ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

IMPLEMENTACIÓN DE COMPUTACIÓN EN LA NUBE EN EL LABORATORIO ADA DE LA FACULTAD DE SISTEMAS PARA ANALÍTICA DE DATOS

IMPLEMENTACIÓN DE UN LAGO DE DATOS EN LOS SERVIDORES DEL LABORATORIO ADA

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

HERNÁN ALEXIS CHUGÁ PORTILLA

hernan.chuga@epn.edu.ec

DIRECTOR: PhD. MARTÍNEZ MOSQUERA SILVIA DIANA

diana.martinez@epn.edu.ec

DMQ, agosto 2023

CERTIFICACIONES

Yo, HERNÁN ALEXIS CHUGÁ PORTILLA declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

HERNÁN ALEXIS CHUGÁ PORTILLA

Certifico que el presente trabajo de integración curricular fue desarrollado por HERNÁN ALEXIS CHUGÁ PORTILLA, bajo mi supervisión.

_____

PhD. MARTÍNEZ MOSQUERA SILVIA DIANA

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

HERNÁN ALEXIS CHUGÁ PORTILLA

PhD. MARTÍNEZ MOSQUERA SILVIA DIANA

AARON ISMAEL BRAVO MENDOZA

ADRIAN OCTAVIO LAJE BARRAGÁN

DEDICATORIA

Dedico este trabajo a mi familia, por su amor y apoyo incondicional, y por ser mi mayor motivación en cada paso que doy. En especial a mis padres por sus palabras de aliento, sus consejos sabios y su amor inquebrantable que han sido mi motor en los momentos difíciles y me han dado la fuerza para seguir adelante. Cada paso que doy, cada meta que alcanzo, es también un tributo a ellos y a todo lo que han hecho por mí.

A mis profesores y mentores, por su sabiduría y guía, y por haberme inspirado a superar mis límites y seguir aprendiendo. A mis amigos, por compartir risas y momentos inolvidables, y por estar siempre en los buenos y malos momentos. A mis compañeros de estudio, por compartir el esfuerzo y la pasión por la investigación y el conocimiento. Y a todas las personas que han sido parte fundamental en mi camino académico y profesional.

AGRADECIMIENTO

Quiero expresar mi más profundo agradecimiento a la Escuela Politécnica Nacional por brindarme la oportunidad de realizar el presente proyecto de integración curricular y por proporcionarme un entorno académico enriquecedor que ha sido fundamental para mi formación como profesional.

Agradezco profundamente a la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación por su apoyo financiero y la concesión de la beca que hizo posible realizar mis estudios en la EPN. Su compromiso con la educación y el desarrollo de talento ha sido fundamental en mi formación y crecimiento académico.

Agradezco de manera especial a la PhD. Martínez Mosquera Silvia Diana, mi directora y guía en este proceso. Gracias a su sabiduría, paciencia y dedicación, pude contar con el apoyo necesario para enfrentar los desafíos de este proyecto y alcanzar mis metas académicas.

A mis queridos padres, Hernán Chugá y Erika Portilla, les agradezco infinitamente por ser mis pilares inquebrantables. Su amor, aliento y constante apoyo han sido mi mayor fortaleza en este camino. Sin ustedes, nada de esto hubiera sido posible.

Y no puedo dejar de mencionar a mis amigos, quienes también merecen mi agradecimiento. Gracias por estar siempre presentes, por brindarme su amistad y por ayudarme a crecer tanto en lo personal como en lo social.

A todos ustedes, mi más sincero agradecimiento por formar parte de este importante capítulo en mi vida y por ser parte fundamental en el cumplimiento de este proyecto. Sin duda, sus palabras de aliento y su compañía han sido el combustible que me ha impulsado a alcanzar cada uno de mis objetivos.

ÍNDICE DE CONTENIDO

RESUMEN

En este proyecto, se llevó a cabo el proceso de diseño e implementación de un lago de datos con Hadoop en un clúster con un nodo maestro y cuatro nodos esclavos utilizando contenedores Docker. Se comenzó con una investigación y análisis de los requerimientos y objetivos del proyecto, centrándose en la importancia de la arquitectura y su adaptación a los recursos del servidor de altas prestaciones del laboratorio ADA.

Para lograr una implementación exitosa, se realizó una experimentación previa con diferentes configuraciones y herramientas, lo que permitió entender el funcionamiento de Hadoop y Docker, así como sus interacciones. Se diseñó la arquitectura del lago de datos, incluyendo el NameNode, ResourceManager y los DataNodes coexistiendo en un solo contenedor con sus NodeManagers para mejorar la eficiencia y la utilización de recursos.

Los recursos de memoria RAM, CPU y almacenamiento se asignaron dinámicamente a cada contenedor para adaptarse a las necesidades cambiantes del sistema, resultando en una arquitectura más flexible y escalable. Además, se creó una red interna que conecta todos los contenedores y permite una comunicación efectiva entre los componentes del lago de datos.

El proceso de guardar datos en el lago de datos se realizó mediante la ejecución de scripts de análisis de datos de los usuarios finales del laboratorio ADA, verificando la división en bloques y su replicación en los DataNodes para garantizar tolerancia a fallos y alta disponibilidad. Se comprobó el funcionamiento de la red interna, el NameNode, el ResourceManager y los DataNodes mediante el acceso a sus interfaces web y verificando que todos los contenedores estuvieran correctamente conectados.

El proyectó culminó con la implementación exitosa de un lago de datos, cumpliendo los objetivos y requerimientos planteados, y, cuya arquitectura fue probada y validada por los usuarios finales pertenecientes al laboratorio ADA

PALABRAS CLAVE: Computación en la nube, Lago de datos, Hadoop, Docker, Almacenamiento de datos.

ABSTRACT

In this project, the process of designing and implementing a data lake with Hadoop was carried out in a cluster with one master node and four slave nodes using Docker containers. It began with an investigation and analysis of the requirements and objectives of the project, focusing on the importance of the architecture and its adaptation to the resources of the high-performance server of the ADA laboratory.

To achieve a successful implementation, a previous experimentation with different configurations and tools was carried out, which allowed understanding the operation of Hadoop and Docker, as well as their interactions. The data lake architecture was designed, including the NameNode, ResourceManager and DataNodes coexisting in a single container with their NodeManagers to improve efficiency and resource utilization.

RAM, CPU, and storage resources were dynamically allocated to each container to adapt to changing system needs, resulting in a more flexible and scalable architecture. In addition, an internal network was created that connects all the containers and allows effective communication between the components of the data lake.

The process of saving data in the data lake was carried out by executing data analysis scripts from the end users of the ADA laboratory, verifying the division into blocks and their replication in the DataNodes to guarantee fault tolerance and high availability. The operation of the internal network, the NameNode, the ResourceManager and the DataNodes was verified by accessing their web interfaces and verifying that all the containers were correctly connected.

The project culminated in the successful implementation of a data lake, meeting the objectives and requirements set, and whose architecture was tested and validated by end users belonging to the ADA laboratory.

KEYWORDS: Cloud Computing, Data Lake, Hadoop, Docker, Data Storage.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El proyecto realizado fue un detallado proceso de diseño e implementación de un lago de datos con Hadoop en un clúster de cinco nodos, un nodo maestro y cuatro nodos esclavos, utilizando contenedores Docker. Inicialmente, se plantearon objetivos específicos que buscaban lograr una arquitectura escalable, flexible, con alta disponibilidad y seguridad, adaptada a los recursos del servidor de altas prestaciones del laboratorio ADA, de la Facultad de Sistemas de la Escuela Politécnica Nacional.

La investigación sobre trabajos relacionados permitió conocer las mejores prácticas y enfoques utilizados en proyectos similares, lo que proporcionó una base para el desarrollo del lago de datos.

La identificación de los requerimientos se enfocó en comprender las necesidades y objetivos específicos del laboratorio con respecto al almacenamiento y gestión de datos. Los principales requerimientos identificados fueron la necesidad de adaptabilidad del lago de datos al servidor que se dispone, y que las herramientas usadas en su implementación sean *open source*. Este fue un proceso colaborativo que involucró a la directora del presente proyecto y al coordinador del laboratorio ADA, con el objetivo de garantizar que el lago de datos se adapte de manera óptima a las necesidades del laboratorio y proporcione una visión clara para la selección de herramientas y el diseño de la arquitectura del lago de datos.

La selección de herramientas se basó en la elección de Hadoop y Docker como tecnologías principales para el proyecto, debido a su robustez, escalabilidad y flexibilidad. Asimismo, se consideró la importancia de crear imágenes base de Hadoop y configurar de manera correcta cada contenedor para lograr un lago de datos eficiente.

El diseño de la arquitectura fue una etapa crítica, donde se definieron los componentes del lago de datos, como el NameNode, ResourceManager y los DataNodes coexistiendo en un solo contenedor con sus NodeManagers, lo que permitió optimizar la utilización de recursos y mejorar el rendimiento. La asignación dinámica de memoria RAM, CPU y almacenamiento en disco fue otro aspecto clave para adaptar el sistema a las necesidades cambiantes.

La implementación del lago de datos se llevó a cabo creando y ejecutando los contenedores Docker, lo que permitió tener un ambiente aislado y preparado para manejar grandes volúmenes de datos. Además, se configuró una red interna para una comunicación efectiva entre los componentes del lago de datos.

La evaluación del funcionamiento del lago de datos incluyó comprobar el acceso a las interfaces web del NameNode y ResourceManager, verificando que los DataNodes estuvieran correctamente conectados y funcionando. También se verificó el proceso de guardado y lectura de datos en el lago, asegurando que la replicación de bloques y la distribución de datos se realizaran de manera eficiente.

El proyecto concluyó con éxito, logrando implementar un lago de datos funcional y eficiente, cumpliendo con los objetivos planteados y brindando una solución tecnológica sólida y preparada para manejar grandes volúmenes de datos en el laboratorio ADA.

1.1 Objetivo general

Desarrollar e Implementar un lago de datos en el servidor de altas prestaciones del laboratorio ADA mediante herramientas de código abierto.

1.2 Objetivos específicos

1. Revisar la literatura existente acerca de los lagos de datos y las herramientas usadas en este campo, con énfasis en su utilización para el almacenamiento efectivo de grandes volúmenes de datos.
2. Recopilar los requerimientos, expectativas y necesidades de las partes interesadas en este lago de datos, así como también las restricciones y consideraciones específicas que se deben tomar en cuenta para garantizar una implementación exitosa.
3. Diseñar la arquitectura integral y detallada para el lago de datos, seleccionando tecnologías y herramientas apropiadas para su implementación, con base en los requerimientos identificados por las partes interesadas, y las especificaciones técnicas del servidor de altas prestaciones del laboratorio ADA.
4. Implementar el lago de datos siguiendo la arquitectura establecida previamente de acuerdo a los especificaciones y requerimientos establecidos por el laboratorio ADA.
5. Evaluar los resultados obtenidos tras la implementación del lago de datos, realizando un análisis comparativo entre los objetivos planteados y los logros alcanzados.

1.3 Alcance

El alcance de este proyecto consiste en implementar un lago de datos, que servirá como herramientas para los múltiples trabajos de ciencias de datos. Este lago de datos se desarrollará en un clúster de varios nodos utilizando tecnologías de Hadoop ejecutadas en contenedores Docker. Se realizará una investigación sobre los lagos de datos para establecer una base sólida de conocimientos y mejores prácticas en el ámbito. Se recopilarán los requisitos y expectativas de las partes interesadas para garantizar una implementación exitosa y se diseñará una arquitectura detallada acorde a los requerimientos técnicos y del servidor de altas prestaciones del laboratorio ADA.

La implementación del lago de datos incluirá la creación y administración de contenedores para el NameNode, ResourceManager, DataNodes y NodeManagers. Se realizará una asignación dinámica de recursos para optimizar el uso de la memoria RAM, capacidad de almacenamiento y capacidad de procesamiento en cada contenedor. Asimismo, se configurará una red interna en Docker para facilitar la comunicación entre los nodos y componentes del lago de datos. Se establecerá la replicación de datos para garantizar la disponibilidad y tolerancia a fallos.

La fase de evaluación incluirá pruebas cualitativas de funcionamiento, para comprobar que se haya cumplido con los requerimientos técnicos y que el lago de datos opera correctamente. Además, se proporcionará un manual técnico que detalla el proceso completo de creación y gestión del lago de datos, para asegurar que tanto los usuarios como los administradores puedan aprovechar al máximo todas las capacidades del lago de datos del laboratorio ADA.

1.4 Marco teórico

Computación en la nube

La computación en la nube o *cloud computing*, representa un nuevo enfoque en el campo de la informática. Para numerosos analistas, esta innovación es equiparable en relevancia a la que tuvo Internet en su época. La computación en la nube es la evolución de un conjunto de tecnologías que están cambiando la manera en que las empresas y organizaciones edifican sus infraestructuras de TI. Al igual que ocurrió con la evolución de la Web, desde la Web 2.0 hasta la Web Semántica, la computación en nube no introduce tecnologías completamente nuevas. Más bien, combina tecnologías potentes e innovadoras para crear un nuevo modelo y arquitectura de la Web (Aguilar, 2018).

Por otro lado, desde el punto de vista empresarial la computación en la nube es el enfoque que posibilita acceder, según necesidad, a una red que ofrece una variedad de servicios informáticos ajustables, como infraestructura, aplicaciones y almacenamiento. En los últimos tiempos, este modelo ha brindado a las empresas la capacidad de establecer presencia en la Web o adquirir servicios informáticos a un costo asequible, sin la necesidad de realizar inversiones en la compra de equipos físicos y software (Del Vecchio, Fabián, & Henríquez, 2015).

