositorio.utmachala.edu.ec/handle/48000/13330>
- Mazon-Olivo, B., Pinta, M., & Redrovan, F. (2020). Desarrollo de competencias en Minería de Datos, una experiencia didáctica. En *Sistematización de experiencias educativas innovadoras* (1.ª ed., pp. 383-406). Universidad Técnica de Machala. <http://repositorio.utmachala.edu.ec/handle/48000/15219>
- Mazon-Olivo, B., Rivas-Asanza, W., Novillo-Vicuña, J., & Flores-Cabrera, C. (2019). Análisis de producción avícola mediante técnicas de inteligencia de negocios y minería de datos. *Alternativas*, 19(2), 80-88. <https://doi.org/10.23878/alternativas.v19i2.203>
- Mazon-Olivo, Bertha, Rivas, Wilmer, Pinta, Maritza, Mosquera, Andrea, Astudillo, Luis, & Gallegos, Herman. (2017). Dashboard para el soporte de decisiones en una empresa del sector minero. *Conference Proceedings - Universidad Técnica de Machala*, 1, 1218-1229.
- Moreno-Carriles, R. M. (2018). Big data, ¿pero qué es? *Angiología*, 70(5), 191-194. <https://doi.org/10.1016/j.angio.2018.05.001>
- Moya, E., Campoverde, E., Tusa, E., Ramirez-Morales, I., Rivas, W., & Mazon, B. (2017). Multi-category Classification of Mammograms by Using Convolutional Neural Networks. *2017 International Conference on Information Systems and Computer Science (INCISCOS)*, 133-140. <https://doi.org/10.1109/INCISCOS.2017.56>
- Oussous, A., Benjelloun, F.-Z., Ait Lahcen, A., & Belfkih, S. (2017). Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2017.06.001>

- Quix, C., Hai, R., & Vatov, I. (2016). Metadata Extraction and Management in Data Lakes With GEMMS. *Complex Systems Informatics and Modeling Quarterly*, 0(9), 67-83. <https://doi.org/10.7250/csimq.2016-9.04>
- Rahul, K., Banyal, R. K., & Arora, N. (2023). A systematic review on big data applications and scope for industrial processing and healthcare sectors. *Journal of Big Data*, 10(1), 133. <https://doi.org/10.1186/s40537-023-00808-2>
- Ramirez, I., & Mazón, B. (2018). *Análisis de Datos Agropecuarios*. Ediciones UTMACH.
- Ramírez-Morales, I., Mazon-Olivo, B., & Pan, A. (2018). Capítulo 1: Ciencia de datos en el sector agropecuario. En I. Ramírez-Morales & B. Mazón-Olivo (Eds.), *Análisis de Datos Agropecuarios* (1.ª ed., pp. 12-44). Universidad Técnica de Machala. <http://repositorio.utmachala.edu.ec/handle/48000/12540>
- Raya, C. V. (2019, diciembre 11). Big Data: El journey del dato. *La innovación distingue a los lideres de los seguidores*. <https://carlosvizoso.com/big-data-el-journey-del-dato/>
- Raza, M. (2021, abril 16). *Big Data: A Big Introduction*. BMC Blogs. <https://www.bmc.com/blogs/big-data/>
- Reinsel, D., Gantz, J., & Rydning, J. (2018). *Data Age 2025. The Digitization of the World From Edge to Core*. Seagate.com. <https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf>
- Reis Digital. (2022, agosto 13). 10 ventajas del big data para tu negocio. *Reis Digital*. <https://reisdigital.es/ventajas-y-desventajas/ventajas-del-big-data/>
- Rivas-Asanza, W., Mazon-Olivo, B., & Mejía-Peñafiel, E. (2018). Generalidades de las redes neuronales artificiales. En W. Rivas-Asanza & B. Mazon-Olivo (Eds.), *Redes neuronales artificiales aplicadas al reconocimiento de patrones* (1.ª ed., pp. 11-35). Universidad Técnica de Machala. <http://repositorio.utmachala.edu.ec/handle/48000/14223>
- Rivas-Asanza, W., Mazon-Olivo, B., & Tusa-Jumbo, E. (2018). Reconocimiento de patrones en imágenes. En W. Rivas-Asanza & B. Mazon-Olivo (Eds.), *Redes neuronales artificiales aplicadas al reconocimiento de patrones* (1.ª ed., pp. 61-126). Universidad Técnica de Machala. <http://repositorio.utmachala.edu.ec/handle/48000/12499>
- Saggi, M. K., & Jain, S. (2018). A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing & Management*, 54(5), 758-790. <https://doi.org/10.1016/j.ipm.2018.01.010>

- Salas, J., de Barros Vidal, F., & Martinez-Trinidad, F. (2019). Deep Learning: Current State. *IEEE Latin America Transactions*, 17(12), 1925-1945. IEEE Latin America Transactions. <https://doi.org/10.1109/TLA.2019.9011537>
- Shadroo, S., & Rahmani, A. M. (2018). Systematic survey of big data and data mining in Internet of Things. *Computer Networks*, 139, 19-47. <https://doi.org/10.1016/j.comnet.2018.04.001>
- Sharma, R., Kamble, S. S., Gunasekaran, A., Kumar, V., & Kumar, A. (2020). A systematic literature review on machine learning applications for sustainable agriculture supply chain performance. *Computers & Operations Research*, 119, 104926. <https://doi.org/10.1016/j.cor.2020.104926>
- Sinha, S. (2020). Hadoop Ecosystem. Hadoop Tools for Crunching Big Data. *Eduureka*. <https://www.edureka.co/blog/hadoop-ecosystem>
- Soni, D., & Kumar, N. (2022). Machine learning techniques in emerging cloud computing integrated paradigms: A survey and taxonomy. *Journal of Network and Computer Applications*, 205, 103419. <https://doi.org/10.1016/j.jnca.2022.103419>
- Țăranu, I. (2015). Big Data Analytics Platforms analyze from startups to traditional database players. *Database Systems Journal*, 6(1), 23-32.
- Tejada, Z. (2018). *Big data architectures—Azure Architecture Center*. <https://learn.microsoft.com/en-us/azure/architecture/databases/guide/big-data-architectures>
- Waehner, K. (2021, septiembre 23). Kappa Architecture is Mainstream Replacing Lambda. *Kai Waehner*. <https://www.kai-waehner.de/blog/2021/09/23/real-time-kappa-architecture-mainstream-replacing-batch-lambda/>
- Wilcox, T., Jin, N., Flach, P., & Thumim, J. (2019). A Big Data platform for smart meter data analytics. *Computers in Industry*, 105, 250-259. <https://doi.org/10.1016/j.compind.2018.12.010>
- Wu, C., Buyya, R., & Ramamohanarao, K. (2016). Big Data Analytics = Machine Learning + Cloud Computing. En *Big Data* (pp. 3-38). Elsevier. <https://doi.org/10.1016/B978-0-12-805394-2.00001-5>
- Zorrilla, M., & García-Saiz, D. (2017). *Arquitecturas y tecnologías para el big data*.

Capítulo II

Instalación en Modo Pseudo-distribuido de Apache Hadoop HDFS, YARN y Spark con Ubuntu Server

- Instalaciones para el Nodo Único (Pseudo-Distribuido)
- Configuración del Nodo Único (Pseudo-Distribuido)
- Configuración del NameNode y ResourceManager
- Instalación de Apache Spark y Jupyter Notebook

Objetivos:

- Instalar y configurar un clúster Hadoop en Ubuntu Server y PuTTY.
- Implementar HDFS, YARN, Spark y MapReduce para procesamiento pseudo-distribuido.
- Integrar Anaconda y Jupyter Notebook como entorno de análisis de datos.
- Configurar la comunicación y servicios esenciales como SSH y variables de entorno.

Las herramientas Apache Hadoop HDFS, MapReduce, YARN y Apache Spark, conforma un software capaz de manejar grandes volúmenes de datos de manera muy eficiente. El ecosistema de Hadoop basa su funcionalidad en la división de datos en múltiples nodos, además de procesar y almacenar la información simultáneamente y de forma distribuida, lo que facilita su localización y acceso. El objetivo principal de este documento es presentar una guía detallada para la configuración e instalación de Apache Hadoop en modo pseudo-distribuido, junto con HDFS, YARN y Apache Spark sobre el sistema operativo Ubuntu Server. La herramienta PuTTY se utiliza para el acceso remoto al servidor. Este modo permite simular un entorno distribuido en una sola máquina, lo que facilita la realización de pruebas y el desarrollo antes de implementar un clúster completo.

Como primer punto, se aborda la instalación de Oracle VM VirtualBox y Ubuntu server, posteriormente, se explica la configuración e instalación de las herramientas de Big data mencionadas anteriormente, y se realizarán pruebas para la verificación de su correcto funcionamiento. La Figura 18, presenta un esquema general del proceso de instalación en modo pseudo-distribuido de Apache Hadoop y Apache Spark, destacando cada una de las etapas clave, desde la configuración del entorno y la conexión remota hasta la integración con Jupyter Notebook para análisis de datos.

"El volumen de datos crece exponencialmente, y nuestra capacidad para darle sentido determinará nuestro éxito."

(Simon, 2013)

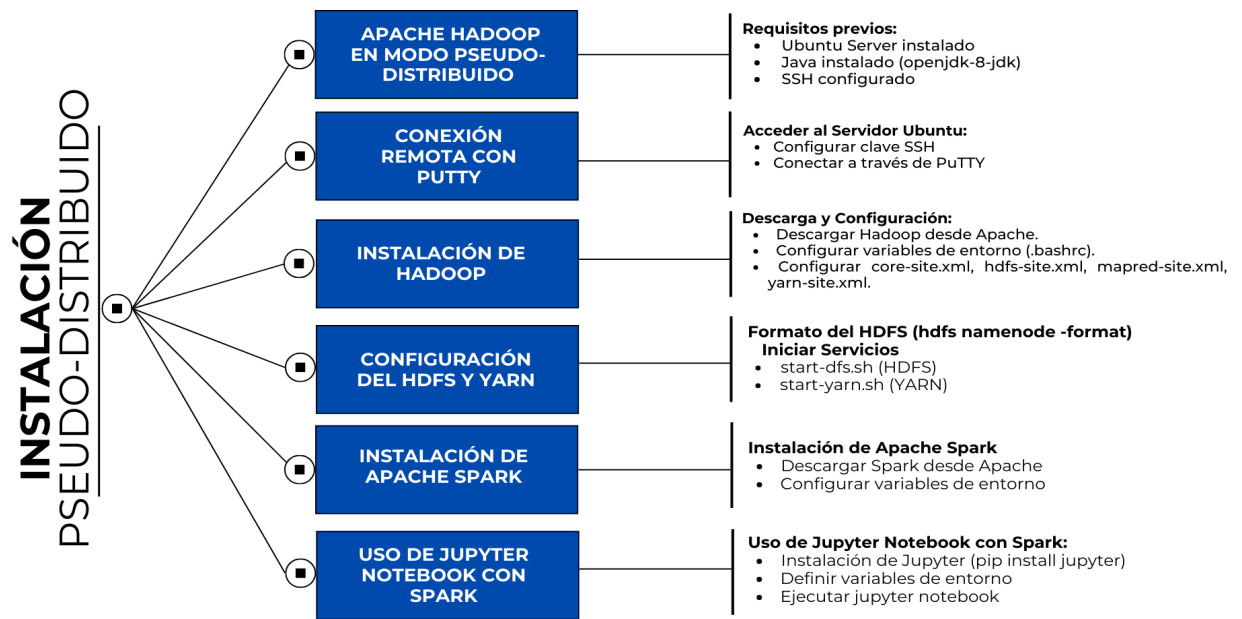


Figura 18. Resumen del Capítulo II
Fuente: Elaboración propia

Modos de instalación de Hadoop

Los modos de instalación de Hadoop definen cómo se configura y ejecuta en un entorno determinado. Existen algunos modos de instalación:

- **Standalone** (local). Hadoop se ejecuta como una aplicación Java única en un solo nodo. Es adecuado para pruebas, desarrollo o aprendizaje.
- Modo **Pseudo-distribuido**. Todo se ejecuta en una sola máquina, pero cada servicio (NameNode, DataNode, ResourceManager, etc.) se ejecuta como proceso separado. Es adecuado para pruebas más realistas porque simula un entorno distribuido real en una sola máquina. Requiere configuración de SSH y HDFS.
- **Modo Distribuido** (Fully-distributed Mode). Hadoop se instala en múltiples nodos (clúster real), donde los servicios están repartidos. Su uso es para poner en producción sistemas de grandes volúmenes de datos. Maneja alta escalabilidad y tolerancia a fallos. Requiere configuración compleja (red, permisos, sincronización). Su implementación puede ser local en equipos físicos, Máquinas Virtuales o contenedores Docker; también puede implementarse en la nube (AWS, Azure, GCP)
- **En línea en Google Colab**, para pruebas.

En la siguiente sección se explica el modo de instalación Pseudo-distribuido.

Configuración de Ubuntu Server en Oracle VM VirtualBox

La Máquina Virtual de Oracle VM VirtualBox permite ejecutar el sistema operativo Ubuntu Server para ejecutar un entorno pseudo-distribuido del ecosistema de Hadoop, ideal para pruebas de big data.

Pasos previos a la instalación de las herramientas de big data:

- Antes de la instalación de las herramientas, se requiere descargar e instalar VirtualBox para Windows, obtener la última versión.

<https://www.virtualbox.org/wiki/Downloads>

- Proceder con la instalación, de no ser posible instalar, es porque se requiere el Microsoft Visual C++ 2019 Redistributable, dependiendo del hardware disponible. Para 64 bits, descargue X64.

Oracle VM VirtualBox 7.0.18 needs the Microsoft Visual C++ 2019 Redistributable Package being installed first. Please install and restart the installation of Oracle VM VirtualBox 7.0.18.

<https://learn.microsoft.com/es-es/cpp/windows/latest-supported-vc-redist?view=msvc-170#visual-studio-2015-2017-2019-and-2022>

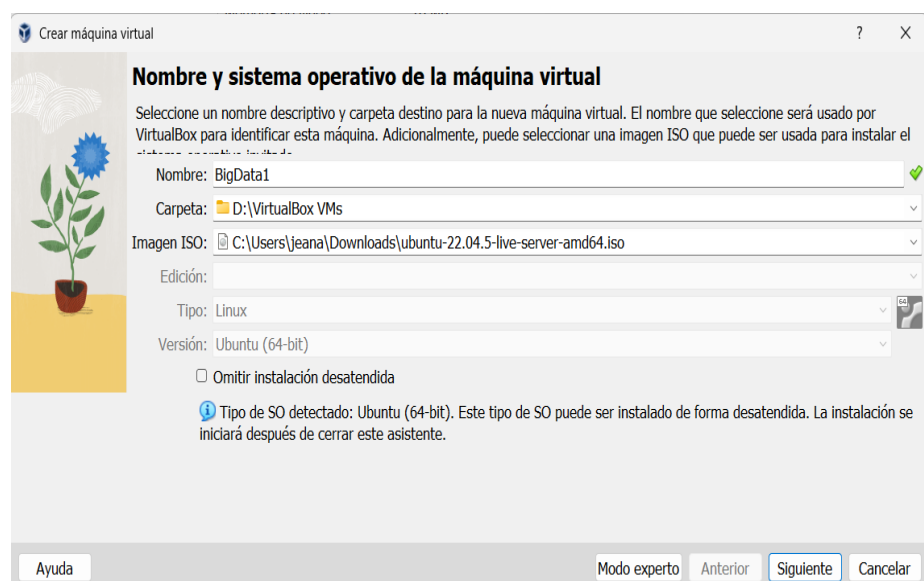
- Descargar Ubuntu, se escogió la versión server Ubuntu 22.04.4 LTS, por ser liviano y sin interfaz gráfica. Si tiene recursos computacionales, considere escoger una versión con interfaz gráfica

<https://releases.ubuntu.com/22.04/>

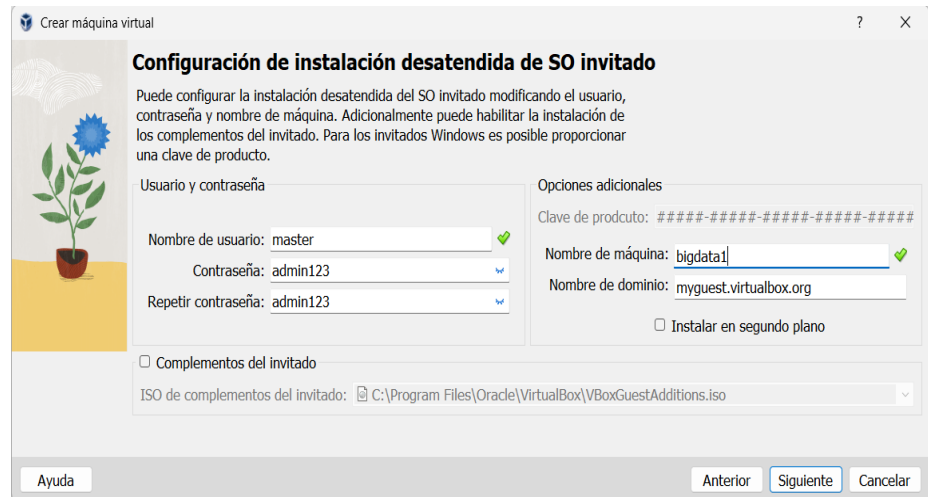
Link directo para descargar Ubuntu sin interfaz gráfica: [ubuntu-22.04.4-live-server-amd64.iso](https://releases.ubuntu.com/22.04.4-live-server-amd64.iso)

Pasos para la configuración de la máquina virtual Oracle VM VirtualBox:

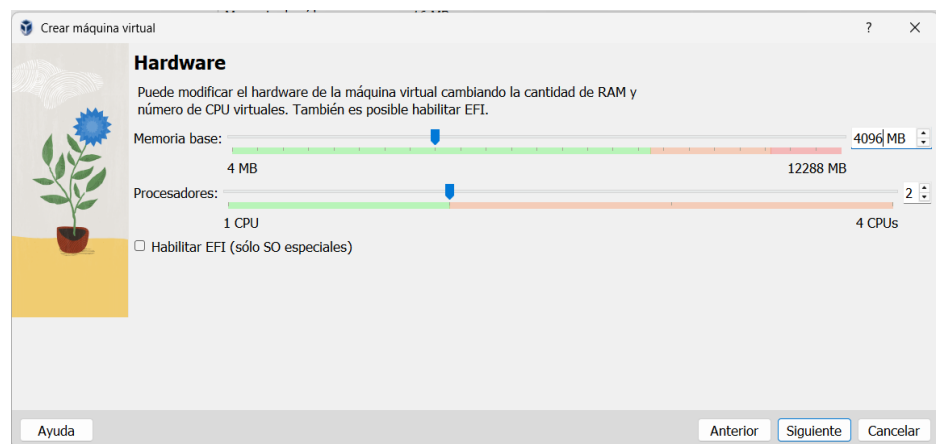
1. Hacer clic en el botón "Nueva" o presiona Ctrl + N. Introducir un nombre para la máquina virtual. Seleccionar el tipo de sistema operativo. Seleccionar la ruta del archivo .iso, del sistema operativo.



2. Introducir un nombre de usuario y de máquina. Asigna una contraseña a tu máquina virtual.



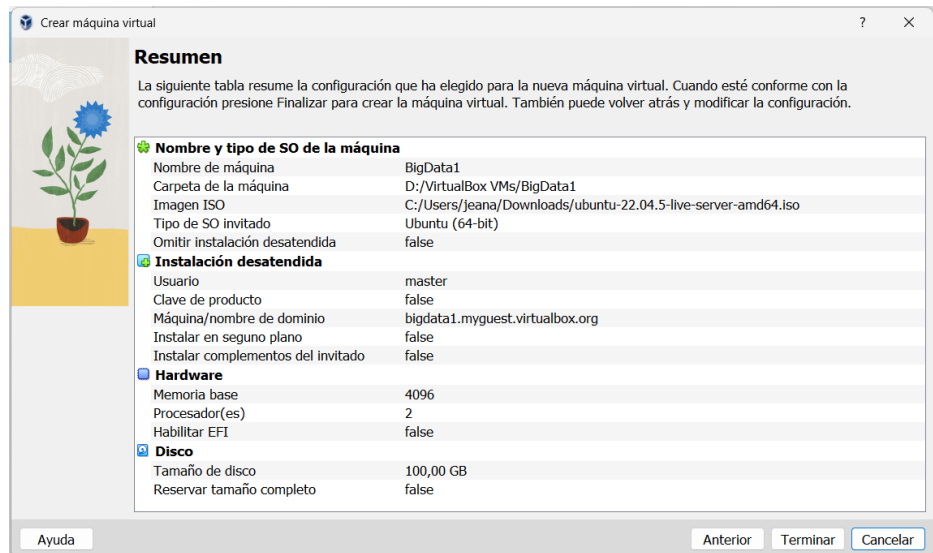
3. Asignar la cantidad de RAM recomendada para el sistema operativo (Ejemplo: 2GB para Linux o más si el sistema lo requiere).



4. Asignar la cantidad de disco duro virtual o almacenamiento para el sistema operativo. (se recomienda 10GB en adelante)



5. Comprobar el resumen de configuración de la máquina virtual y Terminar.



Instalación de servicios de Hadoop en un único nodo (máster/esclavo)

Sistema Operativo Ubuntu Server

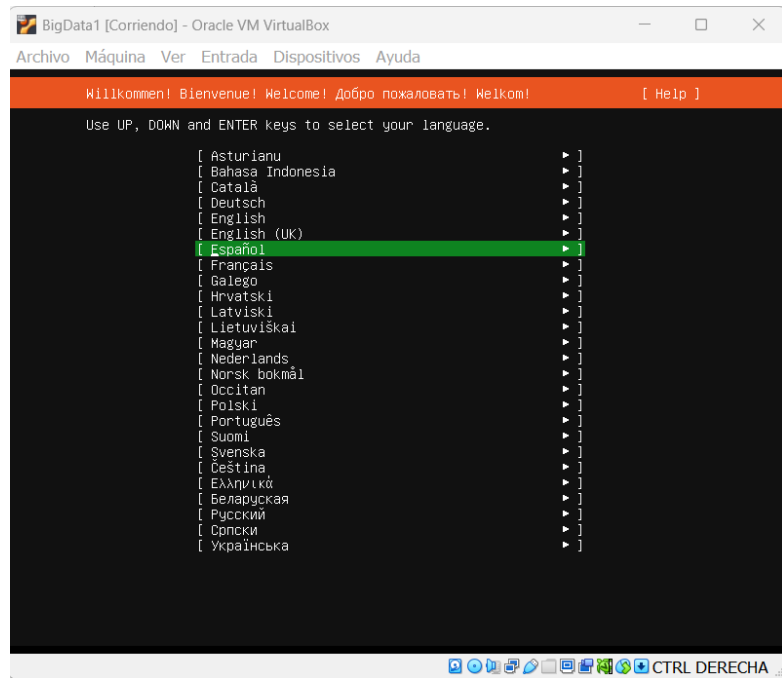
La instalación de Ubuntu Server es necesario y para luego realizar la implementación del ecosistema de Hadoop. Este proceso garantiza un entorno estable y compatible con las herramientas necesarias para el procesamiento de grandes volúmenes de datos.



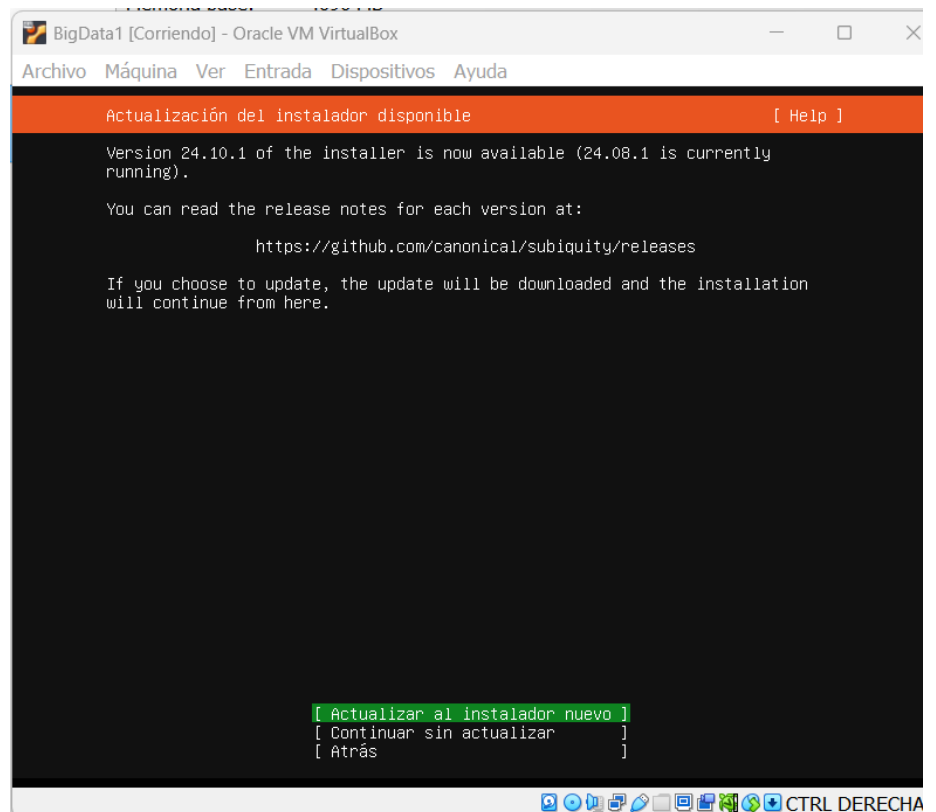
¡IMPORTANTE! Para comenzar, debemos instalar el sistema operativo Ubuntu Server, que actúa como anfitrión del nodo Máster. Dado que la instalación del clúster no tiene una interfaz gráfica, la navegación se realiza mediante el teclado.

Pasos para instalación del Sistema Operativo Ubuntu Server:

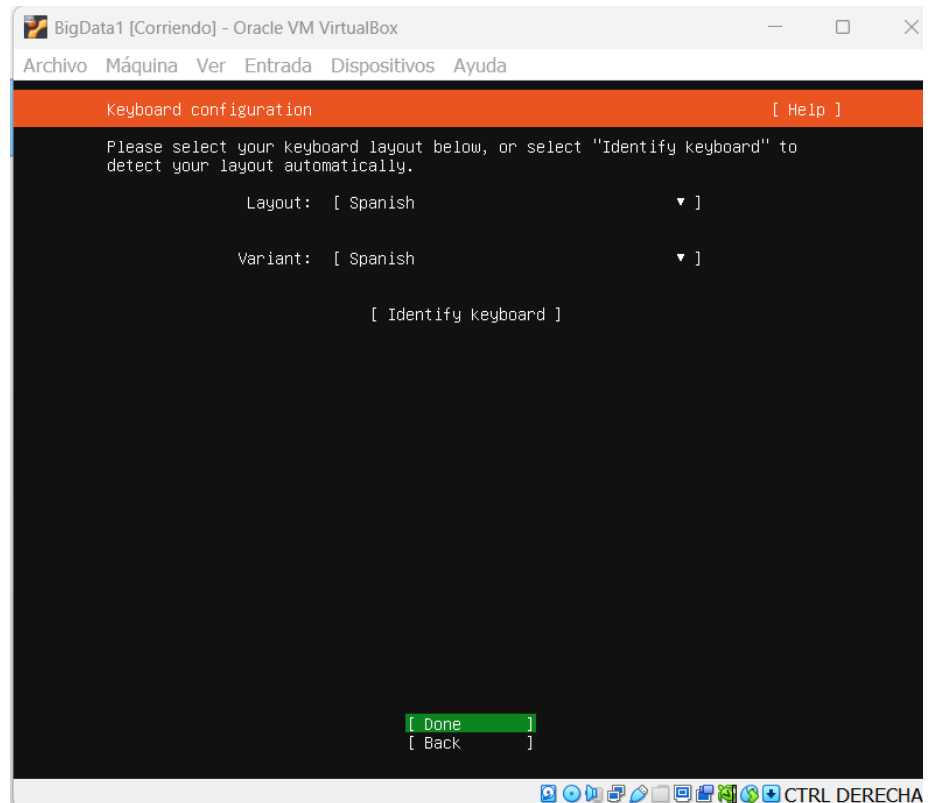
1. Usar las flechas para elegir "Español" y presionar Enter.



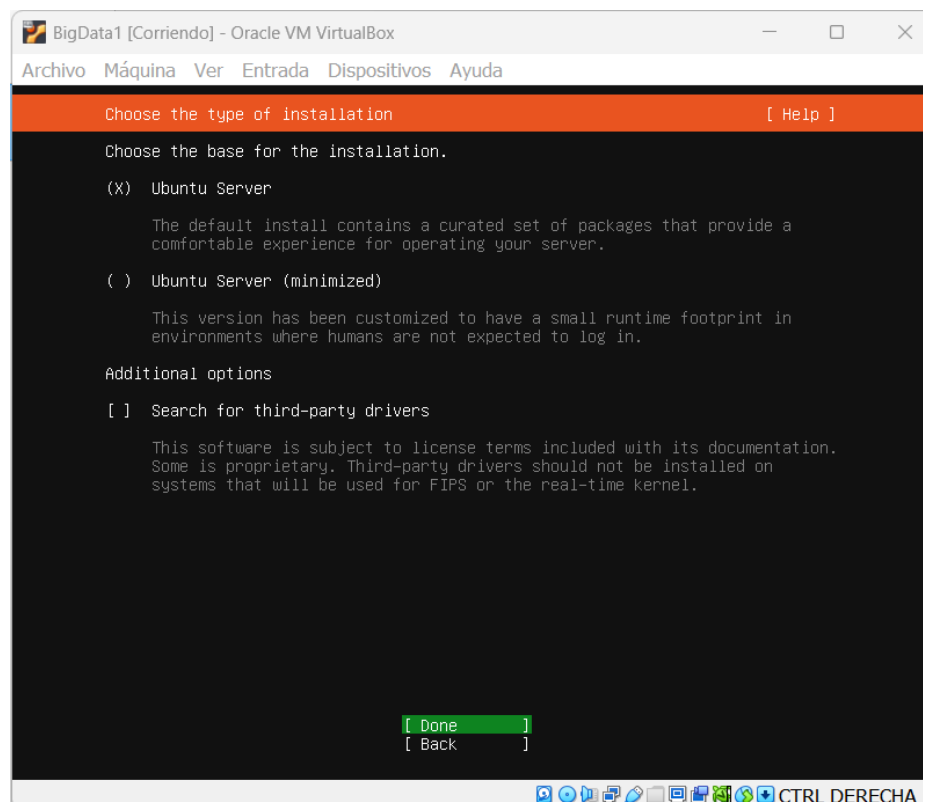
2. Seleccionar la opción "Actualizar al instalador nuevo" y continuar.



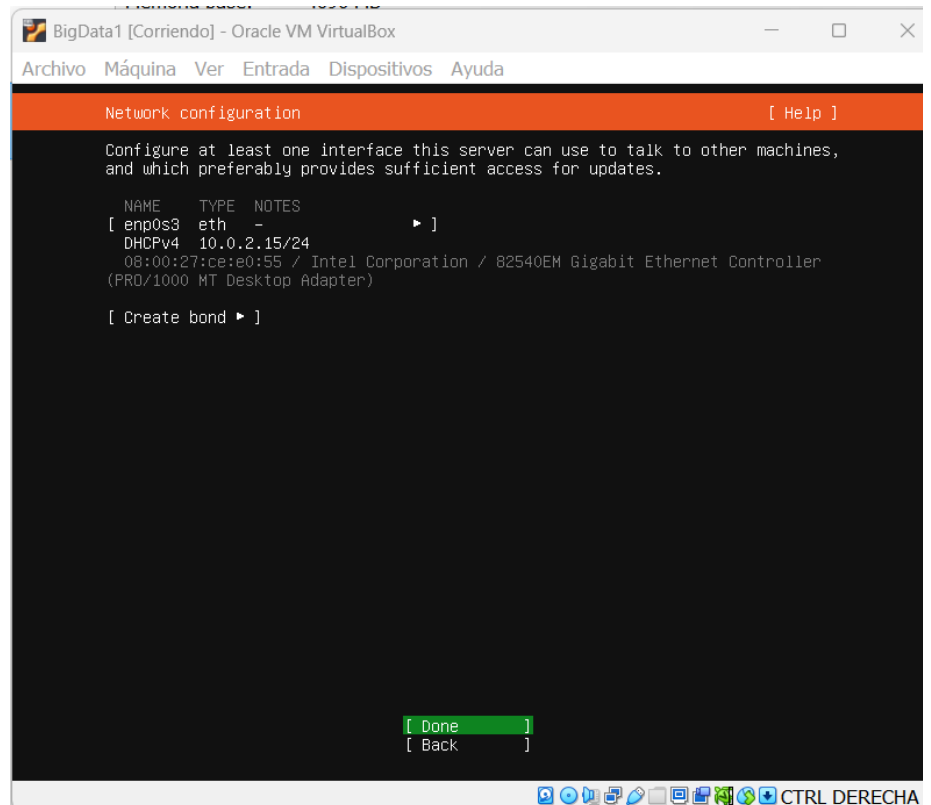
3. Seleccionar el idioma del teclado (español - Latinoamérica o español - España). Presiona Enter para continuar.



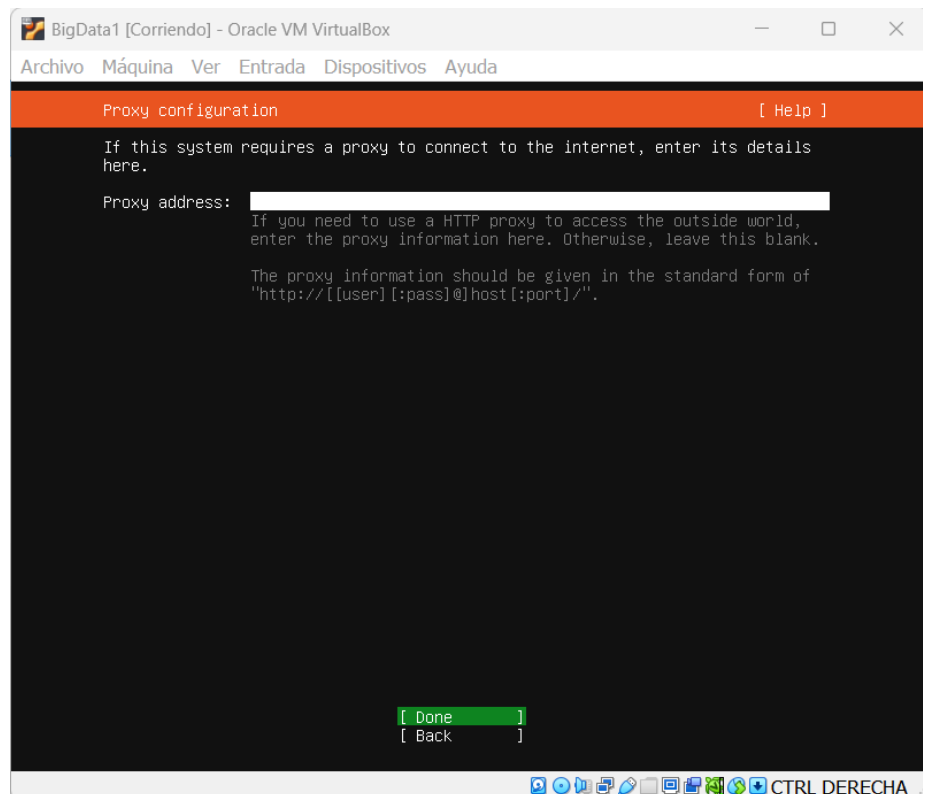
4. Seleccionar la opción "Ubuntu Server". Presiona Enter para continuar.



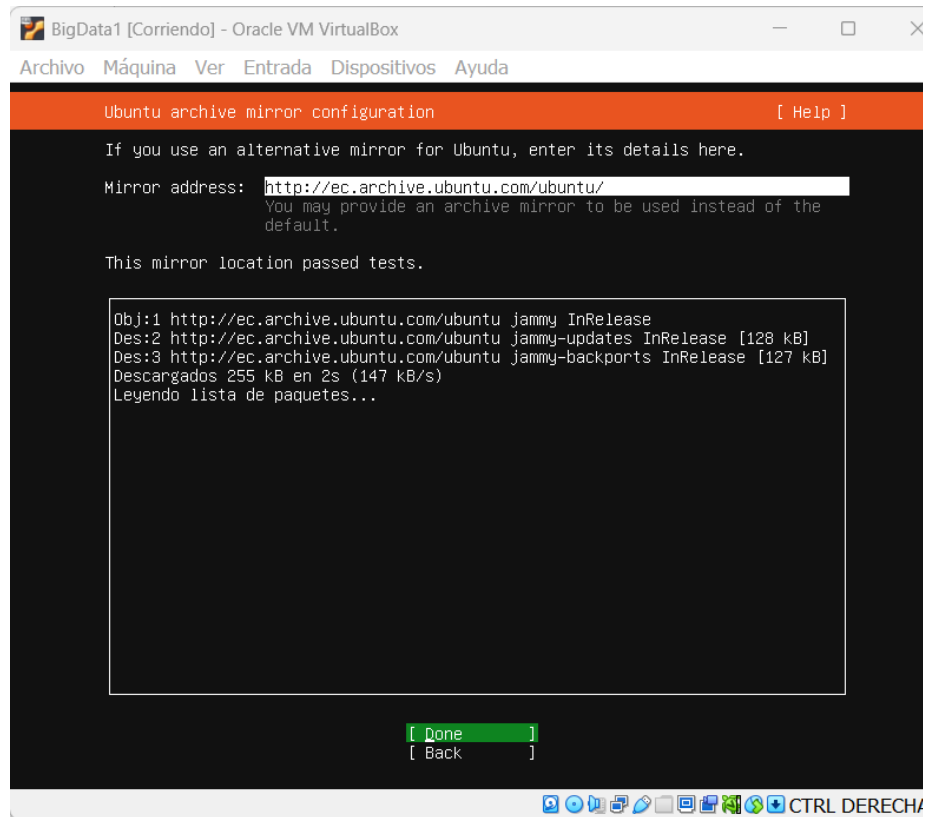
5. Ubuntu detectará de manera automática la conexión por cable. Presionar Enter para continuar.



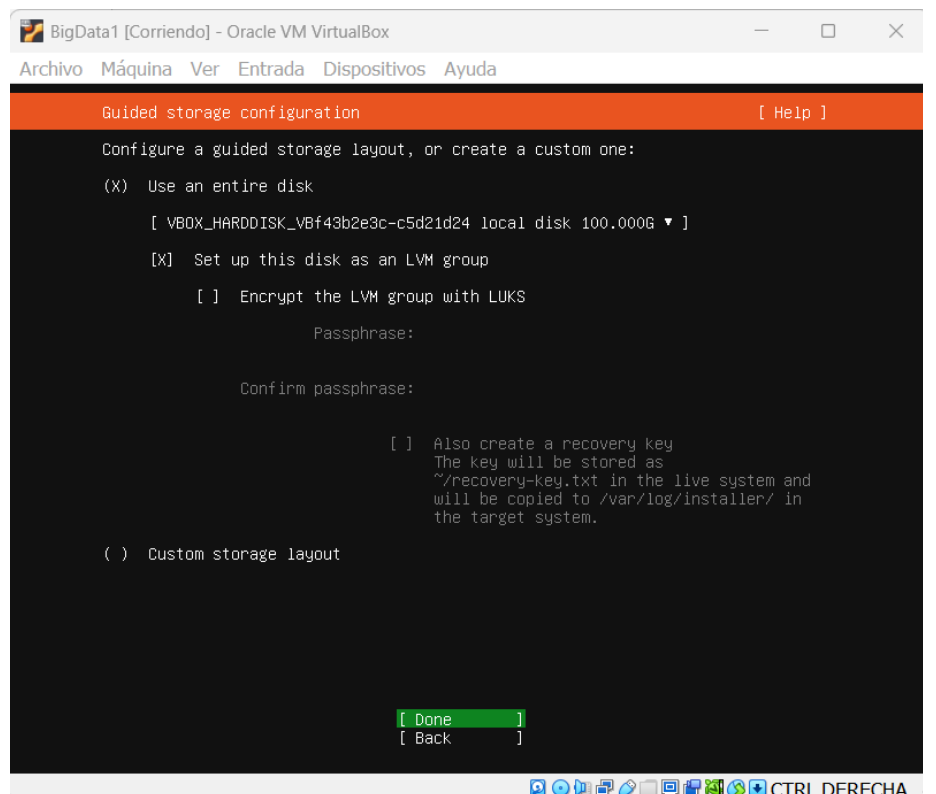
6. Si se usa un proxy, ingresarlo en el campo. Si no, dejarlo vacío y presionar Enter.



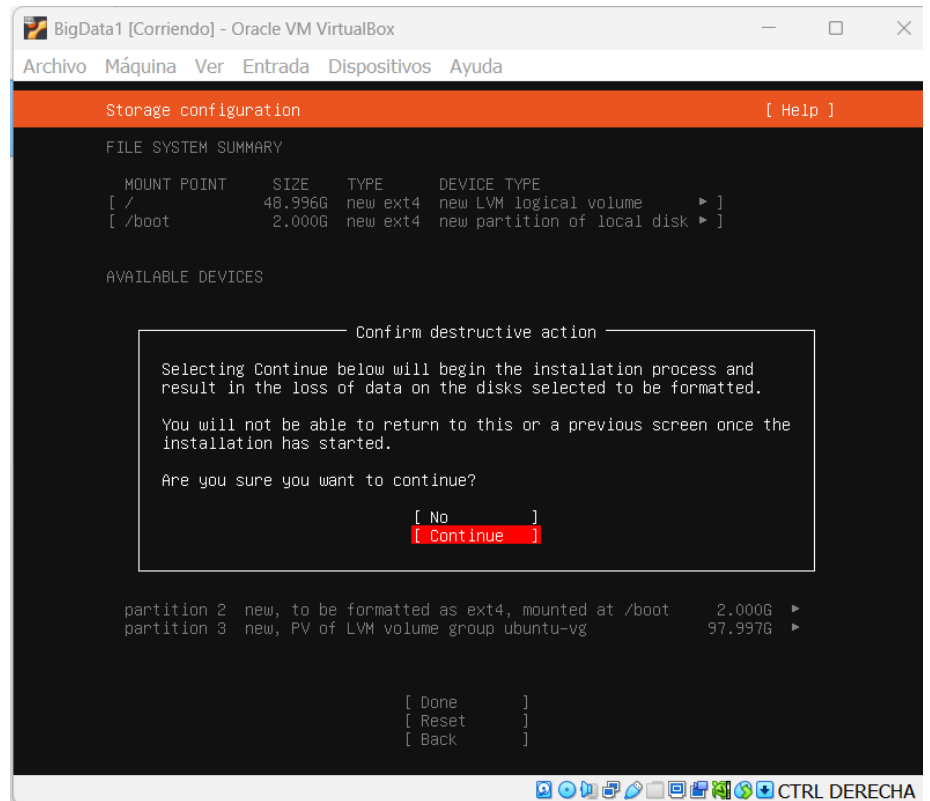
7. El instalador detecta automáticamente el mejor mirror. Presionar Enter para continuar.



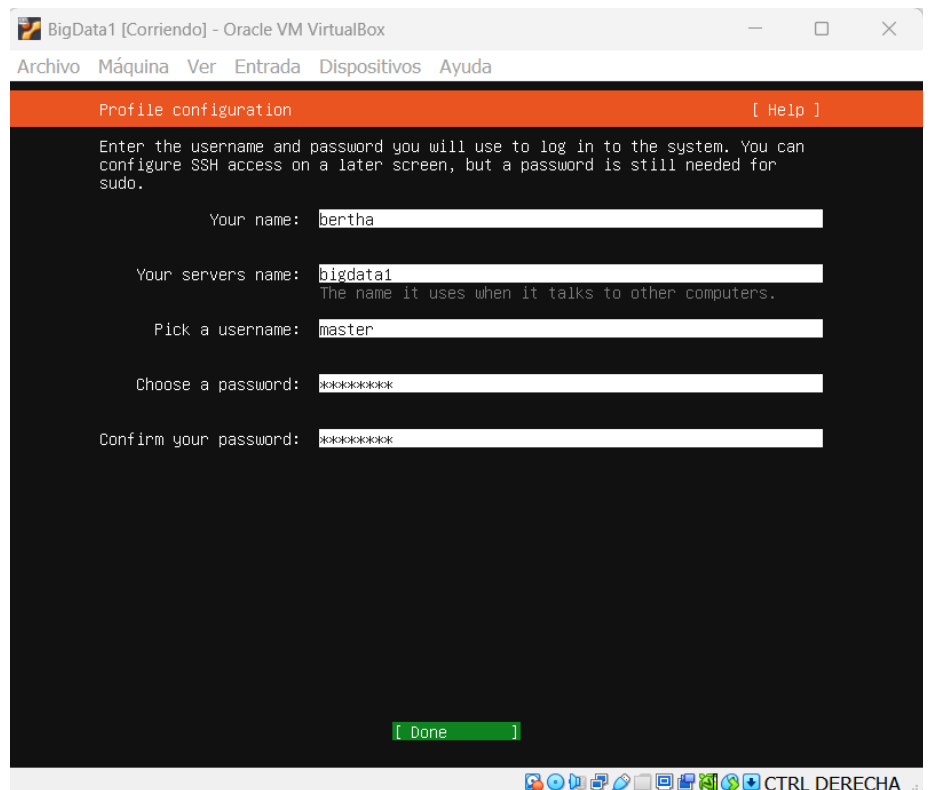
8. Seleccionar una opción de particionado. Confirmar los cambios en el disco con Enter.



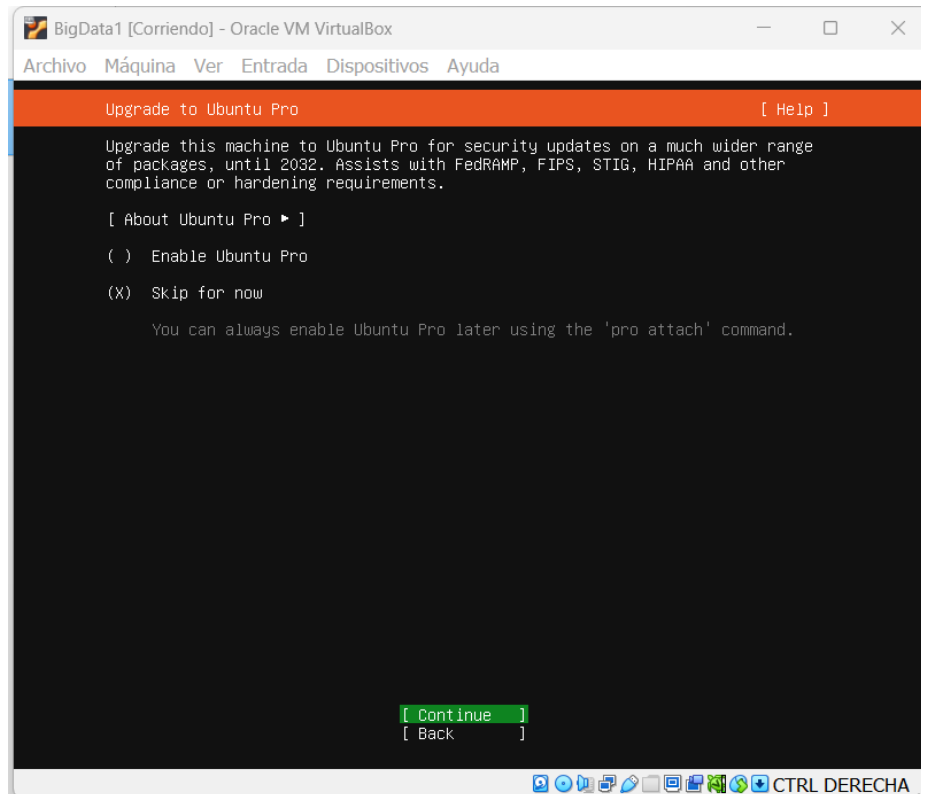
9. Presionar Enter para continuar. Seleccionar "Continue".



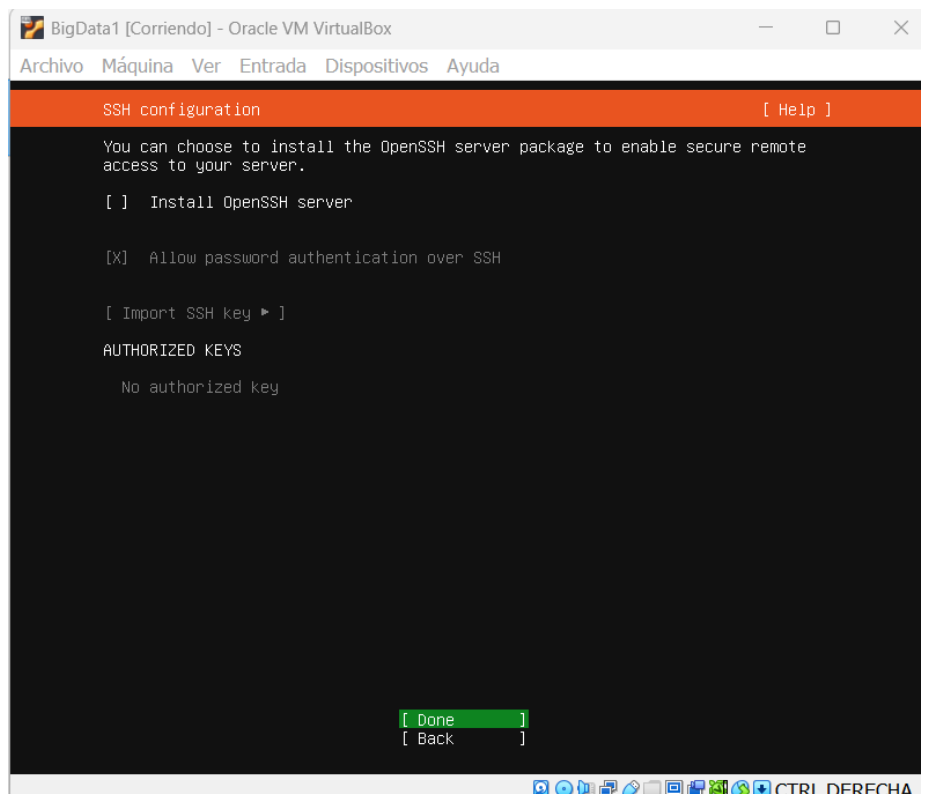
10. Introducir el nombre de usuario de la configuración inicial de la máquina virtual y la misma contraseña.