Es importante mencionar las características principales del *cloud computing*, para ello se toma como referencia al artículo (Mejía, 2011), en el cual se menciona que la computación en la nube es auto reparable, escalable, posee un elevado nivel tanto para la seguridad como para la disponibilidad de la información. Además, para obtener las ventajas de la computación en la nube no es necesario contar con un equipo potente, sino simplemente un dispositivo que tenga conexión a Internet, ya que en ningún momento este dispositivo va a realizar procesos complejos. Por último, no existe la necesidad de que el usuario deba conocer la infraestructura detrás de esta para gozar de sus aplicaciones y servicios.

La arquitectura del *cloud computing* se fundamenta en una separación clara entre el hardware, la plataforma y las aplicaciones (Aguilar, 2018), dando lugar a las siguientes capas:

- **Software como servicio o *Software as a Service (SaaS)*:** Consiste en ofrecer al usuario la capacidad de emplear aplicaciones que operan en la infraestructura de la nube. El usuario no tiene control sobre la infraestructura ni sobre las aplicaciones en sí, a excepción de las personalizaciones o configuraciones permitidas.
- **Plataforma como servicio o *Platform as a Service (PaaS):*** brinda al usuario la posibilidad de implementar sus propias aplicaciones en la infraestructura de la nube, el cual también brinda la plataforma de desarrollo y las herramientas de programación requeridas. En este escenario, el usuario conserva el control sobre la aplicación, aunque no sobre toda la infraestructura subyacente.
- **Infraestructura como servicio o *Infraestructure as a Service (IaaS):*** en este caso el proveedor pone a disposición del usuario recursos como capacidad de almacenamiento, procesamiento y comunicaciones, que pueden emplearse para ejecutar una amplia gama de software, desde sistemas operativos hasta aplicaciones diversas.

Además, la computación en la nube, al ser un paradigma ampliamente reconocido por su escalabilidad y flexibilidad en el procesamiento y almacenamiento de datos, sienta las bases para la evolución de conceptos innovadores como el "lago de datos". Este último, emergiendo como una solución para el manejo y análisis eficiente de grandes conjuntos de datos, se nutre de los principios de la computación en la nube para ofrecer a las organizaciones una infraestructura dinámica que responde a las demandas cambiantes de la ciencia de datos y la investigación.

Lago de datos

En 2010, James Dixon acuñó el concepto de "lago de datos" con el fin de diferenciarlo del enfoque utilizado en Hadoop y los *data marts* o almacenes de datos. Dixon utiliza una metáfora para ilustrar esta distinción: compara el *data mart* con una tienda de agua embotellada y convenientemente empacada para su consumo, mientras que el lago de datos se asemeja a una masa de agua en su estado natural (Mathis, 2017). Esta distinción es importante ya que en un almacén de datos existen políticas regulatorias sobre el formato de los datos y capacidad de almacenamiento, bien definidas. Por el contrario, en un lago de datos, no se cuenta con límites de almacenamiento o restricciones sobre los tipos de los datos, es decir, se pueden almacenar cualquier cantidad de datos y en cualquier formato (Khine & Wang, 2018).

La descripción informal de Dixon deja margen para la interpretación, pero, por lo general un lago de datos, o en inglés, data lake, se define como un modelo eficiente que se enfoca en el almacenamiento de grandes cantidades de datos de diversos tipos, procurando la escalabilidad. Está diseñado para la rápida ingestión de datos sin procesar desde diversas fuentes, y para permitir su exploración, análisis y procesamiento por parte de los usuarios. El lago de datos se apoya en una tecnología de bajo costo y escala masiva. En un lago de datos, se elimina la necesidad de realizar complejas transformaciones, lo que puede reducir los costos de ingesta de datos. En resumen, el concepto fundamental de un lago de datos es que todos los datos generados por la organización se almacenan en una única fuente de datos, preservando su formato original (Karuna & Praseetha, 2020).

Desde el punto de vista de los analistas de datos, un lago de datos es un repositorio único que almacena todos los datos de una organización, desde los datos sin procesar (que son una copia exacta de los datos del sistema de origen) hasta los datos transformados utilizados para diversas tareas. Un lago de datos permite tanto el almacenamiento como el análisis, abarcando informes, visualización, análisis estadístico y aprendizaje automático. En esta forma de almacenamiento se pueden encontrar datos estructurados provenientes

de bases de datos relacionales (tablas con filas y columnas), datos semiestructurados como archivos Comma-Separated Values (CSV), registros, Extensible Markup Language (XML) y JavaScript Object Notation (JSON), datos no estructurados como mensajes de correo electrónico, archivos de documentos y PDF, así como datos en formato binario, como imágenes, archivos de audio y videos. Esto crea un almacén de datos centralizado que puede acomodar todas las formas de datos y que está fácilmente accesible para analistas en toda la organización (Raju, Mital, & Finkelsztein, 2018).

Características de un lago de datos

En lo que se refiere a las características de los lagos de datos, el trabajo (Megdiche, Ravat, & Zhao, 2020) se menciona que los principales atributos de un lago de datos son la capacidad de almacenar datos en su formato nativo, garantizar su ciclo de vida y brindar el acceso de los mismos a diferentes usuarios. Por otro lado, en (Sawadogo & Darmont, 2021) se manifiesta que un lago de datos se caracteriza por poseer herramientas y políticas de gobernanza de datos, contar con una organización lógica y física, y, contar con atributos de escalabilidad en lo que respecta a capacidad de almacenamiento y procesamiento.

Por último, en (Raju, Mital, & Finkelsztein, 2018) se presenta al lago de datos como un modelo de almacenamiento de datos que debe poseer características de Big Data, como el soportar grandes volúmenes y variedad de datos, además de brindar velocidad en su registro y lectura, debe admitir tecnologías de almacenamiento heterogéneas y debe proporcionar un modelo que permite una configuración ágil y adaptativa.

Arquitectura de un lago de datos

Otro aspecto importante a tener en cuenta en un lago de datos es su arquitectura, por lo cual se han tomado como referencia a trabajos que se han enfocado en el estudio de este campo, tales como, (Mehmood, y otros, 2019), (Balboa, 2022) y (Giebler, y otros, 2021), en los cuales se menciona que la arquitectura de los lagos de dato se compone, generalmente, por las capas de recopilación, ingestión, almacenamiento, análisis y visualización de los datos. A continuación, se detalla en la figura 1.1, cada una de las capas mencionadas.

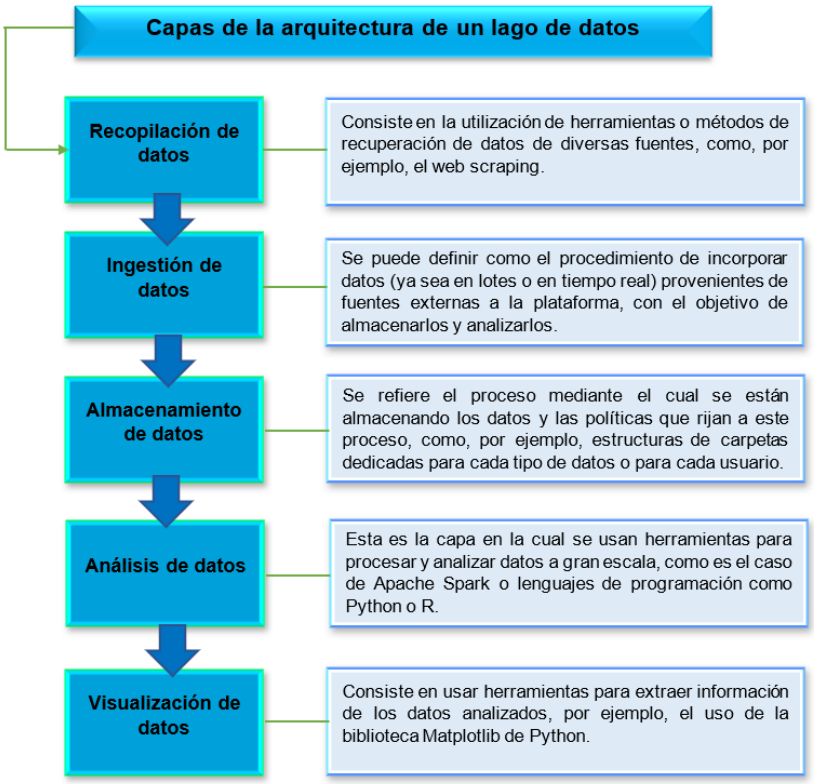

Figura 1.1 Capas de la arquitectura de un lago de datos.

Riesgos de los lagos de datos

Como se ha mencionado, los lagos de datos tienen varias ventajas para los usuarios u organizaciones que decidan usarlos, sin embargo, se han identificado tres riesgos primordiales que presenta este modelo de almacenamiento: la calidad de los datos, su seguridad y los riesgos de control de acceso. La calidad de los datos constituye un riesgo esencial, ya que la eficacia de cualquier análisis o decisión basada en datos depende de la integridad y exactitud de la información utilizada. Los datos incompletos, inconsistentes o erróneos pueden llevar a conclusiones equivocadas y a acciones incorrectas (Madera & Laurent, 2016).

La seguridad de los datos también es importante dentro de sus riesgos, ya que los lagos de datos almacenan una gran cantidad de información sensible y valiosa, lo que los convierte en objetivos de interés para los ciberdelincuentes. Las brechas de seguridad pueden resultar en la exposición de datos confidenciales, el robo de información o la alteración malintencionada de los datos. Además, los riesgos de control de acceso deben ser abordados adecuadamente, debido a que la gestión del lago de datos implica el acceso a datos por parte de múltiples usuarios y grupos, lo que puede dar lugar a problemas de

seguridad y privacidad si no se establecen adecuados controles de acceso. Es necesario definir políticas y roles claros, así como implementar mecanismos de autorización y seguimiento para garantizar que solo usuarios autorizados puedan acceder a los datos apropiados (Madera & Laurent, 2016).

En este contexto, Apache Hadoop, un marco de código abierto, se destaca como el cimiento que posibilita la realización de los lagos de datos.

Como se ha mencionado, el concepto de lago de datos surge como respuesta a la necesidad de almacenar y procesar volúmenes masivos de datos. Este enfoque, con su capacidad de integrar información de diversas fuentes sin predefinir su estructura, redefine cómo se manejan y analizan los datos. En este contexto, Apache Hadoop, un marco de código abierto, se destaca como el cimiento que posibilita la realización de los lagos de datos. Con el tiempo, este marco se ha consolidado, permitiendo el despliegue de soluciones de análisis y almacenamiento de datos en un entorno que abarca múltiples disciplinas y aplicaciones.

Apache Hadoop

Apache Hadoop es una suite de software de código abierto que posibilita el procesamiento y almacenamiento distribuido de conjuntos de datos masivos a través de un clúster compuesto por diversos sistemas computacionales (Munshi & Mohamed, 2018). Es una plataforma de datos altamente disponible y orientada a objetos, empleada en el ámbito del Big Data para la carga, transformación y procesamiento de información en un lago de datos. Hadoop es un marco de trabajo que posibilita el procesamiento distribuido de enormes volúmenes de datos a través de clústeres de computadoras. Su diseño está pensado para escalar desde un único servidor hasta miles de máquinas, brindando capacidad de almacenamiento y potencia de cálculo (Aucancela, Naranjo, & Betún, 2018).

Apache Hadoop fue creado con el propósito de abordar desafíos de datos a gran escala que no pueden ser resueltos por sistemas o software comerciales convencionales. Al ser utilizado, esta herramienta posee características distintivas que le confieren su gran valor (Ojeda, 2020), siendo las principales, las que se muestra en la figura 1.2.

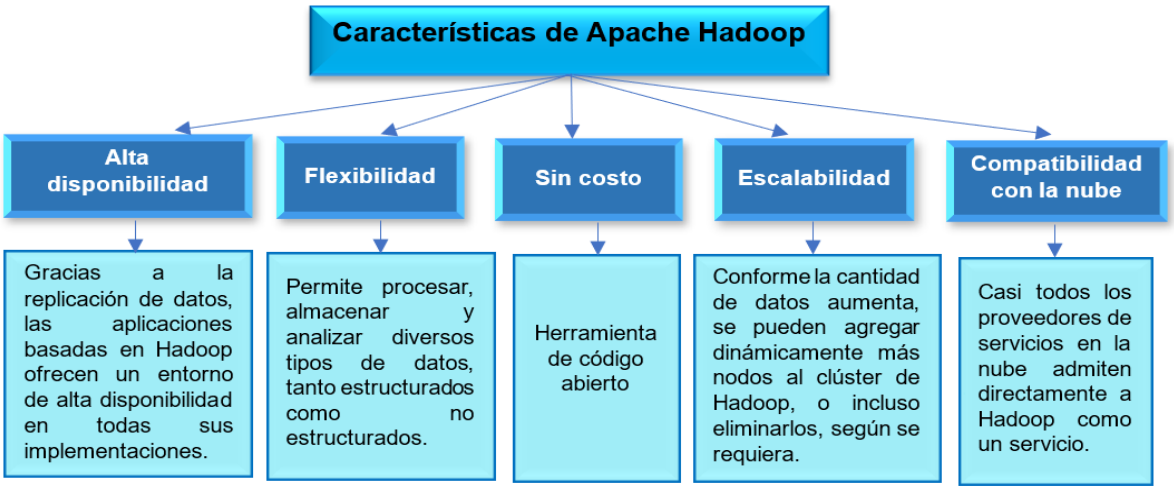

Figura 1.2. Características de la herramienta Apache Hadoop.