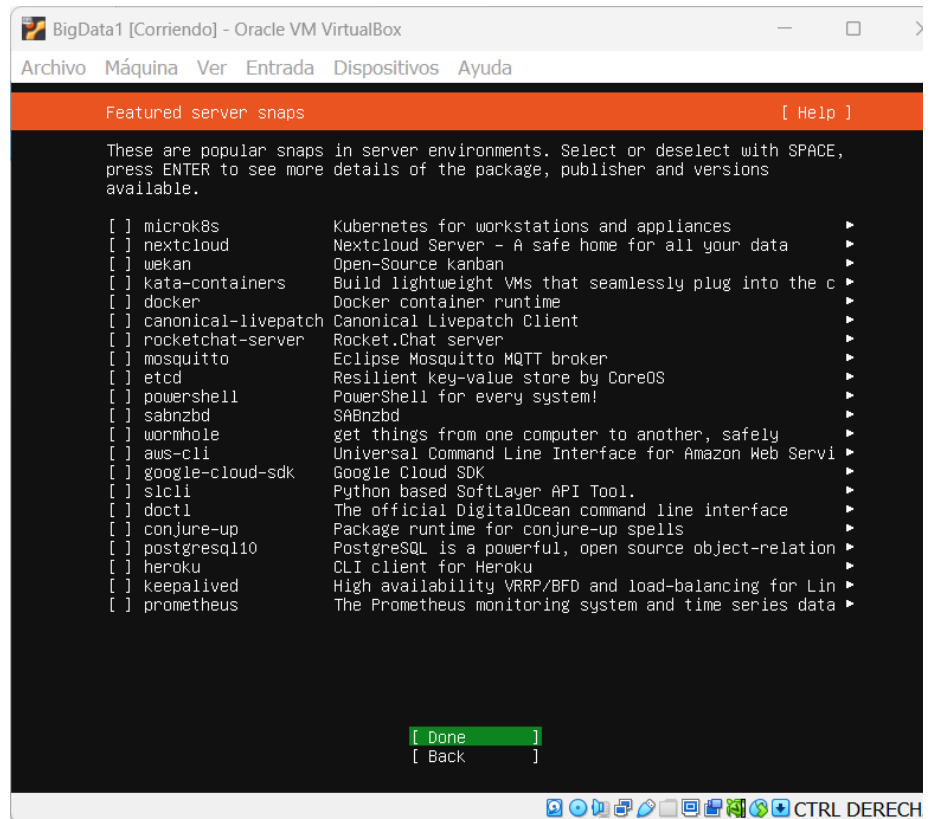
11. Seleccionar la opción "Skip for now". Presionar "Enter" para continuar.



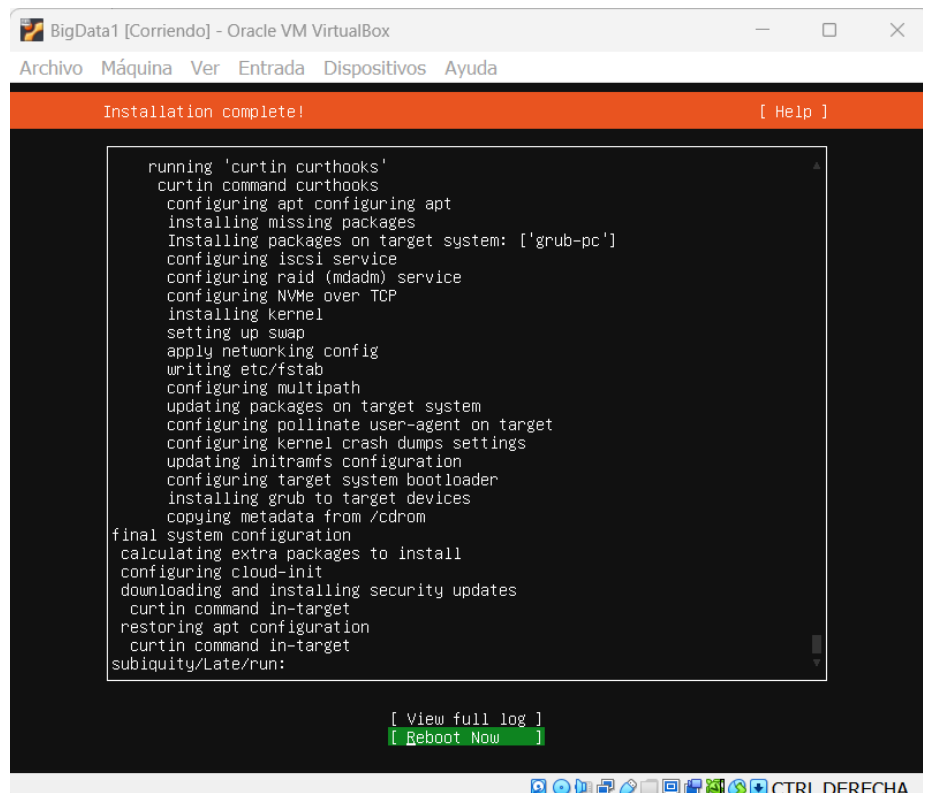
12. Elegir la configuración por defecto. Presionar Enter para continuar.



13. Se puede instalar software adicional. Si no es necesario, mantener por defecto. Presionar Enter para continuar.



14. Cuando finalice la instalación, seleccionar "Reiniciar". Iniciar sesión con el usuario y contraseña creados (En este caso el usuario: **master**)



15. Al Finalizar te la instalación, se muestra la siguiente pantalla.

```

BigData1 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
:02:56 +0000. Datasource DataSourceNone. Up 23.27 seconds
master
Password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-130-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of mar 21 ene 2025 02:03:12 UTC

System load:  0.32          Processes:           117
Usage of /:   14.2% of 47.9GB Users logged in:      0
Memory usage: 5%          IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%

El mantenimiento de seguridad expandido para Applications está desactivado

Se pueden aplicar 23 actualizaciones de forma inmediata.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

master@bigdata1:~$ _

```

16. Actualizar las librerías con el comando **"sudo apt-get update"**

```

BigData1 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda
Usage of /:   14.2% of 47.9GB Users logged in:      0
Memory usage: 5%          IPv4 address for enp0s3: 10.0.2.15
Swap usage:   0%

El mantenimiento de seguridad expandido para Applications está desactivado

Se pueden aplicar 23 actualizaciones de forma inmediata.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

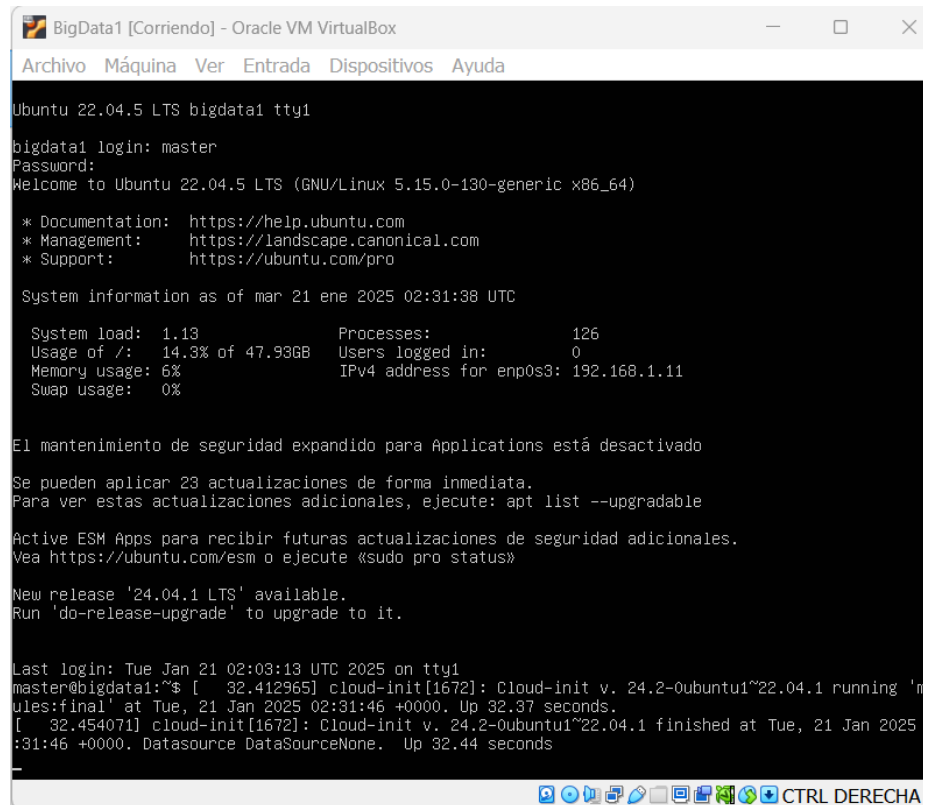
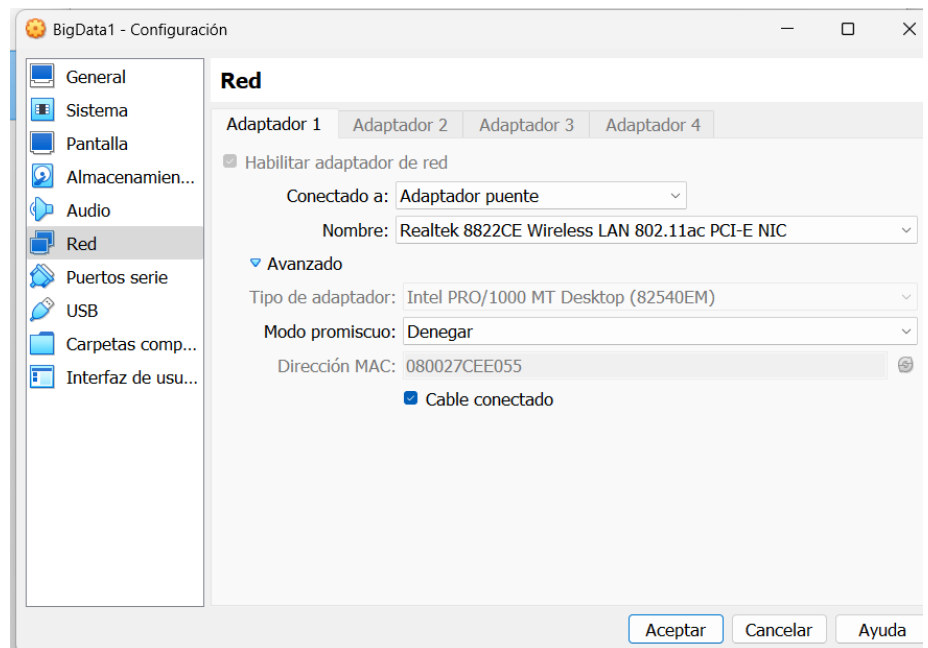
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

master@bigdata1:~$ sudo apt-get update
[sudo] password for master:
Obj:1 http://ec.archive.ubuntu.com/ubuntu jammy InRelease
Des:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Des:3 http://ec.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Obj:4 http://ec.archive.ubuntu.com/ubuntu jammy-backports InRelease
Des:5 http://ec.archive.ubuntu.com/ubuntu jammy/main Translation-es [332 kB]
Des:6 http://ec.archive.ubuntu.com/ubuntu jammy/restricted Translation-es [964 B]
Des:7 http://ec.archive.ubuntu.com/ubuntu jammy/universe Translation-es [1.356 kB]
Des:8 http://ec.archive.ubuntu.com/ubuntu jammy/multiverse Translation-es [68,2 kB]
Descargados 2.015 kB en 2s (858 kB/s)
Leyendo lista de paquetes... Hecho
master@bigdata1:~$

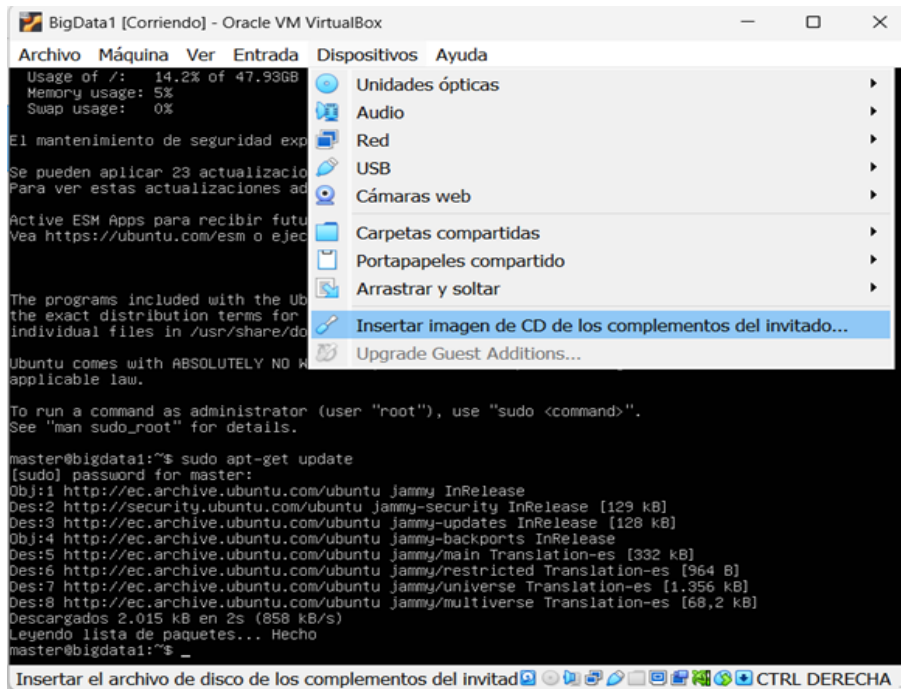
```

17. Para la configuración de red de la máquina virtual, hacer clic en **"Configuración"** (Ctrl + S). Ir a la pestaña **"Red"** en el panel lateral izquierdo. En la pestaña **"Adaptador 1"**, marcar la casilla **"Habilitar adaptador de red"**, y elegir el modo de conexión **"Adaptador puente"** y seleccionar el nombre de la tarjeta de red. Reiniciar la máquina virtual, de esta manera se asigna una ip automáticamente.



Instalación de VBoxLinuxAdditions

En el menú de VirtualBox seleccionar Dispositivo -> Insertar imagen de CD de las **«Guest Additions»**:



Actualizar los paquetes del sistema, y montar dispositivo /mnt

```
sudo apt-get update
sudo mount /dev/cdrom /mnt
```

Acceder al directorio /mnt

```
cd /mnt
sudo apt install bzip2
sudo ./VBoxLinuxAdditions.run
```

Ingresar a Ubuntu desde el sistema operativo anfitrión (Windows) utilizando PuTTY

PuTTY es un emulador gratuito de terminal que soporta SSH y muchos otros protocolos. La mayoría de usuarios, especialmente los que trabajan sobre sistemas operativos Windows, lo encuentran muy útil a la hora de conectar a un servidor Unix o Linux a través de SSH (*Actualidad.es*, 2020).

A continuación, se detallan los pasos para la instalación y configuración de PuTTY.

Descargar PuTTY:

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Elegir la versión acorde a su sistema operativo. Por ejemplo, para Windows de 64-bit x86

<https://the.earth.li/~sgtatham/putty/latest/w64/putty-64bit-0.81-installer.msi>

Acceder al terminal de Ubuntu, proceder con la instalación de OpenSSH Server.

Escribir el siguiente comando en la terminal y presionar Enter:

```
sudo apt install openssh-server
```

Una vez completada la instalación, puedes verificar que el servicio de OpenSSH Server

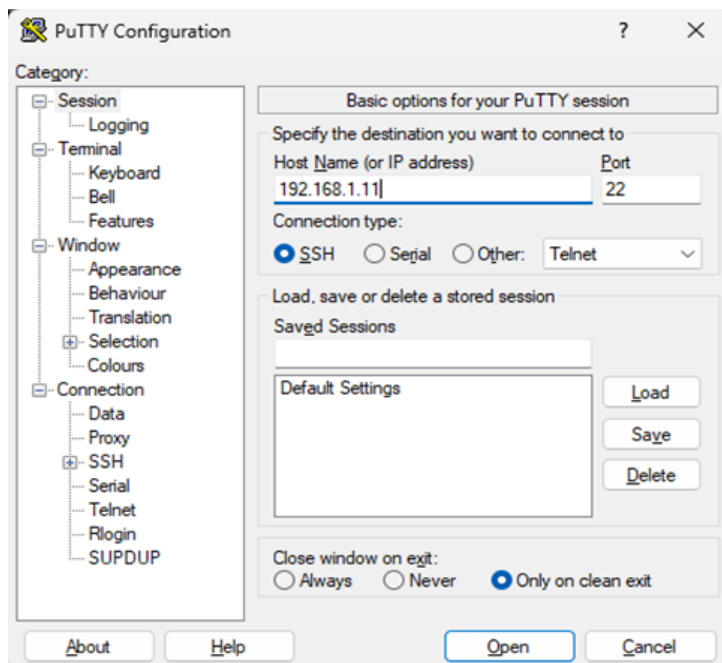
```
sudo systemctl start ssh
sudo systemctl enable ssh
sudo systemctl status ssh
```

Comprobar la conexión a Ubuntu mediante PuTTY. Colocar la IP de Ubuntu que se muestra en pantalla anterior.

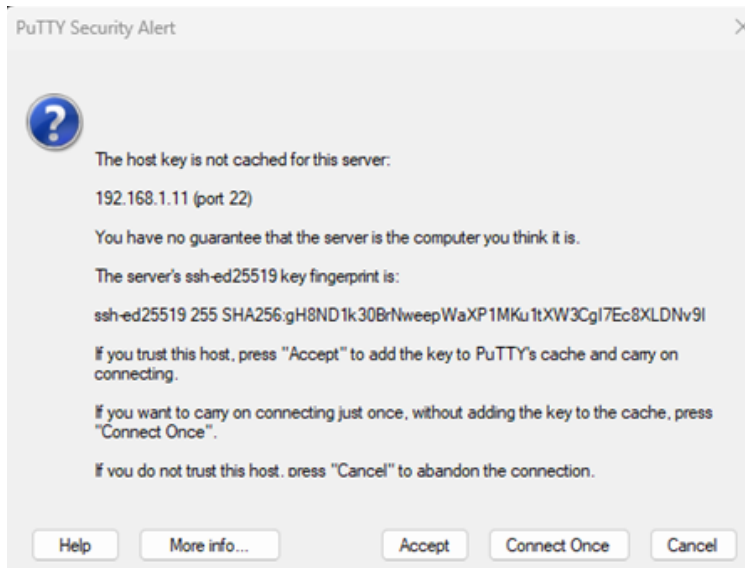


IMPORTANTE!

Asegúrate de que los servicios SSH estén activados y en funcionamiento, de lo contrario, PuTTY no podrá conectarse a Ubuntu Server en VirtualBox.



Seleccionar la opción **"Accept"**, para acceder.



Ingresar con el usuario **master** y su respectiva contraseña **admin**. Este usuario se estableció en la creación de la máquina virtual.



¡IMPORTANTE!

Usa PuTTY para gestionar el nodo Máster en Ubuntu Server con múltiples terminales y monitorear varios procesos al mismo tiempo.

```

master@bigdata1: ~
login as: master
master@192.168.1.11's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 5.15.0-130-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of mar 21 ene 2025 02:42:06 UTC

System load:  0.01               Processes:            117
Usage of /:   14.3% of 47.93GB   Users logged in:     1
Memory usage: 6%                IPv4 address for enp0s3: 192.168.1.11
Swap usage:   0%

El mantenimiento de seguridad expandido para Applications está desactivado
Se pueden aplicar 23 actualizaciones de forma inmediata.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

Active ESM Apps para recibir futuras actualizaciones de seguridad adicionales.
Vea https://ubuntu.com/esm o ejecute «sudo pro status»

New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jan 21 02:31:39 2025
master@bigdata1:~$

```


Instalación de Java JDK Java 8

Según Cruz Alejandro (2021), es importante tener en cuenta la compatibilidad de las dos herramientas, en este caso Apache Hadoop y Java, para un correcto funcionamiento del ecosistema de Hadoop.

Desde este punto en adelante continuaremos el proceso mediante la ventana de PuTTY. Se escogió JAVA 8, debido a que la versión 11 presenta algunos errores, en la gestión posterior de Hadoop.

Instalación y verificación de la **versión de JAVA**

```
sudo apt update && sudo apt install openjdk-8-jdk
java -version
```

Instalación **OpenSSH Client**, y conjuntos de herramientas esenciales

```
sudo apt install openssh-client
sudo apt install wget apt-transport-https
gnupg2 software-properties-common
```

Configurar un usuario Hadoop con acceso SSH

Crear un **usuario hadoop**

```
sudo adduser hadoop
password admin
```

Cambiar al usuario creado, y Agregar al **usuario hadoop** al grupo sudo.

```
sudo su - hadoop
sudo usermod -aG sudo hadoop
```

Generar pares de **claves públicas y privadas**.

```
ssh-keygen -t rsa
```

Agregar la clave pública generada de **id_rsa.pub** a **authorized_keys**.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Cambiar los permisos del archivo **authorized_keys**.

```
sudo chmod 640 ~/.ssh/authorized_keys
```

Verificar si el **SSH** funciona sin contraseña.

```
ssh localhost
```

Instalación de Apache Hadoop (HDFS, MAPREDUCE y YARN)

Para (Candel, 2022b), "Hadoop abarca diversas herramientas diseñadas e implementadas para trabajar juntas. Hadoop es un ecosistema integrado por muchos componentes, que van desde el almacenamiento de datos hasta la integración, el procesamiento de estos o las herramientas especializadas para analistas de datos".

A continuación, se detallan los pasos para la instalación y configuración de Apache Hadoop.

Descargar el archivo .gz de Hadoop

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz
```

Descomprimir el archivo **.gz** de Hadoop, y renombrar el **directorio a Hadoop**

```
tar -xvzf hadoop-3.3.6.tar.gz
mv hadoop-3.3.6 hadoop
```

Ver en qué directorio se ha instalado JAVA

```
dirname $(dirname $(readlink -f $(which java)))
```

Configurar las variables de entorno

```
nano ~/.bashrc
```

Se abre un archivo y colocar al final las siguientes líneas. Guardar con **ctrl+o** <<enter>> y salir con **ctrl+x**.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Activar o **actualizar las variables de entorno**

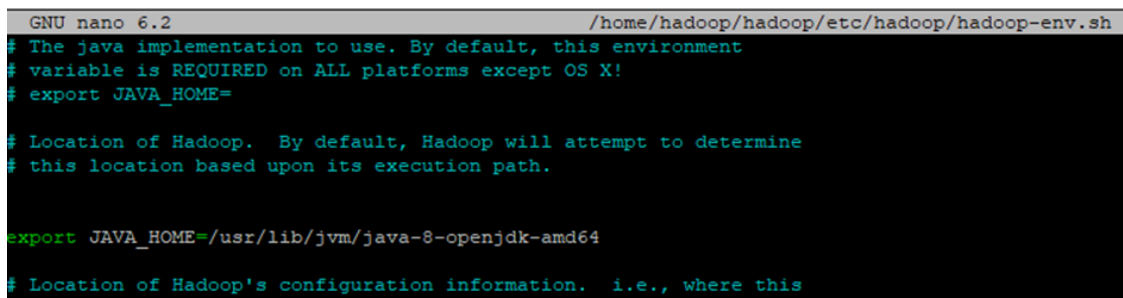
```
source ~/.bashrc
```

Editar el archivo **hadoop-env.sh**

```
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Agregar la línea

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



```
GNU nano 6.2 /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
# export JAVA_HOME=

# Location of Hadoop. By default, Hadoop will attempt to determine
# this location based upon its execution path.

export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop's configuration information. i.e., where this
```

Configuración de Hadoop en modo Pseudo-distribuido.

En el trabajo de (Valdez Menchaca et al., 2020), en un clúster de Hadoop existen dos tipos de nodos:

- **Namenode:** nodo principal o nodo maestro del clúster, contiene los metadatos para HDFS durante el procesamiento de los datos que se distribuyen entre los nodos.
- **Datanode:** estos son los nodos del clúster del sistema del que almacenan los bloques de datos reales de HDFS; estos bloques se replican en varios nodos para brindar soluciones de alta calidad.

Primero, se deberá crear los directorios **namenode** y **datanode** dentro de la carpeta del usuario de Hadoop.

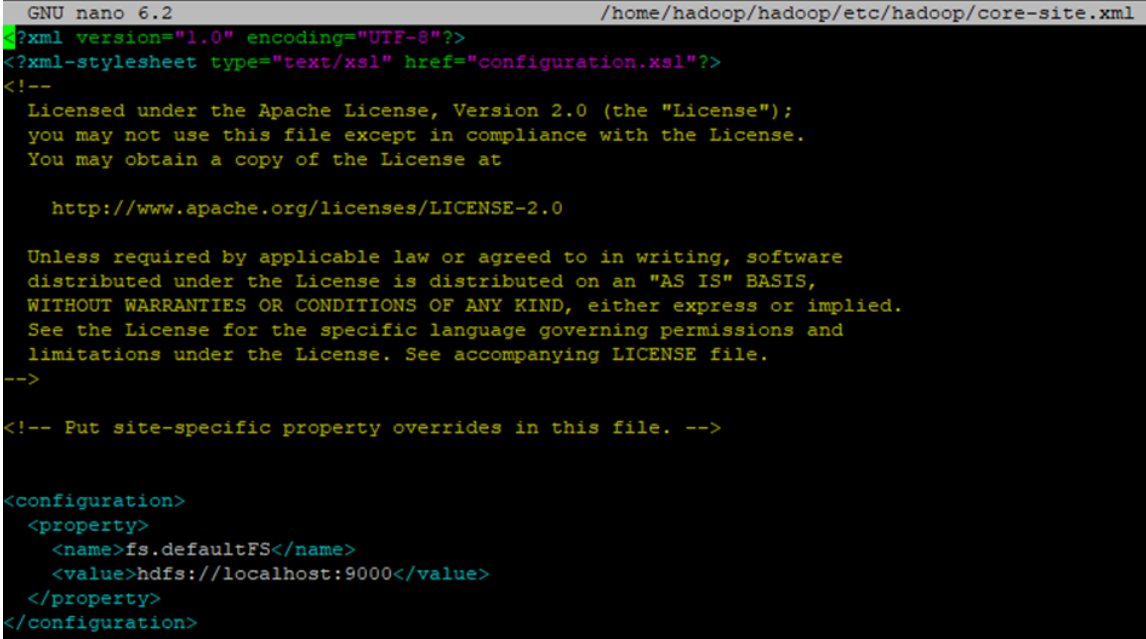
```
mkdir -p ~/hadoopdata/hdfs/{namenode,datanode}
```

Editar el archivo `core-site.xml`

```
nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

Agregar las líneas:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```



```
GNU nano 6.2 /home/hadoop/hadoop/etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Editar el archivo `hdfs-site.xml`

```
nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Agregar las líneas:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

```
GNU nano 6.2 /home/hadoop/hadoop/etc/hadoop/hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

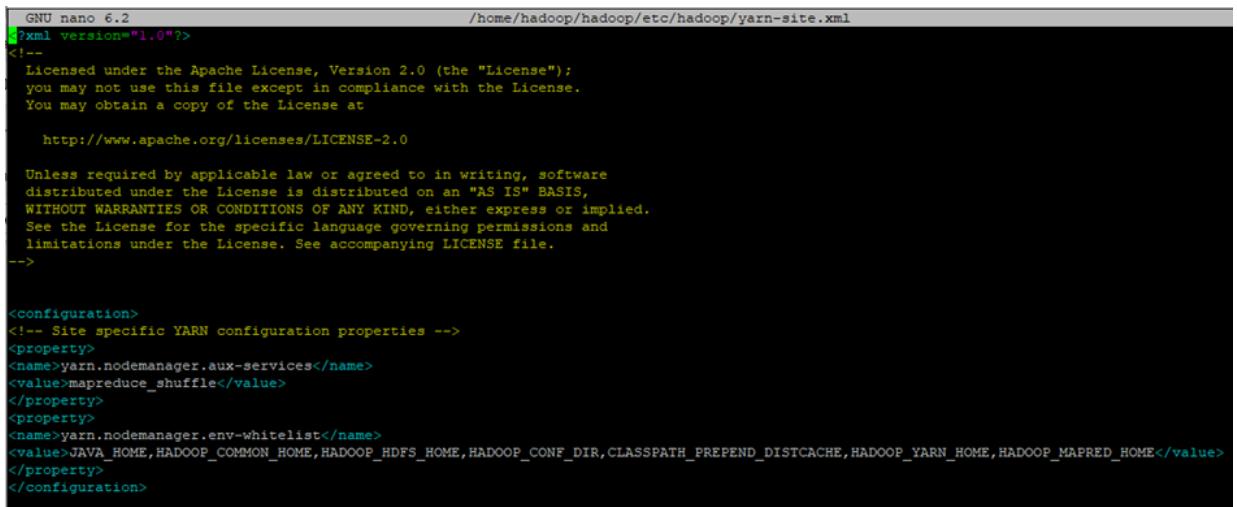
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

Editar el archivo yarn-site.xml

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

Agregar las líneas en el archivo **yarn-site.xml**:

```
<configuration>
  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>
      JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH
      H_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME
    </value>
  </property>
</configuration>
```



```
GNU nano 6.2 /home/hadoop/hadoop/etc/hadoop/yarn-site.xml
<?xml version="1.0"?>
<!--
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License. See accompanying LICENSE file.
-->

<configuration>
  <!-- Site specific YARN configuration properties -->
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.env-whitelist</name>
    <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
  </property>
</configuration>
```

Editar el archivo **mapred-site.xml**

```
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

Agregar las líneas:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

```

GNU nano 6.2 /home/hadoop/hadoop/etc/hadoop/mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>

```

Formatear el HDFS

```
$HADOOP_HOME/bin/hdfs namenode -format
```

Otra forma más simple de ejecutar el comando

```
hdfs namenode -format
```

Instalación y Configuración de **firewall**

```

sudo apt-get install firewallld
sudo firewall-cmd --permanent --add-port=9000/tcp
sudo firewall-cmd --permanent --add-port=9870/tcp
sudo firewall-cmd --permanent --add-port=8042/tcp
sudo firewall-cmd --permanent --add-port=8088/tcp
sudo firewall-cmd -reload

```

Pasos finales, **levantar todos los servicios y comprobar los procesos con jps**

```

start-all.sh
jps

```

Parar todos los servicios

```
stop-all.sh
```

Acceso a las interfaces web de los servicios levantados

Verificar en local a través de un **navegador web**, siempre que el sistema operativo Ubuntu tenga interfaz de escritorio.

**¡IMPORTANTE!**

La IP en la URL (como <http://192.168.1.11:9870>) puede cambiar según la red del equipo. Verifica siempre la IP local del nodo maestro para acceder correctamente a la interfaz de Hadoop.

HDFS NameNode : <http://localhost:9870>

YARN NodeManager : <http://localhost:8042>

YARN Resource Manager (Yarn) : <http://localhost:8088/cluster>

Verificar la ejecución de Hadoop desde el sistema operativo anfitrión

En este caso desde Windows, a través de un navegador web, utilizar la dirección IP con la que se instaló Ubuntu.

Verificar los servicios levantados desde un **navegador de Windows**:

HDFS NameNode: <http://192.168.1.11:9870>

Overview 'localhost:9000' (active)

Started:	Mon Jan 20 23:51:17 -0500 2025
Version:	3.3.6, r1be7823872bda9266a4f8b195058f0d012bf9c
Compiled:	Sun Jun 18 03:22:00 -0500 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-ea20be02-96e2-4c84-86df-f1da93872ec1
Block Pool ID:	BP-1636326706-127.0.1.1-1737434205026

Summary

Security is off.
Safemode is off.
1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
Heap Memory used 106.39 MB of 228 MB Heap Memory. Max Heap Memory is 969.5 MB.
Non Heap Memory used 49.45 MB of 51.11 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	47.53 GB
Configured Remote Capacity:	0 B
DFS Used:	28 KB (0%)
Non DFS Used:	9.53 GB
DFS Remaining:	35.53 GB (74.96%)
Block Pool Used:	28 KB (0%)
DataNodes usages% (Min/Median/Max/1stdDev):	0.00% / 0.00% / 0.00% / 0.00%

YARN NodeManager: <http://192.168.1.11:8042>

NodeManager information

Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
Resource types	memory-mb (unit=Mi), vcores
NodeHealthyStatus	true
LastNodeHealthTime	Tue Jan 21 04:51:36 UTC 2025
NodeHealthReport	
NodeManager started on	Tue Jan 21 04:51:33 UTC 2025
NodeManager Version:	3.3.6 from 1be7823872bda9266a4f8b195058f0d012bf9c by ubuntu source checksum d42eb795a5eadb0febf5e44a787a9 on 2023-06-18T08:31Z
Hadoop Version:	3.3.6 from 1be7823872bda9266a4f8b195058f0d012bf9c by ubuntu source checksum 5652179ad55f76cb287d9c633be53bbd on 2023-06-18T08:22Z

YARN Resource Manager (Yarn) : <http://192.168.1.11:8088/cluster>

No seguro

192.168.1.11:8080/cluster

</

Instalación de Apache Spark

Macías et al. (2016) señalan que “en un sentido metafórico, Apache Spark pretende ser esa chispa de velocidad que permite a las aplicaciones que se ejecutan desde internet ser más eficientes”.

Spark dispone de APIs nativas para los lenguajes Scala (en el que está diseñado), Java, Python y R, que ofrecen un estilo de programación más sencillo. Puede ejecutarse de forma local en sistemas operativos como Windows, Linux y OS X, así como en Hadoop, lo que permite una gran integración con los servicios del ecosistema Hadoop (Ortiz Hernández, 2021). Para realizar la instalación de Apache Spark debemos asegurarnos de que Scala y Java estén instalados correctamente. A continuación, se debe descargar la versión más reciente en su sitio oficial, extraer el archivo y configurar sus variables de entorno.

A continuación, se detallan los pasos para la instalación y configuración de Apache Spark.

Instalar Scala en Ubuntu

```
sudo apt-get install scala
```

Descargar la versión más reciente de Spark:

<https://spark.apache.org/downloads.html>

```
wget https://d1cdn.apache.org/spark/spark-3.5.4/spark-3.5.4-bin-hadoop3.tgz
```

Descomprimir, y mover el directorio spark-3.5.4-bin-hadoop3/ a **/opt/spark**

```
sudo mv spark-3.5.4-bin-hadoop3/ /opt/spark
tar -xvzf spark-3.5.4-bin-hadoop3.tgz
```

Configurar **variables de entorno** de Spark

```
nano ~/.bashrc
```

Se abre un archivo y colocar al final las siguientes líneas. Guardar con **ctrl+o** <<enter>> y salir con **ctrl+x**.

```
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$JAVA_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
```

Activar o **actualizar las variables de entorno**

```
source ~/.bashrc
```

Abrir el puerto 8080 para Spark

```
sudo firewall-cmd --permanent --add-port=8080/tcp
sudo firewall-cmd --reload
```

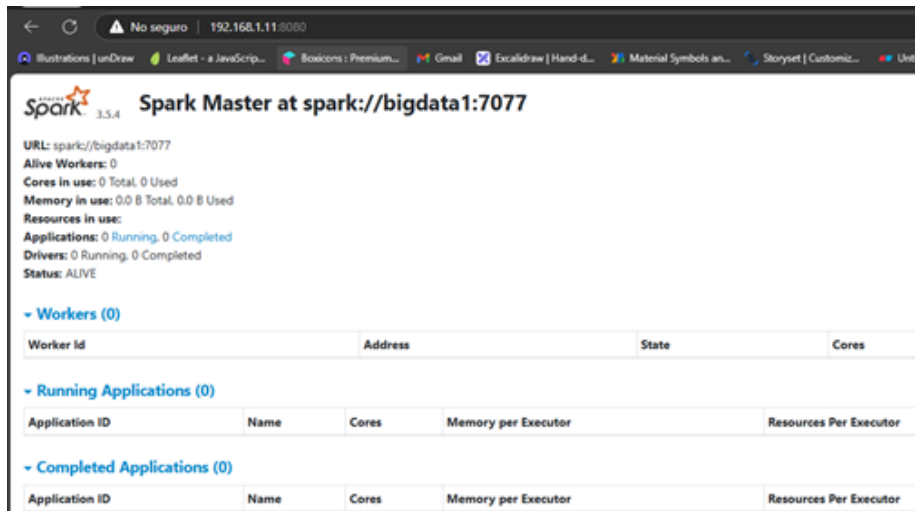
Iniciar o parar **servicios** de Spark

```
start-master.sh
stop-master.sh
```

Verificar desde el sistema operativo anfitrión

En este caso desde Windows, a través de un navegador web, utilizar la dirección IP con la que se instaló Ubuntu.

Verificar Spark desde **un navegador de Windows**: <http://192.168.1.11:8080>



Instalación de Jupyter-notebook

Los autores Díaz García & Cabrera Granado (2018) afirman que “Jupyter Notebook es una interfaz web de código abierto que permite la inclusión de texto, video, audio, imágenes, así como la ejecución de código a través de un navegador en múltiples lenguajes”.

Para ejecutar Jupyter, es necesario instalar algunos paquetes en la máquina donde se instaló Spark. Uno de los primeros paquetes es "**pip**", un sistema de gestión de paquetes escritos en lenguaje Python, en este caso pip3, que es la versión para gestionar Python 3 (Pérez García, 2018). Esto nos da la facilidad de instalar bibliotecas y actualizarlas en el entorno de desarrollo, y sobre todo asegura la compatibilidad con entornos como apache Spark.

A continuación, se detallan los pasos para la instalación y configuración de Jupyter Notebook.

Abrir el puerto 8888 para Jupyter-notebook

```
sudo firewall-cmd --permanent --add-port=8888/tcp
sudo firewall-cmd --reload
```

Instalación y verificación de **pip**

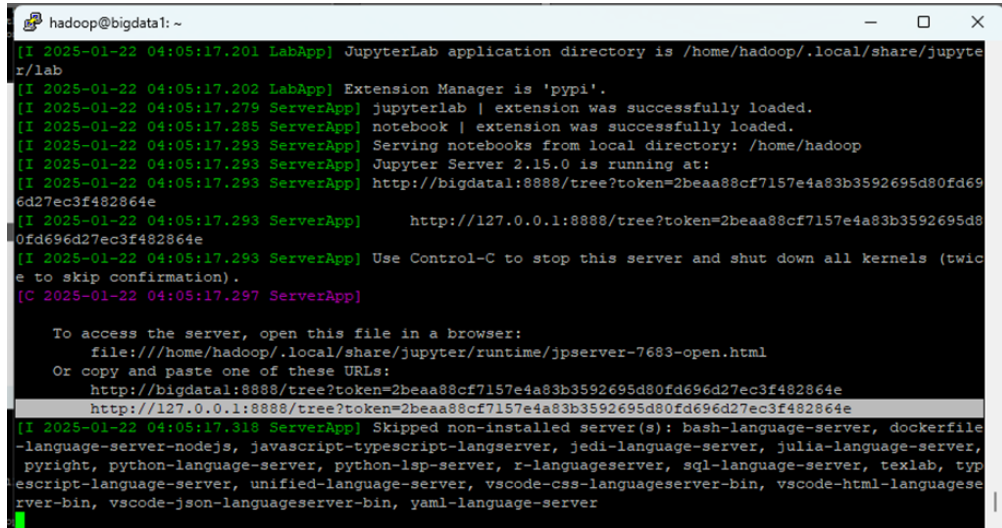
```
sudo apt update
sudo apt install python3-pip -y
pip3 --version
```

Instalación del paquete **Jupyter-core**, y Jupyter Notebook

```
sudo apt-get install jupyter-core
pip3 install notebook
sudo apt install jupyter-notebook -y
```

Una vez completada la instalación, **ejecutar Jupyter**:

```
jupyter-notebook --no-browser --ip=0.0.0.0 --port=8888
```

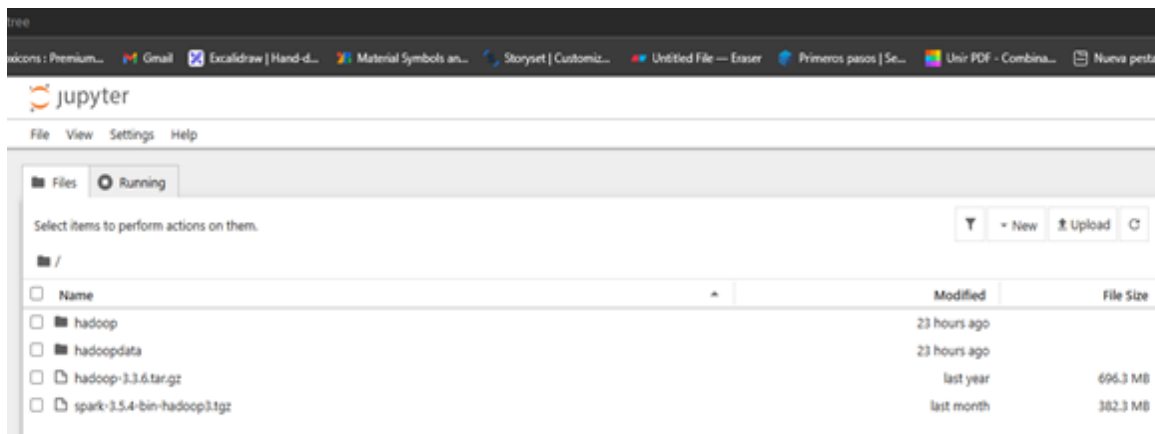


```
hadoop@bigdata1: ~
[I 2025-01-22 04:05:17.201 LabApp] JupyterLab application directory is /home/hadoop/.local/share/jupyter
r/lab
[I 2025-01-22 04:05:17.202 LabApp] Extension Manager is 'pypi'.
[I 2025-01-22 04:05:17.279 ServerApp] jupyterlab | extension was successfully loaded.
[I 2025-01-22 04:05:17.285 ServerApp] notebook | extension was successfully loaded.
[I 2025-01-22 04:05:17.293 ServerApp] Serving notebooks from local directory: /home/hadoop
[I 2025-01-22 04:05:17.293 ServerApp] Jupyter Server 2.15.0 is running at:
[I 2025-01-22 04:05:17.293 ServerApp] http://bigdata1:8888/tree?token=2beaa88cf7157e4a83b3592695d80fd69
6d27ec3f482864e
[I 2025-01-22 04:05:17.293 ServerApp] http://127.0.0.1:8888/tree?token=2beaa88cf7157e4a83b3592695d8
0fd696d27ec3f482864e
[I 2025-01-22 04:05:17.293 ServerApp] Use Control-C to stop this server and shut down all kernels (twic
e to skip confirmation).
[C 2025-01-22 04:05:17.297 ServerApp]