La premisa subyacente en la arquitectura y diseño de Hadoop es que el procesamiento de datos se vuelve más sencillo, rápido y eficiente, pero al mismo tiempo, se reconoce que estos datos pueden generar altos niveles de latencia y congestión en la red. Para remediar esta situación, Hadoop cuenta con sus principales componentes: HDFS (el sistema de archivos distribuido de Hadoop), MapReduce (un paradigma de procesamiento de datos) y YARN (una tecnología para la programación, administración y gestión, de tareas y recursos) (Beltrán, 2022).

Hadoop Distributed File System (HDFS)

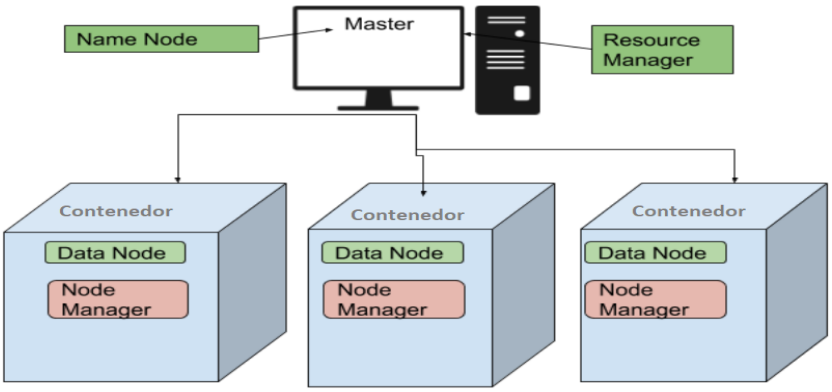

Figura 1.3. Componentes del sistema de archivos HDFS. *(Dikshantmalidev, 2019)*

HDFS es un sistema de archivos distribuido que ha sido especialmente diseñado para funcionar en hardware básico. Aunque comparte similitudes con otros sistemas de archivos distribuidos existentes, presenta diferencias significativas. Una de ellas es su alta tolerancia a fallos y su capacidad para ejecutarse en hardware de bajo costo. HDFS ofrece un rendimiento excepcional en el acceso a los datos de las aplicaciones y resulta especialmente adecuado para aquellas que manejan grandes conjuntos de datos. HDFS

sigue una arquitectura de tipo maestro/esclavo. Como se puede observar en la figura 1.3, un clúster con HDFS consta de un nodo maestro denominado NameNode, que administra el espacio de nombres del sistema de archivos y controla el acceso de clientes a los archivos. Además, hay varios DataNodes, generalmente uno por cada nodo del clúster, encargados del almacenamiento asignado en los nodos. También se encuentran los componentes encargados de la administración de recursos, como son, el ResourceManager, para el lago de datos, y el NodeManager, a nivel de cada nodo (Ojeda, 2020).

NameNode y DataNodes

HDFS brinda un espacio de nombres del sistema de archivos en el que los datos de los usuarios pueden ser almacenados en archivos. Los archivos en HDFS se dividen en bloques, que luego son almacenados en un conjunto de DataNodes. El denominado NameNode se encarga de administrar las acciones ejecutadas en el sistema de archivos, tales como la apertura, el cierre y la modificación de nombres de archivos y directorios. También determina cómo se asignan los bloques a los DataNodes. Por otro lado, el proceso de atender las peticiones de lectura y escritura de los usuarios del sistema de archivos, es responsabilidad de los DataNodes. De la misma manera, se encargan de llevar a cabo la creación, eliminación y duplicación de bloques de acuerdo a las indicaciones del NameNode (Ojeda, 2020).

ResourceManager

Este componente es un servicio encargado de la planificación de los recursos computacionales para las aplicaciones. Su objetivo principal es optimizar la utilización del clúster en términos de memoria, núcleos de CPU, tiempo de respuesta y cumplimiento de acuerdos de nivel de servicio (SLA, por sus siglas en inglés). Este servicio consta de dos componentes: el Planificador, encargado de asignar recursos a las aplicaciones que se envían al clúster, aplicando condiciones basadas en capacidades y colas; y el Administrador de Aplicaciones, utilizado para gestionar las aplicaciones en todo el clúster. El Administrador de Aplicaciones se encarga de recibir solicitudes de aplicaciones, proporcionar los recursos necesarios para que la aplicación pueda iniciar, monitorear el progreso de la aplicación y reiniciarla en caso de fallos (Ojeda, 2020).

NodeManager

El NodeManager es el componente encargado de ejecutar los contenedores en función de la capacidad del nodo. La capacidad del nodo se determina en base a la memoria instalada

y el número de núcleos de la CPU. El NodeManager envía señales periódicas (*heartbeats*) al ResourceManager para mantener actualizado constantemente su estado de salud. Además, también envía información sobre su estado actual, que puede ser el estado del nodo en sí o el estado de las tareas que se están ejecutando en él (Ojeda, 2020). El funcionamiento del NodeManager se encuentra intrínsecamente relacionado con el concepto de contenedor. Este último es la unidad que encapsula recursos como memoria, CPU y almacenamiento.

Contenedor

Un contenedor es una unidad lógica que contiene recursos como memoria, CPU, disco, etc., y está asociado a un nodo específico. El servicio de programación del ResourceManager asigna dinámicamente recursos en forma de contenedores. Cada contenedor otorga los privilegios para utilizar una cantidad específica de recursos de un host en particular (Ojeda, 2020).

En el campo de la gestión de recursos y aplicaciones en entornos distribuidos, los contenedores juegan un papel primordial al encapsular de manera eficiente componentes y dependencias. Y es en este contexto que se presenta Docker, una plataforma de virtualización que proporciona un entorno aislado y portable para la ejecución de aplicaciones. Al emplear tecnologías de contenedores, Docker simplifica el despliegue y escalado de aplicaciones, permitiendo que las mismas funcionen de manera coherente en diversos entornos.

Docker

Docker, como iniciativa de código abierto, posibilita la generación de aplicaciones en contenedores de software, que se caracterizan por ser livianos, independientes y portátiles. En contraste con los enfoques de virtualización convencionales, Docker emplea contenedores en lugar de máquinas virtuales, marcando su diferencia distintiva. Los contenedores son paquetes que permiten crear un entorno donde ejecutar aplicaciones de forma independiente del sistema operativo. El código se ejecuta en estos contenedores de manera aislada, sin interferir con otros contenedores, y todos comparten los recursos de la máquina sin la sobrecarga que implica la capa de virtualización del hipervisor (Ponsico, 2017). A diferencia de la emulación de sistemas operativos completos, en Docker no hay simulación de hardware ni de arquitectura, lo que hace que la ejecución sea más rápida y requiera menos recursos (Fez, Arce, Belda, & Guerri, 2021).

Docker está compuesto por diferentes componentes, como el Docker Daemon, el Docker Client CLI, el Docker Hub, Docker Engine las imágenes y los contenedores (Ponsico, 2017). Los cuales se describen a continuación en la figura 1.4.

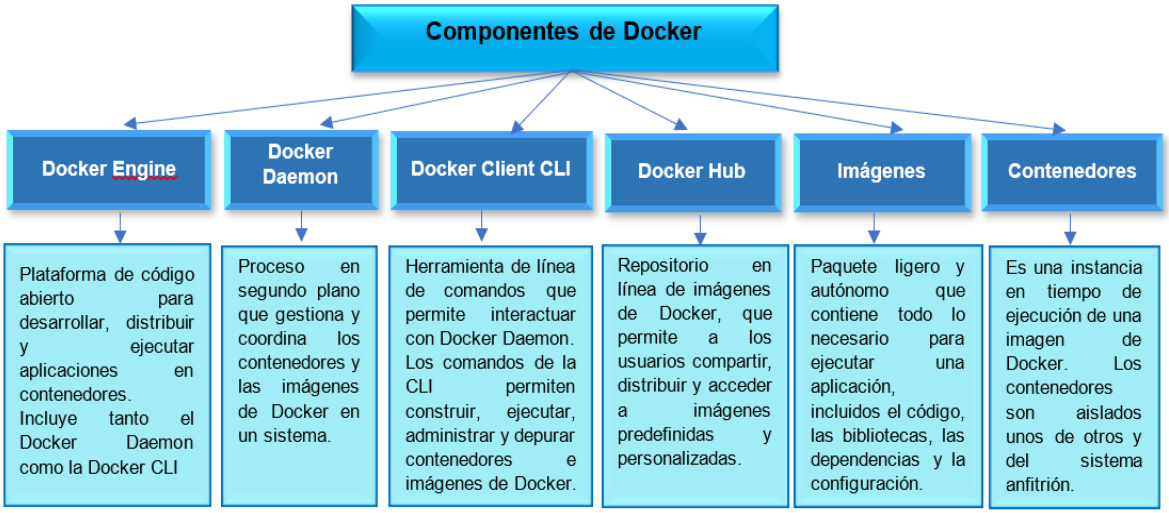

Figura 1.4. Componentes de la herramienta Docker.

Docker Compose es una herramienta que es usada para describir aplicaciones complejas y administrar los contenedores, redes y volúmenes que requieren. Simplifica el proceso de configuración y ejecución de aplicaciones, evitando la necesidad de escribir comandos complejos que podrían conducir a errores de configuración. La descripción de la aplicación y sus requisitos se definen en un archivo de configuración, el cual se procesa utilizando el comando docker-compose. La cantidad de contenedores en una aplicación se puede ajustar utilizando el comando docker-compose scale (Freeman, 2017).

Docker se ha convertido popular muy rápidamente, debido a los beneficios proporcionados. Las ventajas de Docker más valoradas son velocidad, portabilidad, escalabilidad, entrega rápida y densidad (Rad, Bhatti, & Agmadi, 2017).

En la siguiente sección se tratará la revisión de la literatura existente sobre los lagos de datos. Se realizará una revisión de investigaciones, estudios y enfoques relacionados con los lagos de datos y su papel en el almacenamiento efectivo de volúmenes masivos de información. Se explorarán trabajos y conocimientos previos para establecer las bases teóricas y conceptuales de este proyecto. La exploración de trabajos similares resaltará tendencias, desafíos y oportunidades en el ámbito de los lagos de datos. Esta sección es fundamental para las decisiones de implementación y diseño del lago de datos en el laboratorio ADA.

1.5 Revisión de la literatura

Esta sección se enfoca en explorar y analizar la investigación y los estudios previos relevantes relacionados con la implementación de un lago de datos en diferentes contextos. El lago de datos ha surgido como una solución prometedora para gestionar grandes volúmenes de datos heterogéneos en entornos empresariales. En esta sección, se examinarán los conceptos clave, las metodologías y las mejores prácticas identificadas en la literatura científica y técnica, con el objetivo de comprender el estado actual del campo y establecer una base para el desarrollo de esta investigación. A través de la revisión de la literatura existente, se espera identificar las oportunidades, los desafíos y las áreas de mejora en la implementación de un lago de datos.

En la investigación realizada, se encontraron algunos trabajos similares a lo que se desea implementar en este proyecto. Este es el caso del trabajo de (Balón de la Cruz, 2022), el cual proporciona una guía para la adopción de un lago de datos enmarcado en el *cloud computing*, para los datos de la Federación de Organizaciones Populares Peninsulares (FOPOPE), de Santa Elena en Ecuador. Este estudio, es de especial importancia para el presente proyecto, debido a que realiza una investigación y comparación, de los beneficios, viabilidad de implementación, arquitectura y funcionamiento, de los tres principales proveedores del servicio de *cloud computing* de lago de datos. En el trabajo mencionado, se comparan a los servicios de Google Cloud, Microsoft Azure y Amazon Web Services, brindando así una visión general de los requisitos de arquitectura y funcionamiento, que se deben cumplir para la implementación un lago de datos, además, de dar un punto de partida para la selección de herramientas a usar en la implementación del presente proyecto.

Otro de los artículos relevantes, es el trabajo realizado por (Ojeda, 2020), en el que se muestra el proceso de implementación de un lago de datos, destinado análisis de movilidad en la ciudad de México, y creado con servicios de computación en la nube ofrecidos por Amazon Web Services. Este proyecto brinda importantes aportes al presente trabajo, ya que muestra detalladamente el proceso de diseño e implementación de la arquitectura de un lago de datos en la nube, y, lo más importante, presenta el uso y funcionamiento de Apache Hadoop como una de las principales herramientas para la creación de su lago de datos.

Al igual que el trabajo antes mencionado, el proyecto de maestría de (Beltrán, 2022), ofrece una explicación detallada del funcionamiento de cada componente del entorno de Hadoop en el contexto de la implementación de un lago de datos. Este trabajo consiste en la creación de un lago de datos con los datos del Learning Management System (LMS) de

Canvas de la Universidad Internacional SEK, mediante la utilización y aplicación de las herramientas del ecosistema de Hadoop. Además de lo mencionado, en este estudio se integra a Python como una de las herramientas para la exploración y preparación de los datos, lo cual es una parte esencial para el presente proyecto.

Un aspecto fundamental en este proyecto es la orientación que se le debe dar al lago de datos para que funcione como una herramienta integral para el análisis de datos. En este contexto, el trabajo de (Balboa, 2022) es de gran utilidad ya que presenta el proceso de implementación de un lago de datos en el campo empresarial, haciendo énfasis en su uso para el análisis de datos. El objetivo de dicho proyecto es la implementación de un lago de datos en la empresa Belcorp, la cual maneja datos de diversas fuentes y en diversos formatos. Este artículo ofrece información importante sobre el papel que juega el lago de datos y lo que se espera del mismo, desde el punto de vista de los equipos de analistas, que usarán esta forma de almacenamiento como entrada para sus métricas y modelos de predicción.