To access the server, open this file in a browser:
file:///home/hadoop/.local/share/jupyter/runtime/jpserver-7683-open.html
Or copy and paste one of these URLs:
http://bigdata1:8888/tree?token=2beaa88cf7157e4a83b3592695d80fd696d27ec3f482864e
http://127.0.0.1:8888/tree?token=2beaa88cf7157e4a83b3592695d80fd696d27ec3f482864e
[I 2025-01-22 04:05:17.318 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile
-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server,
pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typ
escript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languagese
rver-bin, vscode-json-languageserver-bin, yaml-language-server
```

Verificar la **ejecución de Jupyter** desde el sistema operativo anfitrión, en este caso desde un **navegador web en Windows**, utilizar la dirección IP con la que se instaló Ubuntu.

<http://192.168.1.11:8888/tree?token=2beaa88cf7157e4a83b3592695d80fd696d27ec3f482864e>



Resumen del Capítulo. Este capítulo se centra en la instalación en modo pseudo-distribuido de un ecosistema Big Data compuesto por Apache Hadoop (HDFS, MAPREDUCE y YARN) y Apache Spark, utilizando Ubuntu Server como sistema operativo base, ejecutado en Oracle VM VirtualBox. Se

orienta al lector en la configuración paso a paso del entorno virtual, incluyendo la instalación del sistema operativo, la habilitación de funcionalidades adicionales y el acceso remoto desde Windows a través de PuTTY. Además, se aborda la instalación del JDK de Java 8 como requisito fundamental, la creación de un usuario específico para Hadoop con acceso SSH, y la instalación y configuración de los componentes centrales del ecosistema: Apache HDFS, MAPREDUCE, YARN y Apache Spark. Finalmente, se instala Jupyter Notebook como interfaz para el análisis interactivo de datos, facilitando el uso práctico del entorno Big Data desde una perspectiva educativa y experimental.

Autoevaluación

1. ¿Cuál es el propósito del modo pseudo-distribuido en Hadoop?
 - a) Ejecutar Hadoop en múltiples servidores físicos.
 - b) Simular un clúster Hadoop con un solo nodo para pruebas y desarrollo.
 - c) Mejorar el rendimiento de HDFS en entornos de producción.
 - d) Permitir la conexión con bases de datos relacionales.
2. ¿Qué paquete debe instalarse en Ubuntu Server para gestionar la instalación de nuevos paquetes de Python en Jupyter Notebook dentro del entorno Hadoop?
 - a) pip
 - b) apt-get
 - c) npm
 - d) snap
3. ¿Cuál es el comando utilizado para iniciar el servicio de HDFS en modo pseudo-distribuido?
 - a) `hadoop fs -start`
 - b) `start-hdfs.sh`
 - c) `start-dfs.sh`
 - d) `hdfs namenode -format`
4. ¿Qué configuración debe modificarse en Ubuntu para permitir la comunicación SSH sin contraseña entre nodos en un clúster Hadoop?
 - a) `/etc/network/interfaces`
 - b) `~/.ssh/config`
 - c) `/etc/ssh/sshd_config`
 - d) `~/.ssh/authorized_keys`

5. ¿Qué archivo en la configuración de Hadoop define la ubicación del NameNode y los DataNodes en HDFS?
 - a) core-site.xml
 - b) hdfs-site.xml
 - c) yarn-site.xml
 - d) mapred-site.xml
6. ¿Cuál de las siguientes opciones describe correctamente el rol de YARN en Hadoop?
 - a) Es el sistema de almacenamiento distribuido.
 - b) Administra los recursos y ejecuta tareas en el clúster.
 - c) Procesa datos en paralelo mediante el modelo MapReduce.
 - d) Es un sistema de visualización de datos.
7. ¿Qué puerto se utiliza comúnmente para acceder a la interfaz web de Jupyter Notebook en un entorno Hadoop?
 - a) 8080
 - b) 9870
 - c) 50070
 - d) 8888
8. ¿Cuál es el comando para verificar los archivos almacenados en HDFS?
 - a) `hadoop fs -ls /`
 - b) `hdfs namenode -list`
 - c) `start-hdfs.sh`
 - d) `yarn resource-manager`
9. ¿Qué archivo de configuración se debe modificar para definir la cantidad de réplicas en HDFS?
 - a) core-site.xml
 - b) hdfs-site.xml
 - c) yarn-site.xml
 - d) mapred-site.xml
10. ¿Cómo se inicia Spark en un clúster Hadoop en modo pseudo-distribuido?
 - a) `spark-shell--master yarn`
 - b) `start-spark.sh`
 - c) `hadoop spark start`
 - d) `yarn start-spark`

Respuestas Correctas

1. B, 2. A, 3. C, 4. D, 5. B, 6. B, 7. D, 8. A, 9. B, 10. A

Referencias Bibliográficas

- Candel, J. M. O. (2022). *Big data, machine learning y data science en python*. Ra-Ma Editorial.
- Cruz Alejandro, D. de la. (2021). *Implementación de la herramienta para Big Data "Apache Hadoop", en clústeres de sistemas digitales en el laboratorio B-404* [Thesis, Universidad Autónoma de la Ciudad de México. Colegio de Ciencia y Tecnología. Licenciatura en Ingeniería en Sistemas Electrónicos y de Telecomunicaciones].
<http://repositorioinstitucionaluacm.mx/jspui/handle/123456789/2564>
- Díaz García, E., & Cabrera Granado, E. (2018). *Manual de uso de Jupyter Notebook para aplicaciones docentes*.
<https://hdl.handle.net/20.500.14352/20054>
- Macías, M., Gómez, M., Tous, R., & Torres, J. (2016). *Introducción a Apache Spark*. Editorial UOC.
- Ortiz Hernández, J. (2021). *Optimización de Apache Spark*.
<https://hdl.handle.net/20.500.14352/5348>
- Pérez García, I. (2018). *Learning analytics en cursos sobre TICS utilizando Moodle, Spark y Jupyter notebooks*.
<https://openaccess.uoc.edu/handle/10609/86905>
- ¿Qué es PuTTY y para qué sirve? (2020, abril 4). Actualidad.es.
<https://www.actualidad.es/ciencia-y-tecnologia/2020/04/04/que-es-putty-y-para-que-sirve/>
- Simon, P. (2013). *Too Big to Ignore: The Business Case for Big Data*. John Wiley & Sons.
- Valdez Menchaca, A. G., Castañeda Alvarado, S. R., Zárate Martínez, Á., & Paiz Rivera, A. L. S. (2020). Procesamiento de big data utilizando el software HADOOP. *RES NON VERBA REVISTA CIENTÍFICA*, 10(1), 1-13.
<https://doi.org/10.21855/resnonverba.v10i1.323>

Capítulo III

Instalación en Modo Distribuido de Apache Hadoop HDFS, YARN y Spark con CentOS

- Requisitos de Infraestructura para un Clúster
- Instalaciones para Nodo Máster
- Configuración de Nodos de Datos
- Configuración en Nodo Máster

Objetivos:

- Instalar y configurar un clúster Hadoop de 4 nodos (un máster y 3 esclavos) en CentOS 8.
- Implementar HDFS, YARN y Spark para procesamiento distribuido.
- Integrar Anaconda y Jupyter Notebook como entorno de análisis de datos.
- Configurar la comunicación y servicios esenciales como SSH y variables de entorno.

Apache Hadoop se caracteriza por su procesamiento distribuido, ya que, permite distribuir y paralelizar el tratamiento de los datos de manera eficiente; es popular por ser un framework gratuito, que permite escalar de manera horizontal (añadir nodos de bajo costo) porque proporciona un ambiente de adaptabilidad y personalización para las necesidades de cada organización que implemente una infraestructura distribuida.

Hadoop se relaciona con el concepto de “Commodity Hardware”, que significa hardware de bajo costo; esto permite a las organizaciones, en lugar de comprar grandes servidores para el procesamiento de big data, instalar Hadoop en un conjunto de computadoras de bajo costo conectadas en red y conformando un clúster que actúa como un único servidor.

En este capítulo se instala y configura un clúster desde cero de 4 nodos, con arquitectura base que consta de los módulos principales como lo son Common Utilies, Yarn, HDFS, Procesos MapReduce y Spark; además de implementar Anaconda y Jupyter Notebook.

La **Figura 19** presenta un mapa conceptual que sintetiza los elementos clave abordados en este capítulo, facilitando una visión global del proceso de instalación, configuración y tecnologías involucradas.

"El procesamiento distribuido permite que las operaciones se realicen simultáneamente en diferentes nodos, reduciendo el tiempo de ejecución y aumentando la eficiencia" (Coulouris et al., 2011).



Figura 19. Resumen del Capítulo III

Requisitos de Infraestructura para un Clúster

Para la instalación de un clúster con equipos físicos se debe contar con los requisitos mínimos (Tabla 7) o recomendados (Tabla 8):

Requisitos mínimos

Tabla 7. Requisitos mínimos del Clúster

Procesador	64 bits de 2 núcleos físicos y 4 hilos de procesamiento.
Memoria	6 GB
Almacenamiento	100 GB HDD o SSD
Sistema Operativo	GNU/Linux
Otros	Paquete OpenSSH Server para acceso remoto entre los equipos.

Fuente: Elaboración propia

Requisitos Recomendados

Tabla 8. Requisitos recomendados del clúster

Procesador	64 bits de 4 núcleos físicos y 8 hilos de procesamiento.
Memoria	16 GB
Almacenamiento	250 GB HDD o SSD
Sistema Operativo	GNU/Linux
Otros	Paquete OpenSSH Server para acceso remoto entre los equipos.

Fuente: Elaboración propia

Primero se debe crear un entorno de virtualización usando VMware Workstation 17 Pro con 4 máquinas virtuales conectadas en clúster con el adaptador de red Nat. Las capacidades del hardware que se utilizó para esta instalación se detallan a continuación:

- Procesador 13th Gen Intel(R) Core (TM) i9-13900H 2.60 GHz de 14 núcleos y 20 hilos de procesamiento.
- Memoria RAM de 16 GB 3200 MHz DDR4
- Unidad de almacenamiento: SSD NVMe 1 TB

Requisitos para CentOS 8

El autor Alonso (2023) , expresa que para instalar correctamente el sistema operativo se debe considerar los requisitos que se detallan a continuación:

- Memoria RAM: 2 GB como mínimo.
- Disco Duro: 20 GB, pero es recomendable 40 GB.
- Procesador: 2GHz o superior (Sistema x86 de 64 bits)

Arquitectura de un clúster de Big Data

Definición de Clúster

Un clúster es un conjunto de nodos que trabajan de forma coordinada para almacenar la información y/o realizar el procesamiento (Jeyaraj et al., 2020). Un clúster funciona como un sistema único o monolítico, es decir, para un usuario o para un problema dado todos los nodos (o sistemas miembros) del clúster son vistos como una sola computadora. Esta es la particularidad más representativa de un clúster (Alapati, 2016).

Normalmente hay dos tipos de nodos, que reciben diferentes nombres en función de cuál sea la tecnología Big Data (Hadoop, Spark, Storm, etc), pero que genéricamente vamos a llamarlos maestros y trabajadores (Alapati, 2016).

- Los **nodos maestro** son encargados de dividir, coordinar y distribuir las tareas y los recursos globales en pequeñas partes.
- Por otro lado, los **nodos trabajadores** son los encargados de realizar las diferentes “mini tareas” que se les han sido asignado sobre las “partes” de los datos.

También se pueden levantar o apagar el clúster según la necesidad y de la misma forma se pueden incrementar o disminuir sus nodos.

Diseño de un Clúster Hadoop

Dentro de una arquitectura Hadoop se tiene una o dos máquinas maestras denominadas NameNode y ResourceManager, que se encargan de la gestión del clúster y varios nodos esclavos que son los DataNode y NodeManager (Deshpande & Tanmay, 2016). El esquema para montar el clúster se muestra en la Figura 20.

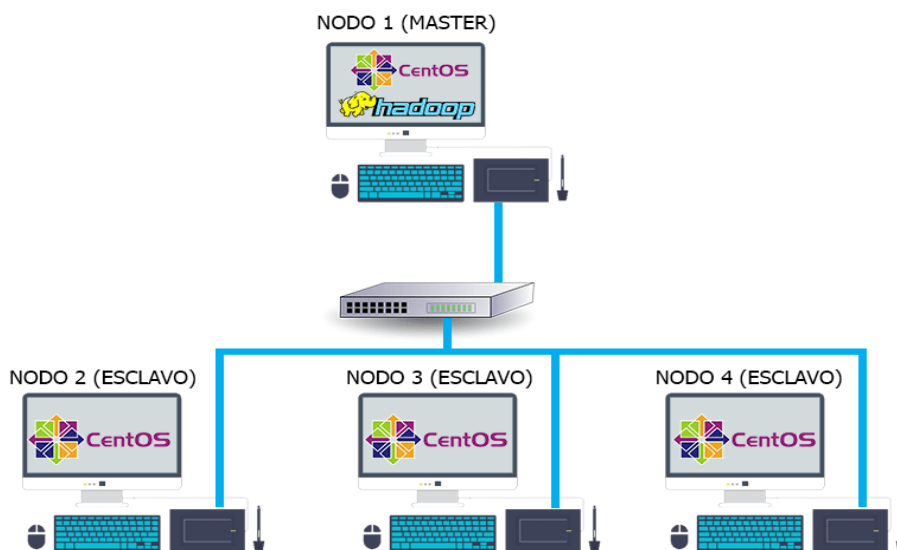


Figura 20. Arquitectura del clúster
Fuente: Elaboración propia

En la **Tabla 9** se detallan los servicios que deben ser instalados en cada equipo del clúster junto con las direcciones IP asignadas y los requisitos específicos para su correcta configuración. Esta arquitectura garantiza una distribución eficiente de las funciones y asegura la interoperabilidad entre los nodos del clúster.

Cada equipo debe configurarse con la IP indicada, respetando los servicios y requisitos asignados para un funcionamiento óptimo del sistema.

Tabla 9. Direccionamiento IP y servicios de cada nodo

Computador	IP address	Servicios	Requerimientos
Nodo1 (Master)	192.168.20.57/24	NameNode Secondary NameNode ResourceManager	Para cada Nodo: - 4 Gb de RAM - 25 Gb de Disco Duro
Nodo2	192.168.20.62/24	DataNode NodeManager	
Nodo3	192.168.20.63/24		
Nodo4	192.168.20.64/24		

Fuente: Elaboración propia

Servicios que se implementan en el clúster

1. Sistema de archivos distribuido de Hadoop (HDFS)

- Arquitectura maestro - esclavo, utilizado para almacenar archivos de manera distribuida.
- Consta de los siguientes componentes (Figura 21):
 - **NameNode** es el servidor maestro que administra el espacio de nombres del sistema de archivos. Realiza operaciones con los archivos y regula el acceso por parte de los clientes. Gestiona la metadata de los archivos y el control de bloques en cada DataNode. Cada archivo es dividido en bloques de datos, por defecto es de 128 MB; aunque es un parámetro configurable.
 - **DataNode** son los nodos esclavos o trabajadores que atienden solicitudes de lectura y escritura de los clientes. Hadoop al ser tolerante a fallos, cada bloque se replica en tres nodos de datos de forma preestablecida.

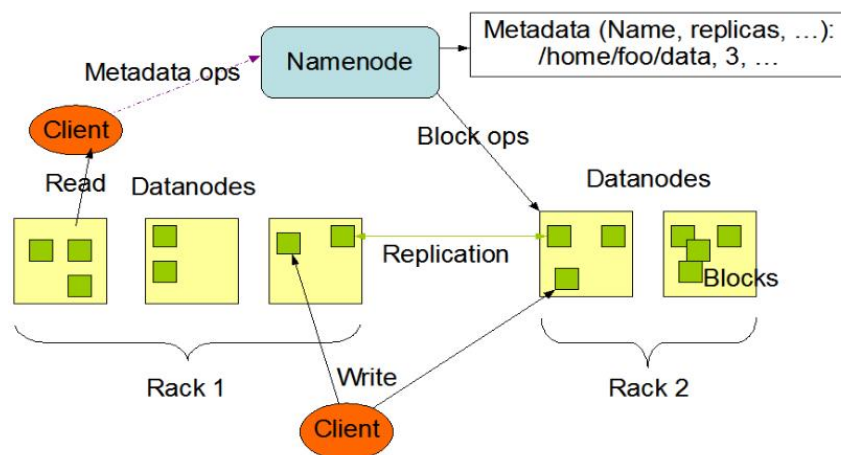


Figura 21. Arquitectura de HDFS

Fuente: (Apache Software Foundation, 2025)

2. YARN (Yet Another Resource Negotiator)

- Actúa como gestor de recursos y planificador de tareas en clústeres distribuidos de Hadoop.
- YARN permite ejecutar múltiples aplicaciones (como MapReduce, Spark, etc.) en un entorno Hadoop al administrar eficientemente los recursos del clúster.
- Consta de los siguientes componentes (Figura 22):
 - ResourceManager (RM): Administra los recursos globales del clúster.
 - NodeManager (NM): Supervisa los recursos en cada nodo del clúster.
 - ApplicationMaster (AM): Coordina la ejecución de una aplicación específica.
 - Container: Unidad lógica que contiene recursos (CPU, memoria, disco, red) para ejecutar tareas.

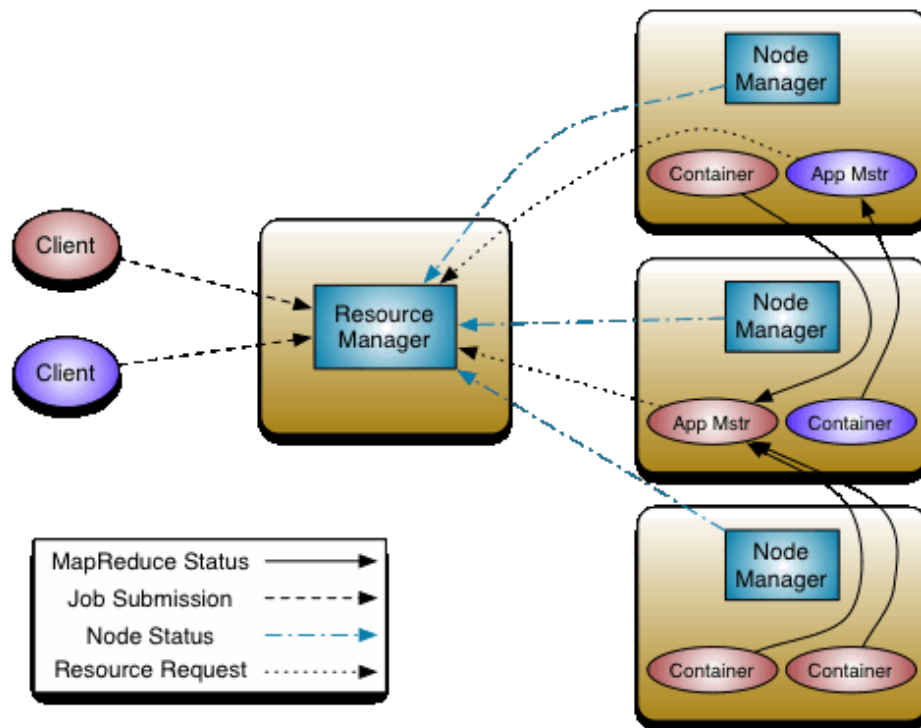


Figura 22. Arquitectura de YARN

Fuente: Documentación de [YARN](#)

3. MapReduce

- Modelo de programación y un motor de procesamiento distribuido desarrollado por Google y adoptado por Apache Hadoop. Permite procesar grandes volúmenes de datos dividiendo las tareas.
- MapReduce se ejecuta en clústeres distribuidos, es tolerante a fallos y adecuado para tareas por lotes, aunque menos eficiente que tecnologías modernas como Spark para ciertos casos.

4. Spark

- Permite el procesamiento en memoria (in-memory computing) de aplicaciones, como consultas interactivas, análisis de datos en tiempo real y aprendizaje automático.
- Tiene soporte para múltiples lenguajes: Python (PySpark), Scala, Java y R.
- Dispone de APIs de alto nivel: para operaciones con DataFrames, SQL, machine learning (MLlib) y procesamiento de flujos (Spark Streaming).
- Se integra con Hadoop y puede trabajar sobre YARN y leer datos desde HDFS.

Instalación y configuración de VMware Workstation

[VMware Workstation Pro](#) ([guía de descarga](#)), es una de las soluciones de virtualización de escritorios de usuario final que permite a los ingenieros de TI construir y probar máquinas virtuales y dispositivos de red similares a los del mundo real desde su PC/portátil (*Introduction to VMware Workstation* | SpringerLink, 2024).

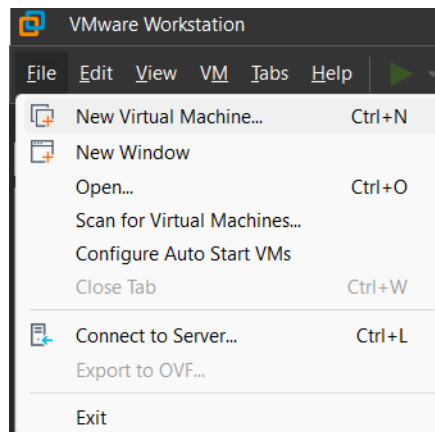
Es necesario tener disponible el archivo .iso del sistema operativo. En nuestro caso se consideró CentOS 8 ([CentOS-8.2.2004-x86_64-dvd1.iso](#)), debido a la presencia de ciertos paquetes necesarios, por ejemplo, el OpenSSH Server, útil para que los diferentes nodos puedan comunicarse entre sí. Se puede probar con una versión más actual de CentOS; para descargar visitar el sitio web <https://www.centos.org/download/>.

En este manual, en la **Tabla 10** se detalla paso a paso la preparación de la máquina virtual de manera que se lo realiza en el programa de virtualización VMware Workstation 17 Pro, debido a que ofrece una mejor compatibilidad con la distribución CentOS 8, el cual favorece a los usuarios principiantes que instalan Linux sin conocimientos avanzados, tales como el particionado de disco o la instalación de más de un sistema operativo en sus equipos.

Se puede realizar este proceso con otro programa de virtualización considerando los requisitos recomendados del clúster en la **Tabla 8** de manera que permita instalar con normalidad la distribución de su elección en su equipo.

Pasos para configuración de la máquina virtual VMware Workstation Pro:

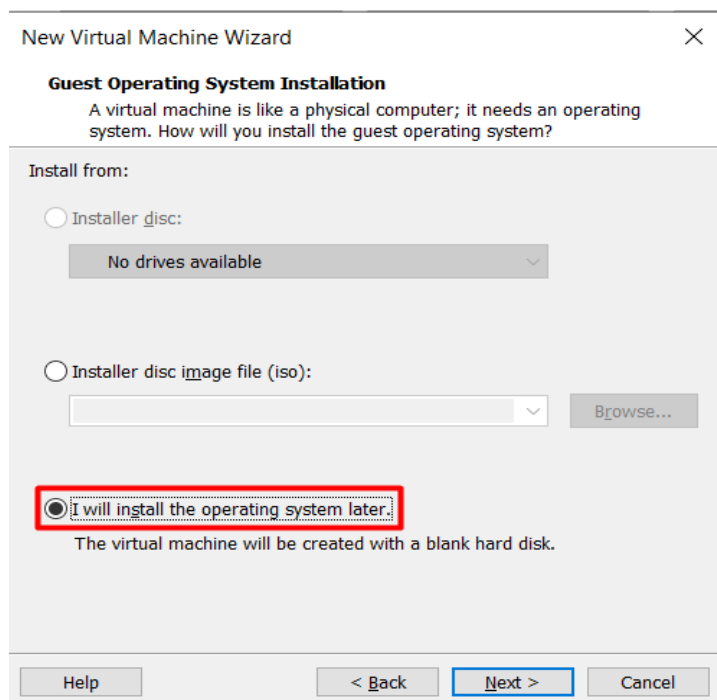
1. Hacer clic en "New Virtual Machine" para crear una nueva máquina virtual.



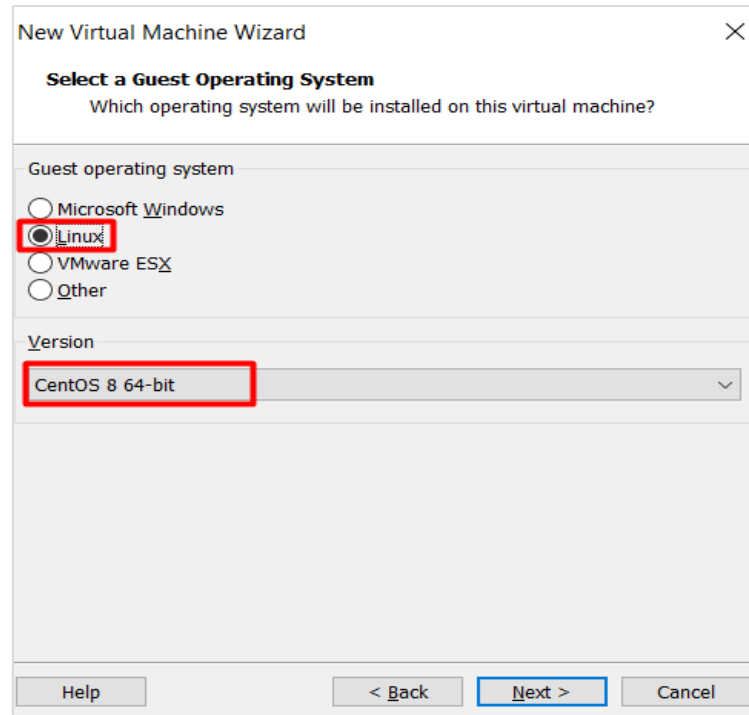
2. Seleccionar la opción de instalación típica o recomendada y hacer clic en siguiente.



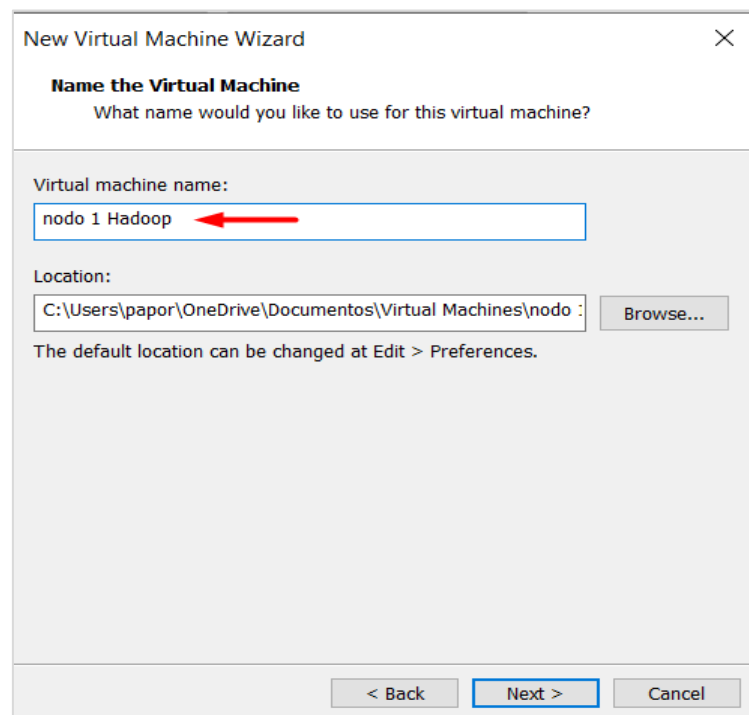
3. Seleccionar instalar el sistema operativo más tarde y dar clic en siguiente.



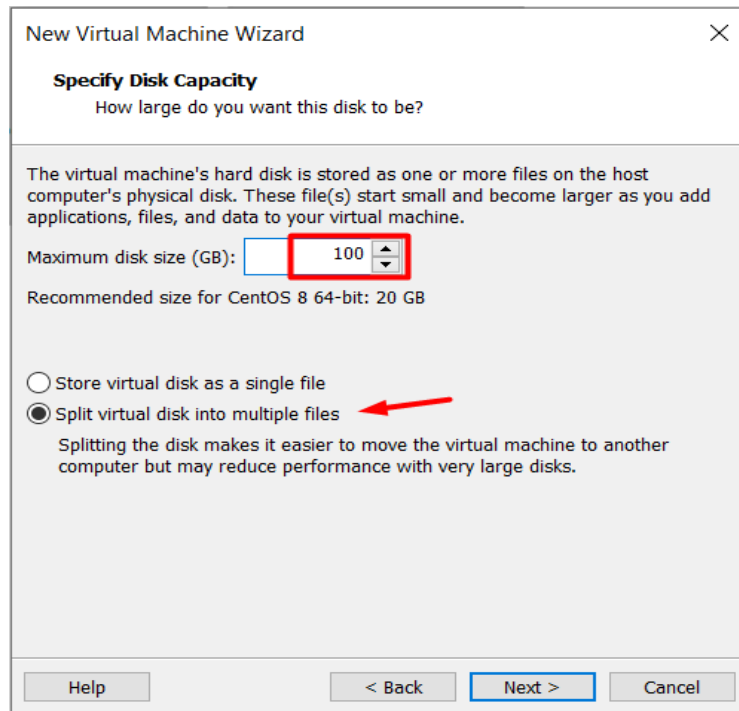
4. Seleccionar la opción de Linux y la versión de CentOS 8 de 64 bits y luego de clic en siguiente.



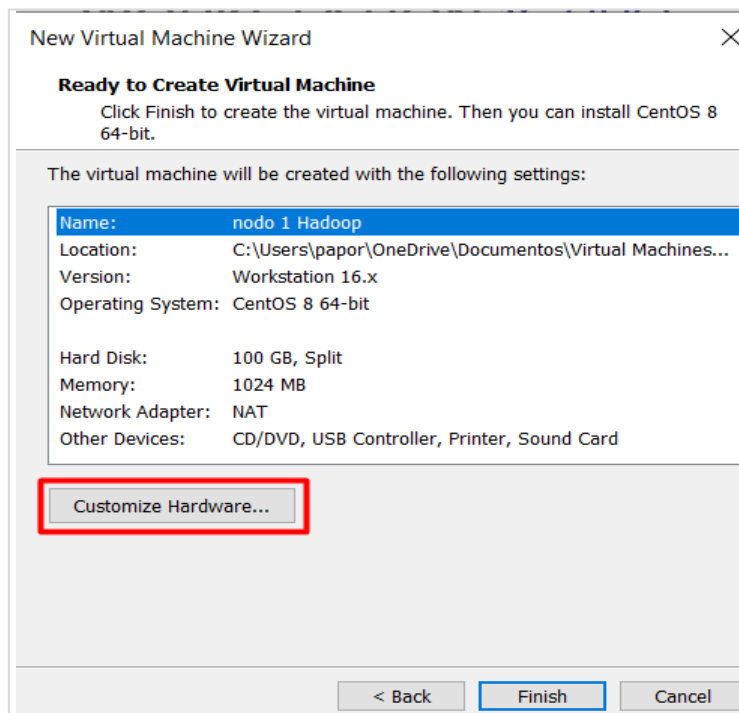
5. Escribir el nombre de la máquina virtual, luego seleccionar la ubicación de instalación y hacer clic en siguiente.



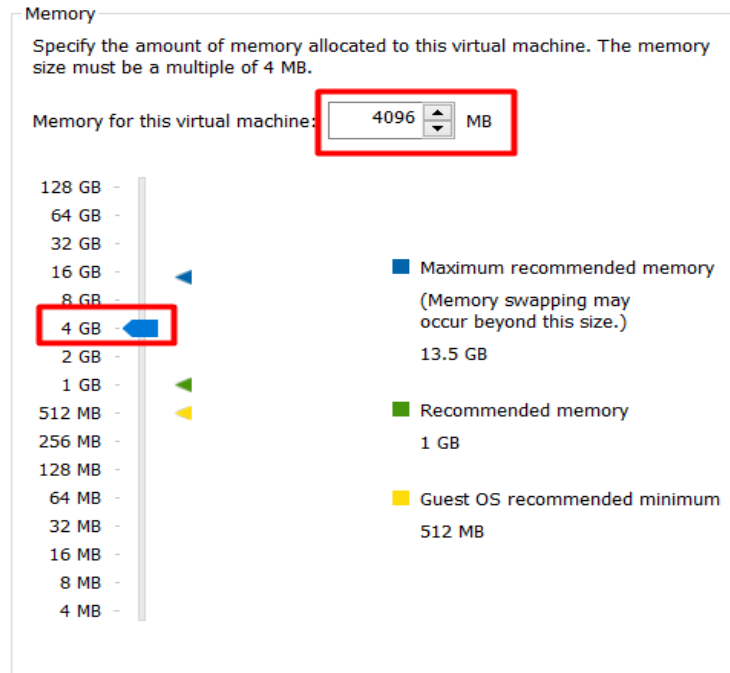
6. Asignar al menos 25 GB de espacio (por cada nodo), seleccionar la opción de dividir el disco virtual en múltiples archivos, y hacer clic en siguiente.



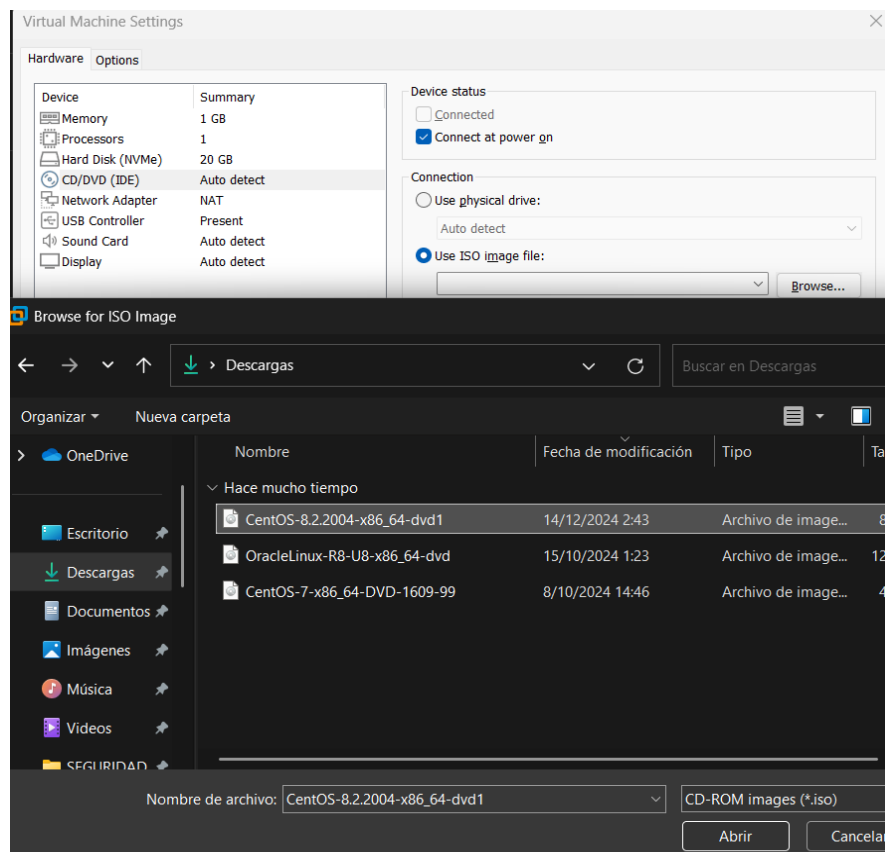
7. Hacer clic en "Personalizar Hardware" para editar las configuraciones de la máquina virtual.



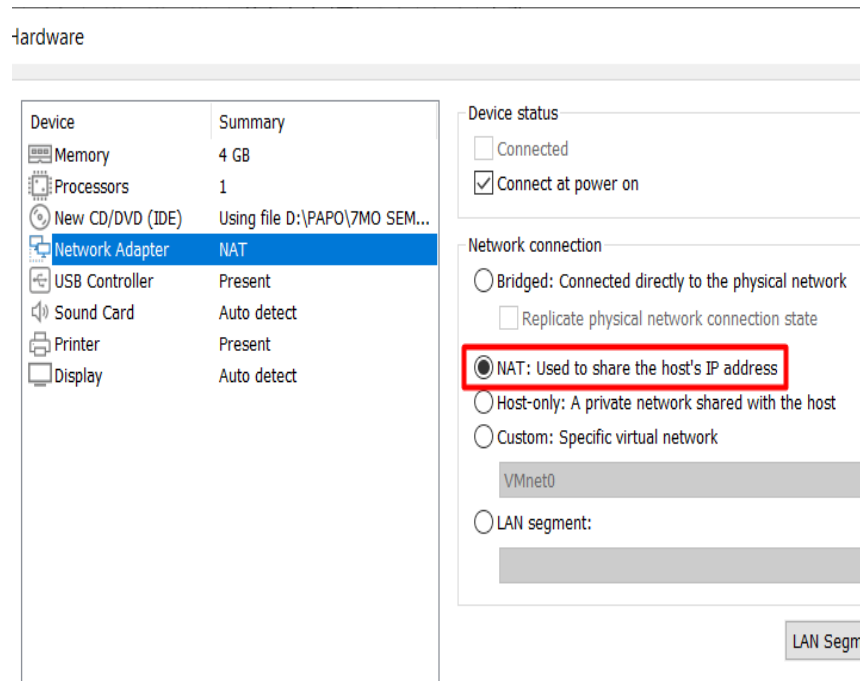
8. En la opción Memory, asignar un mínimo de 4 GB a un máximo de 16 GB, en caso de que su equipo se lo permita, para este manual se asigna 4 GB de RAM.



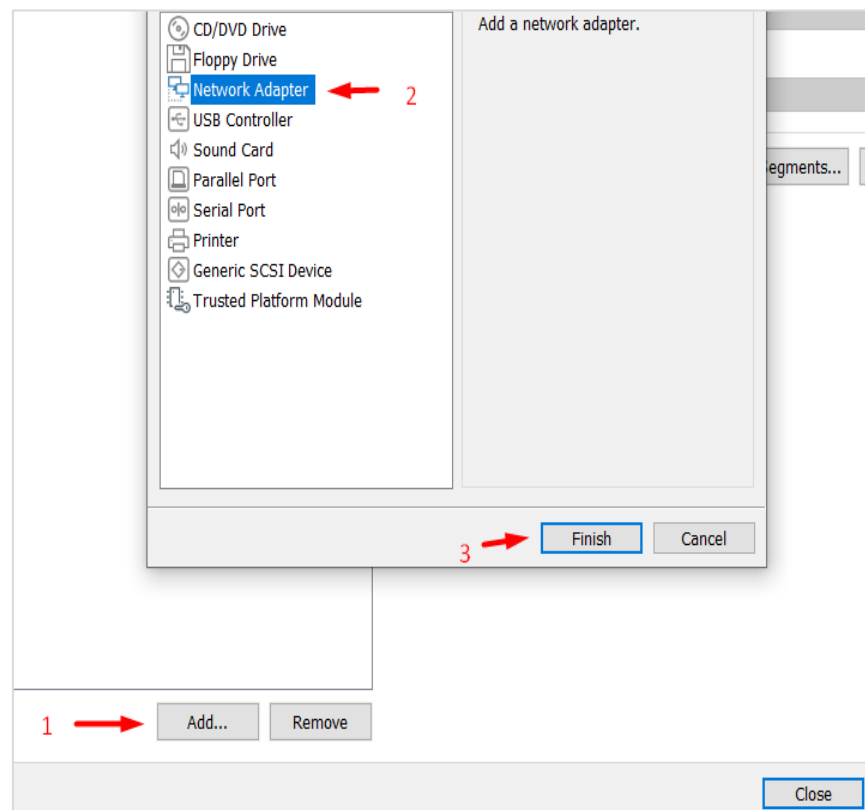
9. En la opción New CD/DVD, seleccionar la imagen ISO de CentOS como medio de instalación.



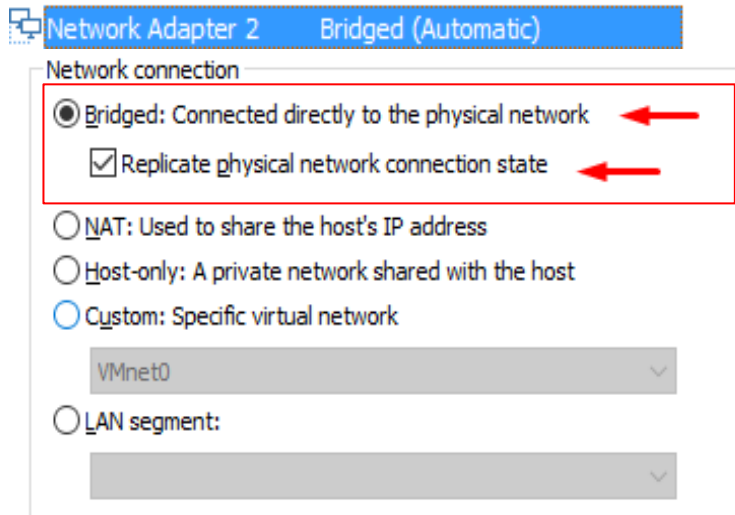
10. En la opción de Network Adapter, seleccionar la conexión NAT para que la máquina virtual tenga conexión a Internet.



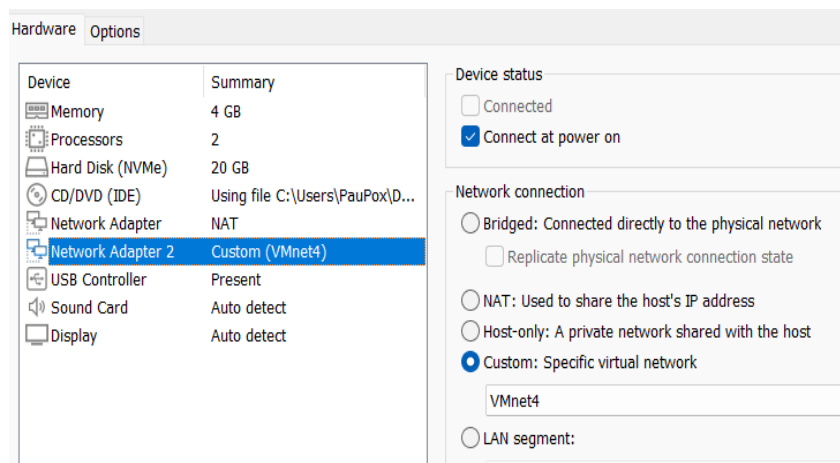
11. Agregar un nuevo adaptador de red haciendo clic en añadir, seleccionar Network Adapter.



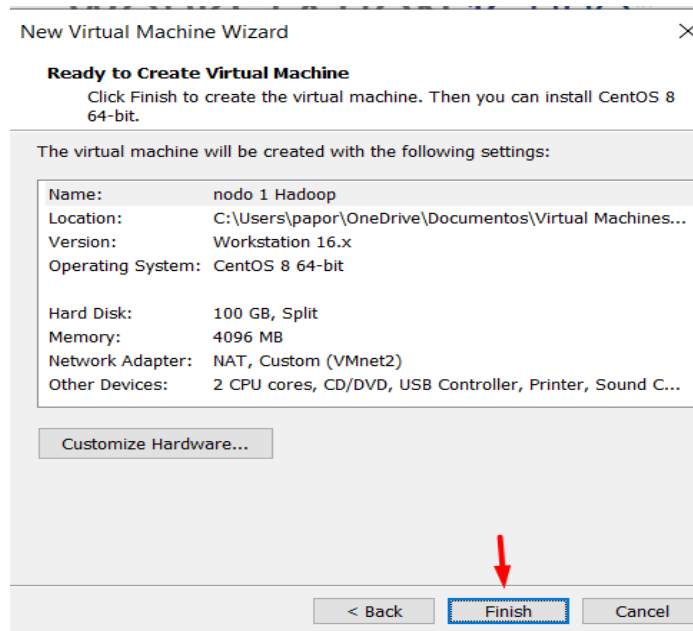
12. En caso de que se vaya a trabajar de manera física con los demás equipos del clúster, configurar el segundo adaptador de red con el tipo de conexión de red BRIDGE y activar (*Replique el estado de la conexión de la red física*).



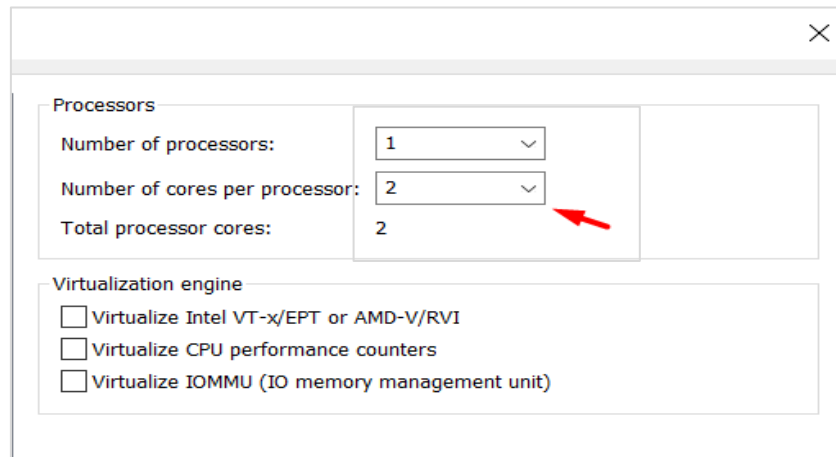
13. Para esta práctica se creará un clúster dentro de un mismo equipo, para eso, se debe configurar el segundo adaptador de red con el tipo de conexión de red Custom, seleccionar la red VMnet0 u otra.



14. Dar clic en “Finish” para culminar la configuración de la máquina virtual.



15. En la opción Processors, seleccionar al menos dos núcleos virtuales para la máquina virtual.



Instalación y configuración del nodo Máster

Instalación del Sistema Operativo CentOS 8

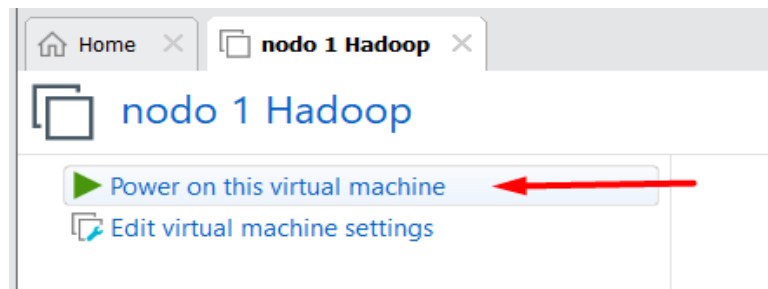


¡IMPORTANTE!

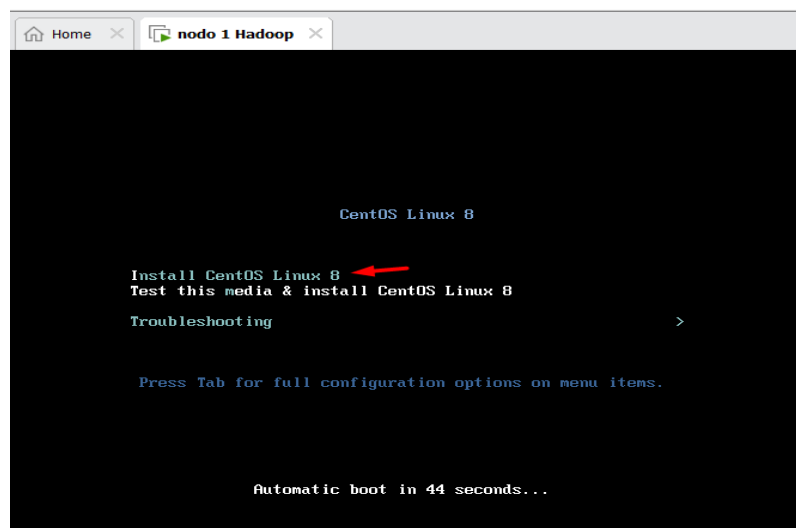
Debe tomar en cuenta que la pantalla inicial de instalación de sistema operativo CentOS no tiene un entorno gráfico; por lo que es necesario desplazarse con el teclado para proseguir con la instalación.

Pasos para la instalación Sistema Operativo CentOS 8:

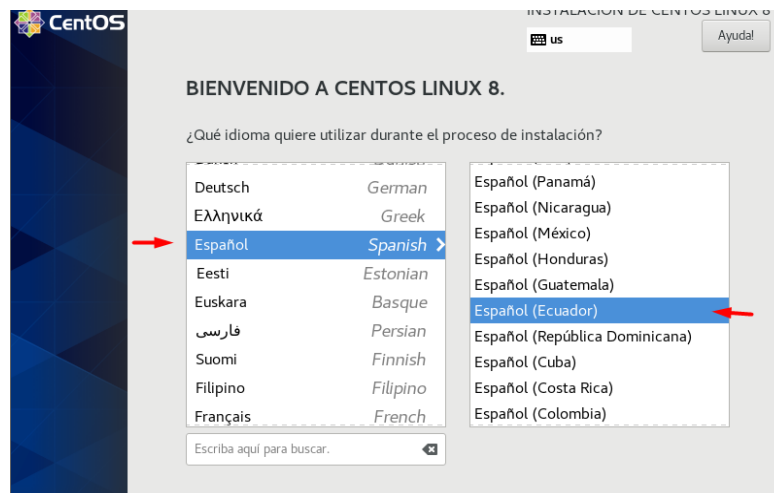
1. Hacer clic en Iniciar esta máquina virtual para proceder con el arranque del sistema de instalación.



2. Seleccionar Install CentOS 8, para proseguir con la instalación, luego debe esperar hasta que nos muestre la ventana de bienvenida.



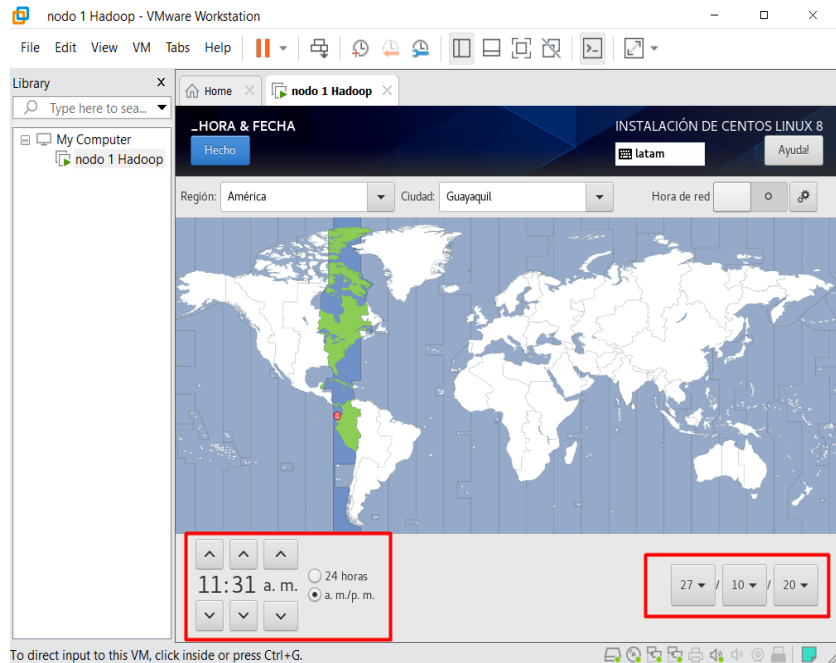
3. Seleccionar idioma con su respectivo paquete a instalar y de clic en continuar.



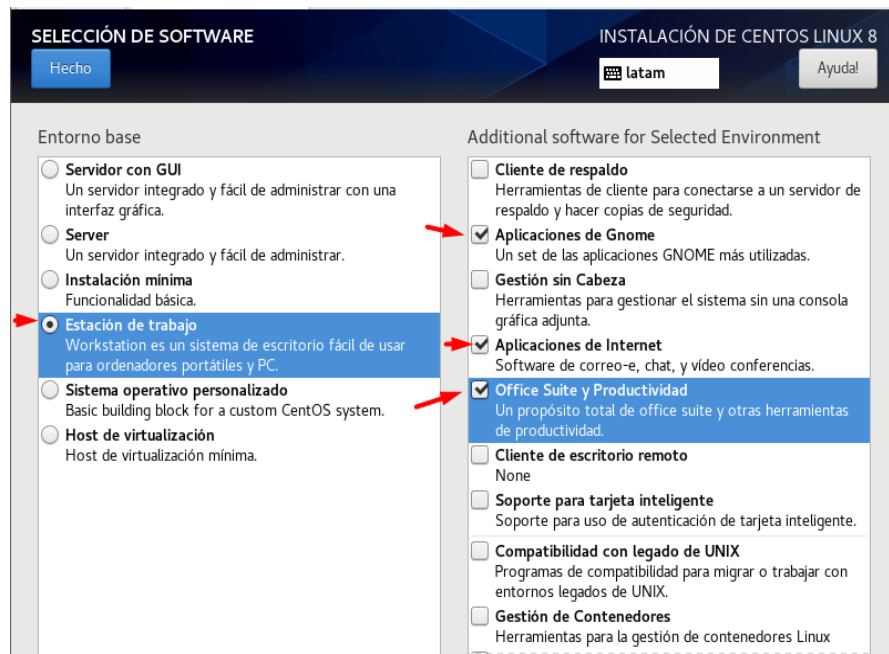
4. Seleccionar fecha y hora para cambiar la configuración regional y zona horaria.



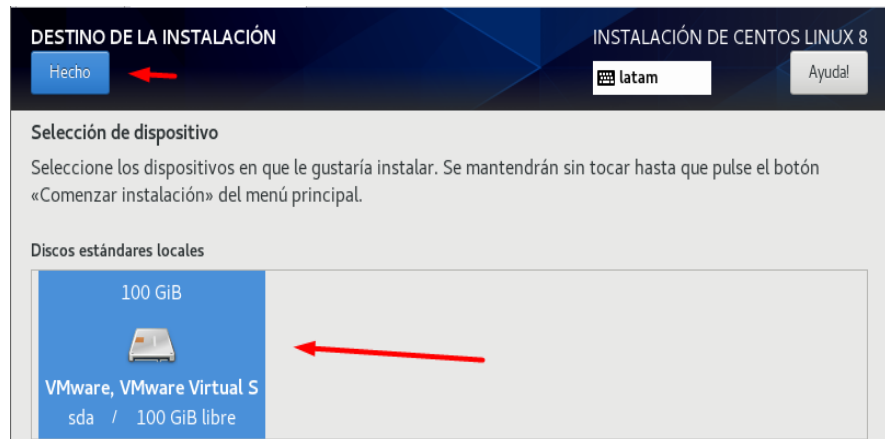
5. Seleccionar la ubicación, configure la fecha y hora. Una vez actualizada la fecha, debe dar clic en Hecho.



6. En el apartado de Software hacer clic en selección de software.
7. Activar: Escritorio Gnome, Aplicaciones de Gnome, Internet y Office Suite de productividad, dar clic en Hecho.



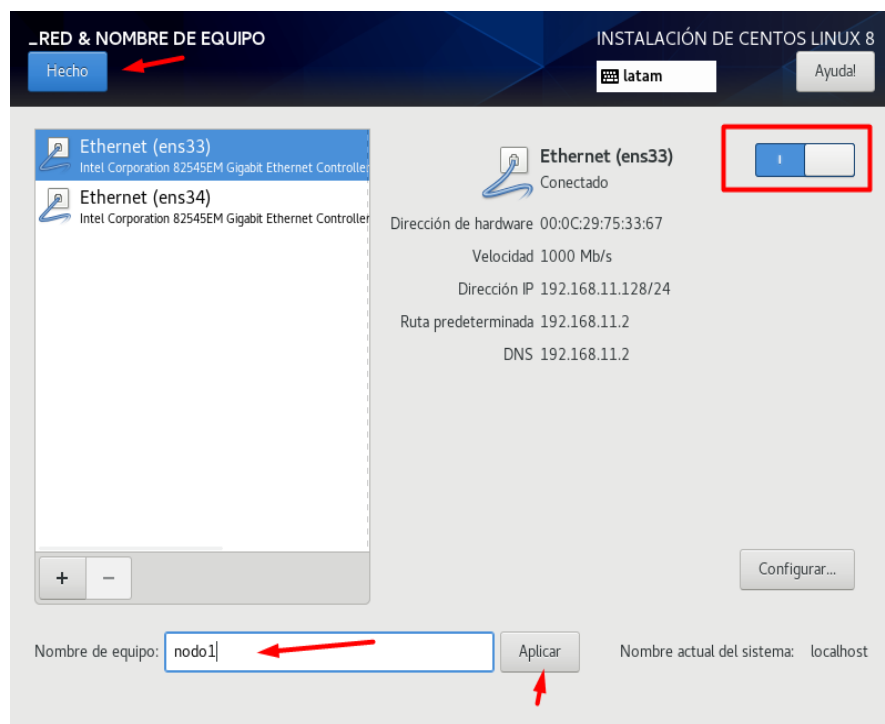
8. En el apartado de Sistema hacer clic en destino de la instalación.
9. Seleccionar el disco donde se va a instalar el sistema y dar clic en Hecho.



10. Hacer clic en red y nombre de equipo dentro del apartado de sistema.



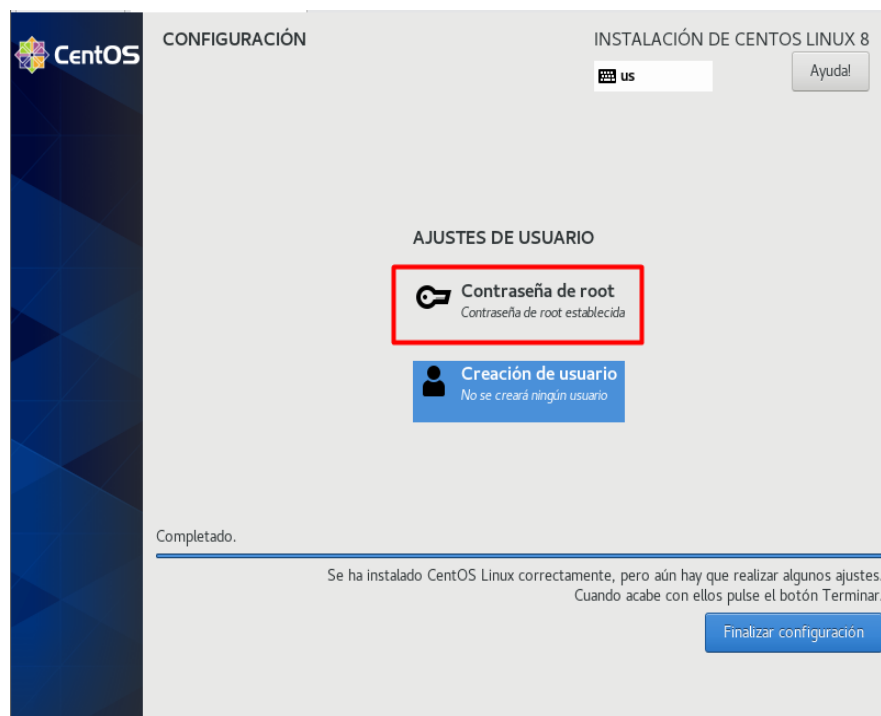
11. Activar la interfaz de conexión a Internet. Escribir el nombre del host, en este caso "nodo1" ya que se está realizando la instalación para el Máster, luego debe dar clic en aplicar y finalmente clic en Hecho.



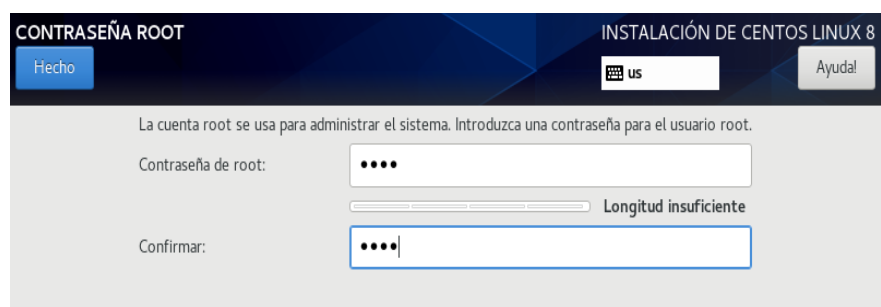
12. Hacer clic en empezar la instalación.



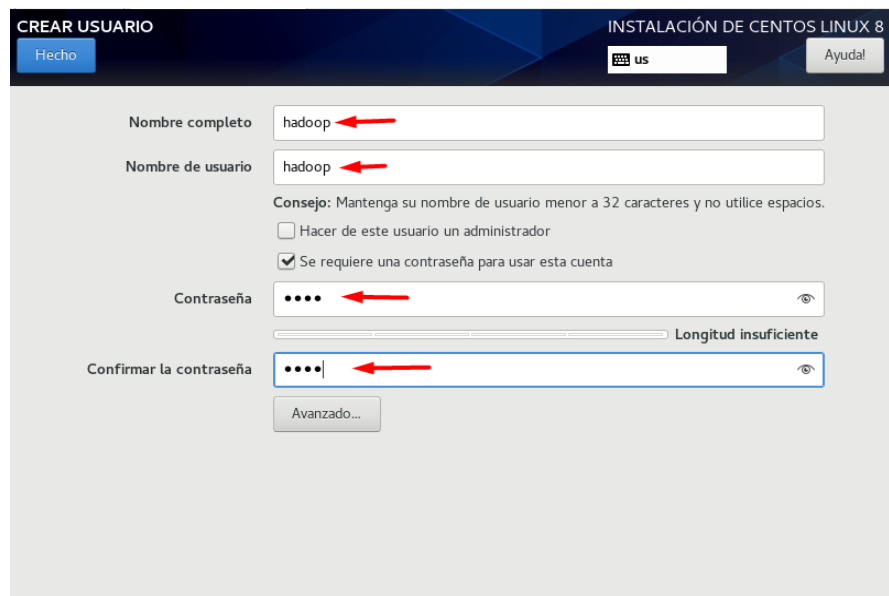
13. Configurar la contraseña para el usuario root y luego crear el usuario cliente mientras se instala.



14. Hacer clic en contraseña de root, escriba la contraseña para el usuario root, y luego hacer clic en Hecho.



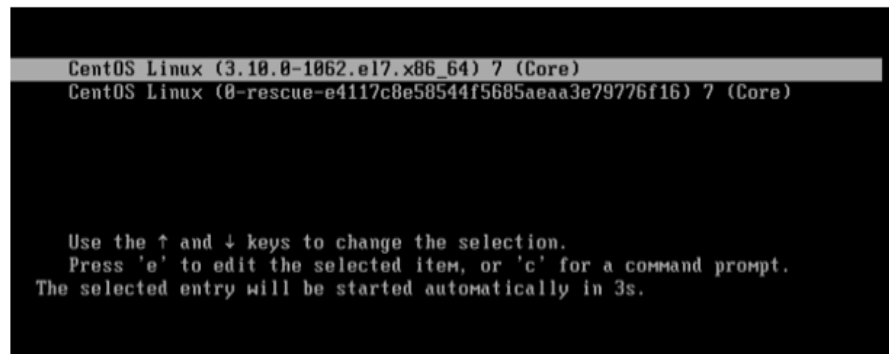
15. Hacer clic en creación de usuario. En el nombre de usuario escribir "hadoop", marcar la casilla "hacer de este usuario sea administrador", escriba la contraseña y hacer clic en Hecho y esperar a que termine la instalación.



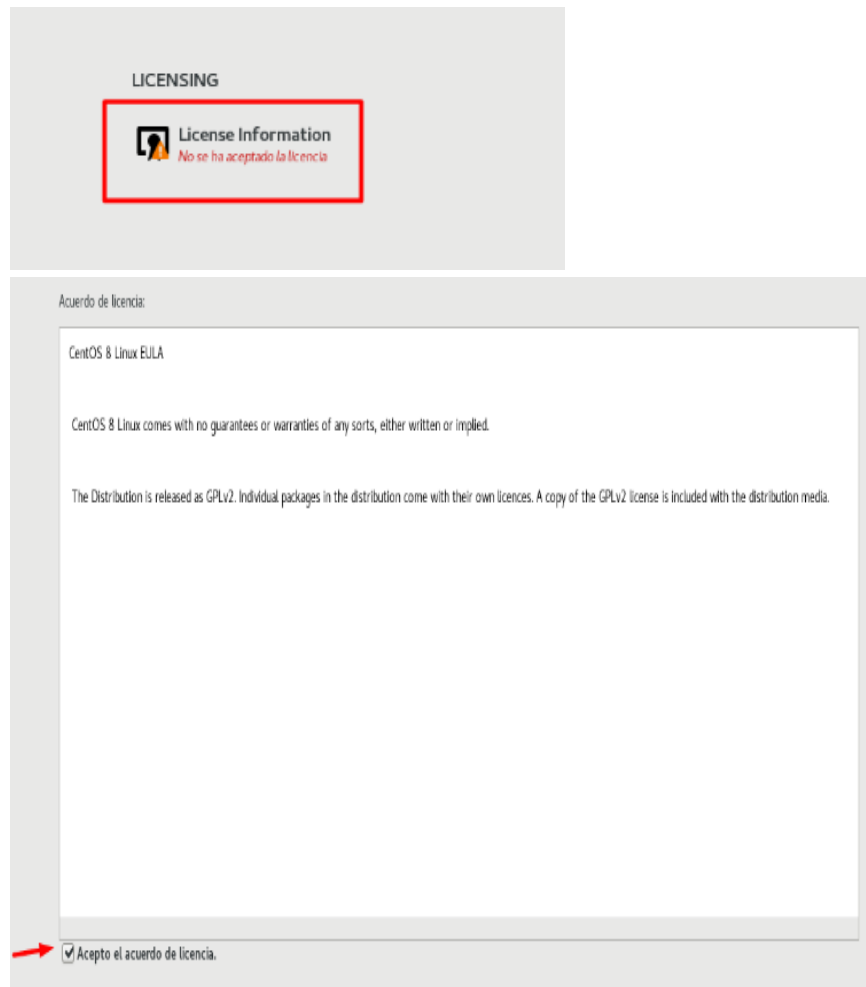
16. Una vez que termine el proceso de instalación debe dar clic en el botón de Finalizar configuración y luego en reiniciar.



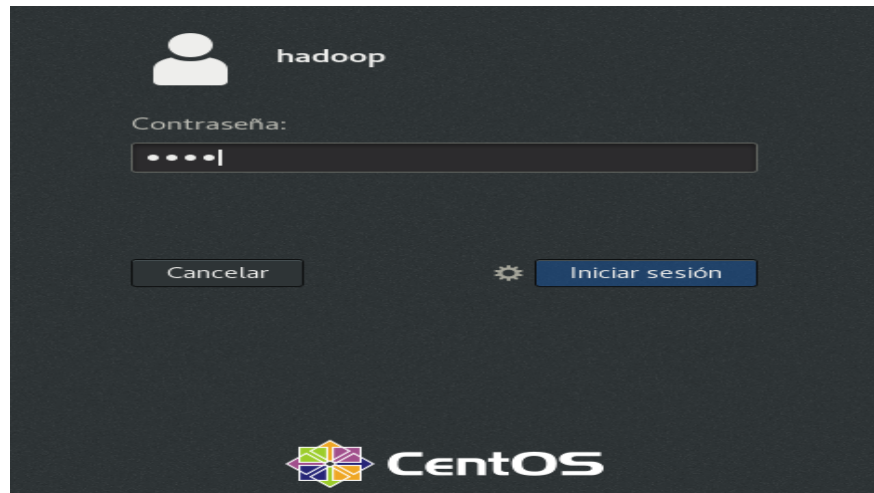
17. Seleccionar la primera opción al momento que muestre las opciones de inicio, de esta manera el sistema CentOS 8 iniciará.



18. Hacer clic en información de licencia para aceptar los términos y condiciones de la misma, luego hacer clic en "Acepto el acuerdo de licencia".



19. Una vez que presione en finalizar configuración, este le indicará en una nueva ventana que debe iniciar sesión con el usuario `hadoop`.



20. Debe entrar al escritorio Gnome del CentOS 8, y se dar por terminada la instalación del sistema.



Instalación y configuración de Hadoop

Ingresar al siguiente enlace para descargar el paquete de Hadoop dentro de la máquina virtual:

<https://hadoop.apache.org/releases.html>

Pasos para la Instalación de Hadoop:

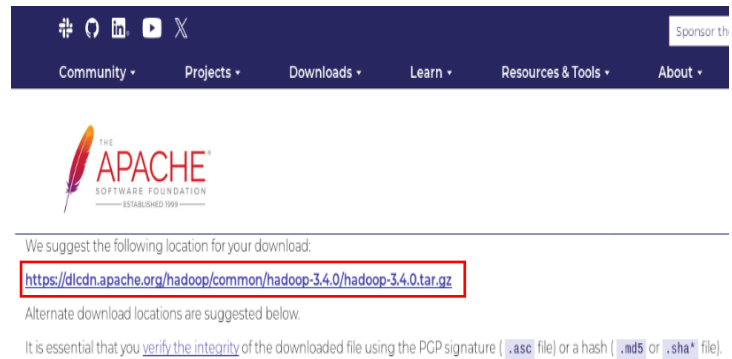
1. Escoger la versión actualizada, en nuestro caso trabajaremos con la 3.4.0 y hacer clic en "binary".

Download

Hadoop is released as source code tarballs with corresponding binary tarballs for convenience. The downloads are distributed using GPG or SHA-512.

Version	Release date	Source download	Binary download
3.4.1	2024 Oct 18	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature) binary-lean (checksum signature)
3.4.0	2024 Mar 17	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)
3.3.6	2023 Jun 23	source (checksum signature)	binary (checksum signature) binary-aarch64 (checksum signature)
2.10.2	2022 May 31	source (checksum signature)	binary (checksum signature)

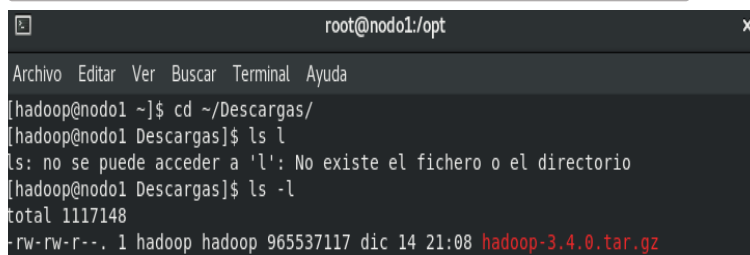
2. Al dar clic en binary, le direccionará a la página donde se encontrará el link de descarga.



3. Abrir una Terminal desde el escritorio.

4. Acceder al directorio Descargas desde la consola, además de visualizar el archivo descargado.