Por último, en el artículo (Madera & Laurent, 2016) se realiza un estudio detallado sobre los lagos de datos desarrollados con la tecnología Hadoop, centrándose en varios criterios como la forma de guardar los datos, la forma correcta en que deben ser usados y, lo más importante, brinda información sobre cuál debe ser el perfil del usuario final del lago de datos. En este trabajo se menciona que el lago de datos no es para todos los usuarios, sino que su diseño está orientado para los perfiles de analista de datos y científicos de datos. Dado que el presente proyecto, el cual tiene características similares al del estudio mencionado, está destinado a servir de ayuda en las investigaciones de los colaboradores del laboratorio ADA, el conocer las habilidades y conocimientos con las que deben contar, es una tarea fundamental a tener en cuenta.

En conclusión, la revisión exhaustiva de la literatura relacionada con el tema del lago de datos ha revelado numerosos trabajos de investigación y estudios que han aportado valiosos conocimientos al campo. Estas investigaciones previas han sentado las bases tanto teóricas como prácticas para comprender los conceptos fundamentales y las metodologías asociadas con los lagos de datos. Sin embargo, lo que distingue este trabajo es la implementación del lago de datos en un entorno virtual utilizando contenedores Docker y la herramienta Hadoop.

Esta aproximación ofrece una mayor flexibilidad, escalabilidad y eficiencia en comparación con los enfoques tradicionales. Además, se añade un componente adicional de valor al proporcionar una interfaz gráfica mediante Jupyter, permitiendo a los usuarios realizar

análisis de datos de manera interactiva y visual. Este estudio no solo se beneficia de la sólida base de conocimientos proporcionada por la literatura existente, sino que también ofrece contribuciones innovadoras en el contexto de una implementación eficiente y un análisis de datos más accesible, en el campo de los lagos de datos.

En la siguiente sección, se abordará la metodología que guiará el desarrollo de este proyecto. En esta fase se recopilarán los requerimientos, se realizará la selección de herramientas, se diseñará la arquitectura del lago de datos y se llevará a cabo su implementación. La metodología establecerá una secuencia ordenada de pasos, desde la conceptualización hasta la materialización del lago de datos en el servidor del laboratorio ADA.

2 METODOLOGÍA

En este proyecto, se ha adoptado una metodología multidisciplinaria que combina diferentes enfoques para abordar de manera integral la implementación del lago de datos. Se han utilizado las metodologías cualitativa, experimental, descriptiva y *design thinking* para asegurar un enfoque completo y efectivo.

El enfoque cualitativo de investigación se presenta como una metodología clave para comprender y describir los fenómenos sociales y humanos asociados al uso del lago de datos. A diferencia de la metodología cuantitativa, que se basa en la recolección y análisis numérico de datos, la metodología cualitativa se enfoca en la interpretación y comprensión de los significados, experiencias y contextos sociales relacionados con el tema que se está investigando, mediante métodos como entrevistas, observaciones y análisis de contenido (Vega, y otros, 2014). Esta metodología cualitativa permitirá explorar y capturar en profundidad las perspectivas y opiniones de los usuarios, así como los desafíos y beneficios experimentados en el contexto de la implementación del lago de datos. Este enfoque fue fundamental en la recopilación de requerimientos en el presente proyecto.

El enfoque experimental implica la planificación y realización de experimentos diseñados específicamente para evaluar el sistema propuesto. Durante estos experimentos, se definen variables relevantes que se controlan cuidadosamente con el fin de medir el impacto de los cambios realizados en el entorno del tema que se está investigando (Murillo, y otros, 2011).

De esta manera, se busca obtener resultados empíricos que permitan comparar y analizar diferentes configuraciones y opciones disponibles en el ecosistema Hadoop. Estos experimentos proporcionarán evidencia concreta sobre el desempeño y las capacidades

del lago de datos implementado, permitiendo tomar decisiones informadas y realizar ajustes pertinentes en función de los resultados obtenidos. El enfoque experimental en este proyecto fue esencial para la parte de experimentación con diferentes configuraciones de la estructura del lago de datos.

En el caso de la metodología descriptiva, esta desempeña un papel fundamental al proporcionar una descripción detallada de los componentes y la configuración del sistema. Esta metodología se utiliza para recopilar información precisa sobre las fundamentales del entorno que se está estudiando, además emplea criterios que posibilitan la definición de la estructura o el comportamiento de los sistemas bajo análisis. (Guevara, Verdesoto, & Castro, 2020).

Mediante un enfoque descriptivo, se busca comprender a fondo el funcionamiento del sistema e identificar áreas que pueden ser mejoradas. A través de la recopilación sistemática de datos y la documentación detallada, se obtendrá una visión completa de la estructura y el rendimiento del servidor de altas prestaciones en la nube, lo que permitirá evaluar su eficiencia y determinar posibles mejoras. La metodología descriptiva es esencial para proporcionar información que fundamente la toma de elecciones y la optimización continua del lago de datos.

Por último, se ha adoptado el enfoque *design thinking* como parte del proceso de diseño conceptual del proyecto. Este enfoque se utiliza para identificar problemas, definir objetivos claros y buscar soluciones adecuadas basadas en investigaciones previas y mejores prácticas. Al integrar el *design thinking* en el proyecto, se busca asegurar que el diseño de la arquitectura del lago de datos se base en una sólida comprensión de las necesidades reales de los usuarios y en una búsqueda activa de soluciones innovadoras (Leopoldo, 2017).

A través de métodos de investigación y diseño centrados en el usuario, este enfoque ayuda a comprender a fondo las expectativas y necesidades de los usuarios del lago de datos. El enfoque design thinking será fundamental para garantizar que el proyecto del lago de datos de cumplimiento a los requerimientos de los usuarios, al tiempo que promueve la innovación y la efectividad en la implementación del sistema.

2.1 Identificación de Requerimientos

La identificación de los requerimientos fue el primer paso para el proceso de implementación del lago de datos en el laboratorio ADA. Esta sección se enfocó en comprender las necesidades y objetivos específicos del laboratorio con respecto al

almacenamiento y gestión de datos, a fin de establecer los fundamentos sólidos para el diseño, desarrollo e implementación del lago de datos. Para ello, se llevó a cabo un análisis de los diferentes aspectos del laboratorio, como los tipos de datos generados, los flujos de trabajo existentes y las necesidades de análisis y visualización de datos.

La identificación de los requerimientos fue un proceso colaborativo que involucró a la directora del presente proyecto y al coordinador del laboratorio ADA, con el objetivo de garantizar que el lago de datos se adapte de manera óptima a las necesidades del laboratorio y proporcione una base sólida para la toma de decisiones basada en datos y la generación de conocimiento científico de calidad.

Para la identificación de los requerimientos para el diseño e implementación del lago de datos en el servidor de altas prestaciones del laboratorio ADA, se ha tomado como referencia a la norma ISO/IEC/IEEE 29148:2018, la cual contiene disposiciones y directrices que abarcan las etapas y elementos vinculados a la ingeniería de requisitos en sistemas, productos y servicios de software a lo largo de su ciclo de vida. Su objetivo principal es establecer los principios para construir requerimientos sólidos, y proporcionar atributos y características específicas para los mismos, lo que garantiza que los requerimientos de un sistema o software estén claramente definidos, sean comprensibles, verificables y trazables (Llanque, 2021).

Uno de los principales procesos que se encuentran en la norma antes mencionada, es la identificación de todas las partes interesadas en la construcción del producto software, que en este caso es el lago de datos. Para realizar el proceso de identificación, se realizó un conversatorio con la directora del presente trabajo para determinar quiénes son las partes interesadas de este componente del proyecto en específico. Como resultado de este conversatorio se obtuvo la Tabla 2.1, que se muestra a continuación.

Tabla 2.1. Lista de partes interesadas o Stakeholders

Lista de Stakeholders	
1	Investigadores de la FIS de la Escuela Politécnica Nacional.
2	Estudiantes de posgrado de la FIS Escuela Politécnica Nacional
3	Personal del laboratorio ADA de la FIS Escuela Politécnica Nacional.

Una vez identificadas las partes interesadas se procedió a comprender las necesidades que se desea cubrir en el laboratorio ADA con la implementación del lago de datos. Para la identificación de los requisitos que se deben cumplir con el presente trabajo, se mantuvieron tres reuniones con la directora del proyecto y una reunión final con el

coordinador del laboratorio ADA. Además, se realizó una investigación teórica para identificar cuáles deben ser los requisitos mínimos que se deben cumplir un lago de datos en estado de producción.

Cabe mencionar que de las reuniones con la directora del proyecto y del coordinador del laboratorio ADA, se obtuvieron como resultado los requerimientos de negocio, operacionales, físicos y de usuarios. Mientras que con la investigación teórica realizada se identificaron requisitos funcionalidad, de uso y *performance*. Todos los requerimientos identificados se agruparon en las categorías antes mencionadas, a la vez que se ordenaron por su relevancia y prioridad. Como resultado de la identificación de requerimientos se obtuvo la Tabla 2.2, la cual se muestra a continuación:

Tabla 2.2. Requerimientos del Lago de datos

REQUERIMIENTOS DE NEGOCIO				
		PRIORIDAD		
NÚM.	REQUERIMIENTO	Alta	Media	Baja
1	Almacenamiento de grandes volúmenes de datos en los servidores del laboratorio ADA.	X		
2	La arquitectura del lago de datos debe adaptarse a las características de hardware y software del servidor de altas prestaciones del laboratorio ADA	X		
REQUERIMIENTOS OPERACIONALES				
3	Implementación del lago de datos sobre un clúster con varios nodos.	X		
4	El formato de datos permitidos debe basarse en políticas establecidas.	X		
5	Las herramientas seleccionadas deben adaptarse a las especificaciones técnicas del servidor del laboratorio.	X		
6	El acceso al lago de datos por parte de usuarios debe basarse en políticas establecidas.	X		
7	El lenguaje o herramientas usadas no deben tener costo de licenciamiento.	X		
REQUERIMIENTOS DE USUARIOS				
8	Que se brinde el uso de los lenguajes Python y R para el análisis de datos.	X		
9	Se requiere una interfaz gráfica para la ejecución de código orientado al análisis de datos.	X		
10	Se debe permitir el procesamiento de grandes volúmenes de datos.	X		
11	Los usuarios deben poder subir los datos al lago de datos mediante políticas establecidas.		X	
12	Los usuarios deben poder descargar sus resultados mediante la interfaz gráfica.		X	

REQUERIMIENTOS DE FUNCIONALIDAD				
13	Escalabilidad		X	
14	Flexibilidad		X	
15	Integración	X		
16	Accesibilidad	X		
17	Seguridad	X		
18	Gobernanza		X	
REQUERIMIENTOS DE PERFORMANCE				
19	Persistencia en el procesamiento de datos	X		
20	Persistencia en la conexión al lago de datos	X		
REQUERIMIENTOS DE MODOS/ESTADOS				
21	Se necesita que el algo de datos y las herramientas de análisis de datos se encuentren en modo de producción.	X		

Con los requerimientos debidamente recopilados, se procedió a la elección de las herramientas idóneas para la implementación del lago de datos. Esta selección debe ser acorde con los objetivos y expectativas de los usuarios, además de garantizar el cumplimiento de los requisitos planteados en la Tabla 2.1. La alineación entre estas herramientas y los objetivos de requerimientos asegura una implementación coherente y eficaz del lago de datos en el contexto del laboratorio ADA.

2.2 Selección de herramientas

La selección de las herramientas adecuadas es un aspecto crítico en la implementación exitosa de un lago de datos en el contexto de este proyecto. Dada la complejidad y el volumen de datos generados en la actualidad, contar con las herramientas adecuadas se vuelve fundamental para capturar, almacenar y procesar eficientemente esta información. En esta sección, se explorarán diversas opciones disponibles, evaluando sus características y funcionalidades con el objetivo de identificar aquellas que mejor se ajusten a los requerimientos específicos del presente proyecto. Como producto del análisis de requerimientos, se tomaron como criterios de selección los siguientes elementos:

- **Escalabilidad:** Debe ser capaz de almacenar grandes cantidades de datos que pueden crecer a medida que la organización crece.
- **Flexibilidad:** Debe ser capaz de manejar diversos tipos de datos, incluyendo datos estructurados, semi-estructurados y no estructurados, así como diferentes formatos de datos, como texto, imágenes, audio y video.

- **Integración:** Debe ser capaz de integrar diferentes fuentes de información, como sistemas de archivos, bases de datos relacionales y sistemas en la nube.
- **Accesibilidad:** Debe ser fácilmente accesible por los usuarios, lo que implica una interfaz de usuario intuitiva y herramientas que permitan a los usuarios realizar consultas y análisis de datos.
- **Seguridad:** Debe ser seguro, lo que implica la implementación de medidas de seguridad, como el control de acceso, la encriptación de datos y la auditoría.
- **Gobernanza:** Debe estar bien gobernado, lo que implica la implementación de políticas y procesos que garanticen la calidad de los datos, la privacidad y la conformidad con las regulaciones.
- ***Open Source*:** La herramienta usada para la implementación del lago de datos debe ser de código abierto y totalmente gratuita.

Con estos parámetros se realizó una investigación de las principales herramientas, programas y sistemas software, usados para la implementación de lago de datos tanto en ámbitos empresariales como educativos. La búsqueda se centró en herramientas que cumplan en su totalidad, o, al menos, parcialmente con los requerimientos antes mencionados. En este contexto se eligieron las siguientes herramientas:

Apache Hadoop: Apache Hadoop se destaca como una plataforma que habilita el procesamiento eficiente de volúmenes masivos de datos. Su modelo de programación sencillo facilita la creación de aplicaciones distribuidas que demandan un uso intensivo de datos y una escalabilidad óptima. La plataforma, escrita en el lenguaje de programación Java, proporciona un sólido framework que sirve de base para el desarrollo de estas aplicaciones distribuidas. Un aspecto relevante es que Apache Hadoop es administrado por la Apache Software Foundation, una reconocida organización involucrada en el desarrollo de software de código abierto (Sánchez, 2015). La presencia y respaldo de esta fundación aportan confianza y garantizan la continua evolución y mantenimiento de la plataforma Hadoop, consolidándola como una opción confiable para abordar el procesamiento y almacenamiento de grandes volúmenes de información en este proyecto de lago de datos.

IBM: En el contexto del lago de datos, IBM proporciona soluciones que abarcan la adquisición, almacenamiento y análisis de datos a gran escala. Sus plataformas, como IBM InfoSphere BigInsights y IBM Cloud Pak for Data, permiten la integración y gestión de diversos tipos de datos. IBM ofrece herramientas para garantizar la seguridad, gobernanza

y cumplimiento normativo de los datos almacenados en el lago. Además, facilita el análisis avanzado y el descubrimiento de información valiosa a través de herramientas de inteligencia artificial y aprendizaje automático. La implementación y uso de soluciones de datos de IBM, generalmente implican una estructura de precios basada en el consumo de recursos, almacenamiento, procesamiento y características adicionales (Cherradi, Bouhafer, & EL Haddadi, 2023).

Cloudera: Cloudera ofrece a las organizaciones almacenar, gestionar y analizar grandes volúmenes de datos de manera eficiente. Brinda una plataforma integral llamada "Cloudera Data Platform" (CDP) que integra diversas tecnologías de código La mayoría de las soluciones y plataformas proporcionadas por Cloudera, como Cloudera Data Platform (CDP) y otras herramientas relacionadas, están disponibles bajo una licencia comercial. abierto, como Hadoop, Spark, Impala, HBase y más, en un entorno unificado. La mayoría de las soluciones y plataformas proporcionadas por Cloudera, como Cloudera Data Platform (CDP) y otras herramientas relacionadas, están disponibles bajo una licencia comercial (Díaz Palacios, 2020).

Amazon S3: Es un servicio de almacenamiento en línea ofrecido por Amazon, dirigido específicamente a desarrolladores. Amazon S3 proporciona un almacenamiento ilimitado a través de una interfaz sencilla. Este servicio no es gratuito, ya que Amazon establece un cargo de $0.15 por gigabyte al mes, además de otros cargos adicionales según el ancho de banda utilizado para recuperar los objetos almacenados. Al utilizar Amazon S3, estas empresas pueden despreocuparse de la gestión del almacenamiento y obtener un servicio confiable que cumple con sus necesidades. La disponibilidad, la simplicidad de uso y el costo razonable de Amazon S3 lo convierten en una opción popular para el alojamiento de contenido de un proyecto de lago de datos (Fonticiella Torre, 2009).

Microsoft Azure Data Lake Store: Azure Data Lake destaca como uno de los principales servicios en la nube para el procesamiento, almacenamiento y análisis de grandes volúmenes de datos. Este servicio proporciona una plataforma robusta y escalable para el manejo de petabytes de información, utilizando lenguajes significativos y versátiles como U-SQL, R, Python y .NET. Azure Data Lake es especializado en el almacenamiento masivo de datos y su posterior análisis. Sin embargo, es importante tener en cuenta que este servicio no es gratuito, ya que existen costos asociados a su utilización. A pesar de esto, Azure Data Lake brinda una solución completa y poderosa para el manejo y análisis de datos a gran escala, ofreciendo la infraestructura necesaria para resolver los desafíos inherentes a los datos presentes en este trabajo (Balón de la Cruz, 2022).

Google Cloud: Google Cloud es una plataforma que ha integrado todas las aplicaciones de desarrollo web que previamente Google ofrecía por separado. Esta plataforma se utiliza para crear soluciones a través de la tecnología almacenada en la nube. Google Cloud Platform proporciona servicios que operan en una infraestructura similar a la que Google utiliza internamente. Destaca por su amplia gama de servicios, incluyendo IaaS, PaaS y SaaS, y ofrece soluciones que abarcan almacenamiento en la nube, bases de datos y *networking*. La parte fundamental de esta herramienta en el lago de datos es el servicio de Google Cloud Storage, el cual tiene costos de licenciamiento si se desea obtener todos sus beneficios (Guayas Espín, 2023).

Una vez realizada esta preselección de las mejores opciones disponibles en el campo de los lagos de datos, se procedió a realizar una comparación de las mismas, con base en el cumplimiento total, parcial y no cumplimiento de los principales requerimientos del lago de datos además de las especificaciones dadas por el laboratorio ADA. Este contraste se resume a continuación en la Tabla 2.3.

Tabla 2.3. Criterios para la selección de herramientas.

 Cumple  Cumple Parcialmente  No Cumple	**Requisito**							
Herramienta	**Escalabilidad**	**Flexibilidad**	**Integración**	**Accesibilidad**	**Seguridad**	**Gobernanza**	***Open Source***	**% de cumplimiento**
Apache Hadoop								100%
Google Cloud								85,71%
Amazon S3								85,71%
Microsoft Azure Data Lake Storage								85,71%
IBM								78,57%
Cloudera								78,57%

Como conclusión, se ha seleccionado Apache Hadoop como la herramienta principal para la implementación del lago de datos. Esta elección se basa en su capacidad para cumplir con los criterios clave establecidos. En primer lugar, Apache Hadoop demuestra una

destacada escalabilidad al poder almacenar grandes cantidades de datos que pueden crecer junto con la organización. Además, ofrece flexibilidad al manejar diversos tipos y formatos de datos, como datos estructurados, semi-estructurados y no estructurados, incluyendo texto, audio, imágenes y video.

Otro aspecto crucial es la integración, y Apache Hadoop destaca al poder integrar diferentes fuentes de datos, como sistemas de archivos, bases de datos relacionales y sistemas en la nube. Asimismo, su accesibilidad se asegura mediante las interfaces de usuario del NameNode y ResourceManager, además su elevada capacidad de adhesión con herramientas de análisis de datos que permiten a los usuarios realizar consultas y análisis de manera sencilla. En términos de seguridad, Apache Hadoop cumple con los requisitos al implementar medidas de seguridad como encriptación de datos, auditoría y control de acceso. También se destaca su capacidad para una buena gobernanza, mediante la implementación de políticas.

Por último, la decisión de seleccionar Apache Hadoop se ve respaldada por su condición de código abierto y totalmente gratuito, cumpliendo así con el requisito de que la herramienta utilizada para la implementación del lago de datos sea de este tipo. En resumen, Apache Hadoop se ha elegido como la herramienta principal para el lago de datos debido a su escalabilidad, flexibilidad, integración, accesibilidad, seguridad, gobernanza y su condición de código abierto. Estas características garantizan que cumpla con los requisitos fundamentales del proyecto y permitirá aprovechar al máximo el potencial del lago de datos.

Se ha seleccionado a Apache Hadoop 3.3.1 como la versión para el lago de datos, lanzada en junio de 2021. Esta decisión se basa en su estabilidad y madurez en comparación con versiones más tempranas. Además, según las investigaciones realizadas, a la fecha de la realización de este proyecto es la versión más estable en cuanto a su integración con la herramienta Docker. Adicionalmente, a partir de su fecha de lanzamiento esta versión tiene 3 años de soporte.

Para el establecimiento del lago de datos de Hadoop, se ha seleccionado Docker como la plataforma principal para la creación de los nodos de clúster. Esta elección se basa en las numerosas ventajas que ofrece Docker, así como en su alta adhesión y compatibilidad con el entorno de Hadoop. Docker proporciona una solución eficiente para la virtualización y el despliegue de aplicaciones en contenedores independientes y aislados. Su capacidad para encapsular aplicaciones y sus dependencias en contenedores portátiles facilita la creación y gestión de los nodos de clúster necesarios para el lago de datos de Hadoop (Shih, Yang,

& Chiang, 2021). Al utilizar Docker, se obtiene un entorno altamente flexible y escalable, permitiendo un despliegue rápido y fácil de nuevos nodos, así como una gestión eficiente de los recursos.

Además, Docker ofrece una gran variedad de imágenes y herramientas preconfiguradas, lo que simplifica la implementación de los componentes de Hadoop en los contenedores. Esto asegura una alta adhesión con Hadoop y facilita la integración de las diversas herramientas y servicios que componen el ecosistema de Hadoop. Otra ventaja clave de Docker es su capacidad para proporcionar un entorno consistente y reproducible. Al utilizar contenedores, se garantiza que cada nodo del clúster tenga una configuración y versión de software coherente, lo que contribuye a la estabilidad y confiabilidad del lago de datos de Hadoop (Shih, Yang, & Chiang, 2021).

Además, Docker ofrece una gestión simplificada de recursos, lo que permite una asignación eficiente de CPU, memoria y almacenamiento a cada contenedor. Esto optimiza el rendimiento del lago de datos y mejora la utilización de los recursos disponibles. En resumen, Docker ha sido seleccionado como la plataforma de contenedores para servir como nodos de clúster en la creación del lago de datos de Hadoop debido a sus ventajas significativas, su alta adhesión con Hadoop y su capacidad para proporcionar un entorno consistente y reproducible. Al aprovechar las características y funcionalidades de Docker, se logra una implementación eficiente y escalable del lago de datos de Hadoop. La versión de Docker seleccionada es la v24.0.5, la cual corresponde al última versión estable y disponible en el sitio oficial de esta herramienta.

Con las herramientas apropiadas en consideración, se avanzó hacia la etapa de diseño de la arquitectura del lago de datos. Esta fase implica la definición de la estructura y la interconexión de los elementos, garantizando la adaptabilidad y disponibilidad requeridas. Además, el diseño se enriquece con la inclusión de mecanismos de replicación, garantizando la integridad y la tolerancia a fallos de los datos almacenados. La cohesión entre las herramientas seleccionadas y el diseño de la arquitectura es fundamental para el adecuado funcionamiento del lago de datos en el laboratorio ADA.

2.3 Diseño de la arquitectura del lago de datos

La sección de diseño de la arquitectura del lago de datos desempeña un papel esencial en la implementación exitosa de esta herramienta basada en Hadoop y Docker. El diseño de la arquitectura del lago de datos es fundamental para garantizar su funcionamiento eficiente, escalabilidad y capacidad de adaptación a las necesidades del proyecto. Un diseño cuidadoso de la arquitectura permite definir la estructura, los componentes y la

integración de los distintos elementos del lago de datos, asegurando su coherencia y optimización.

Fase de experimentación

Durante la fase inicial de este proyecto, se llevó a cabo un proceso de experimentación y aprendizaje con el objetivo de comprender en profundidad las herramientas clave utilizadas en la implementación de un lago de datos. Se estableció un entorno de servidor de pruebas y se comenzó con un lago de datos de un solo nodo, utilizando un DataNode. Con el fin de explorar y familiarizarse con la configuración de Hadoop y la creación de contenedores Docker. En esta primera configuración simplemente se probó la instalación y uso de Hadoop y Docker, sin abordar temas como los bloques y la replicación. A continuación, en la figura 2.1 se presenta la arquitectura configurada en el servidor de desarrollo del laboratorio ADA, en esta primera parte de la experimentación.

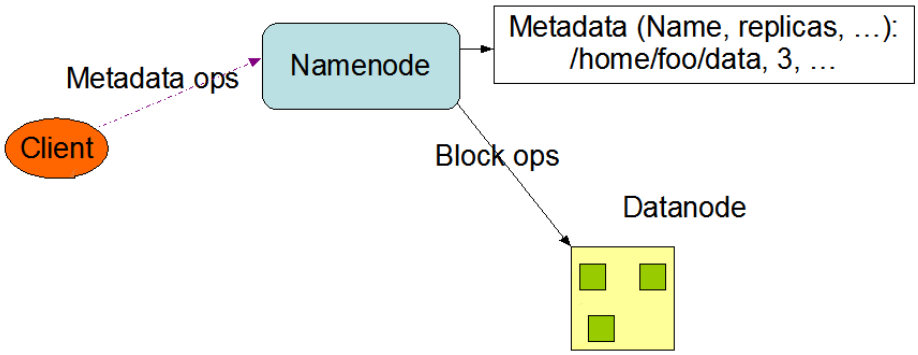

Figura 2.1. Arquitectura del lago de datos con un DataNode *(Hvivani, 2014)*.

A medida que se adquiría experiencia y confianza, se avanzó hacia una implementación con tres DataNodes, ampliando así la escala y complejidad del entorno de prueba, en la figura 2.2 se puede observar la arquitectura configurada en esta primera parte de la experimentación. Durante este proceso de experimentación, se llevaron a cabo diversas actividades, como la carga y descarga de archivos, la configuración de la replicación, tamaño de bloques, la creación de una red en Docker entre los contenedores y la exploración de las interacciones entre los componentes del lago de datos. Estas experiencias permitieron comprender cómo funcionan las herramientas de forma individual y conjunta, sentando así las bases para las etapas posteriores del proyecto.