```
cd ~/Descargas/
ls -l
```



```
root@nodo1:/opt
Archivo Editar Ver Buscar Terminal Ayuda
[hadoop@nodo1 ~]$ cd ~/Descargas/
[hadoop@nodo1 Descargas]$ ls l
ls: no se puede acceder a 'l': No existe el fichero o el directorio
[hadoop@nodo1 Descargas]$ ls -l
total 1117148
-rw-rw-r--. 1 hadoop hadoop 965537117 dic 14 21:08 hadoop-3.4.0.tar.gz
```

5. Abrir una nueva pestaña de la Terminal (Ctrl+Shift+T) y ejecutar los siguientes comandos.

```
su - root
cd /opt
mkdir hadoop
chown hadoop /opt/hadoop
```

6. Ingresar al directorio hadoop y descomprimir el paquete hadoop 3.4.0

```
cd /opt/hadoop/
tar xvf ~/Descargas/hadoop-3.4.0.tar.gz
```

7. Al extraer el archivo, mover el contenido de la subcarpeta generada a /opt/, según la versión descargada.

```
mv hadoop-3.4.0/* .
rmdir hadoop-3.4.0/
```

```
hadoop@nodo1:/opt/hadoop
Archivo  Editar  Ver  Buscar  Terminal  Pestañas  Ayuda
root@nodo1:/opt
hadoop@nodo1:/opt/hadoop
hadoop-3.4.0/sbin/refresh-namenodes.sh
hadoop-3.4.0/sbin/start-balancer.sh
hadoop-3.4.0/sbin/start-all.sh
[hadoop@nodo1 hadoop]$ ls -l
total 0
drwxr-xr-x. 10 hadoop hadoop 215 mar  4  2024 hadoop-3.4.0
[hadoop@nodo1 hadoop]$ mv hadoop-3.4.0/* .
[hadoop@nodo1 hadoop]$ rmdir hadoop-3.4.0/
[hadoop@nodo1 hadoop]$ ls -l
total 84
drwxr-xr-x. 2 hadoop hadoop 203 mar  4  2024 bin
drwxr-xr-x. 3 hadoop hadoop  20 mar  4  2024 etc
drwxr-xr-x. 2 hadoop hadoop 106 mar  4  2024 include
drwxr-xr-x. 3 hadoop hadoop  20 mar  4  2024 lib
drwxr-xr-x. 4 hadoop hadoop 288 mar  4  2024 libexec
-rw-r--r--. 1 hadoop hadoop 23756 mar  4  2024 LICENSE-binary
drwxr-xr-x. 2 hadoop hadoop 4096 mar  4  2024 licenses-binary
-rw-r--r--. 1 hadoop hadoop 15696 mar  4  2024 LICENSE.txt
-rw-r--r--. 1 hadoop hadoop 27165 mar  4  2024 NOTICE-binary
-rw-r--r--. 1 hadoop hadoop 1541 mar  4  2024 NOTICE.txt
-rw-r--r--. 1 hadoop hadoop 175 mar  4  2024 README.txt
drwxr-xr-x. 3 hadoop hadoop 4096 mar  4  2024 sbin
drwxr-xr-x. 4 hadoop hadoop  31 mar  4  2024 share
[hadoop@nodo1 hadoop]$
```

Instalar JDK de Java 8

Java 8 sigue siendo la versión de Java más utilizada por los usuarios encuestados en un estudio reciente de la empresa Perforce Software. A la pregunta de qué lenguaje de programación JDK (Java Development Kit) se utilizaba en su aplicación principal, el 37% de los encuestados en la encuesta patrocinada por Perforce respondió Java 8. El segundo lugar lo ocupa Java 11, utilizado por el 29% de los encuestados (Krill, 2022).

En Centos 8 viene instalado una versión de Java que no cuenta con el JDK, por lo que, deberemos descargar la versión 8 del JDK para ejecutar los comandos y librerías necesarias.

Accederá desde el siguiente enlace:

<https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html>

Product / File Description	File Size	Download
Linux ARM v6/v7 Soft Float ABI	72.86 MB	jdk-8u202-linux-arm32-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	69.75 MB	jdk-8u202-linux-arm64-vfp-hflt.tar.gz
Linux x86	173.08 MB	jdk-8u202-linux-i586.rpm
Linux x86	187.9 MB	jdk-8u202-linux-i586.tar.gz
Linux x64	170.15 MB	jdk-8u202-linux-x64.rpm
Linux x64	185.05 MB	jdk-8u202-linux-x64.tar.gz
Mac OS X x64	249.15 MB	jdk-8u202-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.09 MB	jdk-8u202-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.1 MB	jdk-8u202-solaris-sparcv9.tar.gz

Una vez seleccionado, para concretar la descarga, aceptar los términos, condiciones y licencia, luego ingresar a su cuenta de Oracle y se empezará a descargar el archivo.

Desde una Terminal, acceder como usuario root y ejecutar los siguientes comandos que permitirá la instalación del Java 8.

```
su - root
cd /home/hadoop/Descargas/
rpm -ivh jdk-8u202-linux-x64.rpm
```

Cambiar la versión de java a la que se instaló, con el siguiente comando:

```
alternatives --config java
```

Ingresar el número 2 y dar Enter; de esta manera se cambiará la versión, además para comprobar si la versión es correcta ejecute el siguiente comando:

```
java --version
```

La siguiente pantalla muestra el resultado de los dos comandos anteriores.

```
[root@nod01 Descargas]# alternatives --config java
Hay 2 programas que proporcionan 'java'.

  Selección    Comando
-----
*+ 1          java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.
252.b09-2.el8_1.x86_64/jre/bin/java)
  2           /usr/java/jdk1.8.0_202-amd64/jre/bin/java

Presione Intro para mantener la selección actual[+], o escriba el número de la s
elección: 2
[root@nod01 Descargas]# java -version
java version "1.8.0_202"
Java(TM) SE Runtime Environment (build 1.8.0_202-b08)
Java HotSpot(TM) 64-Bit Server VM (build 25.202-b08, mixed mode)
[root@nod01 Descargas]#
```

Variables de entorno de Hadoop

La configuración de estas variables se las hará en la carpeta raíz del usuario en el archivo **.bashrc**, que contiene todas las variables de entorno que necesitan las aplicaciones para funcionar en el sistema operativo.

Desde **[hadoop@nod01 ~]\$** ejecute lo siguiente:

```
cd
gedit .bashrc
```

Dentro de este archivo exportas 3 variables, **hadoop_home**, **path**, **java_home**.

```
export HADOOP_HOME=/opt/hadoop
export
PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:/opt/hadoop/spark/bin:/opt/hadoop/park/sbin
export JAVA_HOME=/usr/java/jdk1.8.0_202-amd64
```

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
PATH="$HOME/.local/bin:$HOME/bin:$PATH"
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export HADOOP_HOME=/opt/hadoop
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:/opt/hadoop/spark/bin:/opt/hadoop/park/sbin
export JAVA_HOME=/usr/java/jdk1.8.0_202-amd64
```

Ejecuta los comandos en la Terminal para comprobar la versión de Hadoop:

```
./bashrc
hadoop version
```

```
[hadoop@nod01 ~]$ ./bashrc
[hadoop@nod01 ~]$ hadoop version
Hadoop 3.4.0
Source code repository git@github.com:apache/hadoop.git -r bd8b77f398f626bb7791783192ee7a5dfaee760
Compiled by root on 2024-03-04T06:35Z
Compiled on platform linux-x86_64
Compiled with protoc 3.21.12
From source with checksum f7fe694a3613358b38812ae9c31114e
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-3.4.0.jar
[hadoop@nod01 ~]$
```

Antes de verificar el funcionamiento de Hadoop, es necesario configurar el archivo /etc/hosts. Para ello, utilizar el comando ifconfig para identificar la dirección IP del adaptador de red.

Ingresa al archivo de hosts y editar añadiendo la dirección IP correspondiente al adaptador de red Ethernet 33 y al final agregar el nombre del equipo **"nodo1"**.

```
gedit /etc/hosts
```

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.11.128 nodo1
```


Para comprobar el correcto funcionamiento de hadoop, ejecutar los siguientes comandos, tener en cuenta la versión que se haya descargado, en este caso es la 3.4.0:

```
cd /opt/hadoop/share/hadoop/mapreduce/
jar tf hadoop-mapreduce-examples-3.4.0.jar
```

Desde **[hadoop@nodo1 mapreduce]\$** ejecutar lo siguiente:

```
mkdir /tmp/entrada
cp /opt/hadoop/etc/hadoop/*.xml /tmp/entrada
ls -l /tmp/entrada
```

Considerando la versión que ha descargado, debe ejecutar el siguiente comando:

```
hadoop jar hadoop-mapreduce-examples-3.4.0.jar grep /tmp/entrada/
/tmp/salida 'kms[a-z.]+'
```

Visualizar el contenido de la carpeta /tmp/salida y a su vez la salida del trabajo con el comando **cat**.

```
ls -l /tmp/salida/
cat /tmp/salida/part-r-00000
```

Si el proceso está correcto, se muestra información como la pantalla siguiente:

```
Bytes Written=23
[hadoop@nodo1 mapreduce]$ ls -l /tmp/salida/
total 4
-rw-r--r--. 1 hadoop hadoop 11 dic 14 22:20 part-r-00000
-rw-r--r--. 1 hadoop hadoop  0 dic 14 22:20 _SUCCESS
[hadoop@nodo1 mapreduce]$ cat /tmp/salida/part-r-00000
9      kms.acl.
[hadoop@nodo1 mapreduce]$
```

Servicio SSH

El protocolo SSH funciona estableciendo una conexión TCP sobre el puerto 22, aunque este puede modificarse. El Cliente y servidor acuerdan la versión del protocolo y el algoritmo de cifrado. El servidor envía su clave pública al cliente, quien la verifica para confirmar su autenticidad. Luego, el cliente genera una clave de sesión aleatoria y la envía, junto con el algoritmo de cifrado seleccionado, asegurando así una comunicación encriptada (*Qué es el protocolo SSH y cómo configurarlo para mejorar la seguridad de acceso a los servidores Linux* | Hostalia - Pressroom, 2013).

El nodo maestro es el encargado de comunicar las tareas y procesos a cada uno de los nodos esclavos.

Al generar la llave RSA pública y privada, se permitirá la comunicación entre los nodos.

Pasos para la configuración de SSH:

1. Ejecutar los comandos y pulsar Enter cuántas veces lo necesite.

Desde **[hadoop@nodo1 ~]\$**

```
cd
ssh-keygen
```

2. Ejecutar los siguientes comandos para pasar la llave pública entre los nodos que tendrá el clúster.

```
cd ~/.ssh
cp id_rsa.pub authorized_keys
cat authorized_keys
```

```
[hadoop@nodo1 ~]$ cd ~/.ssh
[hadoop@nodo1 .ssh]$ cp id_rsa.pub authorized_keys
[hadoop@nodo1 .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQDPNvRbCMLrjsrie4UJnp+nHFH/Zm+n10GvAZGtwCNQ
MU/p9ahi2+TEQZjnCL59VE/n81UucE0AkW16wHzN/FBoII00tMw18E/v7NEcr1hAM/s3iXsVwiUoAJ0
jJpW0g6K3eAbTBVwyjs0Gw0Xu2SsxKtYh0uo9eMIW6rNcfc+KP91bp0e1Y00IB8icqyLOv6ZZvraXnpD
0ef39qIqUnj7xWcz+3PCgxGfNeGM8P+KKTal7xIFQYsw5o+UEEWPx5x7BJ8lvPMMsIonAvkPPEZZ8Rbf
j3YuyLpcFvyIujZjPL0utfJih7EJNwir3LcFSarfDeH0YcHI3vmtXj2BZuUjGPcbczpDTiDS/TVLEaKs
n9wb0jys/eKCgil+dM0bBFuvRGLfJs07X8WHjhp2LsnNGQYK8TMTK/5H/8nP90t9kV0ojuaoWnoZL
2dL67EVLws8GuhqBiYHf+BDDJIWUW4n0u6+ZdF6MaLdzqJlCVXxZtWv9T1x4fG15k8bMRVc= hadoop@
nodo1
[hadoop@nodo1 .ssh]$
```

3. Ejecutar el comando, saldrá la pregunta **¿Estás seguro de que quieres seguir conectado?**, escribir **“yes”** y dar Enter. Automáticamente conectará con el usuario del equipo sin pedir contraseña sin importar que sea de manera local.

```
ssh nodo1
```

```
[hadoop@nodo1 .ssh]$ ssh nodo1
The authenticity of host 'nodo1 (192.168.11.128)' can't be established.
ECDSA key fingerprint is SHA256:Qa+j0mkKgiZEpj5QKGMaOGIF3St7M5z86MQsipld2ow.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'nodo1,192.168.11.128' (ECDSA) to the list of known h
osts.
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Oct 27 15:36:47 2020
[hadoop@nodo1 ~]$
```

Al finalizar, escriba **“exit”** para volver al nodo original y cerrar la conexión de SSH.

Montaje de Clúster Pseudo distribuido

Configuración del archivo core-site.xml

Desde **[hadoop@nodo1 ~]\$** deberá acceder al archivo core-site.xml, ya que, este fichero es el que contiene las propiedades y configuraciones generales del clúster: NameNode, Secondary NameNode y DataNodes.

Ejecutar los siguientes comandos:

```
cd /opt/hadoop/etc/hadoop/
gedit core-site.xml
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://nodo1:9000</value>
  </property>
</configuration>
```

Configuración del archivo hdfs-site.xml

Desde **[hadoop@nodo1 ~]\$** deberá acceder al archivo hdfs-site.xml, ya que, contiene la configuración para los datos del sistema de ficheros de HDFS.

Ejecutar el siguiente comando:

```
gedit hdfs-site.xml
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/datos/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/datos/datanode</value>
  </property>
</configuration>
```

Abrir una nueva pestaña (Ctrl+Shift+T) e ingresar como usuario **root** para crear un directorio y subdirectorios en la raíz del sistema, además cambiar el propietario del directorio.

```
mkdir /datos
mkdir /datos/namenode
mkdir /datos/datanode
cd /
chown -R hadoop:hadoop /datos
```

Volver a la pestaña **[hadoop@nodo1 ~]\$** y formatear el sistema de ficheros con los siguientes comandos:

```
cd
hadoop namenode -format
```

Pasos para Iniciar y probar HDFS:

1. Ejecutar el comando para iniciar los servicios HDFS.

```
start-dfs.sh
```

```
[hadoop@nodo1 ~]$ cd
[hadoop@nodo1 ~]$ start-dfs.sh
Starting namenodes on [nodo1]
Starting datanodes
Starting secondary namenodes [nodo1]
[hadoop@nodo1 ~]$
```

2. Comprobar el levantamiento de los servicios con **jps**

```
[hadoop@nodo1 ~]$ jps
20146 SecondaryNameNode
19928 DataNode
20300 Jps
19711 NameNode
[hadoop@nodo1 ~]$
```

3. Acceder al navegador y escribir la dirección **nodo1:9870**, se presenta una página web que indica que está correcto el proceso anterior.

Started:	Tue Oct 27 17:39:08 -0500 2020
Version:	3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled:	Mon Jul 06 13:44:00 -0500 2020 by brahma from branch-3.3.0
Cluster ID:	CID-2b5c558b-8df4-4d2b-9ce7-f7e75a3b23b8
Block Pool ID:	BP-1704107792-192.168.11.128-1603838335403

```
stop-dfs.sh
```

```
[hadoop@nodo1 ~]$ stop-dfs.sh
Stopping namenodes on [nodo1]
Stopping datanodes
Stopping secondary namenodes [nodo1]
[hadoop@nodo1 ~]$ cd /opt/hadoop/etc/hadoop/
[hadoop@nodo1 ~]$
```

Comprobado esto, regresar a la terminal y detener los servicios HDFS.

Probar el funcionamiento de HDFS

4. Crear un directorio de ejemplo y luego subir un archivo.

```
start-dfs.sh
echo HolaMundo >> ~/Descargas/holaMundo.txt
cat ~/Descargas/holaMundo.txt
hdfs dfs -mkdir /prueba
hdfs dfs -put ~/Descargas/holaMundo.txt /prueba
hdfs dfs -ls /prueba
```

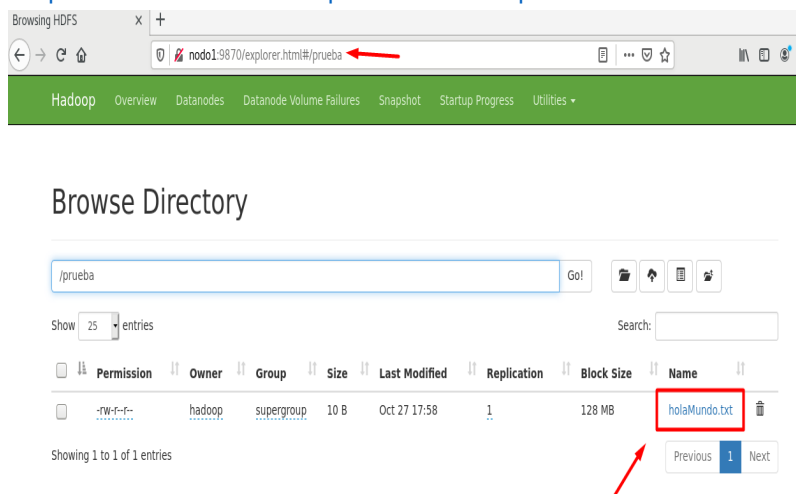
El comando “echo” crea un fichero de texto y el comando “cat” permite visualizar el contenido en el archivo.

Con “hdfs dfs -mkdir” se crea un directorio en HDFS
Con “hdfs dfs -put” sube un archivo del cliente al HDFS.

5. Visualizar y comprobar accediendo al navegador del administrador web HDFS.

```
[hadoop@nodo1 ~]$ start-dfs.sh
Starting namenodes on [nodo1]
Starting datanodes
Starting secondary namenodes [nodo1]
[hadoop@nodo1 ~]$ echo HolaMundo >> ~/Descargas/holaMundo.txt
[hadoop@nodo1 ~]$ cat ~/Descargas/holaMundo.txt
HolaMundo
[hadoop@nodo1 ~]$ hdfs dfs -mkdir /prueba
[hadoop@nodo1 ~]$ hdfs dfs -put ~/Descargas/holaMundo.txt /prueba
[hadoop@nodo1 ~]$ hdfs dfs -ls /prueba
Found 1 items
-rw-r--r-- 1 hadoop supergroup 10 2020-10-27 17:58 /prueba/holaMundo.txt
[hadoop@nodo1 ~]$
```

<http://nodo1:9870/explorer.html#/prueba>



Servicio Yarn

Configuración del archivo yarn-site.xml

Desde `[hadoop@nodo1 ~]$` primero se debe detener los servicios HDFS, para luego acceder al archivo **yarn-site.xml**, ya que, el mismo, configura el modo de trabajo del proceso YARN

Ejecute los siguientes comandos:

```
stop-dfs.sh
cd /opt/hadoop/etc/hadoop/
```

Editar el archivo yarn-site.xml:

```
gedit yarn-site.xml
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

```
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>nodo1</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce_shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.application.classpath</name>
    <value>
      /opt/hadoop/hadoop/etc/hadoop,
      /opt/hadoop/share/hadoop/common/*,
      /opt/hadoop/share/hadoop/common/lib/*,
      /opt/hadoop/share/hadoop/hdfs/*,
      /opt/hadoop/share/hadoop/hdfs/lib/*,
      /opt/hadoop/share/hadoop/mapreduce/*,
      /opt/hadoop/share/hadoop/mapreduce/lib/*,
      /opt/hadoop/share/hadoop/yarn/*,
      /opt/hadoop/share/hadoop/yarn/lib/*
    </value>
  </property>
</configuration>
```

Pasos para iniciar y probar el servicio de Yarn:

1. Ejecutar el comando a continuación para iniciar los servicios HDFS y YARN.

```
start-dfs.sh
start-yarn.sh
```

2. Comprobar la página de administración web de YARN. A la izquierda, una lista de opciones del clúster y, a la derecha, las métricas y la planificación reservada para las aplicaciones que utilizan YARN.

<http://nodo1:8088>

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
0	0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>

Showing 0 to 0 of 0 entries

3. Para observar los nodos activos debe hacer clic en la opción "Nodes" del menú Cluster ubicado en la parte izquierda de la ventana.

Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VC U
0	0	0	0	0	0 B	8 GB	0 B	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebo
1	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Showing 1 to 1 of 1 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Used	Mem Avail
/default-rack		RUNNING	nodo1:41675	nodo1:8042	mar oct 27 18:10:21 -0500 2020		0		0 B	8 GB

Showing 1 to 1 of 1 entries

4. Hacer clic en el link **"nodo1:8042"** para acceder a la información del nodo activo. En esta página se encuentran las opciones del NodeManager, tales como la información del nodo, la lista de aplicaciones ejecutadas y la

Nodes of the cluster - Mozilla Firefox

Browsing HDFS

Nodes of the cluster

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved
0	0	0	0	0	0 B	8 GB	0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
1	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Showing 20 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allocation Tags	Mem Use
/default-rack		RUNNING	nodo1:40513	nodo1:8042	sáb ene 25 23:57:14 -0500 2020		0		0 B

lista de
contenedores.



- ResourceManager
- NodeManager
 - Node Information
 - List of Applications
 - List of Containers
- Tools

NodeManager information	
Total Vmem allocated for Containers	16.80 GB
Vmem enforcement enabled	true
Total Pmem allocated for Container	8 GB
Pmem enforcement enabled	true
Total VCores allocated for Containers	8
Resource types	memory-mb (unit=Mi), vcores
NodeHealthyStatus	true
LastNodeHealthTime	Tue Oct 27 18:12:21 ECT 2020
NodeHealthReport	
NodeManager started on	Tue Oct 27 18:08:17 ECT 2020
NodeManager Version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum e0a276649f889c15d0e8f08eccd0c10 on 2020-07-06T18:58Z
Hadoop Version:	3.3.0 from aa96f1871bfd858f9bac59cf2a81ec470da649af by brahma source checksum 5dc29b802d6ccd77b262ef9d04d19c4 on 2020-07-06T18:44Z

Configuración del archivo mapred-site.xml

Desde **[hadoop@nodo1 ~]\$** se debe acceder al archivo mapred-site.xml, ya que, contiene la configuración para los datos del sistema de ficheros de HDFS.

Ejecutar el siguiente comando:

```
gedit mapred-site.xml
```

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

Inicio y prueba de Yarn a través de MapReduce:

1. Ejecutar el comando a continuación para iniciar los servicios JobHistory Server.

```
mapred --daemon start historyserver
```

Prueba con MapReduce:

Ejemplo que consiste en contar palabras que se repiten en un fichero de texto almacenado en HDFS mediante la clase **"Word-Count"** del archivo **"-mapreduce-examples-3.4.0.jar"**.

- Primero crear un fichero de texto y sirlo al clúster.

```
echo Hola mundo este es un fichero de
texto para el clúster :D >>
/tmp/texto1.txt
cat /tmp/texto1.txt
hdfs dfs -put /tmp/texto1.txt /prueba
hdfs dfs -ls /prueba
hdfs dfs -cat /prueba/texto1.txt
```

```
[hadoop@nod01 hadoop]$ mapred --daemon start historyserver
[hadoop@nod01 hadoop]$ echo Hola mundo este es un fichero de texto para el clúster :D >> /tmp/texto1.txt
[hadoop@nod01 hadoop]$ cat /tmp/texto1.txt
Hola mundo este es un fichero de texto para el clúster :D
[hadoop@nod01 hadoop]$ hdfs dfs -put /tmp/texto1.txt /prueba
[hadoop@nod01 hadoop]$ hdfs dfs -ls /prueba
Found 2 items
-rw-r--r-- 1 hadoop supergroup 10 2020-10-27 17:58 /prueba/holaMundo.txt
-rw-r--r-- 1 hadoop supergroup 59 2020-10-27 18:21 /prueba/texto1.txt
[hadoop@nod01 hadoop]$ hdfs dfs -cat /prueba/texto1.txt
Hola mundo este es un fichero de texto para el clúster :D
[hadoop@nod01 hadoop]$
```

- Ejecutar MapReduce indicando con JAR, clase.jar, carpetas de entrada y salida según la versión de Hadoop.

```
hadoop jar
/opt/hadoop/share/hadoop/mapreduce/hado
op-mapreduce-examples-3.4.0.jar
wordcount /prueba /salida_texto1
```

- Al ejecutar la aplicación en la terminal puede apreciar el proceso que se realiza mediante MapReduce, para observar los resultados escriba los siguientes comandos

```
hdfs dfs -ls /salida_texto1
hdfs dfs -cat /salida_texto1/part-r-00000
```

```
[hadoop@nod01 hadoop]$ hdfs dfs -ls /salida_texto1
Found 2 items
-rw-r--r-- 1 hadoop supergroup 0 2024-12-14 23:21 /salida_texto1/_SUCCESS
-rw-r--r-- 1 hadoop supergroup 95 2024-12-14 23:21 /salida_texto1/part-r-00000
[hadoop@nod01 hadoop]$ hdfs dfs -cat /salida_texto1/part-r-00000
:D 1
Hola 1
HolaMundo 1
clúster 1
de 1
el 1
es 1
este 1
fichero 1
mundo 1
para 1
texto 1
un 1
[hadoop@nod01 hadoop]$
```

5. Comprobar la página de administración web de YARN para observar la aplicación "Word Count"

<http://nodo1:8088>

Cluster Metrics									
1	Apps Submitted	0	Apps Pending	0	Apps Running	1	Apps Completed	0	Containers Running
Cluster Nodes Metrics									
1	Active Nodes	0	Decommissioning Nodes	0	Decommissioned Nodes				
Scheduler Metrics									
Scheduler Type			Scheduling Resource Type				Minimum Allocation		
Capacity Scheduler			[memory-mb (unit=Mi), vcores]				<memory:1024, vCores:1>		
Show 20 entries									
ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1603840095439_0001	hadoop	word count	MAPREDUCE		default	0	Tue Oct 27 18:24:27 -0500 2020	Tue Oct 27 18:28:03 -0500 2020	Tue Oct 27 18:29:45 -0500 2020
Showing 1 to 1 of 1 entries									

6. Hacer clic en el ID de la aplicación para acceder a una información más detallada como el tiempo de ejecución, el estado, el nodo donde se ejecutó etc.

Show: 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime
application_1603840095439_0001	hadoop	word count	MAPREDUCE		default	0	Tue Oct 27 18:24:27 -0500 2020	Tue Oct 27 18:28:03 -0500 2020	Tue Oct 27 18:29:45 -0500 2020

 Application application_1603840095439_0001

Cluster

- About
- Nodes
- Node Labels
- Applications
- NEW
- SAVING
- SUBMITTED
- ACCEPTED
- RUNNING
- FINISHED
- FAILED
- KILLED
- Scheduler

Tools

User: hadoop

Name: word count

Application Type: MAPREDUCE

Application Tags:

Application Priority: 0 (Higher Integer value indicates higher priority)

YarnApplicationState: FINISHED

Queue: default

FinalStatus Reported by AM: SUCCEEDED

Started: mar oct 27 18:24:27 -0500 2020

Launched: mar oct 27 18:28:03 -0500 2020

Finished: mar oct 27 18:29:45 -0500 2020

Elapsed: 5mins, 18sec

Tracking URL: History

Log Aggregation Status: DISABLED

Application Timeout (Remaining Time): Unlimited

Diagnostics:

Unmanaged Application: false

Application Node Label expression: <Not set>

AM container Node Label expression: <DEFAULT_PARTITION>

Escribir JPS para visualizar los servicios que se tiene iniciados. Detener todos los servicios y posteriormente apagar la máquina virtual.

```
jps
mapred --daemon stop historyserver
stop-yarn.sh
stop-dfs.sh
```

Finalmente debe ingresar como usuario **root** para apagar la máquina virtual de esta manera.

```
su - root
init 0
```

Configuración de nodos de Datos

Es necesario tomar en cuenta que, para la instalación de un nodo de datos de forma nativa, se necesita un equipo con sistema operativo CentOS 8; además, para cada nodo se requiere al menos 2 GB de RAM y 25 GB de almacenamiento. En cada equipo (nodo) deberá configurar un nombre de host único como nodo2, nodo3, nodo4, etc., según la cantidad de nodos que se planea implementar.

De lo contrario, como lo es en esta práctica que se desea montar el clúster en máquinas virtuales, se debe clonar la máquina virtual del nodo1 (Master) en estado apagado, las veces que se requiera según la cantidad de nodos de datos del clúster.



¡IMPORTANTE!

Caso 1: Si se va a implementar el clúster en máquinas virtuales de diferentes equipos físicos, cambie el segundo adaptador de red de "Custom" a "Bridged" y marcar la casilla "Replicate physical network connection state" (punto 12 de configuración de VMware), en el nodo1. Esto servirá para conectar en red las máquinas virtuales de los diferentes equipos físicos.

Caso 2: Si se va a montar el clúster en un mismo equipo, no cambie el segundo adaptador de red del nodo 1, dejando la configuración de red 2 en "Custom" (punto 13 de configuración de VMware), esto permite que las máquinas virtuales, se conecten mediante red interna simulando una red cableada.

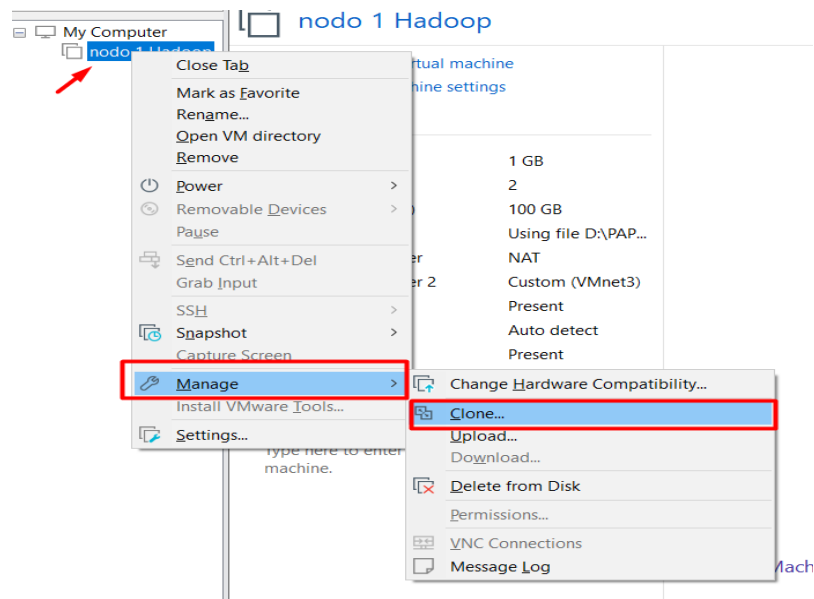
Creación de nodos de datos



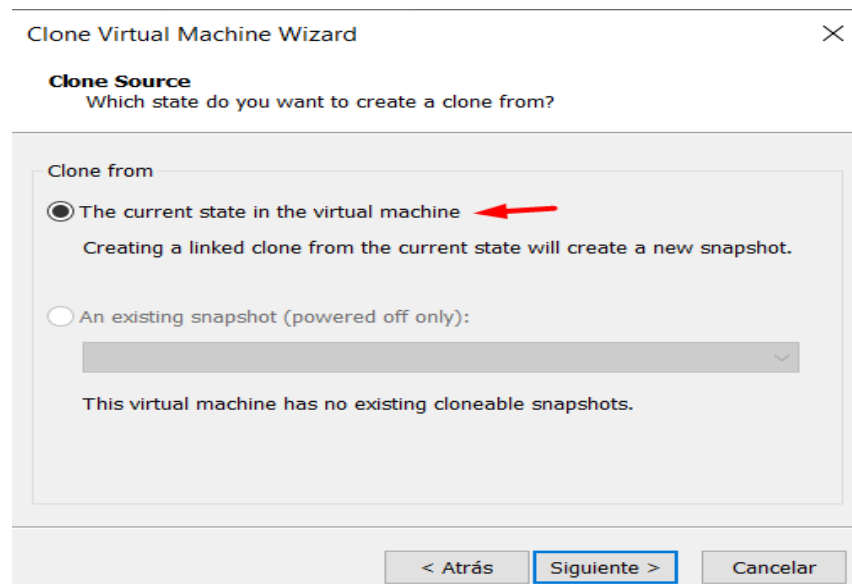
NOTA: Debe asegurarse que la máquina virtual del nodo 1, esté apagada.

Proceso para clonación de máquina virtual del nodo 1 (máster):

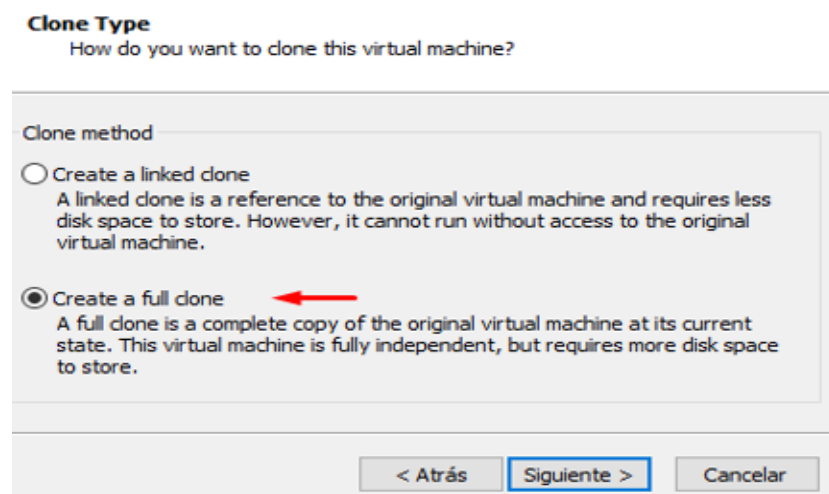
1. Hacer clic derecho sobre la máquina virtual a clonar, seleccionar "Manage" y luego clic en "Clone", al aparecer la asistencia para la clonación, dar clic en "Next" para avanzar.



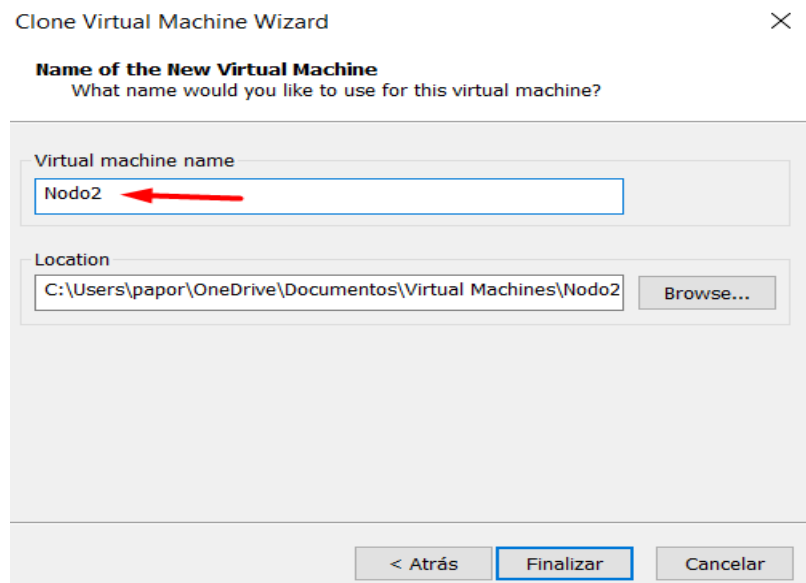
2. Marcar la casilla "The current state in the virtual machine" y hacer clic en "Siguiente" para continuar.



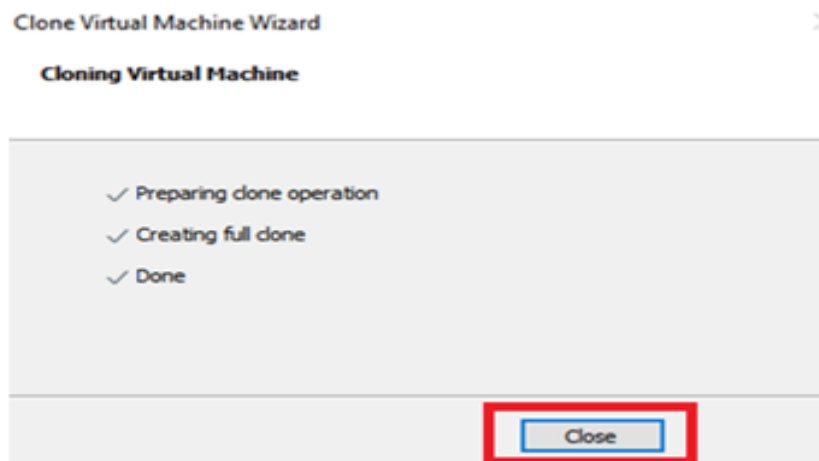
3. Marcar la casilla "Create" a "full done" para realizar una copia completa de la máquina virtual a una nueva totalmente independiente.



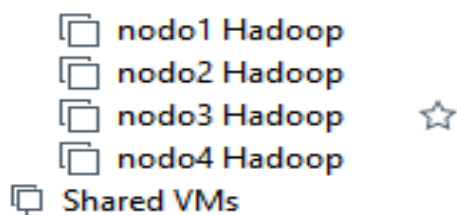
4. Escribir el nombre de la nueva máquina virtual, y elegir la ruta donde se va a almacenar. Hacer clic en "Finalizar" para iniciar el proceso de clonación.



5. Hacer clic en "Close" para finalizar la clonación. Realizar el mismo proceso para clonar los demás nodos que vaya a utilizar en el clúster.



Este proceso debe repetirse de acuerdo con el número de nodos que se desea implementar en el clúster. En nuestro caso se clona 3 veces al nodo Master.



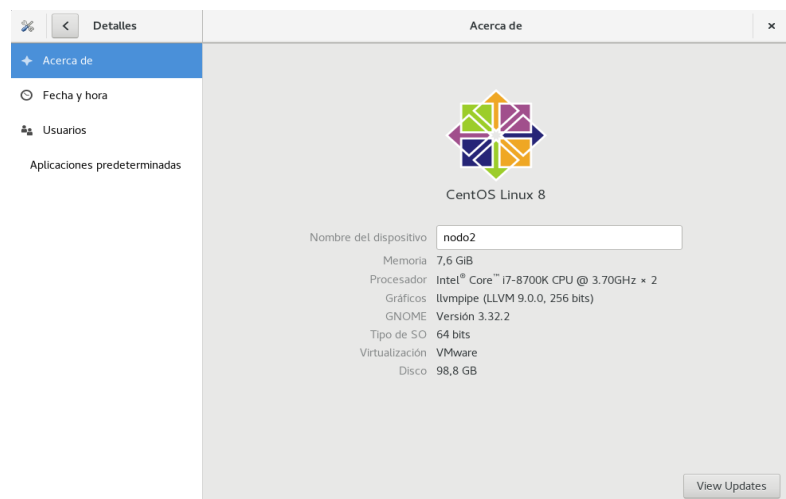
Configuración inicial de nodos

Se debe iniciar todas las máquinas virtuales, en cada una de ellas a excepción del nodo 1. Cambiar el nombre del equipo desde la aplicación de configuración del sistema.

Realizar este procedimiento para cada uno de los nodos clonados del clúster.

Pasos iniciales para configurar los nodos de datos:

1. Diríjase a la configuración y seleccione donde dice "Detalle", haga clic en "Acerca de" y en el apartado del nombre del dispositivo, escriba el nombre correspondiente del equipo (nodo2, nodo3, nodo4).



Configurar la red de comunicación:

2. Modificar el archivo hosts en el **nodo Master y en cada nodo de datos.**

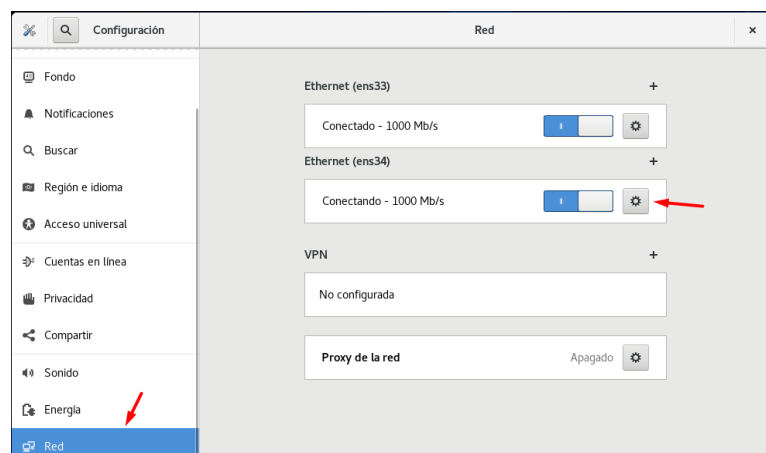
```
su - root
gedit /etc/hosts
```

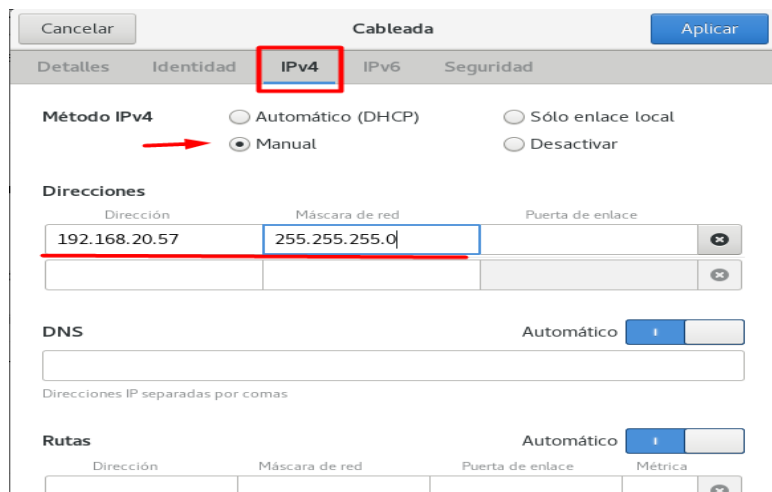
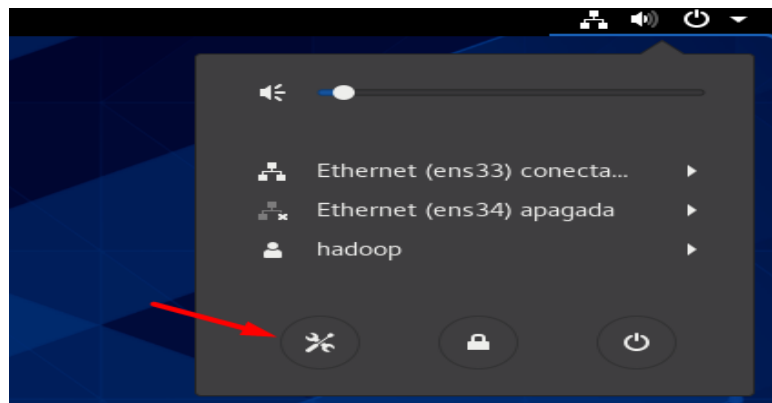
Agregar las direcciones IP propuestas en la Tabla 9 de la siguiente manera:

```
GNU nano 2.9.8 /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
192.168.20.57 nodo1
192.168.20.62 nodo2
192.168.20.63 nodo3
192.168.20.64 nodo4
```

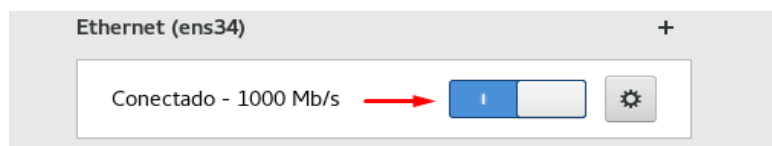
Realizar este procedimiento para cada uno de los nodos del clúster.

3. Configurar la red en todos los nodos, con la dirección ip correspondiente a cada uno, en el **segundo adaptador de red.**





4. Apagar y encender la interfaz de red.



5. Abrir una Terminal y mediante el comando ping realizar las pruebas de conectividad a los demás nodos y viceversa.

```
64 bytes from nodo1 (192.168.20.57): icmp_seq=14 ttl=64 time=0.048 ms
^C
--- nodo1 ping statistics ---
14 packets transmitted, 14 received, 0% packet loss, time 305ms
rtt min/avg/max/mdev = 0.029/0.057/0.273/0.060 ms
[hadoop@nodo1 ~]$ ping nodo2
PING nodo2 (192.168.20.62) 56(84) bytes of data.
64 bytes from nodo2 (192.168.20.62): icmp_seq=1 ttl=64 time=11.6 ms
64 bytes from nodo2 (192.168.20.62): icmp_seq=2 ttl=64 time=0.971 ms
64 bytes from nodo2 (192.168.20.62): icmp_seq=3 ttl=64 time=0.333 ms
64 bytes from nodo2 (192.168.20.62): icmp_seq=4 ttl=64 time=0.610 ms
64 bytes from nodo2 (192.168.20.62): icmp_seq=5 ttl=64 time=0.912 ms
64 bytes from nodo2 (192.168.20.62): icmp_seq=6 ttl=64 time=0.375 ms
^C
--- nodo2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 103ms
rtt min/avg/max/mdev = 0.333/2.460/11.560/4.076 ms
[hadoop@nodo1 ~]$ ping nodo3
PING nodo3 (192.168.20.63) 56(84) bytes of data.
64 bytes from nodo3 (192.168.20.63): icmp_seq=1 ttl=64 time=14.8 ms
64 bytes from nodo3 (192.168.20.63): icmp_seq=2 ttl=64 time=14.9 ms
64 bytes from nodo3 (192.168.20.63): icmp_seq=3 ttl=64 time=0.378 ms
64 bytes from nodo3 (192.168.20.63): icmp_seq=4 ttl=64 time=14.8 ms
^C
--- nodo3 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 74ms
rtt min/avg/max/mdev = 0.378/11.222/14.879/6.260 ms
[hadoop@nodo1 ~]$ ping nodo4
PING nodo4 (192.168.20.64) 56(84) bytes of data.
64 bytes from nodo4 (192.168.20.64): icmp_seq=1 ttl=64 time=0.524 ms
64 bytes from nodo4 (192.168.20.64): icmp_seq=2 ttl=64 time=0.630 ms
64 bytes from nodo4 (192.168.20.64): icmp_seq=3 ttl=64 time=0.268 ms
^C
--- nodo4 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 82ms
rtt min/avg/max/mdev = 0.268/0.474/0.630/0.151 ms
[hadoop@nodo1 ~]$
```

Continuar con la configuración en Nodo Máster

Servicio SSH

Una vez configurada la red entre los nodos, realizar la conexión segura denominada SSH para que los equipos se puedan comunicar entre sí.

Pasos para reconfiguración SSH en el nodo1 **[hadoop@nodo1 ~]\$**:

1. Ingresar al directorio ~/.ssh del nodo1 y borrar todo el contenido de esa carpeta.

```
[hadoop@nodo1 ~]$ cd ~/.ssh/
[hadoop@nodo1 .ssh]$ ls -l
authorized_keys
id_rsa
id_rsa.pub
known_hosts
[hadoop@nodo1 .ssh]$ rm *
[hadoop@nodo1 .ssh]$
```


2. Abrir una nueva pestaña de la Terminal (Ctrl+Shift+T) para conectarse a los demás nodos. Para **cerrar** la sesión de **ssh** escriba **"exit"**.

```
ssh nodo2
yes
cd ~/.ssh/
rm *
exit
```

```
[hadoop@nodo1 .ssh]$ ssh nodo2
The authenticity of host 'nodo2 (192.168.20.62)' can't be established.
ECDSA key fingerprint is SHA256:Qa+j0mkKgiZEj5QKGmA0GIF3St7MSz86MQsipld2ow.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'nodo2,192.168.20.62' (ECDSA) to the list of known hosts.
hadoop@nodo2's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Oct 27 22:29:55 2020
[hadoop@nodo2 ~]$ cd ~/.ssh/
[hadoop@nodo2 .ssh]$ rm *
[hadoop@nodo2 .ssh]$ exit
logout
Connection to nodo2 closed.
[hadoop@nodo1 .ssh]$
```

3. Desde **[hadoop@nodo1 ~]\$** volver a generar la llave **RSA** con el comando.

```
ssh-keygen
```

```
[hadoop@nodo1 .ssh]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa.
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:FXBZUQb0TyiNcQMwLwGtHN/FlrxMJz2GA0g6JONY/Eg hadoop@nodo1
The key's randomart image is:
+---[RSA 3072]-----+
|      .+ .oo+=%OX=. |
|    +E+ .o.o.@+0* |
|   ...oo ..o+0+* |
|    . ... o..=. |
|      S      . |
|-----+
+----[SHA256]-----+
[hadoop@nodo1 .ssh]$
```

4. Hacer una copia de la llave RSA pública al fichero "authorized_keys".

```
cp id_rsa.pub authorized_keys
cat authorized_keys
```

```
[hadoop@nodo1 .ssh]$ cp id_rsa.pub authorized_keys
[hadoop@nodo1 .ssh]$ cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGC4Ubjb2D1jldhlm6dkKvAvKfxt0FblBvDICjuGmKmq
RWpWcpU6I0vF3pNC5AY04Rw06PnFk5sMATzWAZB6nSVBcNSm12iVbsM409JPysaj5L0n6XUz4Jlt/TT
lU4QXgIB6x1ZKH0I79WCsbksAN/oEYm2dDAEsRcFr01U7t1RIe1wWw10lLA6KBtJ0iZCwtow7ia2qQFF
D4/dXvSeH2ZwW8K/+MhVR0i5FQZXq89Hh9KgjJCJ7fEpFgqvXJK9vNwNNdymZ0T2G6K+ikGqvDX22d7T
lIvzv9jsyglDC6Qisp4RfSXGDuB+f0V0serfaZW258+lnKY8LIljU01uxRQ368UtCp0gsqvw8yFw+60
66hmgBWob1NvcljTZ1c6oa88lp8RBCJlvTuvyP/p0vPEKfmhwz4WjwvCPsjiul723B0gRCBpulatEp3F
BeaKo10S+Zqv1G3T0mjXJwoKV3m0KKAHVeY+EC4jnwju4Y74+w7mPnbMSJ12Ffn2B80jiEE= hadoop@
nodo1
[hadoop@nodo1 .ssh]$
```

5. Copiar el fichero "authorized_keys" al directorio ~/.ssh/ del usuario del nodo2.

```
scp authorized_keys nodo2:/home/hadoop/.ssh/
```

```
[hadoop@nodo1 .ssh]$ scp authorized_keys nodo2:/home/hadoop/.ssh/
hadoop@nodo2's password:
authorized_keys                                100% 566   860.6KB/s   00:00
[hadoop@nodo1 .ssh]$
```

Realizar el mismo procedimiento para los otros nodos de datos, abriendo una nueva pestaña.

Utilizar el comando `ssh`, para conectarse al **nodo2** y generar la llave **RSA**, mediante el comando **"cat"** agregar el contenido de la llave pública al fichero **"authorized_keys"**. Luego copiar el archivo **"authorized_keys"** al **nodo3**.

Nodo 2

```
ssh nodo2
ssh-keygen
cd .ssh/
cat id_rsa.pub >> authorized_keys
cat authorized_keys
scp authorized_keys nodo3:/home/hadoop/.ssh/
yes
exit
```

Nodo 3

```
ssh nodo3
ssh-keygen
cd .ssh/
cat id_rsa.pub >> authorized_keys
cat authorized_keys
scp authorized_keys nodo4:/home/hadoop/.ssh/
yes
exit
```

Nodo 4

```
ssh nodo4
ssh-keygen
cd .ssh/
cat id_rsa.pub >> authorized_keys
cat authorized_keys
```

Desde la terminal donde tiene abierta la sesión `ssh` del **nodo4**, copie el fichero **"authorized_keys"** a los demás nodos mediante **scp**.

```
scp authorized_keys nodo3:/home/hadoop/.ssh/
yes
scp authorized_keys nodo2:/home/hadoop/.ssh/
yes
scp authorized_keys nodo1:/home/hadoop/.ssh/
yes
```

Probar el funcionamiento de **ssh** iniciando sesión entre los distintos nodos. **No debe pedir contraseña** al momento de hacer la conexión con el nodo que desea conectarse.

Desde **[hadoop@nodo1 .ssh]\$** ejecutar:

```
ssh nodo2
ssh nodo3
ssh nodo4
```

Archivos de configuración

Para que el clúster funcione correctamente es necesario de que todos los nodos tengan los siguientes requerimientos:

- Todos los nodos deben tener el mismo usuario.
- Accesibles a través del servicio de SSH sin credenciales.
- Los mismos directorios en cada equipo.
- Debe existir la copia de Hadoop en todos los equipos y en el mismo sitio.
- El directorio de datos en todos los nodos y que tengan el permiso correspondiente.

Abrir una Terminal y entrar a los diferentes nodos, ingresar al directorio **/datos**, elimine la carpeta **/datos/namenode**. Entre en el directorio **/datos/datanode** y borre el contenido.

```
ssh nodo2
cd /datos/
rm -rf namenode/
cd datanode/
rm -rf current/
cd ..
exit
```

```
ssh nodo3
cd /datos/
rm -rf namenode/
cd datanode/
rm -rf current/
cd ..
exit
```

```
ssh nodo4
cd /datos/
rm -rf namenode/
cd datanode/
rm -rf current/
cd ..
exit
```

Desde **[hadoop@nodo1 ~]\$**, ingresar al directorio **/datos** y elimine la carpeta **/datos/datanode**.

```
cd /datos/
rm -rf datanode/
```

No se elimina **NameNode**, ya que, en el Nodo Máster deben ir los metadatos que es la configuración del clúster. En los nodos esclavos va el directorio **DataNode** que es donde se guardan los datos del sistema de ficheros HDFS.

Reconfiguración del archivo **hdfs-site.xml**

Desde **[hadoop@nodo1 hadoop]\$**, acceder al archivo **hdfs-site.xml** y modificar la configuración anterior.

Ejecutar los siguientes comandos:

```
cd /opt/hadoop/etc/hadoop/
gedit hdfs-site.xml
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/datos/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/datos/datanode</value>
  </property>

  <property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
  </property>
  <property>
    <name>hadoop.http.staticuser.user</name>
    <value>hadoop</value>
    <final>true</final>
  </property>
</configuration>
```

Configuración del archivo workers

Abrir el fichero "**workers**" para agregar el nombre de los nodos esclavos.

```
gedit workers
```

Dentro del editor de texto, añadir las siguientes etiquetas y guardar el archivo.

```
nodo2
nodo3
nodo4
```

Copiar los ficheros de configuración

Mediante el comando "**scp**" copiar los ficheros de configuración del clúster a los nodos esclavos.

Archivo core-site.xml

```
scp core-site.xml nodo2:/opt/hadoop/etc/hadoop/
scp core-site.xml nodo3:/opt/hadoop/etc/hadoop/
scp core-site.xml nodo4:/opt/hadoop/etc/hadoop/
```

Archivo hdfs-site.xml

```
scp hdfs-site.xml nodo2:/opt/hadoop/etc/hadoop/
scp hdfs-site.xml nodo3:/opt/hadoop/etc/hadoop/
scp hdfs-site.xml nodo4:/opt/hadoop/etc/hadoop/
```

Archivo mapred-site.xml

```
scp mapred-site.xml nodo2:/opt/hadoop/etc/hadoop/
scp mapred-site.xml nodo3:/opt/hadoop/etc/hadoop/
scp mapred-site.xml nodo4:/opt/hadoop/etc/hadoop/
```

Archivo yarn-site.xml

```
scp yarn-site.xml nodo2:/opt/hadoop/etc/hadoop/
scp yarn-site.xml nodo3:/opt/hadoop/etc/hadoop/
scp yarn-site.xml nodo4:/opt/hadoop/etc/hadoop/
```

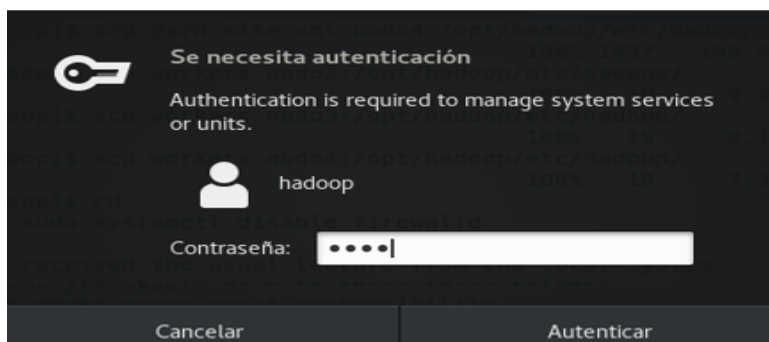
Archivo workers

```
scp workers nodo2:/opt/hadoop/etc/hadoop/
scp workers nodo3:/opt/hadoop/etc/hadoop/
scp workers nodo4:/opt/hadoop/etc/hadoop/
```

Desactivar firewall del sistema

Desactivar el firewall y cortafuegos del sistema en todos los nodos con los siguientes comandos:

```
cd
sudo systemctl disable firewalld
systemctl stop firewalld
```



Para este proceso hay que tener en cuenta que pedirá la autenticación del usuario Hadoop y se debe poner la contraseña establecida.

Este procedimiento se debe realizar para cada uno de los nodos esclavos, entrando desde el nodo1 hacia el **nodo2, 3 y 4** mediante **ssh**.

```
ssh2
sudo systemctl disable firewalld
systemctl stop firewalld
```

```
ssh3
sudo systemctl disable firewalld
systemctl stop firewalld
```

```
ssh4
sudo systemctl disable firewalld
systemctl stop firewalld
```

Hay que recordar que, en un entorno real, no se debe desactivar el firewall y

cortafuegos del sistema, sino que se deben habilitar los puertos necesarios en el cortafuego de su sistema operativo.

Formatear e iniciar el cluster

Desde el **Nodo1**, debe **formatear el Namenode** con el comando **"hdfs namenode -format"** en el nodo maestro.

Proceso para iniciar el clúster:

1. Ejecutar el comando a continuación en la terminal del **nodo1**:

```
hdfs namenode -format
```

```
2020-10-28 00:03:50,311 INFO util.GSet: capacity = 2^13 = 8192 entries
Re-format filesystem in Storage Directory root= /datos/namenode; location= null
? (Y or N) y
2020-10-28 00:03:59,587 INFO namenode.FSImage: Allocated new BlockPoolId: BP-969
678630-192.168.20.57-1603861439579
2020-10-28 00:03:59,587 INFO common.Storage: Will remove files: [/datos/namenode
/current/VERSION, /datos/namenode/current/seen_txid, /datos/namenode/current/fsi
mage_00000000000000000000.md5, /datos/namenode/current/fsimage_0000000000000000
0, /datos/namenode/current/edits_00000000000000000001-0000000000000000002, /datos
/namenode/current/edits_00000000000000000003-0000000000000000003, /datos/namenode
/current/edits_00000000000000000004-0000000000000000005, /datos/namenode/current/
edits_00000000000000000006-00000000000000000013, /datos/namenode/current/edits_000
00000000000000014-00000000000000000015, /datos/namenode/current/edits_inprogress_0
0000000000000000016]
2020-10-28 00:03:59,619 INFO common.Storage: Storage directory /datos/namenode h
as been successfully formatted.
2020-10-28 00:03:59,638 INFO namenode.FSImageFormatProtobuf: Saving image file /
datos/namenode/current/fsimage.ckpt_00000000000000000000 using no compression
2020-10-28 00:03:59,735 INFO namenode.FSImageFormatProtobuf: Image file /datos/n
amenode/current/fsimage.ckpt_00000000000000000000 of size 401 bytes saved in 0 se
conds
2020-10-28 00:03:59,744 INFO namenode.NNStorageRetentionManager: Going to retain
1 images with txid >= 0
2020-10-28 00:03:59,754 INFO namenode.FSImage: FSImageSaver clean checkpoint: tx
id=0 when meet shutdown.
2020-10-28 00:03:59,754 INFO namenode.NameNode: SHUTDOWN MSG:
```

El sistema le preguntará si desea formatear el HDFS, escriba **"Y"** o **"y"** para continuar.

2. Arrancar los servicios HDFS, Yarn y History Server.

Se puede ver que en el **Nodo1** (Master), se levantan los servicios **NameNode**.

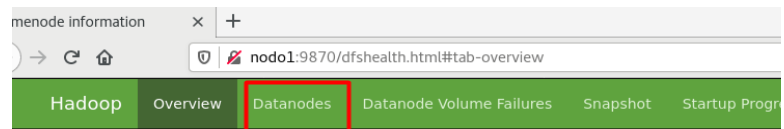
3. Ejecutar **jps** en un nodo esclavo, para verificar que los servicios están iniciados, en este caso solo serán **DataNode**.

```
start-dfs.sh
start-yarn.sh
mapred --daemon start historyserver
```

```
hadoop@nodo2:~
Archivo Editar Ver Buscar Terminal Ayuda
[hadoop@nodo2 ~]$ jps
2752 DataNode
2939 NodeManager
3115 Jps
[hadoop@nodo2 ~]$
```

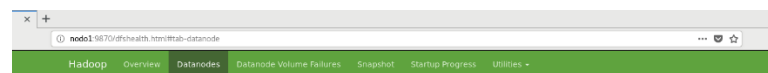
4. Comprobar la página de administración de HDFS y verificar la información de los nodos esclavos (deben estar <http://nodo1:9870/>)

levantados 3
nodos).



Overview 'nodo1:9000' (✓active)

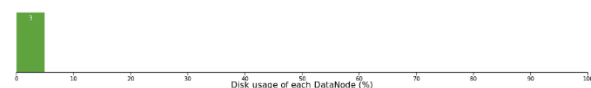
Started:	Wed Oct 28 00:06:36 -0500 2020
Version:	3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled:	Mon Jul 06 13:44:00 -0500 2020 by brahma from branch-3.3.0
Cluster ID:	CID-2149a466-cbf4c0c-95ec-fa20bb9672ed
Block Pool ID:	BP-969578630-192.168.20.57-1603861439579



Datanode Information

✓ In service ⚠ Down ⚠ Decommissioned ⚠ Decommissioned & dead ⚠ In Maintenance & dead

Datanode usage histogram



In operation

Show 25 entries

Node	Http Address	Last contact	Last Block Report	Capacity	Blocks	Block pool used	Version
✓nodo2:9866 (192.168.20.62:9866)	http://nodo2:9866	0s	8m	49.98 GB	0	4 KB (0%)	3.2.0

DataNode State

All

Show

25

entries

Search:

Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity
✓ nodo2:9866 (192.168.20.62:9866)	http://nodo2:9866	1s	6m	4 KB	6.85 GB	49.98 GB
✓ nodo3:9866 (192.168.20.63:9866)	http://nodo3:9866	1s	6m	4 KB	6.85 GB	49.98 GB
✓ nodo4:9866 (192.168.20.64:9866)	http://nodo4:9866	1s	6m	4 KB	6.85 GB	49.98 GB

Showing 1 to 3 of 3 entries

- Comprobar la página de administración de YARN.

Hacer clic en el enlace **"Nodes"**, para visualizar el estado de los nodos esclavos activos.

<http://nodo1:8088/>

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Allo T
	/default-rack	RUNNING	nodo2:46241	nodo2:8042	mié oct 28 00:15:15 -0500 2020		0	
	/default-rack	RUNNING	nodo4:32999	nodo4:8042	mié oct 28 00:15:15 -0500 2020		0	
	/default-rack	RUNNING	nodo3:41565	nodo3:8042	mié oct 28 00:15:14 -0500		0	

Prueba con MapReduce:

- Primero crear un fichero de texto y subirlo al clúster.

```
hdfs dfs -mkdir /prueba
echo Prueba de subida al cluster Hadoop >>
~/Documentos/texto_prueba.txt
hdfs dfs -put ~/Documentos/texto_prueba.txt
/prueba
hdfs dfs -ls /prueba
```

- Ejecutar una librería con **jar**, indicando el JAR, clase, entrada y salida según la versión de Hadoop.

```
hadoop jar
/opt/hadoop/share/hadoop/mapreduce/hado
op-mapreduce-examples-3.4.0.jar
wordcount /prueba /salida_texto1
```

- Comprobar la página de administración web de **YARN**, que se esté ejecutando la **aplicación MapReduce**

- Observar los resultados del conteo en consola con estos comandos.

```
hdfs dfs -ls /salida_texto1
hdfs dfs -cat /salida_texto1/part-r-00000
```

```
[hadoop@nod01 ~]$ hdfs dfs -ls /salida_texto1
Found 2 items
-rw-r--r-- 3 hadoop supergroup 0 2019-08-27 23:44 /salida_texto1/_SUCCESS
-rw-r--r-- 3 hadoop supergroup 47 2019-08-27 23:44 /salida_texto1/part-r-00000
[hadoop@nod01 ~]$ hdfs dfs -cat /salida_texto1/part-r-00000

Hadoop 1
Prueba 1
al 1
cluster 1
de 1
subida 1
[hadoop@nod01 ~]$
```

Instalación Apache Spark

Una de las primeras cosas que llama la atención es que mientras Hadoop fue escrito en Java, Spark ha sido desarrollado en Scala, un lenguaje de programación más funcional y conciso que Java. Una de las propiedades más interesantes de una solución de código abierto es la actividad de su comunidad (Ortega Candel, 2022). Desde el **Nodo1 maestro** ingresar a la página de descarga de Apache Spark desde el siguiente enlace

<https://spark.apache.org/downloads.html>, elija la versión a instalar, para este manual se utilizó la versión 3.4.4.

Pasos para la descarga, instalación y configuración de Apache Spark:

Descarga:

1. Escoger el tipo de paquete **"Pre-built user-provided Apache Hadoop"**.

Hacer clic en el enlace "spark-3.4.4-bin-without-hadoop.tgz" para proceder con la descarga.

Download Apache Spark™

1. Choose a Spark release:
2. Choose a package type:
3. Download Spark: [spark-3.4.4-bin-without-hadoop.tgz](#)
4. Verify this release using the 3.4.4 [signatures](#), [checksums](#) and [project release KEYS](#)



ESTABLISHED 1999

We suggest the following location for your download:

<https://dlcdn.apache.org/spark/spark-3.4.4/spark-3.4.4-bin-without-hadoop.tgz>

Extracción y configuración

2. Ingresar al directorio **/opt/**, y extraer el archivo que se encuentra en la carpeta Descargas

```
cd /opt/hadoop/
tar xvf ~/Descargas/spark-3.4.4-bin-without-hadoop.tgz
```

3. Renombrar la carpeta que se generó de la extracción del fichero de instalación de Spark.

```
mv spark-3.4.4-bin-without-hadoop/ spark
```

4. Editar el archivo `~/bashrc` para agregar las variables de entorno de Spark.

```
gedit ~/.bashrc
```

5. Agregar las siguientes variables de entorno

```
export HADOOP_HOME=/opt/hadoop
export
PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:/opt/hadoop/spark/bin:/opt/hadoop/spark/sbin
export JAVA_HOME=/usr/java/jdk1.8.0_202-amd64
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export SPARK_DIST_CLASSPATH=$(hadoop classpath)
export SPARK_HOME=/opt/hadoop/spark
```

6. Actualizar el archivo `~/bashrc` con el comando `". ~/.bashrc"`.

```
. ~/.bashrc
```

Prueba de Apache Spark:

7. En la Terminal ejecutar el comando “**spark-shell**” que permitirá abrir una sesión local con Scala.

```
spark-shell
```

8. Escribir la siguiente sentencia para leer el fichero README.md

```
val  
fichero=sc.textFile("file:///opt/hadoop/spark/README.md")  
fichero.count()
```

Para cerrar la sesión, escribir el comando “**sys.exit()**”

9. Ejecutar el comando a continuación para iniciar los servicios HDFS y YARN.

```
start-dfs.sh  
start-yarn.sh
```

10. Para probar el funcionamiento de Spark en el clúster, ejecutar el siguiente comando.

```
spark-submit --class  
org.apache.spark.examples.SparkPi --master  
yarn --deploy-mode cluster --name "apli1"  
/opt/hadoop/spark/examples/jars/spark-  
examples_2.12-3.4.4.jar 5
```

11. Acceder al administrador web de YARN, verificar que la aplicación que lanzó al clúster es del tipo “Spark”.

Show 20 entries							
ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime
application_1737506403924_0001	hadoop	apli1	SPARK		root.default	0	Tue Jan 21 19:42:10 -0500 2025

12. Hacer clic en el ID de la aplicación, luego hacer clic en "Logs" para acceder al registro de salida de la aplicación.

Show 20 entries			
Attempt ID	Started	Node	Logs
appattempt_1737506403924_0001_000001	Tue Jan 21 19:42:10 -0500 2025	http://nodo2:8042	Logs
Showing 1 to 1 of 1 entries			

13. Hacer clic en "stdout" para acceder al registro de salida del Log de la aplicación.



▼ ResourceManager
RM Home
► NodeManager
► Tools

Local Logs:

[directory.info](#) : Total file length is 28363 bytes.
[launch_container.sh](#) : Total file length is 6687 bytes.
[prelaunch.err](#) : Total file length is 0 bytes.
[prelaunch.out](#) : Total file length is 100 bytes.
[stderr](#) : Total file length is 0 bytes.
[stdout](#) : Total file length is 33259 bytes.

14. El resultado de la clase "SparkPi" se visualiza en la salida del Log.



Logs for container_16038685078

▼ ResourceManager
RM Home
► NodeManager
► Tools

PI is roughly 3.13091827783656

Instalación Anaconda

La distribución de Anaconda es compatible con sistemas operativos como Windows, Linux, y MacOS, y viene conjuntamente con los paquetes de Python, IPython, Jupyter, Numpy, SciPy, Matplotlib y otros paquetes para el análisis de datos (Rossant, 2018).

Para Rolon-Mérette et al. (2020) "Anaconda es un entorno también conocido como IDE, que facilita el desarrollo de código"; sin embargo, Anaconda contiene más funcionalidades como, por ejemplo, de editar, depurar, inspeccionar y visualizar datos de manera que a un usuario se le haga más fácil entender al momento de presentar resultados. También, "Anaconda es una plataforma en la que se pueden configurar entornos virtuales, donde se pueden lanzar servidores web para Jupyter, gestionar paquetes, etc." (Marturet Rodrigo, 2018).

Para la ejecución de códigos en Python, se emplea Jupyter Notebook, de manera que se permita la vinculación y ejecución normal de código escrito en Python de manera sencilla (Toomey, 2016); el uso de esta herramienta es sumamente sencilla, sin embargo, se debe verificar que se encuentren instalados Anaconda, Python y Jupyter.

En Anaconda, el análisis se puede realizar en modo interactivo o en modo de procesamiento por lotes. En ejecución interactiva se puede seguir en tiempo real la evolución del análisis con la información que se presenta por pantalla y, al término del proceso, se presenta un resumen con la información más relevante. En ejecución por lotes se genera un extenso informe en el que además del diagnóstico y de un resumen se recogen una a una las incidencias detectadas (Martín & Sáenz, 2016). Tiene una interfaz de usuario amigable que permite de forma sencilla introducir la información necesaria para la ejecución del análisis (módulos a analizar, ejecución interactiva o por lotes, el tipo de módulos a analizar, etc.). La distribución de código abierto Anaconda, ofrece un set de herramientas para ciencia de datos, que son utilizadas para interactuar con Spark y Jupyter.

Descarga el paquete de anaconda desde el siguiente enlace:

<https://repo.anaconda.com/archive/>



NOTA: Se puede probar utilizando las versiones recientes de Anaconda, pero no las más actualizadas, podrían ser 2 versiones menos de la que se encuentre vigente.

Proceso de descarga e instalación de Anaconda:

Descarga:																																																									
1. Seleccionar la versión de Anaconda y descárgala	<table><tr><td>Anaconda3-2020.02-Windows-x86.exe</td><td>423.2M</td><td>2020-03-11 10:32:58</td><td>64ae8d0e5095b9a878d4522db4ce751e</td></tr><tr><td>Anaconda3-2020.02-Windows-x86_64.exe</td><td>466.3M</td><td>2020-03-11 10:32:35</td><td>6b02c1c91049d29fc65be68f2443079a</td></tr><tr><td>Anaconda3-2020.07-Linux-ppc64le.sh</td><td>290.4M</td><td>2020-07-23 12:16:47</td><td>daf3de1185a390f435ab80b3c2212205</td></tr><tr><td>Anaconda3-2020.07-Linux-x86_64.sh</td><td>550.1M</td><td>2020-07-23 12:16:50</td><td>1046c40a314ab2531e4c099741530ada</td></tr><tr><td>Anaconda3-2020.07-MacOSX-x86_64.pkg</td><td>462.3M</td><td>2020-07-23 12:16:42</td><td>2941ddbaf0c4b49b342c18cde51fee43</td></tr><tr><td>Anaconda3-2020.07-MacOSX-x86_64.sh</td><td>454.1M</td><td>2020-07-23 12:16:44</td><td>50f20c90b8b5bfcd09759c09e32dce68</td></tr><tr><td>Anaconda3-2020.07-Windows-x86.exe</td><td>397.3M</td><td>2020-07-23 12:16:51</td><td>aa7dcf4d02baa25d14baf5728e29d067</td></tr><tr><td>Anaconda3-2020.07-Windows-x86_64.exe</td><td>467.5M</td><td>2020-07-23 12:16:46</td><td>7c718535a7dd89fa46b147626ada9e46</td></tr><tr><td>Anaconda3-2020.11-Linux-ppc64le.sh</td><td>278.9M</td><td>2020-11-18 16:45:36</td><td>bc09710e65cddbba6868061b149281dc</td></tr><tr><td>Anaconda3-2020.11-Linux-x86_64.sh</td><td>528.8M</td><td>2020-11-18 16:45:36</td><td>4cd48ef23a075e855a8b6d0a8c4bae2</td></tr><tr><td>Anaconda3-2020.11-MacOSX-x86_64.pkg</td><td>435.3M</td><td>2020-11-18 16:45:35</td><td>2f96bb47eb5a949da6f99a71d7d66420</td></tr><tr><td>Anaconda3-2020.11-MacOSX-x86_64.sh</td><td>427.8M</td><td>2020-11-18 16:45:35</td><td>918de9a9936908fe62514e0ca6873a21</td></tr><tr><td>Anaconda3-2020.11-Windows-x86.exe</td><td>403.0M</td><td>2020-11-18 16:45:34</td><td>ca1f6f3e75eb346f5ab2d87bab005878</td></tr><tr><td>Anaconda3-2020.11-Windows-x86_64.exe</td><td>457.2M</td><td>2020-11-18 16:45:34</td><td>0841ffcb927a3c2edbd682520f52e546</td></tr></table>	Anaconda3-2020.02-Windows-x86.exe	423.2M	2020-03-11 10:32:58	64ae8d0e5095b9a878d4522db4ce751e	Anaconda3-2020.02-Windows-x86_64.exe	466.3M	2020-03-11 10:32:35	6b02c1c91049d29fc65be68f2443079a	Anaconda3-2020.07-Linux-ppc64le.sh	290.4M	2020-07-23 12:16:47	daf3de1185a390f435ab80b3c2212205	Anaconda3-2020.07-Linux-x86_64.sh	550.1M	2020-07-23 12:16:50	1046c40a314ab2531e4c099741530ada	Anaconda3-2020.07-MacOSX-x86_64.pkg	462.3M	2020-07-23 12:16:42	2941ddbaf0c4b49b342c18cde51fee43	Anaconda3-2020.07-MacOSX-x86_64.sh	454.1M	2020-07-23 12:16:44	50f20c90b8b5bfcd09759c09e32dce68	Anaconda3-2020.07-Windows-x86.exe	397.3M	2020-07-23 12:16:51	aa7dcf4d02baa25d14baf5728e29d067	Anaconda3-2020.07-Windows-x86_64.exe	467.5M	2020-07-23 12:16:46	7c718535a7dd89fa46b147626ada9e46	Anaconda3-2020.11-Linux-ppc64le.sh	278.9M	2020-11-18 16:45:36	bc09710e65cddbba6868061b149281dc	Anaconda3-2020.11-Linux-x86_64.sh	528.8M	2020-11-18 16:45:36	4cd48ef23a075e855a8b6d0a8c4bae2	Anaconda3-2020.11-MacOSX-x86_64.pkg	435.3M	2020-11-18 16:45:35	2f96bb47eb5a949da6f99a71d7d66420	Anaconda3-2020.11-MacOSX-x86_64.sh	427.8M	2020-11-18 16:45:35	918de9a9936908fe62514e0ca6873a21	Anaconda3-2020.11-Windows-x86.exe	403.0M	2020-11-18 16:45:34	ca1f6f3e75eb346f5ab2d87bab005878	Anaconda3-2020.11-Windows-x86_64.exe	457.2M	2020-11-18 16:45:34	0841ffcb927a3c2edbd682520f52e546
Anaconda3-2020.02-Windows-x86.exe	423.2M	2020-03-11 10:32:58	64ae8d0e5095b9a878d4522db4ce751e																																																						
Anaconda3-2020.02-Windows-x86_64.exe	466.3M	2020-03-11 10:32:35	6b02c1c91049d29fc65be68f2443079a																																																						
Anaconda3-2020.07-Linux-ppc64le.sh	290.4M	2020-07-23 12:16:47	daf3de1185a390f435ab80b3c2212205																																																						
Anaconda3-2020.07-Linux-x86_64.sh	550.1M	2020-07-23 12:16:50	1046c40a314ab2531e4c099741530ada																																																						
Anaconda3-2020.07-MacOSX-x86_64.pkg	462.3M	2020-07-23 12:16:42	2941ddbaf0c4b49b342c18cde51fee43																																																						
Anaconda3-2020.07-MacOSX-x86_64.sh	454.1M	2020-07-23 12:16:44	50f20c90b8b5bfcd09759c09e32dce68																																																						
Anaconda3-2020.07-Windows-x86.exe	397.3M	2020-07-23 12:16:51	aa7dcf4d02baa25d14baf5728e29d067																																																						
Anaconda3-2020.07-Windows-x86_64.exe	467.5M	2020-07-23 12:16:46	7c718535a7dd89fa46b147626ada9e46																																																						
Anaconda3-2020.11-Linux-ppc64le.sh	278.9M	2020-11-18 16:45:36	bc09710e65cddbba6868061b149281dc																																																						
Anaconda3-2020.11-Linux-x86_64.sh	528.8M	2020-11-18 16:45:36	4cd48ef23a075e855a8b6d0a8c4bae2																																																						
Anaconda3-2020.11-MacOSX-x86_64.pkg	435.3M	2020-11-18 16:45:35	2f96bb47eb5a949da6f99a71d7d66420																																																						
Anaconda3-2020.11-MacOSX-x86_64.sh	427.8M	2020-11-18 16:45:35	918de9a9936908fe62514e0ca6873a21																																																						
Anaconda3-2020.11-Windows-x86.exe	403.0M	2020-11-18 16:45:34	ca1f6f3e75eb346f5ab2d87bab005878																																																						
Anaconda3-2020.11-Windows-x86_64.exe	457.2M	2020-11-18 16:45:34	0841ffcb927a3c2edbd682520f52e546																																																						
Instalación																																																									
2. Ingresar al directorio /home/hadoop/Descargas, instalar el archivo que se encuentra en la carpeta Descargas con el comando bash																																																									
<pre>cd /home/hadoop/Descargas bash Anaconda3-2020.11-Linux-x86_64.sh</pre>																																																									
Al dar clic en Enter, aparecerá los términos del contrato para lo cual se debe digitalizar la opción "yes".																																																									
Posteriormente se confirma el directorio de instalación y se presiona Enter, y automáticamente el proceso de instalación comenzará.																																																									

```
[no] >>> yes
no change      /home/hadoop/anaconda3/condabin/conda
no change      /home/hadoop/anaconda3/bin/conda
no change      /home/hadoop/anaconda3/bin/conda-env
no change      /home/hadoop/anaconda3/bin/activate
no change      /home/hadoop/anaconda3/bin/deactivate
no change      /home/hadoop/anaconda3/etc/profile.d/conda.sh
no change      /home/hadoop/anaconda3/etc/fish/conf.d/conda.fish
no change      /home/hadoop/anaconda3/shell/condabin/Conda.ps1
no change      /home/hadoop/anaconda3/shell/condabin/conda-hook.ps1
no change      /home/hadoop/anaconda3/lib/python3.8/site-packages/xontrib/conda.xsh
no change      /home/hadoop/anaconda3/etc/profile.d/conda.csh
modified       /home/hadoop/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,
set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Anaconda3!

=====

Working with Python and Jupyter notebooks is a breeze with PyCharm
Professional! Code completion, Notebook debugger, VCS support, SSH, Docker,
Databases, and more!

Get a free trial at: https://www.anaconda.com/pycharm

[hadoop@nodo1 Descargas]$
```

3. Iniciar y ejecutar el siguiente comando:

```
source ~/.bashrc
```

```
Working with Python and Jupyter notebooks is a breeze with PyCharm
Professional! Code completion, Notebook debugger, VCS support, SSH, Docker,
Databases, and more!

Get a free trial at: https://www.anaconda.com/pycharm

[hadoop@nodo1 Descargas]$ source ~/.bashrc
(base) [hadoop@nodo1 Descargas]$
```

Conexión de Jupyter Notebook con PySpark

Jupyter Notebook "...es un cuaderno basado en la web que se utiliza para una programación interactiva con varios lenguajes, como Python, Octave, Julia y R; es muy popular entre las personas que trabajan en el ámbito de la investigación y Ciencia de Datos. Puede contener código, visualizaciones, resultados y texto enriquecido en un único archivo" (Pajankar, 2020).

Según (Vázquez, 2018), para trabajar con Spark desde Jupyter Notebook se debe realizar la siguiente configuración:

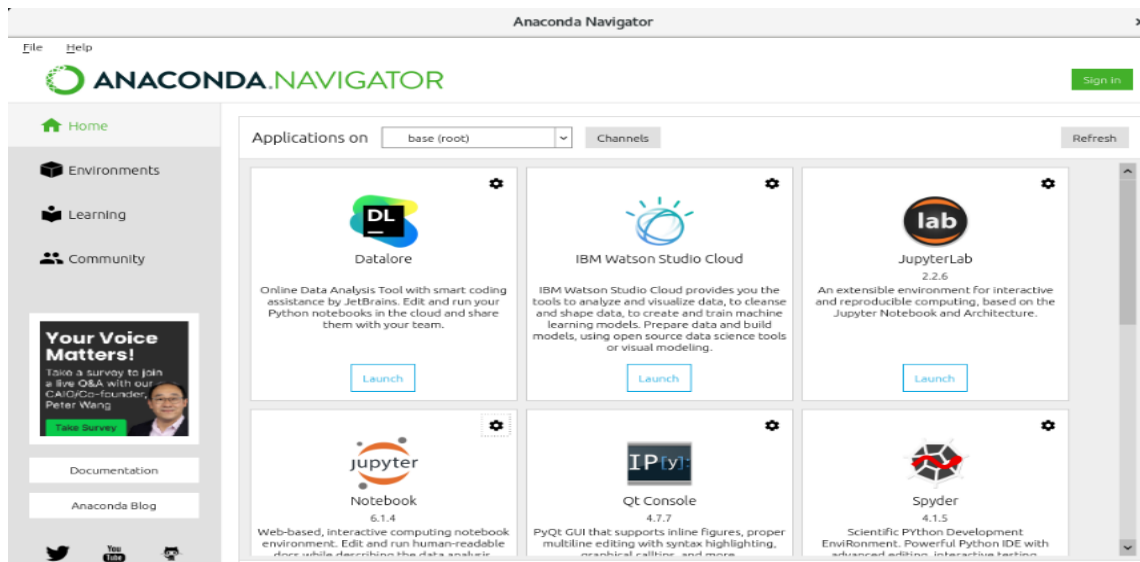
Instalar el paquete **findspark**, el mismo que se encargará de buscar los servicios de Spark de manera interna, para ello se debe ingresar a la Terminal y escribir el siguiente comando:

```
pip install findspark
```

Una vez instalado el paquete findspark, se necesita realizar una conexión o llamada de los servicios de Spark desde Jupyter notebook y así poder trabajar mediante PySpark con toda normalidad.

Para **ejecutar Anaconda**, ingresar el siguiente comando en una Terminal. No cerrar la Terminal, ni manipularla con otros comandos.

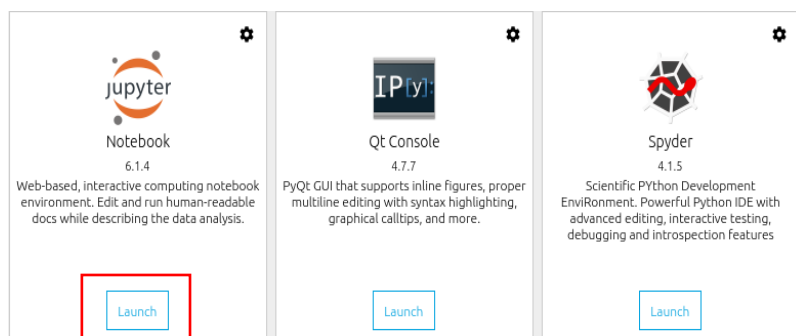
```
anaconda-navigator
```



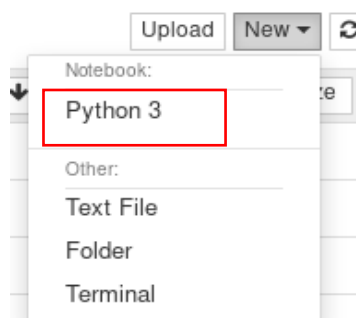
Lanzar una aplicación Python dentro de Jupyter

Proceso para comprobar el funcionamiento de Jupyter:

1. Abrir Jupyter con el botón **launch**.



2. En la interfaz web que se abra, crear un nuevo archivo, eligiendo Python 3



3. Cambiar el nombre del archivo que se generó, por **pruebaPythonSpark.ipynb**



4. En el archivo, escribir los siguientes comandos que importan las librerías. Dar clic en **Run**.

```
import findspark
findspark.init()
import pyspark
from pyspark.sql import SparkSession
sc=pyspark.SparkContext(appName= 'prueba_Spark1',master='yarn')
spark=SparkSession.builder.getOrCreate()
```

5. En el interfaz web de Yarn, verificar que la aplicación de tipo Spark esté corriendo.

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State
application_1603872217249_0001	hadoop	prueba_Spark1	SPARK		default	0	Wed Oct 28 03:07:50 -0500 2020	Wed Oct 28 03:08:02 -0500 2020	N/A	RUNNING

6. Utilizar el siguiente comando para detener el contexto y sesión de Spark

```
sc.stop()
```

7. Observar que la aplicación **prueba_Spark1** ha finalizado en el administrador Web YARN.

Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
prueba_Spark1	SPARK		default	0	Sat Oct 31 01:22:50 -0500 2020	Sat Oct 31 01:22:50 -0500 2020	Sat Oct 31 01:23:20 -0500 2020	FINISHED	SUCCEEDED

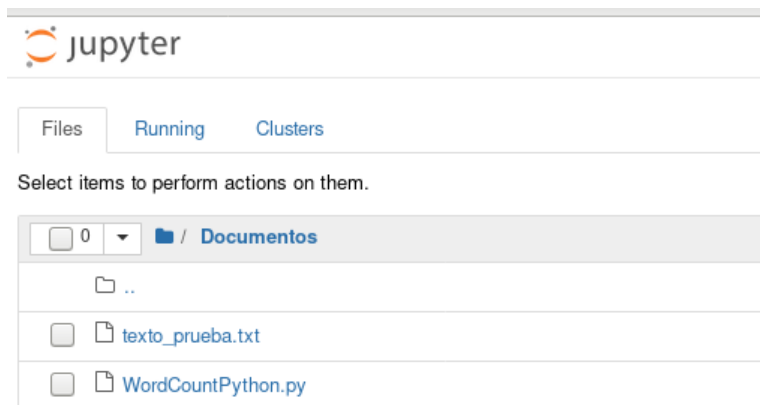
Ejemplo de conteo de palabras - Spark

A continuación se muestra código en Python que permite contar el número de veces que se repite una palabra de un fichero.

Pasos para realizar un ejemplo en Spark:

1. Dentro de Jupyter
ir a la carpeta de
Documentos y
crear un editor de
texto con lenguaje
Python3 y con el
nombre de
archivo:

WordCountPython.py



2. Agregar al archivo el siguiente código Python:

```
import sys

try:
    from pyspark import SparkConf, SparkContext
    # Configuración de Spark
    conf = SparkConf()
    sc = SparkContext(conf=conf)
    # Rutas de entrada y salida
    inputPath = sys.argv[1]
    outputPath = sys.argv[2]
    # Configuración del sistema de archivos
    Path = sc._gateway.jvm.org.apache.hadoop.fs.Path
    FileSystem = sc._gateway.jvm.org.apache.hadoop.fs.FileSystem
    Configuration = sc._gateway.jvm.org.apache.hadoop.conf.Configuration
    fs = FileSystem.get(Configuration())
    # Validación de rutas
    if not fs.exists(Path(inputPath)):
        print("El archivo de ingreso no se encuentra")
    else:
        if fs.exists(Path(outputPath)):
            fs.delete(Path(outputPath), True)
        # Procesamiento con Spark
        sc.textFile(inputPath) \
            .flatMap(lambda l: l.split(" ")) \
            .map(lambda w: (w, 1)) \
            .reduceByKey(lambda t, e: t + e) \
            .saveAsTextFile(outputPath)
        print("Se ha importado los módulos de Spark")
except ImportError as e:
    print("No se ha podido importar los módulos de Spark:", e)
    sys.exit(1)
```


3. Ingresar al directorio donde se guardó el archivo de Python, y ejecutar la siguiente sentencia:

```
cd ~/Documentos/
spark-submit --master yarn --deploy-mode
cluster --name "WordCountPY"
WordCountPython.py /prueba/texto_prueba.txt
/salida_spark_conteo_de_palabras1
```

4. La aplicación se conecta con el **ResourceManager** de Yarn, esta es aceptada y asignada un nodo esclavo para su ejecución.
5. Dentro del Administrador de Yarn, comprobar que se haya lanzado el programa. En nuestro caso la aplicación se ejecutó en el nodo2.

```
2025-01-21 22:21:20,484 INFO yarn.Client: Application report for application_1737506403924_0002/
2025-01-21 22:21:20,484 INFO yarn.Client:
  client token: N/A
  diagnostics: N/A
  ApplicationMaster host: nodo2
  ApplicationMaster RPC port: 42557
  queue: root.default
  start time: 1737516043967
  final status: SUCCEEDED
  tracking URL: http://nodo1:8088/proxy/application_1737506403924_0002/
  user: hadoop
2025-01-21 22:21:20,538 INFO util.ShutdownHookManager: Shutdown hook called
2025-01-21 22:21:20,538 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-4545
2025-01-21 22:21:20,565 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-d553
(base) [hadoop@nodo1 Documentos]$
```

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime
application_1737506403924_0002	hadoop	WordCountPY	SPARK		root.default	0	Tue Jan 21 22:20:43 -0500 2025
application_1737506403924_0001	hadoop	apl1	SPARK		root.default	0	Tue Jan 21 19:42:10 -0500 2025

Comprobar con Administrador HDFS

6. Acceder al administrador de HDFS, y luego al directorio /salida_spark_conteo_de_palabras1

Browse Directory

/salida_spark_conteo_de_palabras1

Go!

Show

25

entries

Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
	-rw-r--r--	hadoop	supergroup	0 B	Jan 21 22:21	3	128 MB	_SUCCESS	
	-rw-r--r--	hadoop	supergroup	40 B	Jan 21 22:21	3	128 MB	part-00000	
	-rw-r--r--	hadoop	supergroup	24 B	Jan 21 22:21	3	128 MB	part-00001	

Showing 1 to 3 of 3 entries

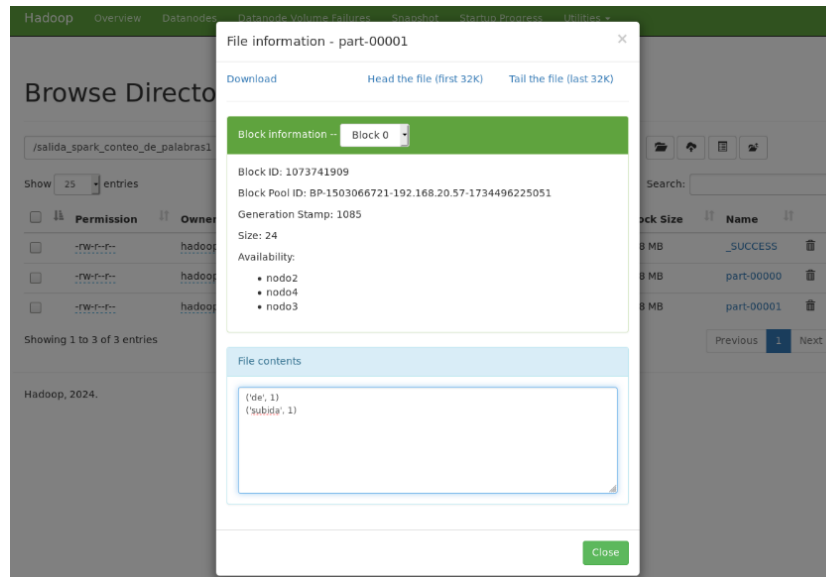
Previous

1

Next

Los archivos part-00000 y part-00001 contienen el resultado de la aplicación.