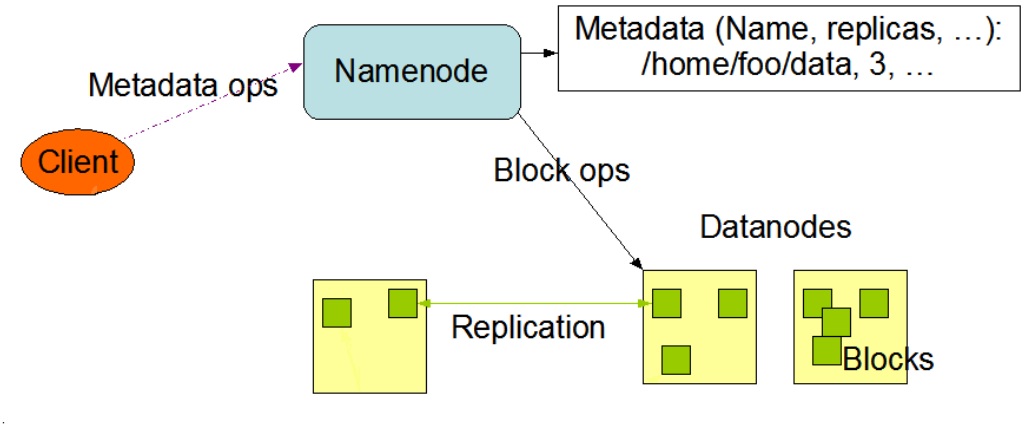

Figura 2.2. Arquitectura del lago de datos con tres DataNodes *(Hvivani, 2014)*.

En la fase final de la experimentación, se dio un paso más al implementar un lago de datos con cuatro DataNodes. Esta configuración permitió evaluar y comprender mejor la escalabilidad y el rendimiento del sistema en un entorno más cercano a la realidad. A través de este proceso gradual de experimentación y ampliación, se pudo adquirir un conocimiento profundo de las herramientas y tecnologías involucradas, lo cual resultó fundamental para el éxito de la implementación final del lago de datos.

En conclusión, la experimentación desempeñó un papel crucial en este proyecto, ya que brindó la oportunidad de aprender y comprender las herramientas necesarias para implementar un lago de datos eficiente y escalable. A través de diversas configuraciones y pruebas, se adquirió experiencia práctica en la configuración de Hadoop, la gestión de contenedores Docker y la interacción entre los componentes del lago de datos. Esta fase de experimentación sentó las bases sólidas de conocimientos y habilidades necesarias para la implementación exitosa del lago de datos en el servidor de producción en el laboratorio ADA.

Fase de diseño

Con el diseño de la arquitectura del lago de datos de este proyecto, se persiguieron varios objetivos específicos para garantizar que esta implementación cumpla con los requerimientos, mencionados anteriormente, y se garantice su eficiencia y confiabilidad. A continuación, se describen los objetivos clave:

- Diseñar una arquitectura de un lago de datos que se adapte a los recursos disponibles en el servidor de altas prestaciones del laboratorio ADA. Esto implica aprovechar al máximo la capacidad de procesamiento, almacenamiento y red

proporcionada por el servidor, optimizando el rendimiento y la eficiencia del lago de datos.

A continuación, en la Tabla 2.4, se muestra un resumen de los principales recursos con los que cuenta el servidor del laboratorio ADA para la implementación del lago de datos.

Tabla 2.4. Recursos disponibles para la implementación del lago de datos.

Característica	Recurso disponible
CPU	4 CPU con 16 núcleos C/U
Memoria RAM	256 GB
Espacio en disco	9.28 TB
Sistema Operativo	Ubuntu 22.04.2 LTS
Recursos de virtualización	Docker v24.0.5
Dirección IP	Conexión con IPv4 para pruebas y desarrollo.
Conectividad de red externa	Conexión a Internet habilitada Descarga: 91.74 Mbit/s Carga: 74.01 Mbit/s

- Diseñar una arquitectura que garantice la disponibilidad del lago de datos. Esto a través de la implementación de mecanismos de replicación y tolerancia a fallos, para minimizar el tiempo de inactividad y garantizar que los datos estén accesibles en todo momento.

Se tomó la decisión de diseñar una arquitectura del lago de datos con un solo nodo maestro (NameNode) y cuatro nodos esclavos (DataNodes). Esta elección se basa en la capacidad del servidor de altas prestaciones de proporcionar un rendimiento óptimo para los nodos del clúster y garantizar la escalabilidad y disponibilidad requeridas para manejar grandes volúmenes de datos. La configuración de un único NameNode permitirá un eficiente control y administración del clúster, mientras que la inclusión de cuatro DataNodes facilitará la replicación de datos para mejorar la tolerancia a fallos. Esta arquitectura, respaldada por la experiencia adquirida durante la fase inicial del proyecto, se considera la opción más adecuada para asegurar el éxito en la implementación del lago de datos con Hadoop en el servidor del laboratorio ADA. A continuación, en la figura 2.3 se muestra un esquema global del diseño de la arquitectura del algo de datos mencionada:

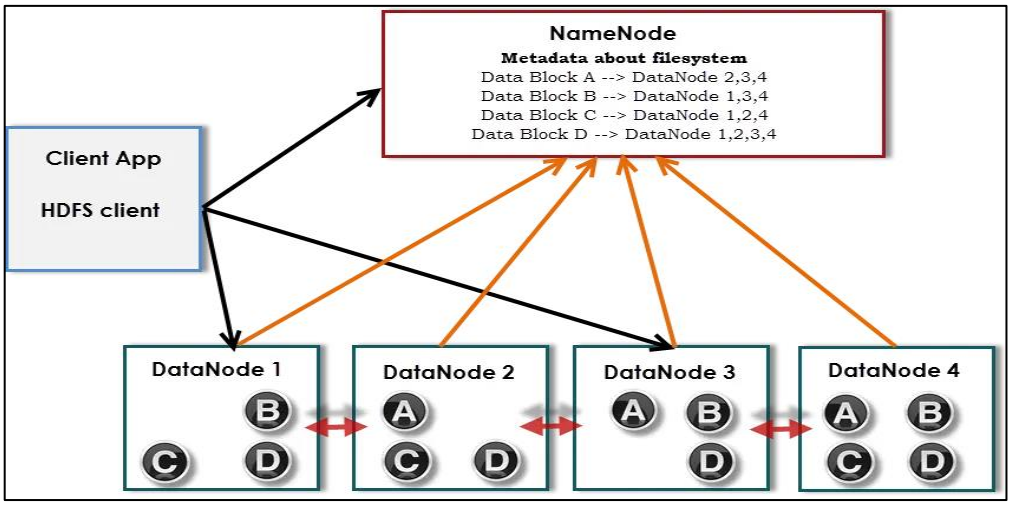

Figura 2.3. Esquema global del diseño de la arquitectura del lago de datos *(Freepng.es, 2023)*.

Además del NameNode y los 4 DataNodes, se debe incorporar el ResourceManager y los NodeManagers a la arquitectura del clúster de 4 nodos con contenedores Docker. El ResourceManager es el componente central del sistema de administración de recursos en Hadoop. Su función principal es coordinar y administrar los recursos del clúster, como la asignación de memoria y CPU a las aplicaciones que se ejecutan en el lago de datos. Los NodeManagers son agentes que se ejecutan en cada DataNode del clúster y son responsables de administrar los recursos locales en ese nodo. Su función es recibir solicitudes de trabajo del ResourceManager y supervisar la ejecución de tareas en ese nodo. A continuación, en la figura 2.4, se muestra el diagrama del funcionamiento del ResourceManager antes descrito.

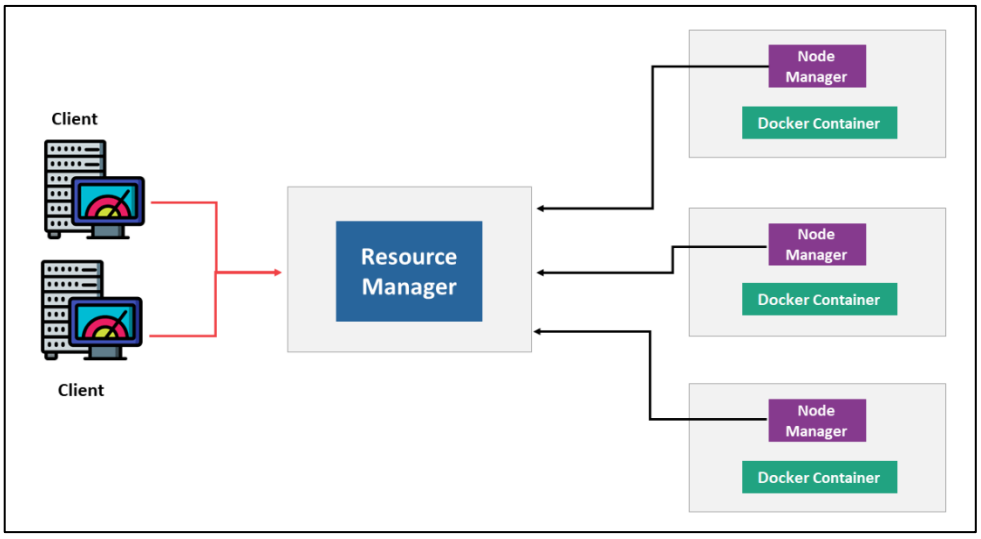

Figura 2.4. Funcionamiento del ResourceManager y NodeManagers *(Subramaniam, 2022)*.

A continuación, se detalla cada componente de la arquitectura del lago de datos, así como también los recursos asignados a los mismos:

NameNode: Su función principal es mantener y administrar los metadatos del sistema de archivos, como la ubicación física y el estado de los bloques de datos distribuidos en el clúster de Hadoop (Ojeda, 2020). También gestiona la replicación de datos para proporcionar tolerancia a fallos.

- Número de contenedores Docker: 1
- Capacidad de almacenamiento: No requiere almacenamiento asignado a parte del establecido por defecto del contenedor Docker (16GB por defecto), ya que no almacena datos de usuario. Solo mantiene los metadatos del sistema de archivos distribuido HDFS
- Memoria RAM: Asignación Dinámica.
- Capacidad de procesamiento: Asignación Dinámica.
- Conectividad de red: Acceso completo a la red interna del clúster y conectividad externa a Internet a través de la red del servidor de altas prestaciones con una velocidad de descarga de 91.74 Mbit/s y una velocidad de carga de 74.01 Mbit/s, ya que debe interactuar con el cliente del lago de datos y este puede encontrarse en una red externa.

ResourceManager: Se encarga de realizar un seguimiento de los recursos de CPU, almacenamiento y memoria RAM disponibles en cada nodo del clúster (representado por los NodeManagers) y administra cómo se asignan estos recursos a las distintas aplicaciones que se despliegan y ejecutan en el clúster (Ojeda, 2020).

- Número de contenedores Docker: 1
- Capacidad de almacenamiento: No requiere almacenamiento asignado a parte del establecido por defecto del contenedor Docker (16GB por defecto), ya que no almacena datos de usuario. Solo mantiene información de los recursos disponibles en cada nodo.
- Memoria RAM: Asignación Dinámica.
- Capacidad de procesamiento: Asignación Dinámica.
- Conectividad de red: Acceso solo a la red interna del clúster.

NodeManager: Su función principal es administrar los recursos locales de cada nodo en el clúster de Hadoop. Se encarga de gestionar la memoria RAM, la capacidad de procesamiento y otros recursos disponibles en el nodo donde se ejecuta (Ojeda, 2020)..

- Número de contenedores Docker: 4
- Capacidad de almacenamiento: No requiere almacenamiento asignado a parte del establecido por defecto del contenedor Docker (16GB por defecto), ya que no almacena datos de usuario. Solo mantiene información de los recursos disponibles en el nodo en que se encuentra.
- Memoria RAM: Asignación Dinámica.
- Capacidad de procesamiento: Asignación Dinámica.
- Conectividad de red: Acceso solo a la red interna del clúster.

DataNodes: Su función principal es almacenar y administrar los bloques de datos de los archivos distribuidos en el clúster. Es responsable de gestionar un conjunto de bloques de datos y garantizar su disponibilidad y replicación en el clúster (Ojeda, 2020).

- Número de contenedores Docker: 4
- Capacidad de almacenamiento: Asignación Dinámica.
- Memoria RAM: Asignación Dinámica.
- Capacidad de procesamiento: Asignación Dinámica.
- Conectividad de red: Acceso solo a la red interna del clúster.

Un aspecto importante a destacar en esta arquitectura es la elección de la asignación dinámica de recursos como memoria RAM, CPU(s) y almacenamiento en disco. La decisión de optar por la asignación dinámica de recursos en la arquitectura del lago de datos fue tomada en base a la experimentación realizada durante la fase inicial del proyecto. A través de las pruebas y análisis llevados a cabo en el servidor de pruebas con diferentes configuraciones de recursos, se pudo comprobar que la asignación dinámica proporcionaba una mayor flexibilidad y eficiencia en comparación con establecer límites y recursos fijos para cada contenedor.

Esta estrategia demostró ser más adaptable a las cargas de trabajo cambiantes y permitió una utilización más óptima de los recursos disponibles. Además, la asignación dinámica

facilitó la escalabilidad y tolerancia a fallos del sistema, lo que respaldó la toma de esta decisión en la arquitectura del lago de datos.

Otra decisión importante en el diseño de esta arquitectura es la de crear un contenedor donde coexistan el DataNode y el NodeManager, es decir, en vez de tener 8 contenedores solo se tendrían 4 contenedores. La elección de crear un contenedor donde coexistan el DataNode y el NodeManager en lugar de colocarlos en contenedores separados se basa en la optimización de recursos y la eficiencia operativa.

Al tener ambos componentes en un mismo contenedor, se minimiza el consumo de recursos y se simplifica la administración y el mantenimiento del clúster (Rosero Abad, 2018). La comunicación interna entre ambos componentes mejora, y en caso de fallo, todo el contenedor se puede reemplazar fácilmente, asegurando la continuidad del servicio. Para agregar más nodos con estos dos servicios, simplemente se debe crear los contenedores Docker con estos servicios y añadirlos a la red interna del lago de datos, de esta manera se solventa la necesidad de escalabilidad del lago de datos.

En el diseño de la arquitectura del lago de datos, es crucial tener en cuenta una serie de aspectos adicionales para garantizar un funcionamiento óptimo y eficiente. Entre estos aspectos se encuentran la inclusión del cliente HDFS, que actúa como interfaz para la manipulación y acceso a los datos en el sistema de archivos distribuido. Asimismo, es necesario considerar la configuración adecuada de la red interna, que permite una comunicación eficiente entre los diferentes componentes del clúster de Hadoop.

El factor de replicación, que determina cuántas copias de cada bloque de datos se almacena en el clúster, debe ajustarse para garantizar la tolerancia a fallos y la disponibilidad de los datos. Además, el tamaño de los bloques de datos influye en el rendimiento y la eficiencia del sistema, ya que bloques más pequeños pueden acelerar la transferencia de datos y la recuperación de fallas. Cabe mencionar que para estos aspectos de la arquitectura no se necesitan recursos del servidor.

De la misma manera, a continuación, se muestra la descripción de los componentes:

Red Interna: Una red interna conecta los contenedores Docker en el lago de datos. Permite comunicación eficiente mediante direcciones IP locales, mejorando el rendimiento y la seguridad al aislar los contenedores en una red privada.

Cliente HDFS: El cliente HDFS desempeña un papel vital al conectar usuarios con el sistema de archivos distribuido HDFS. Funciona como una biblioteca o conjunto de

herramientas que permite acceso y manipulación de datos almacenados. Administra operaciones de lectura/escritura, así como replicación y recuperación de datos en HDFS.

Factor de replicación: El factor de replicación en Hadoop define cuántas copias se guardan por bloque en HDFS. Esto asegura tolerancia a fallos y alta disponibilidad. El factor de replicación de 3 es considerado óptimo en un lago de datos con 4 DataNodes, esto según la documentación oficial de Hadoop, porque ofrece una alta tolerancia a fallos mientras utiliza el espacio de almacenamiento de manera eficiente (Apache Software Foundation, 2023).

Tamaño de bloque: En Hadoop, los archivos se dividen en bloques de tamaño fijo para almacenamiento y procesamiento distribuido. El tamaño de bloque predeterminado de 128 MB, según la documentación oficial de Hadoop, optimiza la eficiencia al reducir metadatos, disminuir sobrecarga de operaciones y mejorar el rendimiento en archivos grandes, minimizando fragmentación y optimizando el espacio de almacenamiento (Apache Software Foundation, 2023).

Descripción del funcionamiento de la arquitectura

Una vez descrito a los elementos que conforman a la arquitectura del lago de datos del presente proyecto, lo siguiente es explicar su funcionamiento. El funcionamiento del lago de datos con Hadoop implica una serie de interacciones y procesos coordinados entre los diferentes componentes. Para lo cual, a continuación, se describe todo el proceso que se lleva a cabo para la escritura y lectura de datos en un lago de datos:

1. Acceso del cliente HDFS al lago de datos.

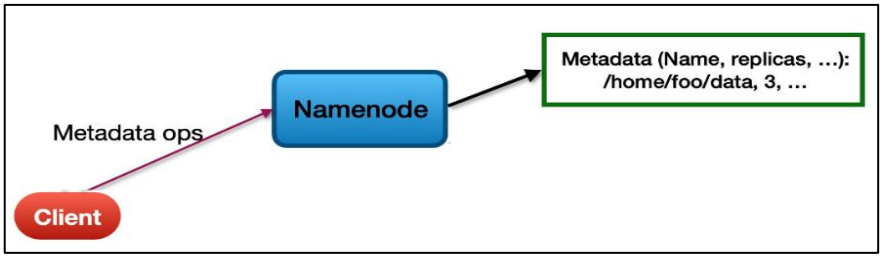

Figura 2.5. Diagrama de interacción del cliente HDFS y el NameNode *(CloudDuggu, 2022)*.

Un usuario o una aplicación que desea almacenar datos en el lago de datos inicia el proceso de escritura enviando una solicitud para guardar un archivo al cliente HDFS. El

cliente HDFS actúa como interfaz para el acceso al HDFS y es responsable de coordinar todas las operaciones relacionadas con el sistema de archivos distribuido. El primer paso del proceso es que el cliente HDFS se ponga en contacto con el NameNode, que es el nodo principal del clúster HDFS y contiene información sobre la ubicación y el estado de los bloques de datos.

Al interactuar con el NameNode, el cliente HDFS solicita espacio de nombres para el archivo que desea almacenar. Esto implica que el cliente proporciona al NameNode el nombre del archivo y su ubicación en el sistema de archivos. A los procedimientos realizados por el NameNode antes mencionados se les conoce como operaciones de metadata. El diagrama de esta interacción se puede observar en la figura 2.5.

2. Selección de DataNodes

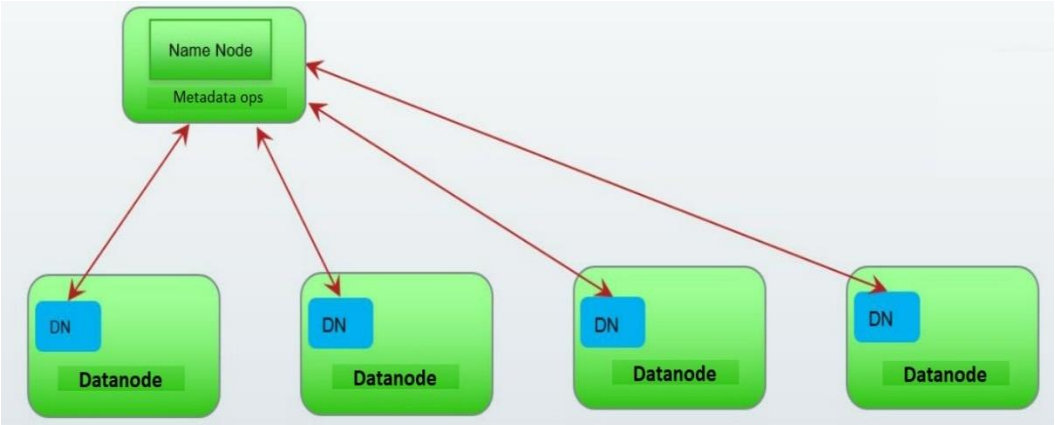

Figura 2.6. Selección de DataNodes para el almacenamiento de los datos *(Kgautam, 2018)*.

El NameNode, después de recibir la solicitud del cliente, realiza la asignación de bloques para el archivo. Esto significa que el NameNode determina qué bloques de datos del sistema de archivos distribuido HDFS se utilizarán para almacenar el archivo en cuestión.

Una vez asignados los bloques, el NameNode selecciona los DataNodes apropiados para almacenar los bloques de datos, como se muestra en la figura 2.6. La selección de DataNodes se realiza de forma aleatoria. Después de tomar todas estas decisiones, el NameNode envía una respuesta al cliente HDFS, que incluye información detallada sobre los bloques asignados, la ubicación de los DataNodes y otros metadatos necesarios para la escritura.

3. Almacenamiento de los datos en bloques.

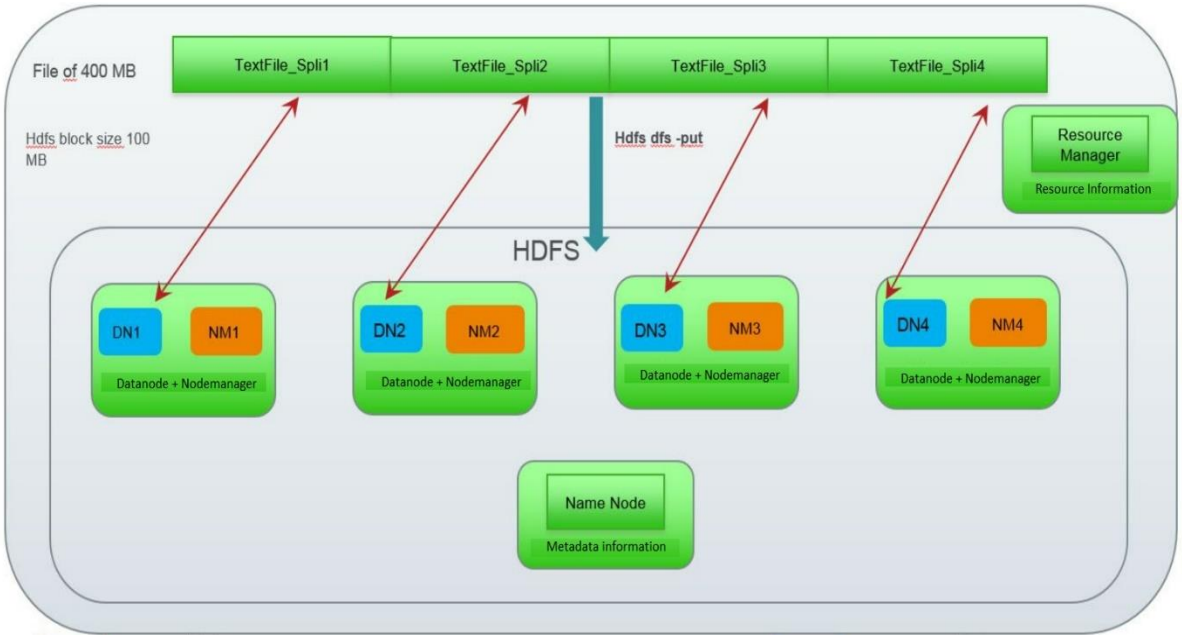

Figura 2.7. División del archivo en bloques para su almacenamiento *(Kgautam, 2018)*.

Una vez que el cliente HDFS recibe la respuesta del NameNode, comienza el proceso de transferencia de datos. El cliente divide el archivo en bloques, y cada bloque se envía secuencialmente al primer DataNode asignado para su almacenamiento, tal y como se muestra en la figura 2.7. El ResourceManager y el NodeManager intervienen en el proceso después de que el cliente HDFS ha realizado la división en bloques y se ha comunicado con el NameNode.

Una vez que el cliente HDFS ha obtenido la información sobre la ubicación de los DataNodes que almacenarán cada bloque de datos, el ResourceManager se encarga de administrar los recursos del clúster y coordinar la ejecución de tareas en los diferentes nodos.

El NodeManager, por otro lado, es responsable de mantener y administrar los metadatos del sistema de archivos en cada DataNode, incluyendo la ubicación física y el estado de los bloques de datos. Además, el NodeManager gestiona la replicación de datos para proporcionar tolerancia a fallos.

4. Proceso de replicación de bloques

Como se mencionó anteriormente, antes de que se realice la replicación, el cliente HDFS divide el archivo en bloques de un tamaño predeterminado (por ejemplo, 128 MB). Cada bloque se trata como una unidad independiente. El cliente HDFS inicia la transferencia del

primer bloque de datos al primer DataNode seleccionado. Una vez que el DataNode ha recibido el bloque, lo almacena en su almacenamiento local. Luego, el DataNode original copia el bloque a los DataNodes adicionales, según el factor de replicación establecido. Por ejemplo, si el factor de replicación es 3, se crearán dos copias adicionales en otros DataNodes, siendo este proceso realizado de forma aleatoria.

La información de la replicación se guarda en el NameNode, que actúa como el punto central de control y metadatos del sistema de archivos distribuido (HDFS). El NameNode mantiene un registro de todos los bloques de datos y su ubicación en el clúster. Esta información se asemeja a una matriz donde se detalla que bloque se encuentra en que nodo.

Un ejemplo de un factor de replicación de 3 con 4 DataNodes, se muestra en la Tabla 2.5. En este caso el Archivo_bloque1 se replicará en los DataNodes 1,3 y 4. De la misma manera, el Archivo_bloque2 se replicará en los DataNodes 2,3 y 4. De igual forma, Archivo_bloque3 se replicará en los DataNodes 1,2 y 3. Por último, el Archivo_bloque4 se replicará en los DataNodes 1,2 y 4.

Tabla 2.5. Ejemplo de información de la replicación almacenada en el NameNode.

Bloque	DataNodes			
	DataNode 1	DataNode 2	DataNode 3	DataNode 4
Archivo_bloque1	Sí	No	Sí	Sí
Archivo_bloque2	No	Sí	Sí	Sí
Archivo_bloque3	Sí	Sí	Sí	No
Archivo_bloque4	Sí	Sí	No	Sí

Para ejemplificar de mejor manera la replicación con un factor de 3, en un lago de datos con 4 DataNodes, se muestra a continuación, en la figura 2.8, de forma gráfica el proceso descrito anteriormente en la Tabla 2.5.