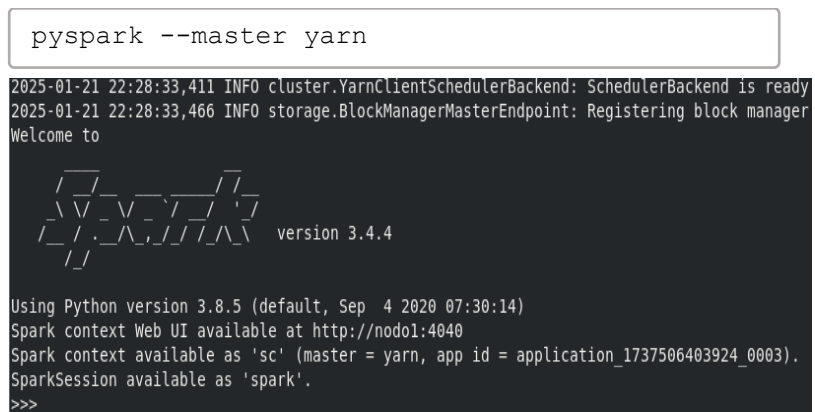
- Ingresar al archivo `part-00001` y hacer clic en **Tail the file (last 32K)** para verificar el resultado.



Resultado de la cantidad de veces que se repite una palabra.

Abrir sesión PySpark con Python:

1. Abrir una Terminal y ejecutar el siguiente comando para crear la sesión.



2. Dentro del Administrador de Yarn, comprobar que se haya lanzado la sesión.

ID	User	Name	Application Type
<u>application_1737506403924_0003</u>	hadoop	PySparkShell	SPARK
<u>application_1737506403924_0002</u>	hadoop	WordCountPY	SPARK
<u>application_1737506403924_0001</u>	hadoop	apli1	SPARK

3. Terminar la sesión.

```
quit()
```

4. Dentro del Administrador de Yarn, verificar que la sesión se haya finalizado y el nodo donde se realizó exitosamente.

User:	hadoop
Name:	PySparkShell
Application Type:	SPARK
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	root.default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Jan 21 22:28:20 -0500 2025
Launched:	Tue Jan 21 22:28:20 -0500 2025
Finished:	Tue Jan 21 22:44:49 -0500 2025
Elapsed:	16mins, 29sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT PARTITION>

Show 20 entries			
Attempt ID	Started	Node	Logs
appattempt_1737506403924_0003_000001	Tue Jan 21 22:28:20 -0500 2025	http://nodo3:8042	Logs

Showing 1 to 1 of 1 entries

Resumen del Capítulo. En este capítulo se detalló el proceso de instalación y configuración de un clúster distribuido con Apache Hadoop en CentOS 8. Se implementaron las herramientas HDFS, MapReduce, YARN y Spark, para la gestión y procesamiento de grandes volúmenes de datos de manera eficiente y escalable. Además, de estas herramientas se integraron con Anaconda y el uso de Jupyter Notebook como plataformas complementarias para el análisis y visualización de datos. Este enfoque práctico permite sentar las bases para una infraestructura robusta de Big Data, en este caso fue el levantamiento con 4 nodos (master: nodo1, esclavos: nodos 2, 3 y 4), para aprovechar las capacidades de procesamiento distribuido de Apache Hadoop; aplicable a las necesidades de cualquier organización o empresa.

Autoevaluación

Responder las siguientes preguntas para evaluar la comprensión de conocimientos del capítulo

- ¿Cuál es el objetivo principal de la instalación distribuida de Hadoop en CentOS 8?
 - Reducir el consumo de memoria en los nodos.
 - Implementar un clúster escalable de Hadoop para procesamiento distribuido.
 - Mejorar la seguridad del sistema operativo CentOS 8.
 - Reemplazar completamente los servidores físicos por máquinas virtuales.
- ¿Qué componente de Hadoop se encarga de la gestión de almacenamiento en el clúster?
 - YARN
 - Spark
 - HDFS

- d. MapReduce
- 3. ¿Cuáles son los requisitos mínimos recomendados para la instalación del clúster?
 - a. 2GB de RAM y 50 GB de almacenamiento.
 - b. 6 GB de RAM y 100 GB de almacenamiento.
 - c. 16 GB de RAM y 250 GB de almacenamiento.
 - d. 4 GB de RAM y 500 GB de almacenamiento.
- 4. ¿Cuál de los siguientes archivos de configuración se usa para definir la replicación y almacenamiento en Hadoop?
 - a. core-site.xml
 - b. hdfs-site.xml
 - c. yarn-site.xml
 - d. mapred-site.xml
- 5. ¿Qué protocolo es necesario para la comunicación entre los nodos del clúster?
 - a. HTTP
 - b. FTP
 - c. SSH
 - d. SMTP
- 6. ¿Qué herramienta sirve para crear scripts y realizar análisis de datos almacenados en Hadoop HDFS?
 - a. PostgreSQL
 - b. Jupyter Notebook
 - c. MySQL
 - d. Excel
- 7. ¿Qué servicio de Hadoop permite la gestión de recursos y ejecución de tareas en el clúster?
 - a. HDFS
 - b. MapReduce
 - c. YARN
 - d. Spark
- 8. ¿Cuál es el comando para iniciar los servicios de HDFS?
 - a. start-yarn.sh
 - b. start-hdfs.sh
 - c. start-dfs.sh

- d. `hadoop namenode -format`
- 9. ¿Qué herramienta se utiliza para la instalación y gestión de paquetes en Python dentro del clúster?
 - a. VirtualBox
 - b. Anaconda
 - c. Docker
 - d. Kubernetes
- 10. ¿En qué archivo se configura el hostname del nodo maestro en Hadoop?
 - a. `/etc/hostname`
 - b. `/opt/hadoop/etc/hadoop/yarn-site.xml`
 - c. `/opt/hadoop/etc/hadoop/core-site.xml`
 - d. `/opt/hadoop/etc/hadoop/mapred-site.xml`

Respuestas correctas

1. B, 2. C, 3. B, 4. B, 5. C, 6. B, 7. C, 8. C, 9. B, 10. C

Referencias Bibliográficas

- Alapati, S. R. (2016). *Expert Hadoop Administration Managing, Tuning, and Securing Spark, YARN, and HDFS*. Addison-Wesley.
- Alonso, V. (2023, mayo 1). CentOS Linux | Características, ventajas y desventajas. *sistemasoperativos.info*.
<https://sistemasoperativos.info/linux/centos/>
- Coulouris, G., Dollimore, J., Kindberg, T., & Blair, G. (2011). *Distributed Systems: Concepts and Design*.
- Deshpande, & Tanmay. (2016). *Hadoop real-work solutions cookbook: Over 90 hands-on recipes to help you learn and master the intricacies of Apache Hadoop 2.X, YARN, Hive, Pig, Oozie, Flume, Sqoop, Apache Spark, and Mahout* (Second edition). Packt Publishing.
- Introduction to VMware Workstation | SpringerLink*. (2024, junio 21).
https://link.springer.com/chapter/10.1007/979-8-8688-0146-4_4
- Jeyaraj, R., Pugalendhi, G., & Paul, A. (2020). *Big Data with Hadoop MapReduce: A Classroom Approach* (1.^a ed.). Apple Academic Press.
<http://gen.lib.rus.ec/book/index.php?md5=A94A8A1623EA8B8AEFB72B90FB229F9D>

- Krill, P. (2022, marzo 4). *Java 8 sigue dominando, pero se acerca la ola de Java 17* | CIO.
<https://www.cio.com/article/2069768/java-8-sigue-dominando-pero-se-acerca-la-ola-de-java-17.html>
- Martín, E., & Sáenz, F. (2016). *Sistema ANACONDA para el análisis automático de la calidad del software*.
- Ortega Candel, J. M. (2022). *Big data, machine learning y data science en Python*. RA-MA Editorial.
<https://0310w0f5l-y-https-elibro-net.basesdedatos.utmachala.edu.ec/es/ereader/utmachala/230290>
- Pajankar, A. (2020). Exploring Jupyter Notebook. *Practical Python Data Visualization*, 17.
- Qué es el protocolo SSH y cómo configurarlo para mejorar la seguridad de acceso a los servidores Linux* | Hostalia - Pressroom. (2013, 05). Hostalía.
<https://pressroom.hostalia.com/white-papers/configurar-protocolo-ssh/>
- Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2020). Introduction to Anaconda and Python: Installation and setup. *The Quantitative Methods for Psychology*, 16(5), S3-S11.
<https://doi.org/10.20982/tqmp.16.5.S003>
- Rossant, C. (2018). *IPython Cookbook—Chapter 3: Mastering the Jupyter Notebook*. <https://ipython-books.github.io/chapter-3-mastering-the-jupyter-notebook/>
- Vázquez, F. (2018, abril 19). *How to use PySpark on your computer*. Medium.
<https://towardsdatascience.com/how-to-use-pyspark-on-your-computer-9c7180075617>

Capítulo IV

Procesamiento de datos distribuidos con MapReduce

- Introducción a MapReduce
- Arquitectura y Funcionamiento de MapReduce
- Ejercicios prácticos

Objetivos:

- Entender los fundamentos y funcionamiento de MapReduce.
- Implementar scripts MapReduce para procesamiento de datos distribuidos en HDFS.
- Desarrollar ejercicios prácticos con MapReduce relacionados con análisis de datos.
- Ejecutar y verificar programas MapReduce usando Hadoop Streaming.

MapReduce es el marco principal para el procesamiento de datos masivos dentro de ecosistemas de Big Data con Apache Hadoop. La estructura de su funcionamiento es elemental, desde la fase de mapeo, donde se dividen los datos en pequeños bloques procesables, hasta la fase de reducción, en la que los resultados se consolidan para obtener insights significativos.

A lo largo de este capítulo, no solo se revisará los fundamentos teóricos de MapReduce, sino también, se resolverán ejercicios prácticos que demuestran cómo este marco puede aplicarse a problemas reales de análisis y procesamiento de datos. Los ejercicios permitirán a los lectores entender cómo MapReduce gestiona tareas de procesamiento de datos de manera eficiente, distribuyendo las cargas de trabajo entre múltiples nodos del clúster y garantizando tolerancia a fallos. La flexibilidad de MapReduce, lo convierte en una herramienta versátil capaz de abordar una amplia gama de casos de uso, desde operaciones básicas hasta análisis más complejos sobre grandes volúmenes de datos.

La Figura 23, resume los aspectos fundamentales de Hadoop MapReduce, incluyendo su funcionamiento básico, aplicaciones en Big Data, así como la realización de ejercicios prácticos y verificación de resultados mediante HDFS y YARN.

“MapReduce es un modelo de programación que simplifica la complejidad del procesamiento de datos a gran escala dividiendo el trabajo en trozos manejables y procesándolos en paralelo”

(Hadoop, 2015).

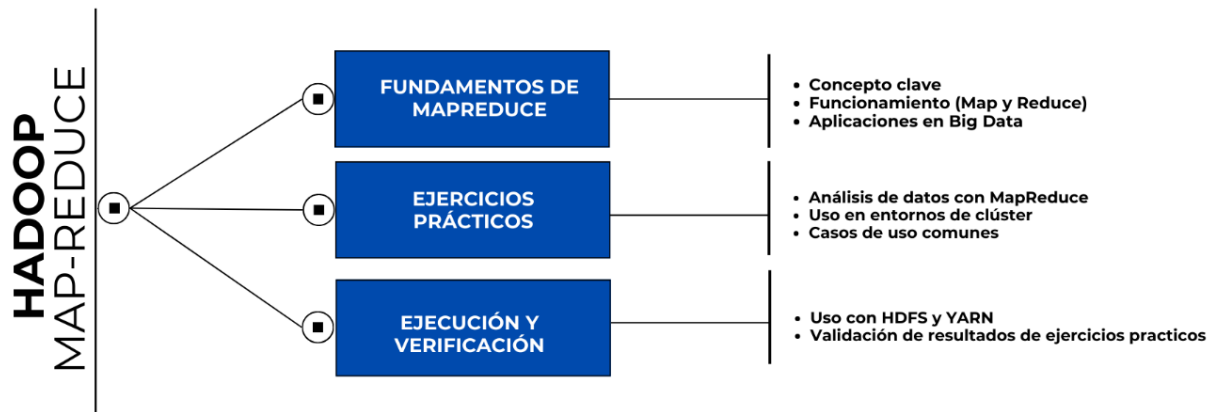


Figura 23. Resumen Capítulo IV
Fuente: Elaboración propia

Fundamentos de MapReduce

¿Qué es MapReduce?

MapReduce es un modelo de programación creado por Google para procesar grandes cantidades de datos de forma paralela en clústeres de computadoras. Existen varias implementaciones de MapReduce, escritas en Java (las más utilizadas), C++, Python y otros lenguajes (Martínez et al., 2013). Se caracteriza por su procesamiento por lotes de datos en paralelo que se encuentran distribuidos en diferentes nodos de Hadoop HDFS.

Se destaca por ser eficiente, distribuido y capaz de ofrecer procesamiento con tolerancia a fallos, que opera sobre el sistema de archivos HDFS. Está diseñado para aplicaciones que manejan grandes volúmenes de datos, proporcionando un rendimiento elevado en el acceso de estos. Este framework se encarga automáticamente de la distribución y paralelización de las tareas, facilitando el trabajo del programador al hacer este proceso transparente (Basgall et al., 2016).

La programación de MapReduce aplica una misma función a un conjunto de elementos o datos de una lista de entrada y genera una lista de datos de salida. En el trabajo de (Gimenez-Garcia et al., 2016), se señala que el objetivo principal de MapReduce es abstraer la complejidad subyacente a estos

entornos y ofrecer un mecanismo eficiente que permita paralelizar el procesamiento de datos de cualquier naturaleza.

Martínez et al. (2013), nos propone una lista de características de la programación MapReduce:

- **Programación funcional:** Funciones map y reduce que producen, ante una lista de datos de entrada, un conjunto de elementos de salida.
- **Orientado a grandes cantidades de datos que se procesan por lotes:** Estos datos se almacenan en bloques entre los distintos nodos del sistema y son controlados por el nodo de datos (DataNode).
- **Sistema de Ficheros fiable y tolerante a fallos:** Replicación de metadatos del nodo de control.
- **Portabilidad del sistema en nodos con hardware y software heterogéneos:** Por tanto, sistema escalable que permite añadir nuevos nodos.
- **Sistema distribuido que lleva a cabo la programación paralela:** Realiza balanceado de la carga de trabajo.
- **Control de los nodos que fallan:** Tolerancia a fallos del sistema.
- Puede implantarse un sistema a **coste cero** ya que todo el software puede encontrarse en licencia libre. Además, el que sea libre aporta más ventajas como personalización del software, adaptabilidad, continua actualización, soporte en comunidades, etc.
- **Alta compatibilidad** ya que puede instalarse en sistemas operativos GNU/Linux y Windows, de forma virtualizada o sobre servidores físicos, con gran variedad de sistemas de ficheros distribuidos.

Arquitectura funcional de MapReduce

MapReduce transforma el enfoque tradicional del procesamiento de datos al dividir automáticamente las tareas en operaciones más pequeñas que pueden ejecutarse en paralelo. Esta arquitectura está compuesta por funciones específicas que coordinan el análisis y la consolidación de los datos a gran escala.

En MapReduce, las funciones principales son:

- 1. Mapper:** Toma los datos de entrada y los convierte en pares clave-valor. Esta función divide los datos grandes en fragmentos más pequeños, procesándolos de forma paralela en diferentes nodos del clúster.
- 2. Combiner:** Es una función opcional que actúa como un "minireducer" local. Se ejecuta después del Mapper para agregar los resultados parciales,

reduciendo la cantidad de datos que deben enviarse al Reducer, optimizando el rendimiento.

- 3. Reducer:** Recibe los pares clave-valor generados por el Mapper y/o Combiner, y realiza la agregación final para producir el resultado deseado, consolidando y procesando los datos de forma eficiente.

Este flujo de Mapper, Combiner, y Reducer permite el procesamiento paralelo de grandes lotes (batch) de datos en clústeres distribuidos (Figura 24).

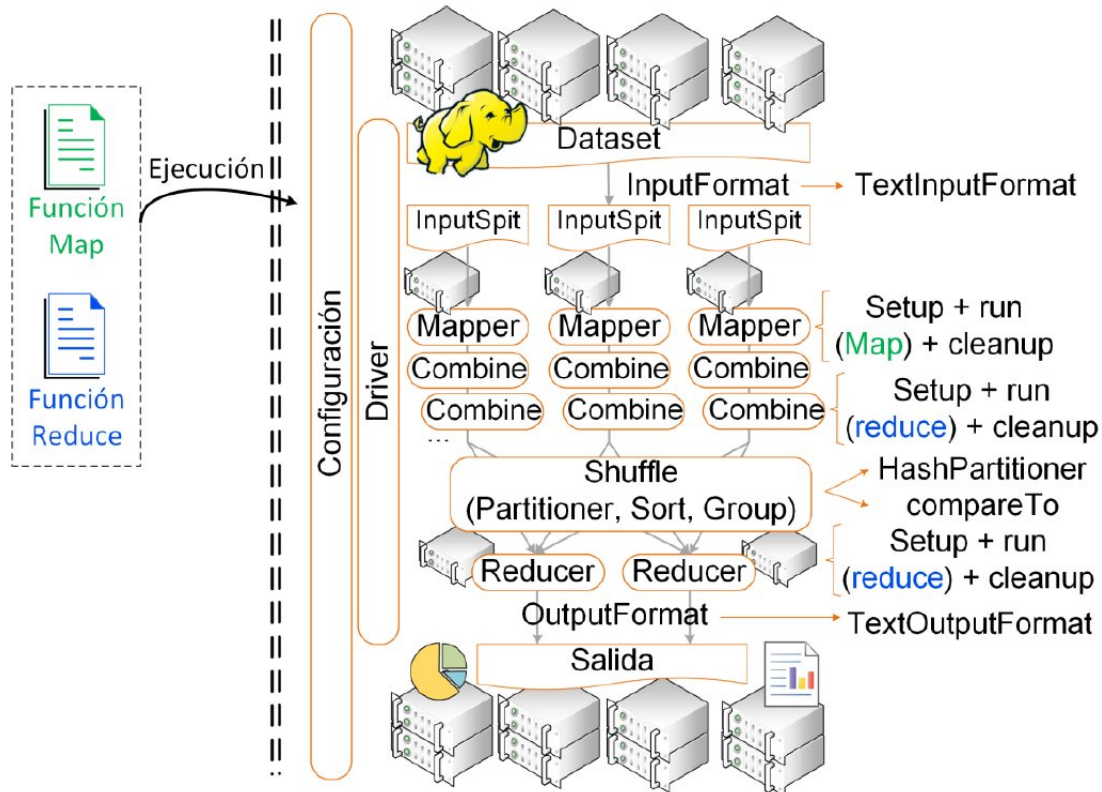


Figura 24. Estructura MapReduce

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Funcionamiento de MapReduce

La fase inicial de MapReduce es Map, que es el encargado de generar el clave valor (dividir el problema en subproblemas), luego viene Reduce, que se basa en la combinación de valores con la misma clave (resolver los subproblemas). Shuffle and Sort tiene como principal propósito agrupar, ordenar y repartir los datos por clave entre los reduce (Figura 25). Según Urrutia & Rivera-Pleitez (2016) la forma en la que los datos se distribuyen en diferentes subtareas y cómo estas se asignan a cada máquina resulta transparente para el desarrollador.

El modelo de programación está basado en la computación de los siguientes conceptos:

1. **Iteración** sobre los datos de entradas.
2. **Computación de los pares** clave-valor para cada dato de entrada.
3. **Iteración** sobre los grupos resultantes.
4. **Reducción** de cada grupo.

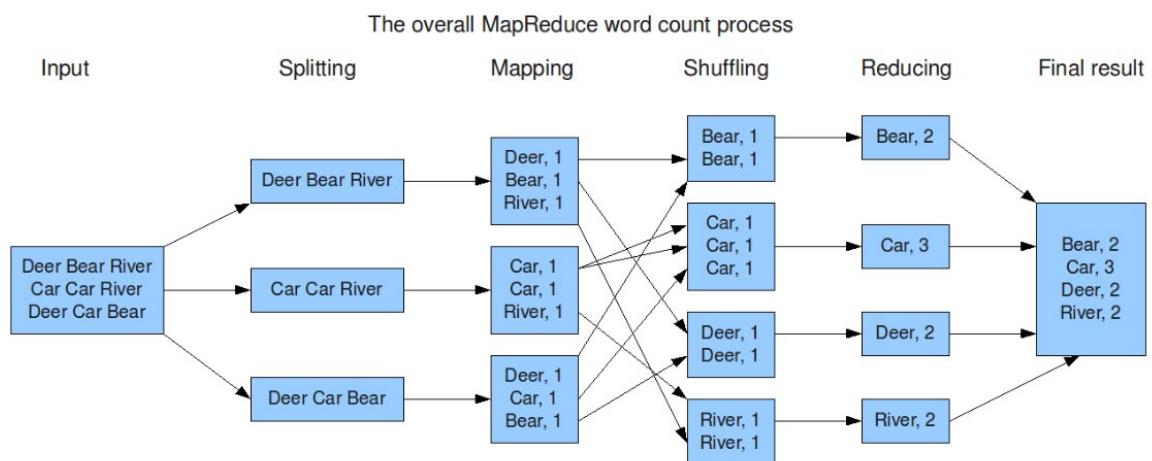


Figura 25. Algoritmo MapReduce

Fuente: (Lăpușan, 2022)

Integración de MapReduce en HDFS (Sistemas de Almacenamiento Distribuido de Hadoop).

MapReduce ha sido una herramienta clave en el procesamiento de grandes conjuntos de datos, ya que permite escribir programas que pueden ejecutarse en paralelo en un clúster de cientos o incluso miles de nodos. Ofrece una escalabilidad prácticamente lineal, permitiendo añadir más nodos informáticos a medida que crecen los conjuntos de datos (Alapati, 2017).

En contraste, es fácil desarrollar algoritmos escalables con MapReduce sin esfuerzo adicional para gestionar un sistema distribuido. Un programa desarrollado para un solo nodo puede utilizarse en miles de nodos sin necesidad de modificar el código. MapReduce paraleliza la ejecución automáticamente, por lo que los usuarios no necesitan invertir esfuerzo en la ejecución paralela (Jeyaraj et al., 2021).

MapReduce es un framework de aplicaciones que se ejecuta sobre YARN junto con otros frameworks distribuidos. YARN se abstrae sobre MapReduce para proporcionar gestión de recursos y, al mismo tiempo, admitir otros procesamiento distribuidos como Spark, Tez y Giraph (Kumar, 2021b). Es decir que YARN hace que Hadoop sea más flexible y eficiente al permitir la ejecución de varios motores de procesamiento en un mismo clúster

Así mismo, su modelo escalable lo convierte en una solución ideal para adaptarse a las crecientes demandas de procesamiento en ecosistemas de Big Data, ofreciendo un balance entre eficiencia, fiabilidad y flexibilidad.

Roles de los nodos en Hadoop HDFS y MapReduce

Los servicios HDFS y MapReduce se integran en el clúster Hadoop, donde se identifican los siguientes roles: nodos maestros, nodos esclavos y máquina cliente (Figura 26) (Hedlund, 2011). A continuación, se describen los roles:

- **Los Nodos maestros:**
 - **El Name Node.** Supervisa y coordina el almacenamiento distribuido de datos (HDFS); también gestiona la metadata de archivos y bloques.
 - Realizan el procesamiento de cálculos en paralelo con los datos distribuidos (Map Reduce).
 - **Job Tracker** (rastreador de tareas) de MapReduce supervisa y coordina el procesamiento en paralelo de lotes de datos distribuidos en HDFS.
- **Los nodos esclavos.** La mayoría de estas máquinas y hacen el trabajo sucio de almacenar los datos (**Data Node**) y ejecutar los cálculos (**Task Tracker**).
- **La máquina cliente.** Es la que tiene Hadoop instalado con todas las configuraciones del clúster, pero no es ni maestra ni esclava. Su función es cargar datos en el clúster, enviar trabajos de Map Reduce y luego recuperar o ver los resultados del trabajo cuando finalice.

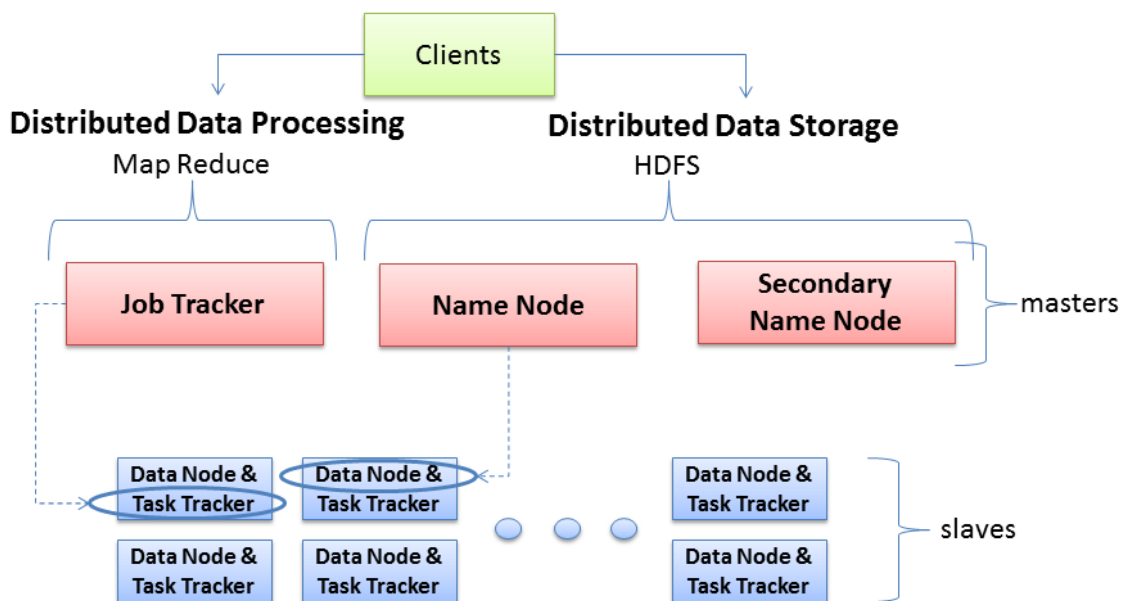


Figura 26. Roles de Hadoop HDFS y MapReduce

Fuente: (Hedlund, 2011)

Levantamiento de Servicios

Antes de realizar todos los ejercicios propuestos en este capítulo, es importante tener en cuenta qué, debe iniciar los servicios.

levantar todos los servicios y comprobar los procesos con **jps**.

```
start-all.sh
jps
```

También podemos levantar los servicios individualmente

```
start-dfs.sh
start-yarn.sh
mapred --daemon start historyserver
```

Ejercicios prácticos utilizando MapReduce

Ejercicio 1. Cálculo de ventas máximas por mes

Dado un conjunto de registros de ventas por mes, se requiere identificar el valor máximo de ventas registrado en cada mes.

Pasos para calcular las ventas máximas por mes. Utilizar una Terminal:

1. Crear el archivo de entrada con los datos.


```
%%bash
{
echo -e "Enero\t1000"
echo -e "Febrero\t1200"
echo -e "Enero\t1500"
echo -e "Febrero\t1300"
echo -e "Marzo\t1400"
} > ./ventas.txt
```

2. Crear el script Python del Mapper.

```
nano mapperVentas.py
```

```
#!/usr/bin/python3
import sys
# Mapper para calcular las ventas máximas
for linea in sys.stdin:
    linea = linea.strip()
    mes, ventas = linea.split("\t", 1)
    print(f"{mes}\t{ventas}")
```

3. Crear el script Python del Reducer.

```
nano reducerVentas.py
```

```
#!/usr/bin/python3
import sys
# Reducer para calcular las ventas máximas
subproblema = None
ventasMax = None

for claveValor in sys.stdin:
    mes, ventas = claveValor.split("\t", 1)
    ventas = float(ventas)

    if subproblema == None:
        subproblema = mes
        ventasMax = ventas

    if subproblema == mes:
        if ventas > ventasMax:
            ventasMax = ventas
    else:
        print(f"{subproblema}\t{ventasMax}")
        subproblema = mes
        ventasMax = ventas
print(f"{subproblema}\t{ventasMax}")
```

4. Otorgar los permisos de ejecución a los archivos creados.

```
chmod u+x mapperVentas.py
chmod u+x reducerVentas.py
```

5. Copiar el archivo local a HDFS

Verificar el archivo local:

```
ls /home/hadoop/ventas.txt
```

Verificar los permisos del archivo local:

```
ls -l /home/hadoop/ventas.txt
```

Crear el directorio en HDFS (si no existe):

```
hdfs dfs -mkdir -p /user/hadoop
```

Copiar o mover el archivo a HDFS:

```
hdfs dfs -put /home/hadoop/ventas.txt /user/hadoop/
```

Verificar que el archivo se haya movido correctamente:

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar los scripts Map y Reduce junto con el archivo de datos de entrada y carpeta de salida.
- ```
hadoop jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar -file
/home/hadoop/mapperVentas.py -mapper
/home/hadoop/mapperVentas.py -file
/home/hadoop/reducerVentas.py -reducer
/home/hadoop/reducerVentas.py -input
/user/hadoop/ventas.txt -output salidaVentas
```

7. Verificar en consola y en la interfaz de salida de HDFS.

```
hdfs dfs -cat /user/hadoop/salidaVentas/*
```

```
(base) [hadoop@nodo1 ~]$ hdfs dfs -cat /user/hadoop/salidaVentas/*
Enero 1500.0
Febrero 1300.0
Marzo 1400.0
(base) [hadoop@nodo1 ~]$
```

```
Block ID: 1073741992
Block Pool ID: BP-1503066721-192.168.20.57-1734496225051
Generation Stamp: 1168
Size: 41
Availability:
 • nodo4
 • nodo2
 • nodo3
```

#### File contents

```
Enero 1500.0
Febrero 1300.0
Marzo 1400.0
```

Este ejercicio de MapReduce muestra cómo procesar datos distribuidos para calcular las ventas máximas por mes. El Mapper emite pares clave valor < mes, ventas> cada venta con su mes correspondiente. Luego el Reducer identifica el valor máximo por mes, aprovechando la ordenación previa de Hadoop.

## Ejercicio 2. Cálculo de temperatura máxima por año

A partir de registros de temperatura clasificados por año y mes, obtener la temperatura más alta registrada cada año.

Pasos para calcular la temperatura máxima por año. Utilizar una Terminal:

1. Crear el archivo de entrada con los datos.

```
%%bash
{
echo -e "1999\tEnero\t1"
echo -e "1999\tEnero\t5"
echo -e "1999\tEnero\t3"
echo -e "1999\tEnero\t2"
echo -e "1999\tFebrero\t4"
echo -e "1999\tFebrero\t2"
echo -e "2000\tEnero\t3"
echo -e "2000\tEnero\t6"
echo -e "2000\tFebrero\t6"
echo -e "2000\tFebrero\t2"
echo -e "2001\tAbril\t3"
} > ./medidas.txt
```

2. Crea el archivo Mapper.

```
nano mapperMaxTemp.py
```

```
#!/usr/bin/python3
import sys
'''
Mapper de maxTemp
'''

#Por cada medida de temp emitimos los pares <anyo,
temp>
for linea in sys.stdin:
 linea = linea.strip()
 anyo , mes, temp = linea.split("\t", 2)
 print("%s\t%s" % (anyo, temp))
```

3. Crear el archivo Reducer.

```
nano reducerMaxTemp.py
```

```
#!/usr/bin/python3
import sys
'''
Reducer de MaxTemp
'''

subproblema = None
tempMaxima = None
for claveValor in sys.stdin:
 anyo, temp = claveValor.split("\t", 1)
 #convertir la temp a float
 temp = float(temp)
 #El primer subproblema de reducer es el primer anio, temp max
 if subproblema == None:
 subproblema = anyo
 tempMaxima = temp
 #si anio es del subproblema actual, comprobar si es la temp max
 if subproblema == anyo:
 if temp > tempMaxima:
 tempMaxima = temp
 else: #si ya acabamos con el subproblema, emitimos
```



```

print("%s\t%s" % (subproblema, tempMaxima))
#Pasar al siguiente subproblema (de momento la temp es la maxima)
subproblema = anyo
tempMaxima = temp
#el anterior bucle no emite el último subproblema
print("%s\t%s" % (subproblema, tempMaxima))

```

- Otorgar los permisos de ejecución a los archivos creados.

```

chmod u+x mapperMaxTemp.py
chmod u+x reducerMaxTemp.py

```

- Copiar el archivo local a HDFS

#### Verificar el archivo local

```
ls /home/hadoop/medidas.txt
```

#### Verificar los permisos del archivo local:

```
ls -l /home/hadoop/medidas.txt
```

#### Crear el directorio en HDFS (si no existe)

```
hdfs dfs -mkdir -p /user/hadoop
```

#### Copiar o mover un archivo a HDFS

```
hdfs dfs -put /home/hadoop/medidas.txt /user/hadoop/
```

#### Verificar que el archivo se haya movido correctamente

```
hdfs dfs -ls /user/hadoop/
```

- Ejecutar los scripts Map y Reduce junto con el archivo de datos de entrada y carpeta de salida

```

hadoop jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-
streaming-3.4.0.jar \
-input /user/hadoop/medidas.txt \
-output salidaMaxTemp \
-mapper /home/hadoop/mapperMaxTemp.py \
-reducer /home/hadoop/reducerMaxTemp.py

```

- Comprobar la carpeta de salida

```
hdfs dfs -cat /user/hadoop/salidaMaxTemp/*
```

```

(base) [hadoop@nodol ~]$ hdfs dfs -cat /user/hadoop/salidaMaxTemp/*
1999 5.0
2000 6.0
2001 3.0
(base) [hadoop@nodol ~]$

```

Block ID: 1073741888

Block Pool ID: BP-1503066721-192.168.20.57-1734496225051

Generation Stamp: 1064

Size: 27

Availability:

- nodo4
- nodo3
- nodo2

File contents

|      |     |
|------|-----|
| 1999 | 5.0 |
| 2000 | 6.0 |
| 2001 | 3.0 |

Este ejercicio de MapReduce ilustra cómo procesar datos distribuidos para calcular la temperatura máxima de cada año. El Mapper emite pares clave valor < año, temp> y el Reducer encuentra el valor máximo por año, aprovechando la ordenación automática de Hadoop.

### Ejercicio 3. Conteo de palabras de un archivo de texto

Pasos para dado un archivo de texto, contar la frecuencia de cada palabra:

1. Crear el archivo entrada con los datos.
 

```
%%bash
{
echo "hola mundo hola universo"
echo "mundo universo hola"
echo "universo universo universo"
} > ./palabras.txt
```

2. Crear el archivo Mapper.

```
nano mapperwordcount.py

#!/usr/bin/python3
import sys

Emitir cada palabra con el conteo de 1
for linea in sys.stdin:
 linea = linea.strip()
 palabras = linea.split()
 for palabra in palabras:
 print(f"{palabra}\t1")
```

3. Crear el archivo Reducer.

```
nano reducerWordCount.py
```

```
#!/usr/bin/python3
import sys

palabra_actual = None
conteo_actual = 0

for linea in sys.stdin:
 palabra, conteo = linea.split("\t")
 conteo = int(conteo)
 if palabra == palabra_actual:
 conteo_actual += conteo
 else:
 if palabra_actual:
 print(f"{palabra_actual}\t{conteo_actual}")
 palabra_actual = palabra
 conteo_actual = conteo
 if palabra_actual:
 print(f"{palabra_actual}\t{conteo_actual}")
```

4. Otorgar los permisos de ejecución a los archivos creados.

```
chmod u+x mapperwordcount.py
chmod u+x reducerWordCount.py
```

5. Copiar el archivo local a HDFS

#### Verificar el archivo local

```
ls /home/hadoop/palabras.txt
```

#### Verificar los permisos del archivo local:

```
ls -l /home/hadoop/palabras.txt
```

#### Crear el directorio en HDFS (si no existe)

```
hdfs dfs -mkdir -p /user/hadoop
```

#### Copiar o mover un archivo a HDFS

```
hdfs dfs -put /home/hadoop/palabras.txt
/user/hadoop/
```

#### Verificar que el archivo se haya movido correctamente

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar los scripts Map y Reduce junto con el archivo de datos de entrada y carpeta de salida
- ```
hadoop jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
-file /home/hadoop/mapperwordcount.py \
-mapper /home/hadoop/mapperwordcount.py \
-file /home/hadoop/reducerWordCount.py \
-reducer /home/hadoop/reducerWordCount.py \
-input /user/hadoop/palabras.txt \
-output salidaWordCount
```
7. Comprobar la salida.
- ```
hdfs dfs -cat /user/hadoop/salidaWordCount/*
```

```
2025-01-27 19:11:52,182 INFO streaming.StreamJob: Output directory: salidaWordCount
(base) [hadoop@nodo1 ~]$ hdfs dfs -cat /user/hadoop/salidaWordCount/*
hola 3
mundo 2
universo 5
(base) [hadoop@nodo1 ~]$
```

```
Block ID: 1073742005
Block Pool ID: BP-1503066721-192.168.20.57-1734496225051
Generation Stamp: 1181
Size: 26
Availability:
 • nodo2
 • nodo3
 • nodo4
```

#### File contents

```
hola 3
mundo 2
universo 5
```

Este ejercicio de MapReduce cuenta las ocurrencias de cada palabra en un archivo de texto. El Mapper emite pares clave valor, cada palabra con un conteo de 1 <palabra, 1>, y el Reducer suma las ocurrencias por palabra. Gracias a la ordenación automática de Hadoop, las palabras se agrupan eficientemente.

### Ejercicio 4. Productos más vendidos

Dada una lista de productos con su cantidad vendida, calcular el total de ventas por producto.

Pasos para resolver el ejercicio:

1. Crear la variable de entrada con los datos.

```
%%bash
{
echo -e "Producto1\t20"
echo -e "Producto2\t30"
echo -e "Producto1\t15"
echo -e "Producto3\t50"
echo -e "Producto2\t40"
} > ./productos.txt
```

2. Crear el archivo Mapper.

```
nano mapperProductos.py
```

```
#!/usr/bin/env python3
import sys

Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 producto, cantidad = line.split("\t")
 print(f"{producto}\t{cantidad}")
```

3. Crear el archivo Reducer.

```
nano reducerProductos.py
```

```
#!/usr/bin/env python3
import sys
current_product = None
current_total = 0
Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 producto, cantidad = line.split("\t")
 cantidad = int(cantidad)

 if current_product == producto:
 current_total += cantidad
 else:
 if current_product:
 print(f"{current_product}\t{current_total}")
 current_product = producto
 current_total = cantidad

if current_product:
 print(f"{current_product}\t{current_total}")
```

4. Otorgar los permisos de ejecución a los archivos creados.

```
chmod u+x mapperProductos.py
chmod u+x reducerProductos.py
```

5. Copiar el archivo local a HDFS

**Verificar el archivo local**

```
ls /home/hadoop/productos.txt
```

**Verificar los permisos del archivo local:**

```
ls -l /home/hadoop/productos.txt
```

**Crear el directorio en HDFS (si no existe)**

```
hdfs dfs -mkdir -p /user/hadoop
```

**Copiar o mover un archivo a HDFS**

```
hdfs dfs -put /home/hadoop/productos.txt /user/hadoop/
```

**Verificar que el archivo se haya movido correctamente**

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar los scripts Map y Reduce junto con el archivo de datos de entrada y carpeta de salida

```
hadoop jar
$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-
3.4.0.jar \
-file mapperProductos.py -mapper mapperProductos.py \
-file reducerProductos.py -reducer
reducerProductos.py \
-input /user/hadoop/productos.txt -output
salidaProductos
```

7. Comprueba la salida.

```
hdfs dfs -cat /user/hadoop/salidaProductos/*
```

```
(base) [hadoop@nodo1 ~]$ hdfs dfs -cat /user/hadoop/salidaProductos/*
Producto1 35
Producto2 70
Producto3 50
(base) [hadoop@nodo1 ~]$
```

Block ID: 1073742018

Block Pool ID: BP-1503066721-192.168.20.57-1734496225051

Generation Stamp: 1194

Size: 39

Availability:

- nodo2
- nodo4
- nodo3

**File contents**

```
Producto1 35
Producto2 70
Producto3 50
```

Este ejercicio de MapReduce ilustra cómo sumar las cantidades de productos en un conjunto de datos distribuidos. El Mapper emite pares clave valor, en

este caso cada producto junto con su cantidad, y el Reducer acumula las cantidades para cada producto, sumándolas eficientemente.

### Ejercicio 5. Cálculo de precio mínimo y máximo por producto.

En base a una lista de precios de diversos productos, determinar el precio mínimo y máximo registrado para cada uno.

Pasos para calcular el mínimo y máximo de precios:

1. Crear la variable de entrada con los datos.
 

```
%%bash
{
echo -e "Producto1\t10"
echo -e "Producto2\t20"
echo -e "Producto3\t5"
echo -e "Producto1\t15"
echo -e "Producto2\t25"
} > ./precios.txt
```
2. Crear el archivo Mapper.
 

```
nano mapperPrecios.py

#!/usr/bin/env python3
import sys

Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 producto, precio = line.split("\t")
 print(f"{producto}\t{precio}")
```
3. Crear el archivo Reducer.
 

```
nano reducerPrecios.py

#!/usr/bin/env python3
import sys

current_product = None
min_price = float('inf')
max_price = float('-inf')

Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 producto, precio = line.split("\t")
 precio = int(precio)

 if current_product == producto:
 min_price = min(min_price, precio)
 max_price = max(max_price, precio)
 else:
```

```

if current_product:
 print(f"{current_product}\t{min_price}\t{max_price}")
current_product = producto
min_price = precio
max_price = precio

```

```

if current_product:
 print(f"{current_product}\t{min_price}\t{max_price}")

```

4. Otorgar los permisos de ejecución a los archivos creados.

```

chmod +x mapperPrecios.py
chmod +x reducerPrecios.py

```

5. Copiar el archivo local a HDFS

#### Verificar el archivo local

```
ls /home/hadoop/precios.txt
```

#### Verificar los permisos del archivo local:

```
ls -l /home/hadoop/precios.txt
```

#### Crear el directorio en HDFS (si no existe)

```
hdfs dfs -mkdir -p /user/hadoop
```

#### Copiar o mover un archivo a HDFS

```
hdfs dfs -put /home/hadoop/precios.txt /user/hadoop/
```

#### Verificar que el archivo se haya movido correctamente

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar el programa

```

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
 -file mapperPrecios.py -mapper mapperPrecios.py \
 -file reducerPrecios.py -reducer reducerPrecios.py \
 -input /user/hadoop/precios.txt -output salidaPrecios

```

7. Comprobar la salida.

```
hdfs dfs -cat /user/hadoop/ salidaPrecios/*
```

```

(base) [hadoop@nodo1 ~]$ hdfs dfs -cat /user/hadoop/salidaPrecios/*
Producto1 10 15
Producto2 20 25
Producto3 5 5
(base) [hadoop@nodo1 ~]$

```



Block ID: 1073742031

Block Pool ID: BP-1503066721-192.168.20.57-1734496225051

Generation Stamp: 1207

Size: 46

Availability:

- nodo4
- nodo2
- nodo3

File contents

|           |    |    |
|-----------|----|----|
| Producto1 | 10 | 15 |
| Producto2 | 20 | 25 |
| Producto3 | 5  | 5  |

Este ejercicio de MapReduce muestra cómo calcular los precios mínimo y máximo de productos en un conjunto de datos. El Mapper emite pares clave valor, en este caso cada producto junto con su precio, y el Reducer calcula el precio mínimo y máximo para cada producto, utilizando las funciones `min()` y `max()` para obtener los valores correspondientes.

### Ejercicio 6. Promedio de calificaciones de estudiantes

Dada una lista de calificaciones por estudiante, calcular el promedio de cada uno.

Procedimiento para calcular el promedio de calificaciones de estudiantes:

1. Crear la variable de entrada con los datos.
 

```
%%bash
{
echo -e "Juan\t85"
echo -e "Maria\t90"
echo -e "Pedro\t78"
echo -e "Juan\t88"
echo -e "Maria\t95"
echo -e "Pedro\t82"
echo -e "Juan\t92"
echo -e "Maria\t87"
} > ./calificaciones.txt
```
2. Crear el archivo Mapper.
 

```
nano mapperCalificaciones.py

#!/usr/bin/env python3
import sys

Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 estudiante, calificacion = line.split("\t")
 print(f"{estudiante}\t{calificacion}\t1")
```

3. Crea el archivo Reducer.

```
nano reducerCalificaciones.py
```

```
#!/usr/bin/env python3
import sys

current_student = None
total_score = 0
count = 0

Procesa cada línea de entrada
for line in sys.stdin:
 line = line.strip()
 estudiante, calificacion, cnt = line.split("\t")
 calificacion = float(calificacion)
 cnt = int(cnt)

 if current_student == estudiante:
 total_score += calificacion
 count += cnt
 else:
 if current_student:
 avg_score = total_score / count
 print(f"{current_student}\t{avg_score:.2f}")
 current_student = estudiante
 total_score = calificacion
 count = cnt

if current_student:
 avg_score = total_score / count
 print(f"{current_student}\t{avg_score:.2f}")
```

4. Otorgar los permisos de ejecución a los archivos creados.

```
chmod +x mapperCalificaciones.py
chmod +x reducerCalificaciones.py
```

5. Copiar el archivo local a HDFS

**Crear el directorio en HDFS (si no existe)**

```
hdfs dfs -mkdir -p /user/hadoop
```

**Copiar o mover un archivo a HDFS**

```
hdfs dfs -put calificaciones.txt /user/hadoop/
```

**Verificar que el archivo se haya movido correctamente**

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar el programa
- ```
hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
  -file mapperCalificaciones.py -mapper mapperCalificaciones.py \
  -file reducerCalificaciones.py -reducer reducerCalificaciones.py \
  -input /user/hadoop/calificaciones.txt -output salidaCalificaciones
```

7. Comprobar la salida.

```
hdfs dfs -cat salidaCalificaciones/part-*
```

```
hadoop@bigdata1:~$ hdfs dfs -cat salidaCalificaciones/part-*
Juan      88.33
Maria     90.67
Pedro     80.00
hadoop@bigdata1:~$
```

```
Block ID: 1073741840
Block Pool ID: BP-1636326706-127.0.1.1-1737434205526
Generation Stamp: 1016
Size: 35
Availability:
  • bigdata1
```

File contents

```
Juan 88.33
Maria 90.67
Pedro 80.00
```

Este ejercicio de MapReduce calcula el promedio de calificaciones por estudiante. El Mapper emite pares clave valor, en este caso las calificaciones junto con un contador de 1 por cada entrada, y el Reducer acumula las calificaciones y calcula el promedio por estudiante.

Ejercicio 7. Promedio de temperatura por ciudad

A partir de registros de temperatura por ciudad, obtener el promedio de temperatura de cada una.

Pasos para calcular el promedio de temperatura por ciudad:

1. Crear la variable de entrada con los datos.

```
%%bash
{
echo -e "Quito\t18"
echo -e "Guayaquil\t30"
echo -e "Cuenca\t16"
echo -e "Quito\t20"
echo -e "Guayaquil\t32"
echo -e "Cuenca\t17"
echo -e "Quito\t22"
echo -e "Guayaquil\t29"
echo -e "Cuenca\t15"
} > ./temperaturas.txt
```

2. Crear el archivo Mapper.

```
nano mapperTemperatura.py

#!/usr/bin/env python3
import sys

for line in sys.stdin:
    line = line.strip()
    if not line:
        continue
    partes = line.split()
    if len(partes) != 2:
        continue # Evitar líneas mal formateadas

    ciudad, temperatura = partes
    print(f"{ciudad}\t{temperatura}\t1") # Agregar '1'
    como contador
```

3. Crear el archivo Reducer.

```
nano reducerTemperatura.py

#!/usr/bin/env python3

import sys

current_city = None
total_temp = 0
total_count = 0

# Procesa cada línea de entrada
for line in sys.stdin:
    line = line.strip()
    ciudad, temperatura, count = line.split("\t")
    temperatura = int(temperatura)
    count = int(count)

    if current_city == ciudad:
        total_temp += temperatura
        total_count += count
```

```

else:
    if current_city:
        promedio = total_temp / total_count
        print(f"{current_city}\t{promedio:.2f}")
        current_city = ciudad
        total_temp = temperatura
        total_count = count

if current_city:
    promedio = total_temp / total_count
    print(f"{current_city}\t{promedio:.2f}")

```

4. Otorgar los permisos de ejecución a los archivos creados.

```

chmod +x mapperTemperatura.py
chmod +x reducerTemperatura.py

```

5. Copiar el archivo local a HDFS

Crear el directorio en HDFS (si no existe)

```
hdfs dfs -mkdir -p /user/hadoop
```

Copiar o mover un archivo a HDFS

```
hdfs dfs -put temperaturas.txt /user/hadoop/
```

Verificar que el archivo se haya movido correctamente

```
hdfs dfs -ls /user/hadoop/
```

6. Ejecutar el programa

```

hadoop jar $HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.4.0.jar \
    -file mapperTemperatura.py -mapper mapperTemperatura.py \
    -file reducerTemperatura.py -reducer reducerTemperatura.py \
    -input /user/hadoop/temperaturas.txt -output salidaTemperaturas

```

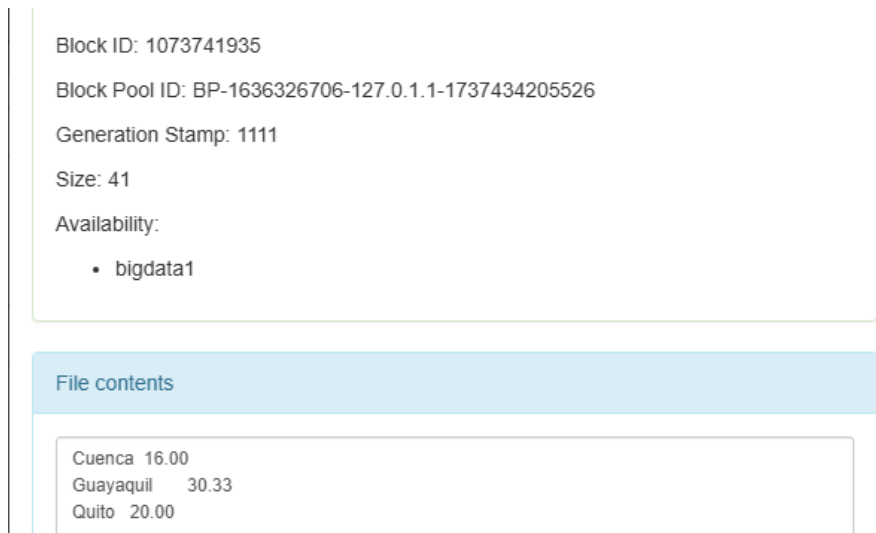
7. Comprobar la salida.

```
hdfs dfs -cat salidaTemperaturas/part-00000
```

```

hadoop@bigdatal:~$ hdfs dfs -cat salidaTemperaturas/part-00000
Cuenca 16.00
Guayaquil 30.33
Quito 20.00
hadoop@bigdatal:~$

```



Este ejercicio de MapReduce calcula el promedio de temperaturas por ciudad. El Mapper emite pares clave valor, en este caso las temperaturas por ciudad junto con un contador, mientras que el Reducer acumula las temperaturas y calcula el promedio para cada ciudad.

Ejercicio 8. Ventas de Productos

A partir de registros de ventas de Productos, obtener el total de ventas por producto.

Pasos para calcular el total de ventas por producto:

1. Crear la variable de entrada con los datos.


```
%%bash
{
echo -e "Manzanas\t30"
echo -e "Bananas\t25"
echo -e "Manzanas\t45"
echo -e "Peras\t20"
echo -e "Bananas\t30"
echo -e "Peras\t25"
echo -e "Manzanas\t20"
echo -e "Bananas\t15"
} > ./ventas.txt
```
2. Crear el archivo Mapper.


```
nano mapperVentas.py

#!/usr/bin/env python3
import sys

for line in sys.stdin:
    line = line.strip()
    producto, monto = line.split("\t")
    print(f"{producto}\t{monto}\t1")
```

3. Crear el
archivo
Combiner

```
nano combinerVentas.py
```

```
#!/usr/bin/env python3
import sys

current_product = None
total_amount = 0
sales_count = 0

for line in sys.stdin:
    line = line.strip()
    producto, monto, cnt = line.split("\t")
    monto = float(monto)
    cnt = int(cnt)

    if current_product == producto:
        total_amount += monto
        sales_count += cnt
    else:
        if current_product:
            print(f"{current_product}\t{total_amount}\t{sales_count}")
            current_product = producto
            total_amount = monto
            sales_count = cnt

if current_product:
    print(f"{current_product}\t{total_amount}\t{sales_count}")
```

4. Crear el
archivo
Reducer.

```
nano reducerVentas.py
```

```
#!/usr/bin/env python3
import sys

current_product = None
total_amount = 0
sales_count = 0

for line in sys.stdin:
    line = line.strip()
    producto, monto, cnt = line.split("\t")
    monto = float(monto)
    cnt = int(cnt)

    if current_product == producto:
        total_amount += monto
        sales_count += cnt
    else:
        if current_product:
            print(f"{current_product}\tTotal Ventas:
{total_amount:.2f}\tNúmero de Ventas: {sales_count}")
            current_product = producto
            total_amount = monto
            sales_count = cnt
```

```
if current_product:
    print(f"{current_product}\tTotal Ventas:
{total_amount:.2f}\tNúmero de Ventas: {sales_count}")
```

5. Otorgar los permisos de ejecución a los archivos creados.

```
chmod +x mapperVentas.py
chmod +x combinerVentas.py
chmod +x reducerVentas.py
```

6. Copiar el archivo local a HDFS

Crear el directorio en HDFS (si no existe)

```
hdfs dfs -mkdir -p /user/hadoop
```

Copiar o mover un archivo a HDFS

```
hdfs dfs -put ventas.txt /user/hadoop/
```

Verificar que el archivo se haya movido correctamente

```
hdfs dfs -ls /user/hadoop/
```

7. Ejecutar el programa

```
hadoop jar
/home/hadoop/hadoop/share/hadoop/tools/lib/hadoop-streaming-
3.3.6.jar \
  -file mapperVentas.py -mapper mapperVentas.py \
  -file combinerVentas.py -combiner combinerVentas.py \
  -file reducerVentas.py -reducer reducerVentas.py \
  -input /user/hadoop/ventas.txt -output
salidaVentasCombiner
```

8. Comprobar la salida.

```
hdfs dfs -cat salidaVentasCombiner/part-*
```

```
hadoop@bigdata1:~$ hdfs dfs -cat salidaVentasCombiner/part-*
Bananas Total Ventas: 70.00      Número de Ventas: 3
Manzanas      Total Ventas: 95.00      Número de Ventas: 3
Peras   Total Ventas: 45.00      Número de Ventas: 2
hadoop@bigdata1:~$
```

Block ID: 1073741888

Block Pool ID: BP-1636326706-127.0.1.1-1737434205526

Generation Stamp: 1064

Size: 146

Availability:

- bigdata1

File contents

```
Bananas Total Ventas: 70.00  Número de Ventas: 3
Manzanas      Total Ventas: 95.00  Número de Ventas: 3
Peras   Total Ventas: 45.00  Número de Ventas: 2
```


Este ejercicio de MapReduce calcula el total o promedio de ventas por producto. El Mapper emite pares clave valor, en este caso los montos de venta por producto junto con un contador, el Combiner realiza una agregación local de los datos para optimizar el procesamiento, y el Reducer acumula los montos y calcula el resultado final para cada producto.

Comandos HDFS aplicados en Interfaz en línea de comandos (I)

Los comandos básicos de [Hadoop](#) y de HDFS son similares a los que se utiliza en una terminal de un sistema Linux. En la Tabla 10, se presenta una lista con los comandos de HDFS más importantes divididos por secciones en función de su utilidad.

Tabla 10. Comandos HDFS

Comando	Significado
hdfs dfs -ls <path>	Lista ficheros
hdfs dfs -ls -R <path>	Lista recursivamente
hdfs dfs -cp <src> <dst>	Copia ficheros HDFS a HDFS
hdfs dfs -mv <src> <dst>	Mueve ficheros HDFS a HDFS
hdfs dfs -rm <path>	Borra ficheros en HDFS
hdfs dfs -rm -r <path>	Borra recursivamente
hdfs dfs -cat <path>	Muestra fichero en HDFS
hdfs dfs -tail <path>	Muestra el final del fichero
hdfs dfs -stat <path>	Muestra estadísticas del fichero
hdfs dfs -mkdir <path>	Crea directorio en HDFS
hdfs dfs -chmod ...	Cambia permisos de fichero
hdfs dfs -chown ...	Cambia propietario/grupo de fichero
hdfs dfs -du <path>	Espacio en bytes ocupado por ficheros
hdfs dfs -du -s <path>	Espacio ocupado acumulado
hdfs dfs -count <paths>	Cuenta nº dirs/ficheros/bytes
hdfs dfs -put <local> <dst>	Copia de local a HDFS
hdfs dfs -copyFromLocal ...	Igual que -put
hdfs dfs -moveFromLocal ...	Mueve de local a HDFS
hdfs dfs -get <src> <loc>	Copia de HDFS a local
hdfs dfs -copyToLocal ...	Copia de HDFS a local
hdfs dfs -getmerge ...	Copia y concatena de HDFS a local
hdfs dfs -text <path>	Muestra el fichero en texto

Fuente: (Fernandez, 2020; Kumar, 2021)

Resumen del capítulo. Se abordó una visión integral sobre el procesamiento MapReduce, abarcando tanto sus fundamentos como su aplicación práctica en escenarios de Big Data. A través de varios ejercicios, se ha explorado cómo este algoritmo puede utilizarse para resolver problemas básicos y complejos de análisis de datos, desde calcular valores máximos y mínimos hasta realizar conteos y comparaciones en grandes conjuntos de datos. Con una comprensión sólida de MapReduce, los lectores ahora cuentan con las herramientas necesarias para implementar soluciones eficientes y escalables en sus propios proyectos, aprovechando al máximo las capacidades de plataformas como Apache Hadoop.

Ejercicios Propuestos

A continuación, se presentan un conjunto de ejercicios prácticos para reforzar los conocimientos adquiridos sobre procesamiento MapReduce. Estos ejercicios están diseñados para reforzar habilidades en el procesamiento de grandes volúmenes de datos y la creación de soluciones distribuidas eficientes.

1. Total de ventas por categoría de producto. Dado un conjunto de productos categorizados, calcular el total de ventas por categoría.
2. Frecuencia de usuarios activos por mes. A partir de registros de actividad de usuarios, contar cuántos usuarios estuvieron activos cada mes.
3. Promedio de ingresos por cliente. Dado un conjunto de transacciones, calcular el ingreso promedio generado por cada cliente.
4. Máxima puntuación por juego. De una lista de puntuaciones obtenidas en diferentes juegos, identificar la puntuación más alta por juego.
5. Total de productos vendidos por región. Dado un conjunto de ventas clasificado por región, calcular el total de productos vendidos en cada una.
6. Promedio de calificaciones por materia. A partir de registros de calificaciones, obtener el promedio por cada materia.
7. Clientes con mayor número de compras. Identificar los clientes que realizaron el mayor número de compras en un periodo determinado.
8. Porcentaje de participación de cada producto en ventas totales. Calcular qué porcentaje de las ventas totales corresponde a cada producto.
9. Detección de registros duplicados. A partir de una lista de registros, identificar y contar aquellos que estén duplicados.
10. Identificación de clientes frecuentes. A partir de registros de ventas, identificar a los clientes que realizaron compras de manera frecuente.

Referencias Bibliográficas

- Alapati, S. R. (2017). *Expert Hadoop administration: Managing, tuning, and securing Spark, YARN, and HDFS*. Addison-Wesley.
- Basgall, M. J., Hasperué, W., Estrebou, C. A., & Naiouf, M. (2016). *Clustering de un flujo de datos usando MapReduce*. XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016).
<http://sedici.unlp.edu.ar/handle/10915/56748>
- Fernandez, O. (2020, abril 27). HDFS: Guía de Comandos Básicos 2025. *Aprender BIG DATA*. <https://aprenderbigdata.com/comandos-hdfs/>
- Gimenez-Garcia, J. M., Fernandez, J. D., & Martinez-Prieto, M. A. (2016). *Compresión de Big Semantic Data basada en HDT y MapReduce*.
- Hadoop: The Definitive Guide, 4th Edition*[Book]. (2015).
<https://www.oreilly.com/library/view/hadoop-the-definitive/9781491901687/>
- Hedlund, B. (2011, septiembre 10). *Understanding Hadoop Clusters and the Network*. Brad Hedlund.
<http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/>
- Jeyaraj, R., Pugalendhi, G., & Paul, A. (2021). *Big data with Hadoop MapReduce: A classroom approach*. Apple Academic Press.
- Kumar, N. (2021). *Big Data Using Hadoop and Hive*. Mercury Learning & Information.
- Lăpușan, T. (2022). *Hadoop MapReduce deep diving and tuning*. Today Software Magazine.
<http://www.todaysoftmag.com/article/1358/hadoop-mapreduce-deep-diving-and-tuning>
- Martínez, C. G., Soriano, N. P., López, J. G., & de, A. (2013). *PROCESAMIENTO DE GRANDES VOLÚMENES DE DATOS EN ENTORNOS CLOUD COMPUTING UTILIZANDO HADOOP MAPREDUCE*.
- Urrutia, V. I. R., & Rivera-Pleitez, J. G. (2016). Big Data, análisis de datos en la nube. *Entorno*, 61, Article 61.
<https://doi.org/10.5377/entorno.v0i61.6126>

Capítulo V

Procesamiento y análisis de Big Data con Apache Spark

- Introducción a Apache Spark
- Bases Teóricas
- Componentes Principales
- Lenguajes de programación Spark
- Ejercicios prácticos con RDD y DataFrame

Objetivos:

- Comprender el funcionamiento de Apache Spark y su importancia en el procesamiento de datos.
- Describir los componentes principales de Apache Spark y sus funciones.
- Procesar datos de gran volumen utilizando RDDs y DataFrames en Apache Spark.

Apache Spark es conocido como un motor de procesamiento de datos distribuido el cual nos permite la manipulación y el análisis de grandes volúmenes de información de una manera rápida y eficiente. Originalmente fue desarrollado en los laboratorios de la Universidad de California, Berkeley, y ahora ha evolucionado para ser una de las herramientas más populares en lo que respecta al procesamiento de datos masivo. A diferencia de sistemas como Hadoop MapReduce, Spark es conocido por destacar su capacidad de realizar procesamiento en memoria, lo que reduce significativamente el tiempo de aplicaciones que requieren múltiples etapas de cómputo. Su característica más grande es su versatilidad ya que soporta tareas de análisis de datos, procesamiento en tiempo real, machine learning y análisis de grafos. Spark ofrece una interfaz de fácil manejo y una capacidad para ser escalable en grandes clústeres, convirtiéndose en una de las herramientas adecuadas para la gestión y análisis de datos a gran escala. Además, Spark facilita la integración con varias fuentes de datos como HDFS, S3, JDBC o Cassandra, entre otras; y también, posee compatibilidad con lenguajes de programación como Java, Scala, Python y R, permitiendo que los usuarios puedan realizar análisis de datos complejos, usando herramientas con las cuales se encuentran familiarizados.

Como se observa en la Figura 27, se ilustra un resumen gráfico sobre las funcionalidades de Apache Spark, características, arquitectura y las aplicaciones en las que se utiliza, y, su integración con diversas fuentes de datos y lenguajes de programación.

«Spark permite a los programadores cargar datos en la memoria caché de un clúster y consultarlos repetidamente, esto lo hace ideal para algoritmos de aprendizaje automático iterativos.» -

Karau, Konwinski, Wendell & Zaharia

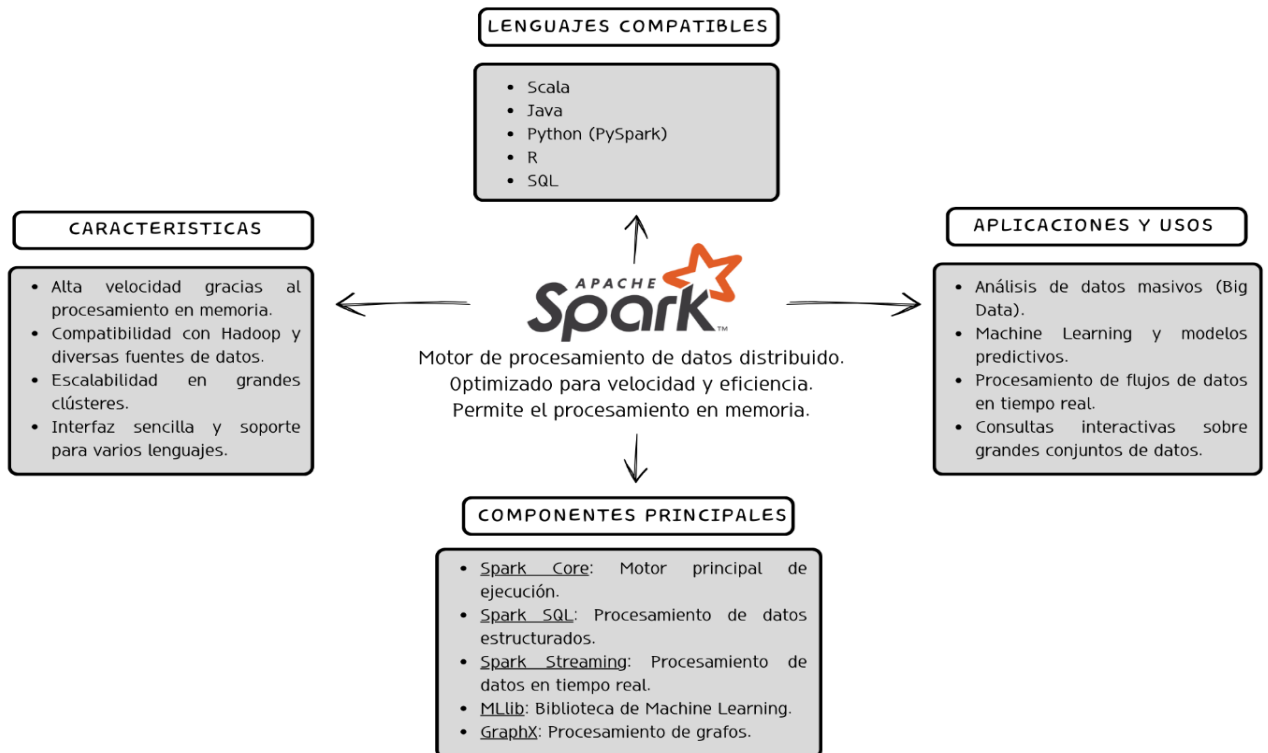


Figura 27. Resumen del Capítulo
Fuente: Elaboración Propia

Fundamentos de Apache Spark

¿Qué es Apache Spark?

Apache Spark es una plataforma la cual ha sido desarrollada para la optimización de velocidad y rendimiento de aplicaciones de Big Data. El término "Spark", en inglés hace referencia a una partícula incandescente o chispa, esto simboliza metafóricamente lo que puede ser capaz esta tecnología para mejorar la eficiencia de las aplicaciones que operan en internet. Spark busca ser esa chispa que acelera el procesamiento de datos, permitiendo consultas interactivas y algoritmos iterativos con respuestas más rápidas; esto se debe a que Spark distribuye los procesos en memoria de múltiples maquinas, lo que hace que mejore significativamente el rendimiento (Macias et al., 2016; Panchikala, 2018).

Historia y evolución de Spark

Los orígenes de Spark se dieron en los laboratorios de investigación de la Universidad de Berkeley en 2009, en donde surgió como un proyecto de código abierto, alcanzó su popularidad en el 2013, cuando se convirtió en un proyecto respaldado por Apache Software Foundation. Desde entonces, su desarrollo ha experimentado un crecimiento masivo (Macias et al., 2016).

Mapreduce vs Spark

A continuación, en la Tabla 11, se presenta una comparación entre MapReduce y Apache Spark, destacando sus principales diferencias en cuanto a eficiencia, rendimiento y funcionalidad.

Tabla 11. MapReduce vs Spark

MAPREDUCE	SPARK
No es eficiente en aplicaciones que necesitan múltiples pasos y comparten datos con baja latencia entre operaciones paralelas.	Facilita el procesamiento de flujos de datos grandes, es ideal para el aprendizaje automático en línea y análisis que requieren de respuestas inmediatas
Es más lento debido a que los resultados intermedios se guardan en el disco duro.	Es más rápido, especialmente en tareas repetitivas, ya que los resultados intermedios se mantienen en memoria.
Tiene un enfoque al procesamiento de datos por lotes y necesita otras herramientas para tareas como el procesamiento de flujos, aprendizaje automático o consultas a bases de datos.	Aparte del procesamiento por lotes, tiene herramientas integradas para flujos de datos (Spark Streaming), aprendizaje automático (MLlib), procesamiento de grafos (GraphX) y consultas SQL (Spark SQL).
Necesita menos memoria para poder funcionar.	Requiere más memoria, y su rendimiento puede reducirse si los datos no caben en la memoria que se encuentra disponible.
MapReduce producen datos en formas de pares Clave-Valor, los cuales se almacenan en un buffer circular en lugar de escribirse en el disco directamente.	Los resultados de Map se guardan en el caché del sistema operativo, que decide si los datos permanecen allí o se mueven hacia el disco.

Fuente: (Chauhan & Sharma, 2016)

¿Cómo funciona Apache Spark?

Apache Spark está diseñado para dirigir y supervisar aplicaciones que están compuestas por múltiples tareas de procesamiento de datos el cual se ejecuta en un clúster. Como hemos mencionado anteriormente, Spark tiene la capacidad de leer datos de diversas soluciones de almacenamiento persistente, como Amazon S3 o Google Storage, o sistemas de almacenamiento distribuido como HDFS e incluso bases de datos clave-valor como Pache Cassandra (Apache Spark, 2018; Luu, 2021).

Sin embargo, es de suma importancia resaltar que Spark no almacena datos por sí mismo, si no que está centrado exclusivamente en el procesamiento. Esta es una de las principales diferencias con Hadoop, el cual integra un sistema de almacenamiento persistente (HDFS) como su motor de procesamiento (MapReduce). Uno de los aspectos de los cuales destaca Spark es su velocidad de procesamiento, la clave de su velocidad radica en su capacidad de realizar operaciones en memoria, lo que hace que su rendimiento sea más eficiente significativamente. Además, Spark extiende las funcionalidades de MapReduce, permitiéndonos otros tipos de operaciones como consultas interactivas y procesamiento en tiempo real (streaming). Todas estas características hacen que se convierta en una de las herramientas más poderosas y versátiles para el manejo de grandes volúmenes de datos (Rajpurohit et al., 2024).

Apache Spark trabaja en un marco diseñado para la computación en clúster, lo que la hace una herramienta muy versátil para una gran variedad de aplicaciones. Los perfiles profesionales que comúnmente usan Spark son los científicos de datos e ingenieros de datos (Damji et al., 2020).

Tareas en Ciencias de Datos

Es una disciplina relativamente nueva que se enfoca en el análisis y modelado de datos para extraer conocimiento valioso. Un profesional que se dedica a la ciencia de datos, se considera un científico de datos; su labor principal se centra en analizar grandes volúmenes de información y el desarrollar modelos predictivos. Estos profesionales habitualmente cuentan con conocimientos en SQL, estadística, lenguajes de programación modernos como Phyton, Matlab o R; además, realizan tareas de limpieza de datos como transformar y prepararlos, lo cual es necesario para obtener resultados valiosos en sus análisis (Damji et al., 2020). Es decir, el trabajo de un científico de datos se basa en dar respuestas a preguntas específicas o descubrir patrones que se encuentran ocultos entre los datos. Su trabajo también implica el análisis

exploratorio, realizar consultas de manera interactiva y obtener resultados de manera rápida. En este sentido, Spark se ha destacado por su rapidez y su API sencilla, de manera que proporciona un entorno eficiente para la exploración y el modelado de datos. Además, cuenta con bibliotecas integradas con algoritmos fáciles de utilizarse sin necesidad de desarrollarlos desde cero, lo que facilita y reduce el tiempo de análisis y experimentación (Damji et al., 2020).

Aplicaciones de Spark en procesamiento de datos

Spark es empleado para desarrollar aplicaciones de procesamiento de datos a nivel empresarial, que sean eficientes, escalables y que permitan procesar grandes volúmenes de datos de manera automatizada y estructurada (Damji et al., 2020). Spark simplifica la ejecución de tareas en entornos distribuidos sin necesidad de gestionar aspectos complejos como la comunicación entre nodos, la tolerancia a fallos o la sincronización de procesos. A su vez también les otorgan el control que necesitan para supervisar, optimizar y ajustar el rendimiento de sus aplicaciones. La arquitectura modular de Spark, facilita la creación de bibliotecas reutilizables y permite probar los componentes de manera local antes de implementarlos en producción (Damji et al., 2020).

En términos generales, las aplicaciones Spark pueden ser clasificadas en dos grandes áreas: la ciencia de datos y las aplicaciones de procesamiento de datos. Los científicos de datos habitualmente usan Spark para el análisis avanzado de un gran volumen de datos, creaciones de modelos predictivos, entre otros. Por otra parte, los ingenieros de datos adaptan Spark para gestionar y procesar los datos de forma más eficiente en entornos distribuidos.

Componentes de Apache Spark

Apache Spark es una herramienta equipada con componentes que están diseñados para su fácil interoperabilidad y gestionar diversas cargas de trabajo, como son SQL o MLib. Esto significa que al integrar estos componentes podemos usar sus funcionalidades como si se trataran de bibliotecas dentro de un proyecto de software. Al tener todo en un solo lugar, Spark ofrece varias ventajas como pueden ser la optimización de los costos de ejecución y el desarrollo de aplicaciones más versátiles, ya solo necesitamos mantener un solo sistema en lugar de tener varios. Esto puede ayudarnos a simplificar el trabajo cuando se tratan de proyectos un poco complejos (Gowda, 2024; Luu, 2021).

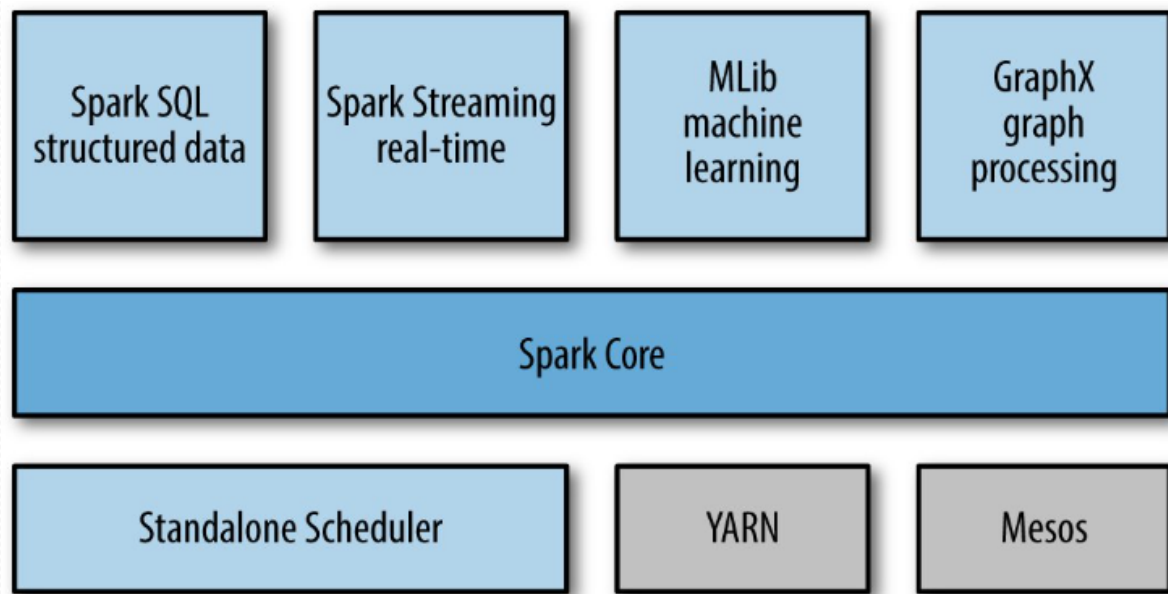


Figura 28. Componentes Principales de Spark

Fuente: [Kdnuggets](#)

En la Figura 28 podemos observar cómo se organizan y ejecutan las tareas (es decir, la gestión del clúster), Spark es muy flexible. Podemos elegir opciones, como Standalone, Yarn o Mesos, esto dependiendo de cómo mejor se adapten a tus necesidades. Estos sistemas nos ayudan a administrar los recursos de manera eficiente. Spark también cuenta con varios módulos que componen su arquitectura principal, cada uno de estos módulos se especializa en una tarea específica, lo que hace que Spark sea una herramienta muy completa y adaptable (Damji et al., 2020).

El framework de Spark se compone de los siguientes componentes (Prado, 2020):

- **Spark Core:** Es el núcleo o base sobre el cual el resto de los módulos se construyen. Contiene librerías importantes que soportan el funcionamiento del sistema.
- **Spark SQL:** Este módulo está diseñado para el procesamiento de datos estructurados y semiestructurados. Facilita la manipulación y el análisis de datos mediante consultas SQL y otras operaciones.
- **Spark Streaming:** Ayuda en el procesamiento de datos en tiempo real; si se tiene una fuente de datos en continuo flujo, este módulo permite capturar esos datos, transformarlos en caso de ser necesarios para posteriormente enviarlos a un destino en específico.
- **Spark MLlib:** Es una librería muy extensa y robusta que contiene una gran variedad de algoritmos de Machine Learning, como clustering,

clasificación, regresión, entre otros. Ofrece una interfaz fácil de usar para implementar y usar estos algoritmos en proyectos de análisis de datos.

- **Spark GraphX:** Este es un componente especializado en el procesamiento de Grafos (DAG), nos permite realizar operaciones sobre grafos, trabajando con sus nodos y aristas.

Estos componentes permiten abordar una amplia gama de problemas y aplicaciones en lo que respecta al procesamiento de datos.

Lenguajes de programación compatibles con Spark

Apache Spark tiene compatibilidad con múltiples lenguajes de programación, esto ofrece a los desarrolladores la flexibilidad de elegir el lenguaje al que mejor se ajuste a sus necesidades y experiencia. Esta versatilidad es una de las razones por las que Spark se ha convertido en una de las herramientas más populares en aplicaciones de Big Data (*Apache Spark*, 2024). Los lenguajes que soporta Spark son:

Scala: Es el lenguaje principal que se usa en el desarrollo de Apache Spark y que, de hecho, Spark está escrito en Scala, permitiéndonos así una integración más natural y eficiente. La API principal de Spark en Scala nos permite escribir código claro y expresivo para el procesamiento de datos distribuidos. Además, Scala es conocido por la capacidad de combinar programación funcional y orientada a objetos, lo que la hace ideal para tareas de procesamiento de datos complejas (*Apache Spark*, 2024).

Java: Spark también tiene compatibilidad con este lenguaje de programación, es uno de los lenguajes más populares y que se ha adaptado ampliamente en el desarrollo de software. Los desarrolladores pueden utilizar Java para crear aplicaciones Spark, aprovechando su amplia comunidad de soporte. Esto es muy útil para las empresas que cuentan ya con una base de código existente en Java y que desean integrar Spark en sus sistemas sin la necesidad de migrar a otro lenguaje (*Apache Spark*, 2024).

R: Spark también cuenta con soporte para R, es un lenguaje especializado en análisis estadístico y la visualización de datos. Esta compatibilidad permite aprovechar a los usuarios de R la escalabilidad y velocidad de Spark en sus proyectos de análisis de datos de forma que combina lo mejor de ambos mundos. Los desarrolladores que ya tienen experiencia y están familiarizados con R pueden usarlo en Spark para el manejo de grandes volúmenes de datos sin tener que aprender uno nuevo (*Apache Spark*, 2024).

SQL: Spark nos permite ejecutar consultas SQL a través de su módulo Spark SQL. Esto nos facilita la ejecución de consultas SQL sobre los conjuntos de datos distribuidos, esto es muy útil para aquellos desarrolladores que cuentan

con experiencia con el manejo de bases de datos relacionales. Spark SQL también nos permite integrar consultas SQL con código que se encuentra escrito en otros lenguajes, esto nos brinda una gran flexibilidad en el desarrollo de aplicaciones (*Apache Spark*, 2024).

Phyton: Este es uno de los lenguajes que ha sido utilizado ampliamente en la ciencia de datos y análisis, es un lenguaje que a través de la API PySpark, los desarrolladores podrán escribir código en Phyton para poder interactuar con las funcionalidades de Spark, lo que lo convierte en una opción muy popular para proyectos de análisis y machine learning. Phyton ofrece simplicidad y legibilidad que combinada con la potencia de Spark, hacen que este lenguaje sea una opción ideal para los científicos de datos (*Apache Spark*, 2024).

Esta compatibilidad de Spark con varios lenguajes de programación hace que Spark sea accesible para una gama extensa de desarrolladores y profesionales de la ciencia de datos, esto a su vez ayuda a las organizaciones a aprovechar las habilidades existentes en sus equipos de analistas.

Modelo de Ejecución Distribuida

Apache Spark opera bajo una arquitectura maestro-esclavo con un administrador de clúster. En un clúster de Spark, existe un único nodo maestro y múltiples nodos trabajadores o workers (esclavos). En la Figura 29, se observa la arquitectura maestro-esclavo de Apache Spark, donde el Driver Program coordina los ejecutores en los nodos trabajadores mediante el Cluster Manager (Buitrago, 2020).

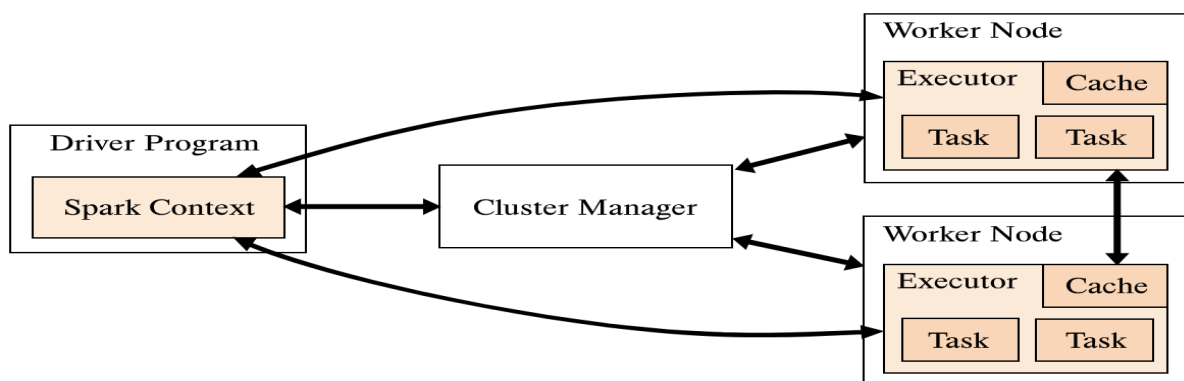


Figura 29. Arquitectura de un Clúster de Apache Spark

Fuente: (Buitrago, 2020)

El nodo maestro actúa como la instancia principal que gestiona el Driver Program, mientras que los workers se encargan de alojar y ejecutar las tareas asignadas. Estos workers pueden ser ejecutados en un solo servidor o distribuirse en varios nodos dentro de un clúster. En conclusión, la arquitectura de Spark se componen de un controlador, varios trabajadores y un

administrador del clúster que gestiona la ejecución de los procesos (Buitrago, 2020).

Spark Driver: Es el programa que contiene el método principal, el cual marca el inicio de la ejecución. Este proceso se ejecuta en la JVM (Java Virtual Machine), el cual es el responsable de coordinar y llevar a cabo las tareas que se encuentran dentro del programa. Esto sería algo equivalente a la clase que contiene el método `public static void main(String[] args [])` en Java. El Driver Program se ejecuta en el nodo maestro y es el que se encarga de crear y gestionar el contexto de Spark (Buitrago, 2020).

Woker: Son las instancias en las que residen los ejecutores, son los encargados de ejecutar el código de la aplicación que el usuario ha escrito para el clúster (Buitrago, 2020).

Executors: Son los procesos que se ejecutan en la memoria del nodo trabajador (worker). En estos se almacenan tanto los bloques de datos como tareas o el código que se va a ejecutar. Cada aplicación de Spark contiene su propio proceso de ejecución, el cual es manejado a través de un executor (Buitrago, 2020).

Cluster Manager: Es el componente que es responsable de la distribución de los recursos entre las aplicaciones de Spark. El contexto de Spark puede conectarse a diversos administradores de clústeres, como Mesos, YARN o Kubernetes, además del administrador de clúster independiente de Spark (Buitrago, 2020).

Modelos de datos en Spark

Spark utiliza tres herramientas clave que ayudan en la gestión de datos masivos: los RDDs, los DataFrames y los Datasets (Figura 30). Estas estructuras permiten desde análisis simples hasta operaciones complejas, adaptándose a diferentes necesidades mientras garantizan que los datos nunca se pierdan, incluso si falla un componente del sistema. A continuación, se describen cómo funcionan.

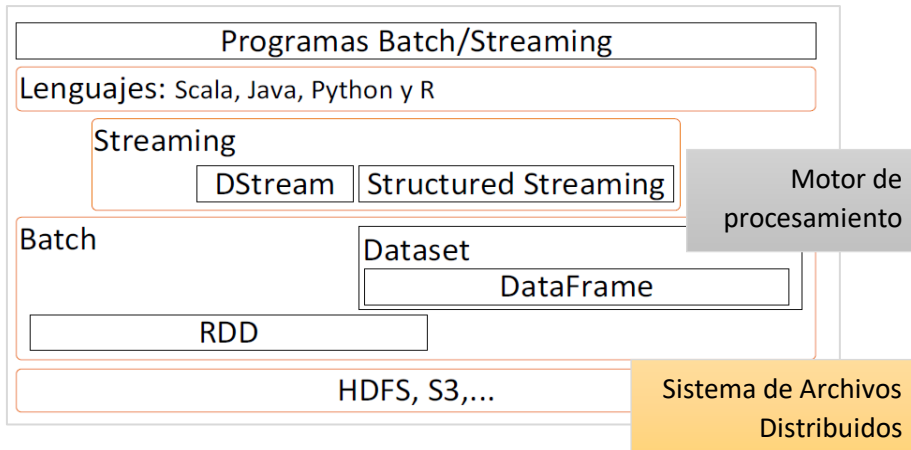


Figura 30. Modelos de datos y procesamiento de Spark

Fuente: (Damji et al., 2020; Luu, 2021).

Conjuntos de Datos Resilientes (RDDs)

Los RDDs son una estructura central de Apache Spark que representa una colección inmutable de elementos o datos distribuidos a través de un clúster, que pueden ser procesados en paralelo; es decir, que una vez que se crean no pueden modificarse, pero si pueden ser procesados en paralelo a través de varios nodos. Su diseño permite realizar operaciones eficientes sobre grandes volúmenes de datos mediante transformaciones funcionales y acciones.

Los RDDs proporcionan tolerancia a fallos mediante un mecanismo de seguimiento de linaje, lo que permite reconstruir los datos perdidos si un nodo falla (Rajpurohit et al., 2024). Los RDDs se crean a partir de archivos o transformaciones de otros RDDs, y pueden procesarse mediante:

- **Transformaciones** (como map, filter, flatMap) que generan nuevos RDDs sin que se altere el RDD original.
- **Acciones** (como count, reduce, save) que producen resultados concretos o tangible, como devolver un valor o guardar los datos en un sistema externo.

Los RDDs son adecuados para tareas que requieren control explícito del flujo de datos, como algoritmos iterativos o procesamiento a bajo nivel.

Dataframes

Un DataFrame en Apache Spark es una abstracción de datos estructurados, similar a una tabla en una base de datos relacional o a un DataFrame de pandas en Python. Representa una colección distribuida de datos organizados en columnas con nombres y tipos de datos, lo que permite consultas y transformaciones de manera declarativa utilizando **Spark SQL** (Rajpurohit et al., 2024).

A diferencia de los RDDs, los DataFrames permiten optimizaciones automáticas a través del **Catalyst Optimizer**, lo que mejora el rendimiento en tareas de análisis de datos a gran escala. Se pueden crear a partir de diversas fuentes como archivos CSV, JSON, Parquet, Avro, bases de datos JDBC relacionales y NoSql (Apache HBase o Apache Cassandra, etc.) o incluso desde RDDs. Los DataFrame tienen las siguientes características:

- Esquema definido con nombres de columnas y tipos de datos.
- Operaciones expresadas en SQL o en una API funcional (como select, filter, groupBy, etc.).
- Mayor rendimiento gracias a la ejecución optimizada y a la planificación lógica y física.

Cuando algún usuario ejecuta una consulta u operación sobre un DataFrame, Spark analiza esa solicitud y genera un plan para una ejecución eficiente, que ayuda a mejorar el rendimiento significativamente en comparación con los RDDs. Los DataFrame ofrecen una gran cantidad de funciones que nos facilita las transformaciones y análisis de datos, como select(), filter(), groupBy(), join(), distinct(), orderBy() y agg() (Rajpurohit et al., 2024).

Dataset

Un Dataset en Apache Spark es una estructura de datos distribuida que combina las ventajas de los RDDs (tipado fuerte y manejo explícito de objetos) con las optimizaciones de los DataFrames (consultas declarativas y Catalyst Optimizer). Los DataSets solo se encuentran disponibles en lenguajes de Java y Scala (Rajpurohit et al., 2024).

Operaciones en RDDs

En Spark las abstracciones de datos son estructuras inmutables, esto quiere decir que una vez creadas ya no pueden ser modificadas. El procesamiento de los datos en Spark está basado en la aplicación de varias operaciones sobre estos datos. Estas operaciones están divididas en dos categorías: Transformaciones, las cuales permiten generar nuevos conjuntos de datos a partir de existentes, y las acciones, los cuales ejecutan cálculos y nos entregan un resultado (Fajardo, 2022).

Transformaciones

Las transformaciones en Apache Spark son funciones que están diseñadas para la manipulación de datos. Estas son definidas de manera abstracta y no son ejecutadas inmediatamente cuando las declaramos, ya que Spark empieza

el proceso cuando se invoca una acción. Este comportamiento, se conoce como Lazy Evaluation o evaluación perezosa la cual garantiza que estas transformaciones sean ejecutadas solo cuando sean necesario (Fajardo, 2022).

Algunas de las transformaciones que suelen ser las más comunes son: Map, Filter, FlatMap, Distinct y Union. A continuación, se detallan cada una de ellas:

Map: Aplica una función a cada elemento de un conjunto de datos, transformándolos en un nuevo valor. Por cada elemento de entrada hay exactamente un elemento de salida (Figura 31).

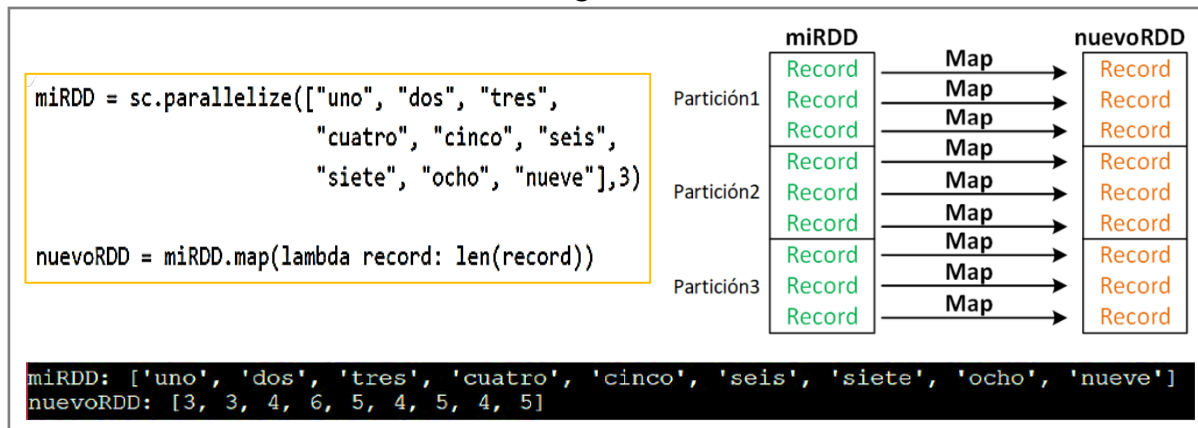


Figura 31. Funcionamiento de Map

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Filter: Filtra elementos de la colección conservando solo aquellos que cumpla con la condición especificada (Figura 32).

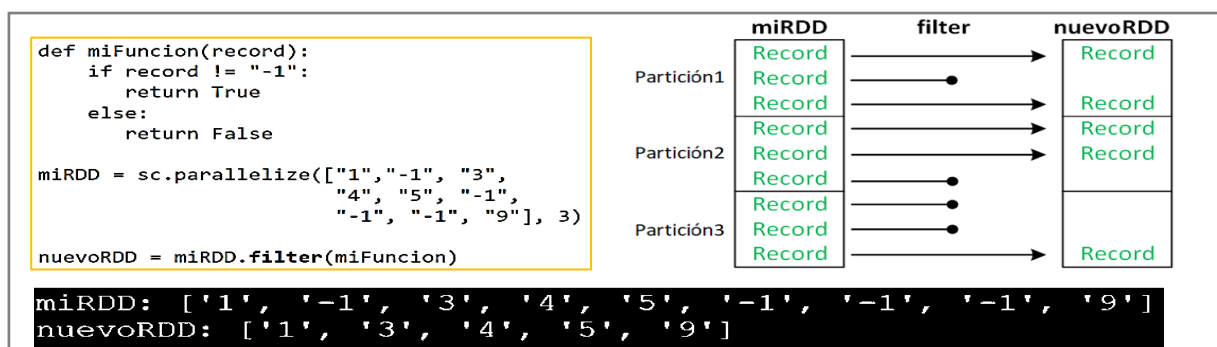


Figura 32. Funcionamiento de Filter

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Distinct: Remueve elementos que se encuentra duplicados y se conservan aquellos que contengan un valor único en la colección (Figura 33).

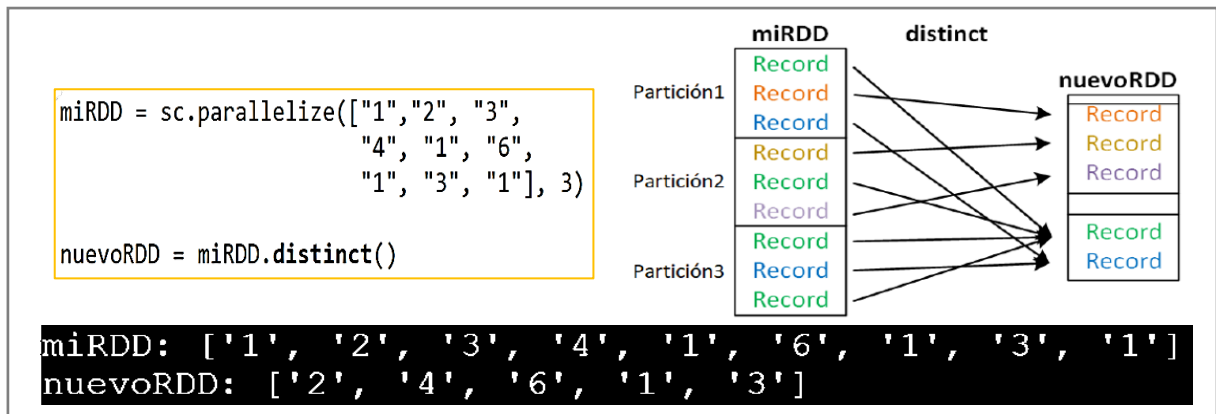


Figura 33. Funcionamiento de Distinct

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Union: Combina dos colecciones en una sola la cual contiene todos los elementos de ambas colecciones (Figura 34).

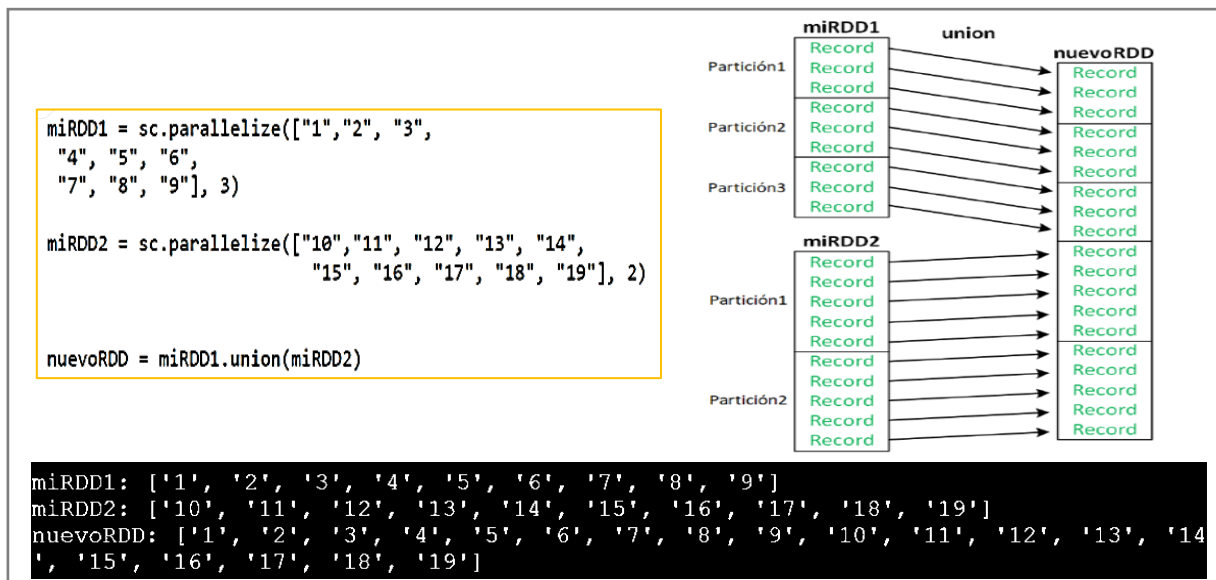


Figura 34. Funcionamiento de Union

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Según (Fajardo, 2022), las transformaciones que se consideran como las más comunes en Apache Spark son:

FlatMap, MapPartitions, MapPartitionsWithIndex, Sample, GroupByKey, Intersection.

transformaciones se pueden clasificar en dos categorías según la dependencia de los datos: estrechas (*narrow*) y amplias (*wide*). Como se muestra en la **Figura 35**, las transformaciones estrechas son las que en cada partición de entrada ayuda exclusivamente a una sola partición de salida. Estas operaciones pueden

ser ejecutadas localmente dentro de un solo nodo, sin tener la necesidad de transferir datos entre los distintos nodos del clúster. Las transformaciones amplias en cambio involucran particiones de entrada las cuales afectan a múltiples particiones de salida. Para que estas puedan ser ejecutadas es necesario intercambiar datos entre los diferentes nodos del clúster, este proceso se suele conocer como shuffle (Fajardo, 2022).

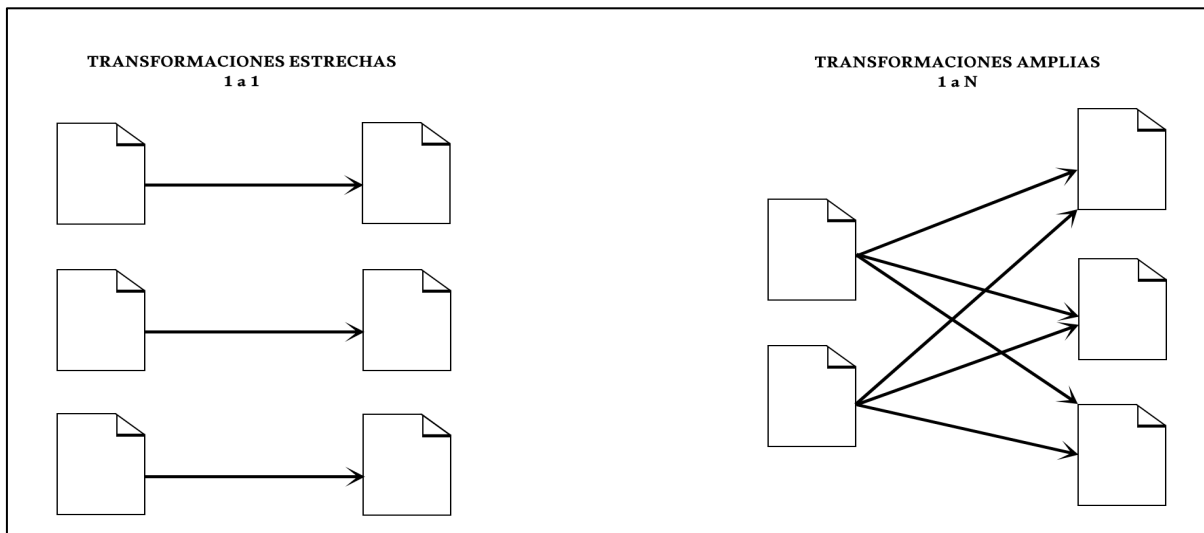


Figura 35. Ejemplo de Transformaciones Estrechas y Amplias

Fuente: (Fajardo, 2022)

Acciones

Las acciones son funciones las cuales nos permiten obtener un resultado a partir de los datos que hemos procesado. Dependiendo de que acción usemos el resultado puede variar ya que puede ser un valor único, una colección de valores o hasta el almacenamiento de los datos en un sistema externo (Fajardo, 2022).

Algunas de las acciones que suelen ser las más comunes son: Collect, Count, First, Take, TakeSample. A continuación, se detallan cada una de ellas:



Nota: Manejar con cuidado algunos comandos como: collect(), take(n), repartition(n), coalesce(n), cache() / persist(), join(), groupBy(), ya que pueden provocar problemas como: afectar el rendimiento, generar cuellos de botella, provocar un desbordamiento de memoria en el driver, entre otros problemas.

Collect: Recupera todos los datos de un RDD y los devuelve en forma de lista en el driver. Manejarlo con cuidado y solo en conjuntos pequeños de datos, ya que puede causar desbordamiento de memoria si el conjunto de datos es muy grande (Figura 36).

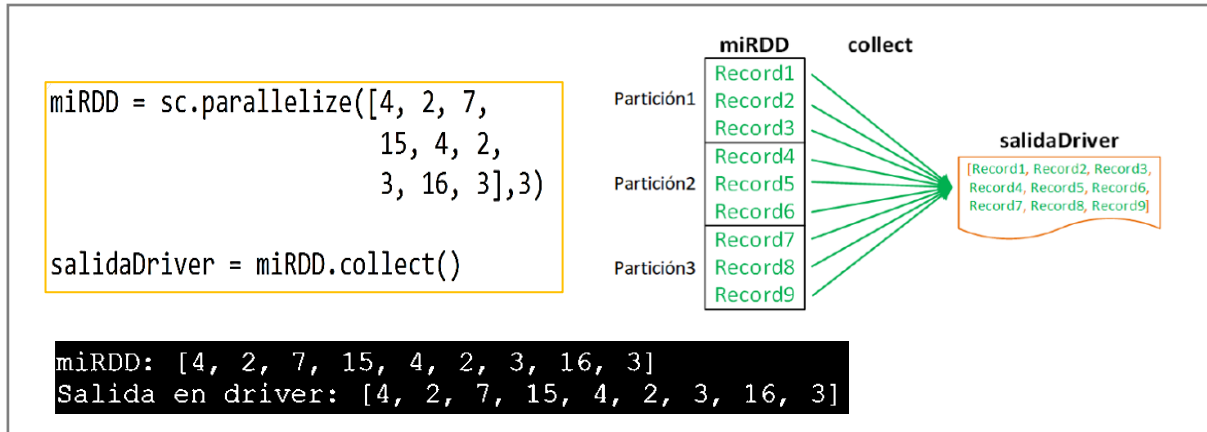


Figura 36. Funcionamiento de Collect

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Count: Devuelve el número total de un RDD. Manejarlo con cuidado ya que puede forzar la evaluación completa del RDD/DataFrame. Es Costoso para grandes volúmenes (Figura 37).

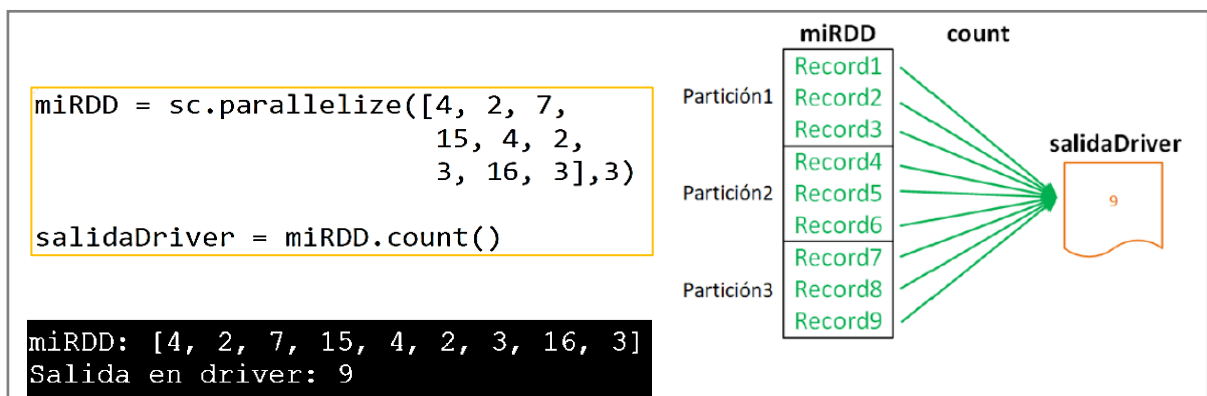


Figura 37. Funcionamiento de Count

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

First: Devuelve el primer elemento de un RDD (Figura 38).

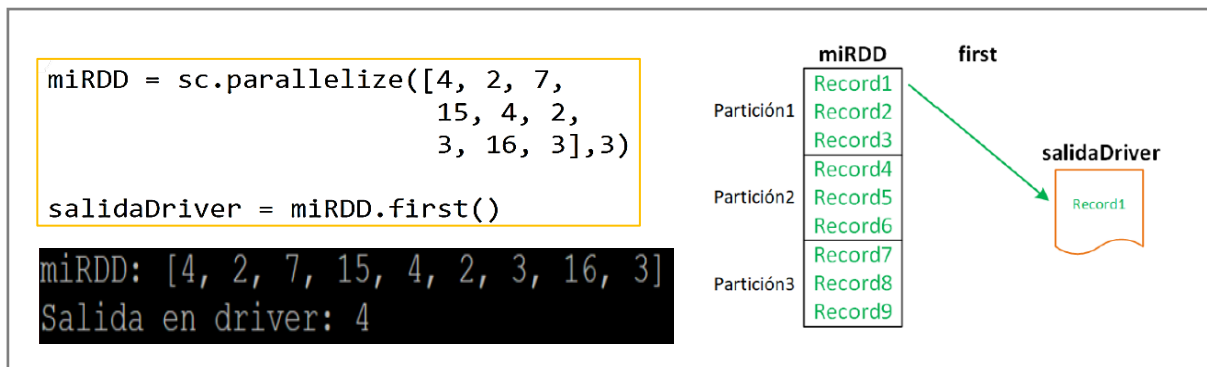


Figura 38. Funcionamiento de First

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Take: Devuelve n elementos en forma de lista de un RDD (Figura 39).

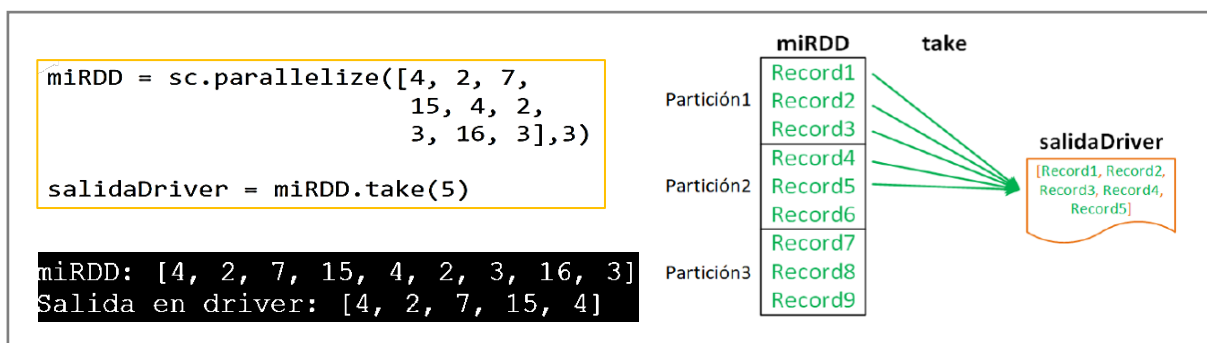


Figura 39. Funcionamiento de Take

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

TakeOrdered: Devuelve n elementos de un RDD ordenados de menor a mayor (Figura 40).

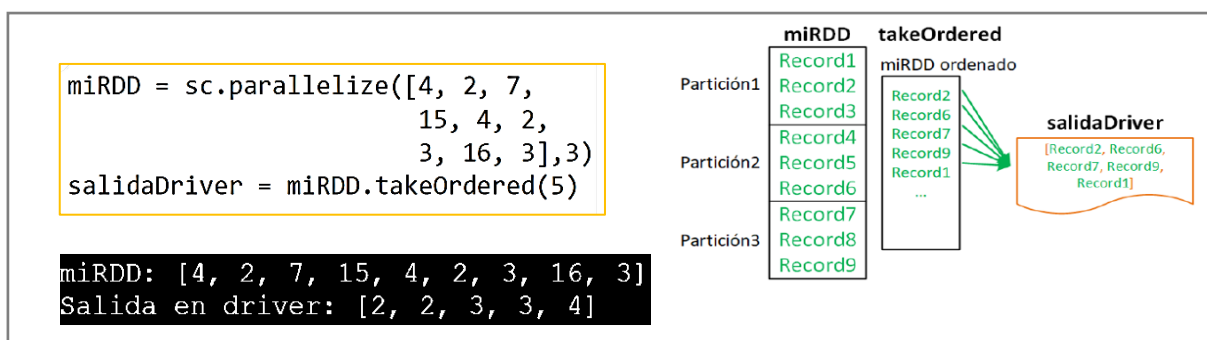


Figura 40. Funcionamiento de TakeOrdered

Fuente: Apuntes asignatura de Procesamiento de Big Data. [Maestría de Big Data y Ciencia de Datos. VIU](#)

Existen más acciones que según (Fajardo, 2022) en Spark como: Reduce, Foreach, CountByKey, TakeSample, SaveAsTextFile, SaveAsObjectFile.



Nota: Como se ilustra en la **Figura 41**, una transformación no se ejecuta de manera inmediata, sino que se registra en el DAG (**Grafo Acíclico Dirigido**). Solo cuando se invoca una acción, Spark evalúa y ejecuta las transformaciones para poder proporcionar un resultado.

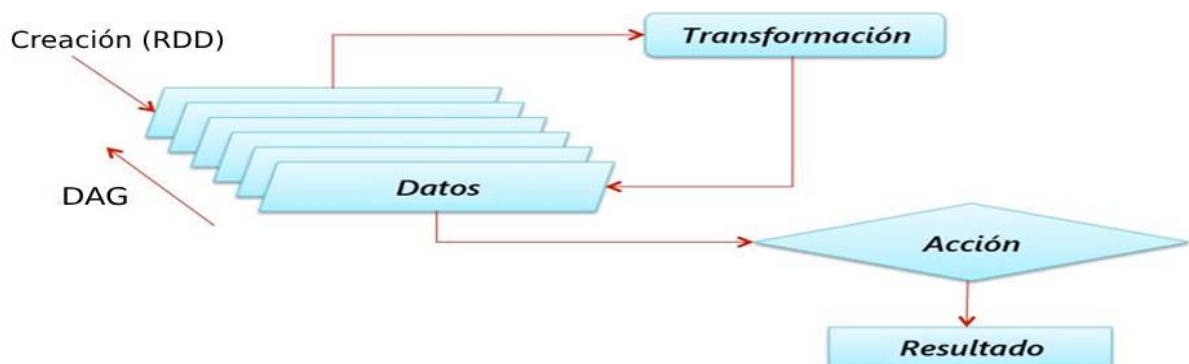


Figura 41. Transformaciones y Acciones

Fuente: (Fajardo, 2022)

Spark sigue el modelo MapReduce, en la cual Reduce es una de las acciones más relevantes. Además existen otras acciones que están diseñadas para la recuperar datos específicos o almacenarlos de manera eficiente en varios formatos y ubicaciones (Fajardo, 2022).

Ejercicios prácticos

Antes de empezar a realizar los ejercicios es importante iniciar todos los servicios del clúster, con el comando `start-dfs.sh`, `start-yarn.sh` y `start-master.sh`.

`start-dfs.sh` (Inicia el sistema de archivos distribuido HDFS).

```

hadoop@bigdata:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bigdata]
hadoop@bigdata:~$
  
```

`start-yarn.sh` (Inicia el sistema de gestión de recursos YARN).

```
hadoop@bigdata:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@bigdata:~$
```

start-master.sh (Script de Apache Spark para iniciar el nodo maestro en modo standalone).

```
hadoop@bigdata:~$ start-master.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark/logs/spark-hadoop-org.apache.spark.
deploy.master.Master-1-bigdata.out
hadoop@bigdata:~$
```

Para verificar los servicios que se encuentran activos lo hacemos mediante el comando **jps**.

```
hadoop@bigdata:~$ jps
2209 DataNode
2388 SecondaryNameNode
2629 ResourceManager
2760 NodeManager
2088 NameNode
3261 Jps
3118 SparkSubmit
```

Para poder realizar los ejercicios tendremos que entrar a través del comando **pyspark**, la cual es la interfaz de Python de Apache Spark.

```

  ____  _
 / ___|| | | |
 \___ \| |_| |
  ___) | __| |
 |___)_| |_| |
  version 3.5.4

Using Python version 3.12.7 (main, Oct 4 2024 13:27:36)
Spark context Web UI available at http://192.168.1.174:4040
Spark context available as 'sc' (master = local[*], app id = local-1738287982879)
SparkSession available as 'spark'.
```



Nota: Los códigos proporcionados son Python, para pegar el código solución de los ejercicios en el clúster se lo puede hacer simplemente haciendo clic derecho en el clúster (PySpark).

Ejercicio 1. Gastos con tarjeta de crédito

Dado un dataset que contenga entradas con la forma “persona; método_pago; dinero_gastado”, crea un programa llamado personaGastosConTarjetaCredito

que para cada persona indique la suma del dinero gastado con tarjeta de crédito, con el formato persona; gastoconTDC.

Ejemplo de entrada y salida de datos:

Entrada

Alice; Tarjeta de crédito; 100
 Alice; Tarjeta de crédito; 150
 Alice; Bizum; 200
 Bob; Tarjeta de crédito; 201
 Luis; Bizum; 300

Salida

Alice; 250
 Bob; 201
 Luis; 0



Nota: Alice gasta en total 450 euros, pero sólo 250 son con tarjeta de crédito (100 + 150). Se valorará positivamente la eficiencia del programa, por ejemplo, no usar transformaciones innecesarias.

A continuación, se presenta la solución de dos maneras: **1) mediante de RDD**
2) mediante DataFrame de Apache Spark.

Pasos para la resolución **Ejercicio 1 usando RDD:**

```
1. En la from pyspark.sql import SparkSession
   consola
   de
   PySpark,
   pegamos
   el
   siguiente
   código y
   damos
   enter

# Iniciar la sesión de Spark
spark =
SparkSession.builder.appName("RDD_PagoPorMetodo").getOrCreate()
sc = spark.sparkContext # Contexto de Spark

# Crear un RDD con los datos
data = [
    ("Alice", "Tarjeta de crédito", 100),
    ("Alice", "Tarjeta de crédito", 150),
    ("Alice", "Bizum", 200),
    ("Bob", "Tarjeta de crédito", 201),
    ("Luis", "Bizum", 300)
]
rdd = sc.parallelize(data)

# Filtrar solo los pagos con "Tarjeta de crédito"
rdd_filtered = rdd.filter(lambda x: x[1] == "Tarjeta de crédito")

# Mapear (persona, dinero_gastado) y reducir sumando por persona
rdd_summed = rdd_filtered.map(lambda x: (x[0],
x[2])).reduceByKey(lambda x, y: x + y)

# Obtener todas las personas con gasto 0 si no usaron tarjeta
personas_rdd = rdd.map(lambda x: (x[0], 0)).distinct()
final_rdd =
personas_rdd.leftOuterJoin(rdd_summed).mapValues(lambda x: x[1]
if x[1] else 0)

# Mostrar resultados
for persona, gasto in final_rdd.collect():
    print(f"{persona}: {gasto}")
```

2. Código y Salida en consola

```
>>> # Crear un RDD con los datos
>>> data = [
...     ("Alice", "Tarjeta de crédito", 100),
...     ("Alice", "Tarjeta de crédito", 150),
...     ("Alice", "Bizum", 200),
...     ("Bob", "Tarjeta de crédito", 201),
...     ("Luis", "Bizum", 300)
... ]
>>>
>>> rdd = spark.sparkContext.parallelize(data)
>>>
>>> # Filtrar solo los pagos con "Tarjeta de crédito"
>>> rdd_filtered = rdd.filter(lambda x: x[1] == "Tarjeta de crédito")
>>>
>>> # Mapear a (persona, dinero gastado) y reducir sumando por persona
>>> rdd_summed = rdd_filtered.map(lambda x: (x[0], x[2])).reduceByKey(lambda x,
y: x + y)
>>>
>>> # Obtener todas las personas con gasto 0 si no usaron tarjeta
>>> personas_rdd = rdd.map(lambda x: (x[0], 0)).distinct()
>>> final_rdd = personas_rdd.leftOuterJoin(rdd_summed).mapValues(lambda x: x[1]
if x[1] else 0)
>>>
>>> # Mostrar resultados
>>> for persona, gasto in final_rdd.collect():
...     print(f"{persona};{gasto}")
...
Alice;250
Bob;201
Luis;0
```

3. Para volver a ver la salida tenemos dos formas

Forma 1: `print(final_rdd.collect())`

```
>>> print(final_rdd.collect())
[('Alice', 250), ('Bob', 201), ('Luis', 0)]
```

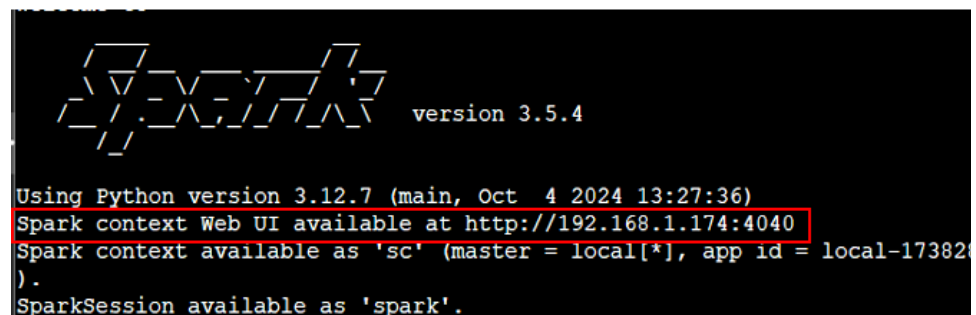
Si quieres ver la salida en un formato más claro dentro de la consola, usa:

Forma 2: `final_rdd.foreach(lambda x: print(f"{x[0]};{x[1]}"))`

```
>>> final_rdd.foreach(lambda x: print(f"{x[0]};{x[1]}"))
Alice;250
Bob;201
Luis;0
```

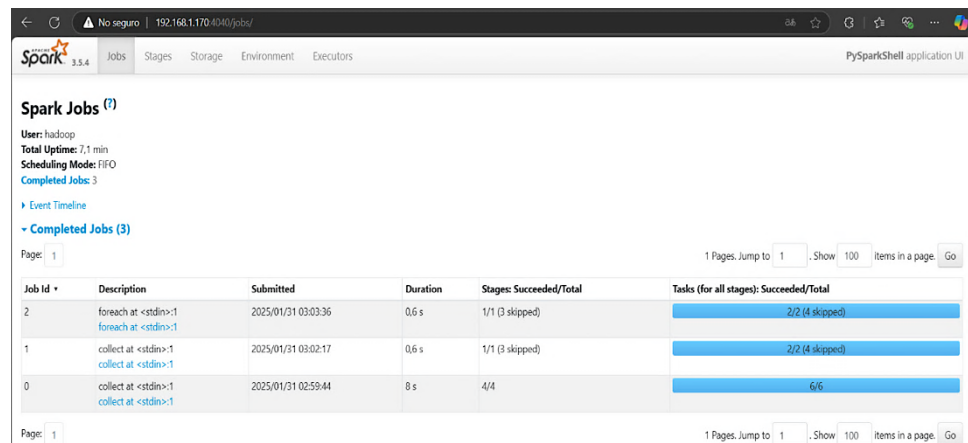
4. Para ver las aplicaciones de Spark que hemos ejecutado podemos usar la interfaz web de PySpark

Al iniciar pyspark nos arroja la ip y puerto con la cual ingresar a la interfaz



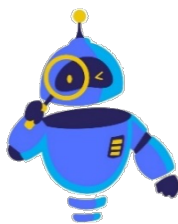
```
version 3.5.4

Using Python version 3.12.7 (main, Oct 4 2024 13:27:36)
Spark context Web UI available at http://192.168.1.174:4040
Spark context available as 'sc' (master = local[*], app id = local-17382
).
SparkSession available as 'spark'.
```



The screenshot shows the Databricks Spark Jobs interface. At the top, it says 'Spark 3.5.4' and 'PySparkShell application UI'. Below that, it shows 'Spark Jobs (7)' with details: 'User: hadoop', 'Total Uptime: 7,1 min', 'Scheduling Mode: FIFO', and 'Completed Jobs: 3'. There is a link to 'Event Timeline' and a section for 'Completed Jobs (3)'. A table lists the jobs with columns: Job Id, Description, Submitted, Duration, Stages: Succeeded/Total, and Tasks (for all stages): Succeeded/Total. The table shows three jobs, all of which are completed.

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	foreach at <stdin>-1 foreach at <stdin>-1	2025/01/31 03:03:36	0,6 s	1/1 (3 skipped)	2/2 (4 skipped)
1	collect at <stdin>-1 collect at <stdin>-1	2025/01/31 03:02:17	0,6 s	1/1 (3 skipped)	2/2 (4 skipped)
0	collect at <stdin>-1 collect at <stdin>-1	2025/01/31 02:59:44	8 s	4/4	6/6



Funcionamiento del código: Procesa un conjunto de datos de pagos realizados por distintas personas con diferentes métodos de pago. Primero, crea un RDD con los datos de pagos y luego filtra solo aquellos realizados con "Tarjeta de crédito". Después, mapea los datos para extraer el nombre de la persona y el monto gastado, sumando el total gastado por cada persona que usó tarjeta de crédito. Luego, se genera un RDD con todas las personas y un gasto inicial de 0, asegurando que también se incluyan quienes no usaron tarjeta. Se realiza una *left outer join* con los datos filtrados y se reemplazan valores *None* por 0. Finalmente, se imprimen los nombres de las personas junto con el monto total gastado con tarjeta de crédito.

Pasos para resolver el **Ejercicio 1 usando DataFrame:**

- En la consola de PySpark, pegamos el siguiente código y damos enter


```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, sum

# Iniciar la sesión de Spark
spark = SparkSession.builder.appName("PagoPorMetodo").getOrCreate()

# Crear los datos en formato (persona, método de pago, dinero gastado)
data = [("Alice", "Tarjeta de crédito", 100),
        ("Alice", "Tarjeta de crédito", 150),
        ("Alice", "Bizum", 200),
        ("Bob", "Tarjeta de crédito", 201),
        ("Luis", "Bizum", 300)]

# Definir columnas
columns = ["persona", "catego_pago", "dinero_gastado"]

# Crear DataFrame
df = spark.createDataFrame(data, columns)

# Filtrar solo pagos con Tarjeta de Crédito y agrupar por persona sumando el gasto
```



```

result_df = (df.filter(col("catego_pago") == "Tarjeta de crédito")
              .groupBy("persona")
              .agg(sum("dinero_gastado").alias("gastoconTDC")))

# Obtener todas las personas con gasto 0 si no usaron tarjeta
personas_df = df.select("persona").distinct()
final_df = personas_df.join(result_df, "persona",
                             "left").fillna(0)

# Mostrar resultados
final_df.show()

```

2. Código y
Salida en
consola

```

>>> from pyspark.sql import SparkSession
>>> from pyspark.sql.functions import col, sum
>>>
>>> # Crear los datos en formato (persona, método de pago, dinero gastado)
>>> data = [("Alice", "Tarjeta de crédito", 100),
...         ("Alice", "Tarjeta de crédito", 150),
...         ("Alice", "Bizum", 200),
...         ("Bob", "Tarjeta de crédito", 201),
...         ("Luis", "Bizum", 300)]
>>>
>>> # Definir columnas
>>> columns = ["persona", "metodo_pago", "dinero_gastado"]
>>>
>>> # Crear DataFrame
>>> df = spark.createDataFrame(data, columns)
>>>
>>> # Filtrar solo pagos con Tarjeta de Crédito y agrupar por persona sumando el
gasto
>>> result_df = (df.filter(col("metodo_pago") == "Tarjeta de crédito")
...              .groupBy("persona")
...              .agg(sum("dinero_gastado").alias("gastoconTDC")))
>>>
>>> # Obtener todas las personas con gasto 0 si no usaron tarjeta
>>> personas_df = df.select("persona").distinct()
>>> final_df = personas_df.join(result_df, "persona", "left").fillna(0)
>>>
>>> # Mostrar resultados
>>> final_df.show()
[Stage 12:>
(0 + 1) / 1]
+-----+-----+
|persona|gastoconTDC|
+-----+-----+
|   Bob|         201|
|  Alice|         250|
|   Luis|           0|
+-----+-----+

```

3. Para volver a ver la salida: `print(final_df.show())`

volver a
ver la
salida
colocam
os el
siguiente
código

```

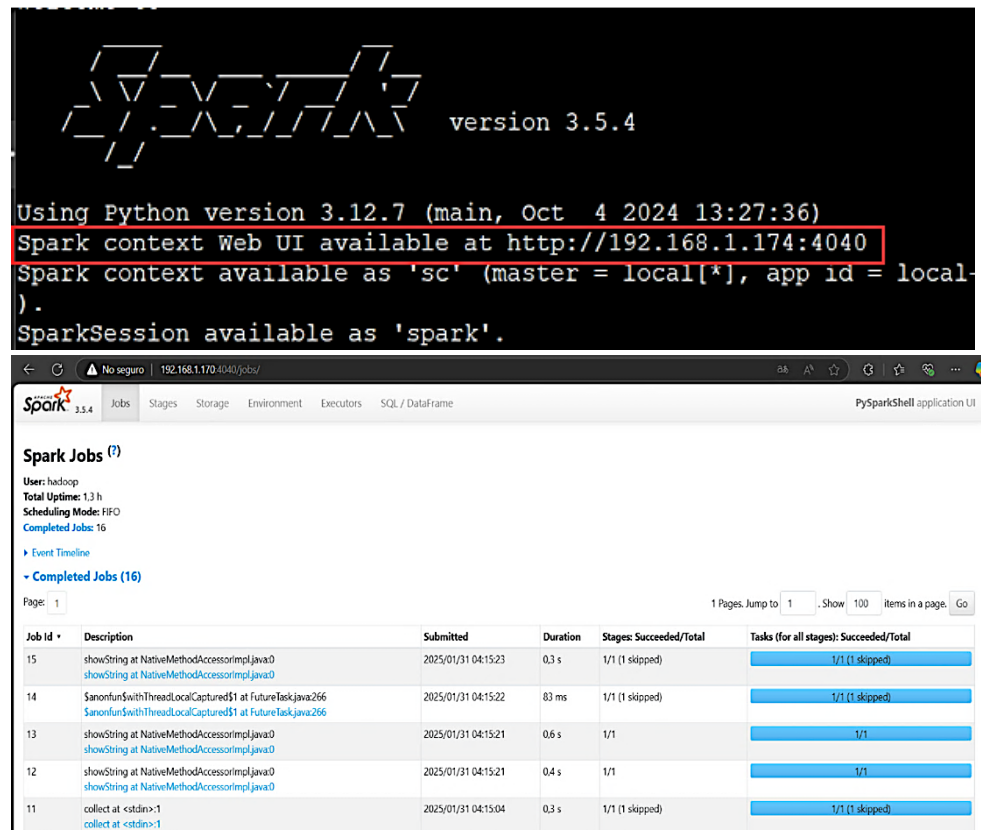
>>> print(final_df.show())
+-----+-----+
|persona|gastoconTDC|
+-----+-----+
|   Bob|         201|
|  Alice|         250|
|   Luis|           0|
+-----+-----+

```

4. Para ver las aplicaciones de spark

Al iniciar pyspark nos arroja la ip y puerto con la cual ingresar a la interfaz

que
hemos
ejecutado
o
podemos
usar la
interfaz
web de
PySpark



version 3.5.4

Using Python version 3.12.7 (main, Oct 4 2024 13:27:36)

Spark context Web UI available at <http://192.168.1.174:4040>

Spark context available as 'sc' (master = local[*], app id = local-...)

SparkSession available as 'spark'.

Spark Jobs (16)

User: hadoop
Total Uptime: 1.3 h
Scheduling Mode: FIFO
Completed Jobs: 16

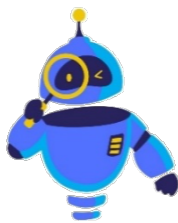
Event Timeline

Completed Jobs (16)

Pages: 1

1 Pages. Jump to 1. Show 100 items in a page. Go

Job ID	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
15	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/31 04:15:23	0.3 s	1/1 (1 skipped)	1/1 (1 skipped)
14	\$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266 \$anonfun\$withThreadLocalCaptured\$1 at FutureTask.java:266	2025/01/31 04:15:22	83 ms	1/1 (1 skipped)	1/1 (1 skipped)
13	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/31 04:15:21	0.6 s	1/1	1/1
12	showString at NativeMethodAccessorImpl.java:0 showString at NativeMethodAccessorImpl.java:0	2025/01/31 04:15:21	0.4 s	1/1	1/1
11	collect at <stdin>:1 collect at <stdin>:1	2025/01/31 04:15:04	0.3 s	1/1 (1 skipped)	1/1 (1 skipped)



Funcionamiento del código: Utiliza DataFrames para procesar datos de pagos. Primero, crea un DataFrame con información sobre personas, métodos de pago y montos gastados. Luego, filtra los pagos realizados con "Tarjeta de crédito" y agrupa por persona, sumando el total gastado con este método. Después, extrae la lista de todas las personas y realiza una *left join* con los datos de gastos, asegurando que quienes no usaron tarjeta de crédito aparezcan con un gasto de 0. Finalmente, muestra el DataFrame resultante con los nombres de las personas y su gasto total con tarjeta de crédito.

Ejercicio 2. Calcular el total de ventas y ganancias por categoría de producto

En base al DataSet de "Global_SuperStore", calcular el total de ventas y ganancias por categoría de producto; también el promedio de ventas y ganancias.

DataSet: [Global_SuperStore.csv](#)



Nota: Antes de realizar los siguientes ejercicios primero debemos tener la data en el clúster, para el traspaso de archivos desde Windows al clúster podemos hacerlo de una manera fácil con el siguiente comando en el CMD:

```
scp -o StrictHostKeyChecking=no "ruta al documento"
hadoop@IpDelCluster:/home/hadoop/
```

Luego nos pedirá la contraseña de nuestro usuario **Hadoop**, una vez ingresada el proceso de traspaso está concluido

```
C:\Users\Steven>scp -o StrictHostKeyChecking=no "C:/Users/Steven/Downloads/Inteligencia de Negocios/Global_Superstore.csv" hadoop@192.168.1.170:/home/hadoop/
Warning: Permanently added '192.168.1.170' (ED25519) to the list of known hosts.
hadoop@192.168.1.170's password:
Global_Superstore.csv 100% 11MB 27.3MB/s 00:00
```

Podemos revisar el archivo en el clúster con **ls -l**

```
-rw-rw-r-- 1 hadoop hadoop 12012342 feb 3 00:31 Global_Superstore.csv
drwxr-xr-x 11 hadoop hadoop 4096 dic 16 03:56 hadoop
-rw-rw-r-- 1 hadoop hadoop 730107476 jun 25 2023 hadoop-3.3.6.tar.gz
drwxrwxr-x 3 hadoop hadoop 4096 dic 16 03:36 hadoopdata
-rwxrwxr-x 1 hadoop hadoop 263 dic 23 04:35 mapperMaxTemp.py
-rw-rw-r-- 1 hadoop hadoop 151 dic 23 03:32 medidas.txt
-rwxrwxr-x 1 hadoop hadoop 983 dic 23 04:36 reducerMaxTemp.py
-rw-rw-r-- 1 hadoop hadoop 400879762 dic 17 05:16 spark-3.5.4-bin-hadoop3.tgz
hadoop@bigdata:~$
```

A continuación, se presenta la solución de dos maneras: **1) mediante de RDD**
2) mediante DataFrame de Apache Spark

Pasos para resolver el **Ejercicio 2 usando RDD**:

```
1. En la # Iniciar la sesión de Spark
   consola from pyspark.sql import SparkSession
   de spark =
   PySpark, SparkSession.builder.appName("GlobalSuperstoreRDD").getOrCreate()
   pegamos sc = spark.sparkContext # Contexto de Spark
   el file_path = "/home/hadoop/Global_Superstore.csv"
   siguiente
   código y # Cargar archivo como RDD
   damos rdd = sc.textFile(file_path)
   enter. cabecera = rdd.first()
          data_rdd = rdd.filter(lambda row: row != cabecera).map(lambda row:
                               row.split(","))

          # Filtrar filas con datos numéricos válidos
          def es_fila_valida(row):
              try:
                  float(row[18]) # Intentar convertir Ventas a float
                  float(row[21]) # Intentar convertir Ganancia a float
                  return True
              except ValueError:
                  return False

          data_rdd = data_rdd.filter(es_fila_valida)

          # Mostrar primeras filas para verificar
```

```

print("Primeras 5 filas de ventas y ganancias:")
print(data_rdd.map(lambda row: (row[15], row[18],
row[21])).take(5))

# Total y Promedio de Ventas y Ganancias por Categoría (RDD)

ventas_y_ganancias_por_categoria = (
    data_rdd.map(lambda row: (row[15], (float(row[18]),
float(row[21]), 1))) # (Categoría, (Ventas, Ganancia, 1))
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1], x[2] +
y[2])) # Sumar ventas, ganancias y contar registros
)

# Calcular promedio
promedios_por_categoria =
ventas_y_ganancias_por_categoria.mapValues(
    lambda x: (x[0], x[1], x[0] / x[2], x[1] / x[2]) # (Total
Ventas, Total Ganancia, Promedio Ventas, Promedio Ganancia)
)

# Mostrar resultados
print("Total y Promedio de Ventas y Ganancias por Categoría:")
for categoria, valores in promedios_por_categoria.collect():
    print(f"{categoria}: Total Ventas={valores[0]}, Total
Ganancia={valores[1]}, Promedio Ventas={valores[2]:.2f}, Promedio
Ganancia={valores[3]:.2f}")

# Finalizar sesión de Spark
spark.stop()

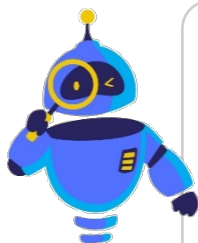
```

2. Salida en consola.

```

Technology: Total Ventas=4742630.562080022, Total Ganancia=663369.9891799991, Pr
omedio Ventas=467.95, Promedio Ganancia=65.45
Furniture: Total Ventas=4110874.18629997, Total Ganancia=285204.72380000126, Pro
medio Ventas=416.25, Promedio Ganancia=28.88
Office Supplies: Total Ventas=3787070.2254999117, Total Ganancia=518473.83430000
057, Promedio Ventas=121.10, Promedio Ganancia=16.58
>>> # Finalizar sesión de Spark
>>> spark.stop()

```



Funcionamiento del código: Carga un archivo CSV llamado *Global_Superstore.csv* como un RDD y analiza las ventas y ganancias por categoría de productos. Primero, inicia una sesión de Spark y lee el archivo, eliminando la cabecera y dividiendo cada fila en columnas. Luego, filtra las filas con datos numéricos válidos en las columnas de ventas y ganancias. Después, agrupa los datos por categoría y calcula el total de ventas y ganancias, así como sus promedios. Finalmente, imprime los resultados y cierra la sesión de Spark.

Pasos para resolver el **Ejercicio 2 usando DataFrames**:

- En la consola de PySpark, pegamos el siguiente código y damos enter


```
# Iniciar la sesión de Spark
from pyspark.sql import SparkSession
from pyspark.sql.functions import avg, sum

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

file_path = "/home/hadoop/Global_Superstore.csv"

# Cargar archivo como DataFrame
df = spark.read.csv(file_path, header=True, inferSchema=True)

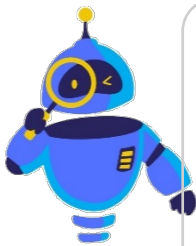
# Mostrar estructura del DataFrame
df.printSchema()

# Total y Promedio de Ventas y Ganancias por Categoría
(DataFrame)
df.groupBy("Category").agg(
    sum("Sales").alias("Total Sales"),
    sum("Profit").alias("Total Profit"),
    avg("Sales").alias("Average Sales"),
    avg("Profit").alias("Average Profit")
).show()

# Finalizar sesión de Spark
spark.stop()
```

- Salida en consola.

Category	Total Sales	Total Profit	Average Sales	Average Profit
Office Supplies	3787070.2254999117	518473.83430000057	121.09711973587157	16.578960582611217
Furniture	4110874.18629997	285204.72380000126	416.24890505264983	28.878566605913452
Technology	4744557.498080019	663778.7331799992	467.85893877132617	65.45495840449651



Funcionamiento del código: Utiliza DataFrames para analizar un archivo CSV llamado *Global_Superstore.csv*. Primero, inicia una sesión de Spark y carga el archivo en un DataFrame, infiriendo automáticamente los tipos de datos y considerando la primera fila como cabecera. Luego, muestra la estructura del DataFrame para verificar las columnas y tipos de datos. Después, agrupa los datos por categoría de productos y calcula el total y el promedio de ventas y ganancias para cada categoría. Finalmente, muestra los resultados y cierra la sesión de Spark. Es una forma más estructurada y eficiente de procesar los datos en comparación con los RDDs

Ejercicio 3. Cantidad Total de Pedidos y Promedio de Productos Vendidos por Categoría

En base al DataSet de "Global_SuperStore", Cantidad Total de Pedidos y Promedio de Productos Vendidos por Categoría, calcular:

- Cantidad total de pedidos por categoría.
- Cantidad total de productos vendidos por categoría.
- Promedio de productos vendidos por pedido en cada categoría.

Pasos para resolver el **Ejercicio 3 usando RDD**:

1. En la consola de PySpark, pegamos el siguiente código y damos Enter

```
# Iniciar la sesión de Spark
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("GlobalSuperstoreRDD").getOrCreate()
sc = spark.sparkContext # Contexto de Spark

file_path = "/home/hadoop/Global_Superstore.csv"

# Cargar archivo como RDD
rdd = sc.textFile(file_path)
cabecera = rdd.first()
data_rdd = rdd.filter(lambda row: row != cabecera).map(lambda row:
row.split(","))

# Filtrar filas con datos numéricos válidos
def es_fila_valida(row):
    try:
        int(row[19]) # Intentar convertir Quantity a entero
        return True
    except ValueError:
        return False

data_rdd = data_rdd.filter(es_fila_valida)

# Mostrar primeras filas para verificar
print("Primeras 5 filas de cantidad de productos vendidos:")
print(data_rdd.map(lambda row: (row[15], row[19])).take(5))

# Total de pedidos y productos vendidos por categoría
pedidos_y_productos_por_categoria = (
    data_rdd.map(lambda row: (row[15], (int(row[19]), 1))) # (Categoría,
(Cantidad, 1))
    .reduceByKey(lambda x, y: (x[0] + y[0], x[1] + y[1])) # Sumar cantidad y
    contar pedidos
)

# Calcular promedio de productos por pedido
promedio_productos_por_pedido = pedidos_y_productos_por_categoria.mapValues(
```

```

    lambda x: (x[0], x[1], x[0] / x[1]) # (Total Productos, Total Pedidos,
Promedio Productos por Pedido)
)

# Mostrar resultados
print("Total de Pedidos y Promedio de Productos Vendidos por Categoría:")
for categoria, valores in promedio_productos_por_pedido.collect():
    print(f"{categoria}: Total Productos={valores[0]}, Total
Pedidos={valores[1]}, Promedio Productos por Pedido={valores[2]:.2f}")

# Finalizar sesión de Spark
spark.stop()

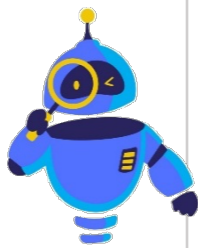
```

2. Salida en consola.

```

Technology: Total Productos=35144, Total Pedidos=10135, Promedio Productos por Pedido=3.47
Furniture: Total Productos=34954, Total Pedidos=9876, Promedio Productos por Pedido=3.54
Office Supplies: Total Productos=108182, Total Pedidos=31273, Promedio Productos por Pedido=3.46
>>> # Finalizar sesión de Spark
>>> spark.stop()

```



Funcionamiento del código: Este código en PySpark carga un archivo CSV llamado *Global_Superstore.csv* como un RDD y analiza la cantidad de productos vendidos por categoría. Primero, inicia una sesión de Spark, lee el archivo y elimina la cabecera, dividiendo cada fila en columnas. Luego, filtra las filas que contienen datos numéricos válidos en la columna de cantidad de productos (*Quantity*). A continuación, agrupa los datos por categoría, sumando la cantidad total de productos vendidos y contando el número de pedidos. Finalmente, calcula el promedio de productos vendidos por pedido en cada categoría, muestra los resultados y cierra la sesión de Spark.

Pasos para resolver el **Ejercicio 3 usando DataFrames**:

1. En la consola de PySpark, pegamos el siguiente código y damos enter

```

# Iniciar la sesión de Spark
from pyspark.sql import SparkSession
from pyspark.sql.functions import count, sum, avg

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

file_path = "/home/hadoop/Global_Superstore.csv"

# Cargar archivo como DataFrame
df = spark.read.csv(file_path, header=True, inferSchema=True)

# Mostrar estructura del DataFrame

```



```
df.printSchema()

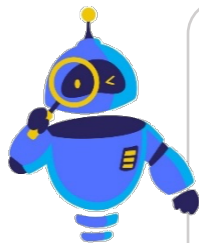
# Total de pedidos y productos vendidos por categoría
df.groupBy("Category").agg(
    sum("Quantity").alias("Total Products Sold"),
    count("Order ID").alias("Total Orders"),
    (sum("Quantity") / count("Order ID")).alias("Avg Products per Order")
).show()

# Finalizar sesión de Spark
spark.stop()
```

2. Salida en consola:

```
+-----+-----+-----+-----+
|      Category|Total Products Sold|Total Orders|Avg Products per Order|
+-----+-----+-----+-----+
|Office Supplies|          108182|        31273|      3.45927797141304|
|      Furniture|          34954|         9876|      3.539287160793844|
|    Technology|          35176|        10141|      3.4686914505472832|
+-----+-----+-----+-----+

>>>
>>> # Finalizar sesión de Spark
>>> spark.stop()
```



Funcionamiento del código: Este código en PySpark utiliza DataFrames para analizar el número de productos vendidos y pedidos en el archivo `Global_Superstore.csv`. Primero, inicia una sesión de Spark, carga el archivo en un DataFrame infiriendo los tipos de datos y considerando la primera fila como cabecera. Luego, muestra la estructura del DataFrame para verificar las columnas. Después, agrupa los datos por categoría de productos y calcula el total de productos vendidos, el número total de pedidos y el promedio de productos vendidos por pedido. Finalmente, muestra los resultados y cierra la sesión de Spark.

Resumen del capítulo. Se presentó un panorama general de Apache Spark resaltando que es una poderosa herramienta que nos ayuda en el procesamiento de grandes volúmenes de datos, se destaca por su rapidez gracias a su procesamiento en memoria. Mediante su arquitectura distribuida nos permite manejar los datos de manera eficiente en clústeres, superando a Hadoop MapReduce al evitar escrituras repetitivas en el disco. Se usa ampliamente en análisis de datos, machine learning y el procesamiento en

tiempo real, también es compatible con lenguajes como Python, Scala Java R y SQL.

Entre los componentes claves de Spark se encuentran los RDDs (estructuras resilientes y distribuidas), los DataFrames (optimizados y organizados en columnas) Y módulos especializados como son Spark SQL, Spark Streaming, MLlib y GraphX. Spark Divide las operaciones en transformaciones, el cual genera nuevos conjuntos de datos sin afectar a los datos originales y acciones, los cuales ejecutan cálculos y devuelven un resultado.

En este capítulo se presentaron ejercicios prácticos usando RDDs y DataFrames para calcular gastos, ventas y pedidos en un determinado DataSet. Gracias a la flexibilidad y rendimiento de Spark se ha convertido en una de las principales herramientas esenciales en el mundo de la Big Data.

Referencias Bibliográficas

- Apache Spark: Introducción, qué es y cómo funciona | ESIC.* (2018).
https://www.esic.edu/rethink/tecnologia/apache-spark-introduccion-que-es-y-como-funciona?utm_source=chatgpt.com
- Apache Spark: Qué es, Cómo funciona y Beneficios.* (2024). EBIS Education.
<https://www.ebiseducation.com/apache-spark-que-es-como-funciona-y-beneficios>
- Buitrago, B. (2020, octubre 18). ¿Qué hay detrás del procesamiento de Apache Spark? II. *iWannaBeDataDriven*.
<https://medium.com/iwannabedatadriven/qu%C3%A9-hay-detr%C3%A1s-del-procesamiento-de-apache-spark-ii-474402939ca5>
- Chauhan, V., & Sharma, M. (2016). "A REVIEW: MAPREDUCE AND SPARK FOR BIG DATA ANALYTICS.
- Damji, J. S., Wenig, B., Das, T., & Lee, D. (2020). *Learning Spark: Lightning-Fast Data Analytics*.
- Fajardo, H. M. (2022). *Estudio Comparativo entre Apache Spark y Apache Flink en el procesamiento de streaming en entornos Big Data*.
https://sedici.unlp.edu.ar/bitstream/handle/10915/155135/Documento_completo.pdf?sequence=1&isAllowed=y
- Gowda, D. (2024). *Apache Spark for Machine Learning* (1.^a ed.). Packt Publishing.
<http://gen.lib.rus.ec/book/index.php?md5=B40816015A9835648EAC2B186DAC82D1>
- Luu, H. (2021). *Beginning Apache Spark 3: With DataFrame, Spark SQL, Structured Streaming, and Spark Machine Learning Library* (2.^a ed.). Apress.

<http://gen.lib.rus.ec/book/index.php?md5=A6ED0A122181730EE28E88A5B0C5DEFE>

Macias, M., Gómez, M., Tous, R., & Torres, J. (2016). *Introduccion a Apache Spark*. Editorial UOC.

https://books.google.com.ec/books/about/Introducci%C3%B3n_a_Apache_Spark.html?id=JqunDAAAQBAJ&redir_esc=y

Penchikala, S. (2018). *Big Data Processing with Apache Spark* (Primera). Lulu.com.

<https://books.google.com.ec/books?id=eodaDwAAQBAJ&printsec>

Prado, V. A. J. (2020). *Apache spark*.

https://www.academia.edu/42202911/Apache_spark

Rajpurohit, A. M., Kumar, P., Kumar, R. R., & Kumar, R. (2024). A Review on Apache Spark. *SSRN Electronic Journal*.

<https://doi.org/10.2139/ssrn.4492445>



Bertha Eugenia Mazón Olivo

<https://orcid.org/0000-0002-2749-8561>

bmazon@utmachala.edu.ec

Es docente investigadora titular de la Universidad Técnica de Machala - Ecuador. Tiene el título de Doctora en Tecnologías de la Información y las Comunicaciones por la Universidad de A Coruña de España. Es Magíster en Informática Aplicada e Ingeniera en Sistemas por la Escuela Superior Politécnica de Chimborazo. Cuenta con un Diplomado Internacional en Ingeniería de Datos por la Universidad Nacional de Ucayali de Perú. Actualmente, cursa una Maestría en Big Data y Data Science en la Universidad Internacional de Valencia, España. Es docente en varios programas de postgrado. Es miembro del grupo de investigación AutoMathTIC y ha participado en varios proyectos de investigación; en la actualidad es directora del proyecto "Adopción de las tecnologías de Inteligencia Artificial e Internet de las Cosas en el sector Agropecuario de la provincia de El Oro". Formó parte del equipo de diseño curricular para la creación de la carrera de Ciencia de Datos e Inteligencia Artificial en la UTMACH. Sus intereses son Internet de las Cosas, Ciencia de Datos, Big Data, Machine Learning y Deep Learning. Cuenta con varias publicaciones de artículos en revistas indexadas, libros y capítulos de libros.



Wilmer Braulio Rivas Asanza

<https://orcid.org/0000-0002-2239-3664>

wrivas@utmachala.edu.ec

Es Ingeniero en Informática por la Universidad Católica de Cuenca, con una formación académica que incluye un Diplomado en Auditoría de TI por la Escuela Superior Politécnica del Litoral (ESPOL), un Diplomado en Inteligencia Artificial, una Maestría en Docencia y Gestión en la Educación Superior por la Universidad Estatal de Guayaquil y una Maestría en Gestión Estratégica de las Tecnologías de la Información por la Universidad Estatal de Cuenca. Además, es Doctor en Tecnologías de la Información y las Comunicaciones por la Universidad de A Coruña, España. Cuenta con experiencia profesional tanto en el sector público como en el privado. Es profesor titular en la Universidad Técnica de Machala durante aproximadamente 14 años, impartiendo asignaturas como Sistemas Operativos, Gestión de Centros de Cómputo e Inteligencia Artificial. Actualmente, cuenta con varias publicaciones y libros indexados que reflejan su trayectoria académica y profesional.



Joffre Jeorwin Cartuche Calva

<https://orcid.org/0000-0002-1633-2291>

jcartuche@utmachala.edu.ec

Ecuatoriano, Analista e Ingeniero en Sistemas Informáticos por la ESPOCH, con una Maestría en Ingeniería de Software por la ESPE. Candidato a Doctor en Dirección de Proyectos por la Universidad de Investigación e Innovación de México. Docente titular agregado en la Universidad Técnica de Machala, donde imparte asignaturas de pregrado como Programación Orientada a Objetos, Programación Avanzada, Programación Móvil con Despliegues en la Nube, y en posgrado Gestión de Proyectos de Software. Cuenta con experiencia en desarrollo de software, soluciones móviles e integradas, y gestión tecnológica. Sus intereses incluyen arquitectura de software, metodologías ágiles, inteligencia artificial aplicada, microservicios, contenedores, computación en la nube (AWS y Azure) y despliegue de datos. Su enfoque está orientado a la innovación tecnológica, el desarrollo de software escalable y la gestión eficiente de proyectos de TI.



Ashley Paulina Aguilar Serrano

<https://orcid.org/0009-0003-8468-7793>

aaguilar31@utmachala.edu.ec

Ingeniera de Tecnologías de la Información por la Universidad Técnica de Machala. Su interés se enfoca en Big Data, Ciencia de Datos y Análisis de Datos, para explorar cómo el procesamiento y la interpretación de grandes volúmenes de datos pueden generar valor en distintos sectores empresariales. Además, busca innovar en la gestión y aprovechamiento de datos para la toma de decisiones estratégicas y el desarrollo de soluciones tecnológicas.



Jean Carlos Ávila Villaprado

<https://orcid.org/0009-0002-4091-6032>

javila3@utmachala.edu.ec

Ingeniero de Tecnologías de la Información por la Universidad Técnica de Machala. A lo largo de su formación, ha desarrollado habilidades en programación de computadoras, bases de datos y soporte técnico, participando en proyectos locales que le han permitido consolidar sus conocimientos y mejorar su capacidad de resolución de problemas. Apasionado por la tecnología, se mantiene en constante aprendizaje, con un enfoque especial en Big Data y la Inteligencia Artificial.

ISBN: 978-9942-33-954-6



9 789942 339546

Compás
capacitación e investigación