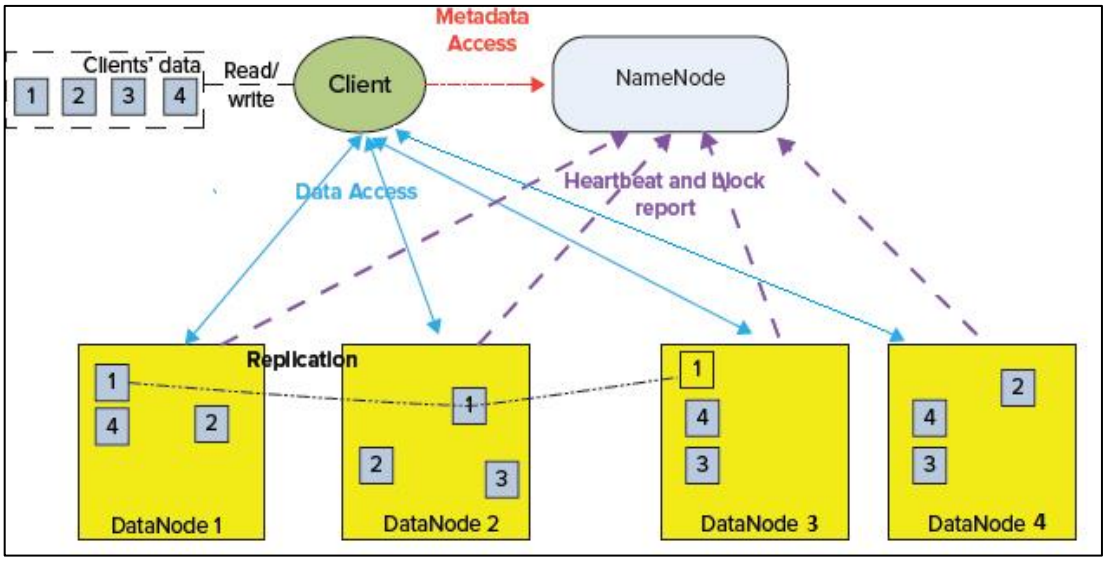

Figura 2.8. Proceso de replicación de bloques en HDFS *(Hvivani, 2014)*.

5. Confirmación de almacenamiento y disponibilidad de uso.

Una vez que todas las réplicas del bloque de datos han sido copiadas en los DataNodes seleccionados, el cliente HDFS recibe una confirmación de que los datos han sido almacenados con éxito en el HDFS. Esta confirmación asegura que el proceso de replicación y almacenamiento se ha realizado de manera exitosa y que los datos están seguros y disponibles para su uso. Después de la confirmación de almacenamiento, el NameNode actualiza los metadatos del sistema de archivos para reflejar la ubicación y el estado de los bloques de datos recién guardados.

Los metadatos incluyen información esencial sobre cada bloque, como su identificador, ubicación, tamaño y factor de replicación. Estos metadatos son esenciales para el correcto funcionamiento del sistema de archivos distribuido, ya que permiten al NameNode mantener un registro actualizado de la topología del clúster y de la ubicación de los datos.

Una vez que los metadatos han sido actualizados, los datos están listos para ser accedidos y procesados por los usuarios y aplicaciones que necesiten trabajar con ellos dentro del lago de datos. Los bloques de datos replicados están distribuidos en los DataNodes, lo que permite un acceso distribuido y paralelo a los datos, mejorando el rendimiento y la eficiencia del sistema. Los usuarios y aplicaciones pueden acceder a los bloques de datos directamente desde los DataNodes más cercanos, lo que reduce la latencia y mejora el tiempo de respuesta en la lectura y escritura de datos.

6. Acceso y Lectura de datos

La lectura de datos en un lago de datos con esta arquitectura sigue un proceso similar al de escritura, pero en sentido inverso. A continuación, se describe cómo se da la lectura de datos:

1. El usuario o la aplicación que desea acceder a los datos interactúa con el cliente HDFS, que actúa como interfaz para acceder al HDFS.
2. El cliente HDFS consulta al NameNode para determinar la ubicación de los bloques de datos que conforman el archivo que se desea leer.
3. El NameNode responde al cliente HDFS proporcionando la ubicación de los bloques de datos.
4. El cliente HDFS inicia la transferencia de datos solicitando a los DataNodes que almacenan los bloques que necesita.
5. Cada DataNode que almacena los bloques de datos solicitados responde al cliente HDFS enviando los datos del bloque.
6. El cliente HDFS recopila los datos de los bloques individuales y los ensambla para reconstruir el archivo original que se desea leer.
7. Una vez que se han recopilado todos los bloques de datos necesarios y el archivo se ha reconstruido, el cliente HDFS entrega los datos al usuario o la aplicación que realizó la solicitud.

Este proceso se resume de forma gráfica a continuación en la figura 2.9.

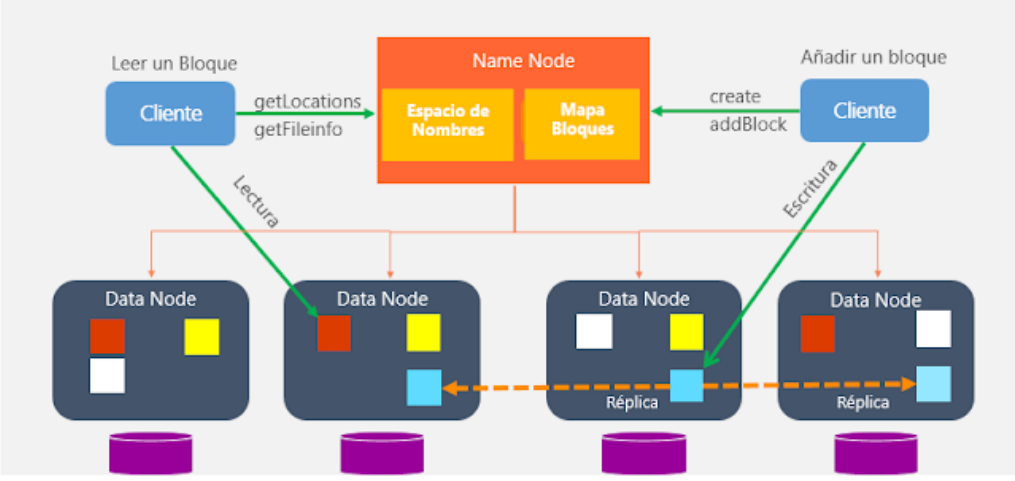

Figura 2.9. Proceso de lectura de datos en el lago de datos *(Recuero, 2021)*.

2.4 Implementación del lago de datos

En esta sección, se abordará el proceso de implementación del lago de datos, utilizando contenedores Docker en el servidor de altas prestaciones del laboratorio ADA. La implementación de esta arquitectura del lago de datos con Hadoop representa un paso crucial en el desarrollo de este proyecto, ya que permitirá almacenar y gestionar grandes cantidades de datos de manera distribuida y escalable.

A lo largo de esta sección, se describirá en detalle cada componente de la arquitectura, su configuración y su funcionamiento, destacando la asignación dinámica de recursos y la integración de NameNode, ResourceManager, DataNodes y NodeManagers.

Además, se abordarán aspectos clave como el factor de replicación y el tamaño de bloques, que garantizan la disponibilidad y la eficiencia del sistema. Con el objetivo de lograr una implementación exitosa, se tendrán en cuenta las especificaciones técnicas del servidor, la interacción con el cliente HDFS, y se describirá el proceso de lectura y escritura de datos en el lago de datos.

Configuración inicial del servidor

Antes de proceder con la implementación del lago de datos, fue necesario realizar una configuración previa del servidor. Se instaló el sistema operativo Ubuntu 22.04.3 LTS para garantizar una base estable y compatible con las herramientas y tecnologías a utilizar. Luego, se procedió a instalar Docker v24.0.5 para facilitar la creación y administración de los contenedores que compondrían el lago de datos.

Además, se realizó una partición del almacenamiento en disco, asignando el suficiente espacio para dar cabida a los datos que se almacenarían en el lago. Esta partición fue de 9.28 TB de capacidad en el servidor del laboratorio ADA, asegurando así que el lago de datos pudiera manejar grandes volúmenes de información de manera eficiente. Cabe mencionar que esta partición se montó sobre el directorio /var/lib que es el directorio donde se instala Docker por defecto, y, por tanto, es en donde se crearan los contenedores Docker de los componentes del lago de datos.

Con estas configuraciones iniciales, se sentó una base sólida para la implementación exitosa del lago de datos, asegurando que el servidor estuviera listo para desplegar y ejecutar adecuadamente todos los componentes de la arquitectura Hadoop en contenedores Docker. Las configuraciones iniciales para la implementación del lago de datos se pueden evidenciar, a continuación, en la figura 2.10.

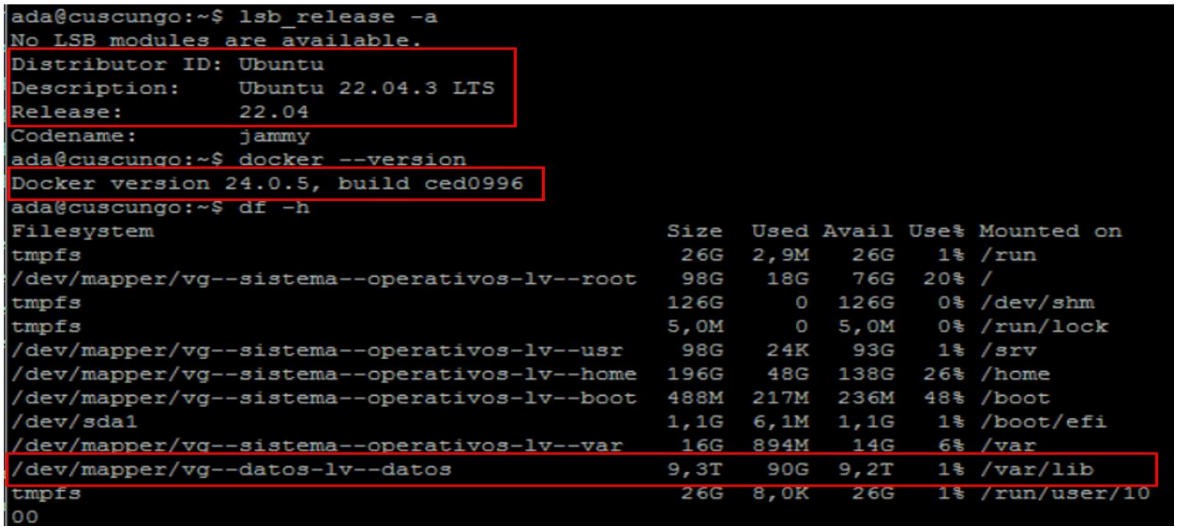

Figura 2.10. Configuraciones previas a la implementación del lago de datos.

Descarga de recursos necesarios

Los recursos necesarios para la implementación del lago de datos se obtuvieron del sitio web oficial de la Universidad Internacional de Valencia, los cuales se pueden descargar de https://recursos.universidadviu.es/int-guia-gratuita-hadoop. Además, para facilitar la posible replicación de este proyecto en un futuro, se ha decidido comprimir todos los recursos necesarios en una sola carpeta, que se puede encontrar en el Anexo I, para su descarga y uso de ser necesario. La carpeta mencionada se le ha denominado como “Hadoop_cluster”, y su contenido se muestra en la figura 2.11.

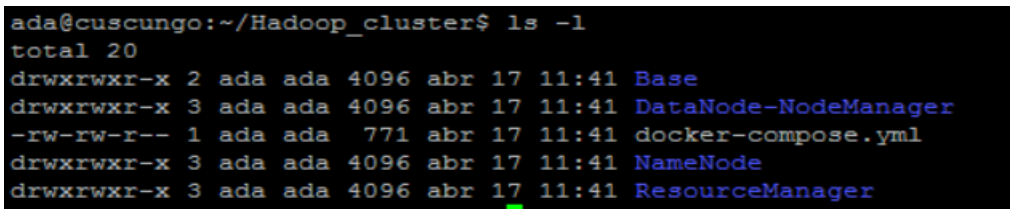

Figura 2.11. Recursos necesarios para la implementación del lago de datos.

Como se puede observar en la figura 2.11, en esta carpeta se encuentra el archivo docker-compose.yml, el cual es un archivo de configuración utilizado en Docker Compose, una herramienta que permite crear y gestionar aplicaciones multi-contenedor de una manera más sencilla. Además, se encuentran carpetas con los nombres de cada componente de la arquitectura del lago de datos, dentro de las cuales se encuentran los archivos necesarios para construir las imágenes Docker de cada componente y archivos adicionales de configuración de cada contenedor, que se explicaran en las siguientes secciones.

Cabe mencionar que, para las secciones posteriores, si se va a realizar la creación de la imagen de determinado componente, primero se debe ingresar a la carpeta con el nombre

de ese componente, así, por ejemplo, si se va a crear la imagen del NameNode primero hay que ingresar al directorio Hadoop_cluster/NameNode/ y ejecutar en ese directorio el comando correspondiente.

Creación de la imagen base de Hadoop en Docker

Una imagen Docker es una unidad de ejecución que contiene todas las dependencias necesarias para ejecutar una aplicación de forma aislada y reproducible. En el contexto de la implementación del lago de datos, crear una imagen base de Hadoop es de vital importancia, ya que proporciona una base sólida y consistente para desplegar el sistema de almacenamiento y procesamiento distribuido.

Esta imagen encapsula el entorno Hadoop con todas sus configuraciones y herramientas en un archivo denominado Dockerfile, lo que permite una rápida y confiable replicación en múltiples contenedores. Al crear una imagen base de Hadoop, se garantiza una implementación coherente y libre de errores, facilitando el despliegue y escalado del lago de datos con eficiencia y manteniendo la compatibilidad con las demás herramientas utilizadas en el proyecto.

En esta imagen se especifican muchos de los aspectos importantes del entorno Hadoop, como es el caso de la versión de Hadoop, la imagen base de la que parte, la instalación del software necesario, se crean variables de entorno y se establecen el directorio donde se instalará Hadoop. Otro aspecto importante es que es en esta imagen donde se debe crear un grupo y un usuario específicos para el manejo de Hadoop a lo largo de la implementación del lago de datos. Todos los datos antes mencionados se resumen a continuación en la Tabla 2.6. Además, en el Anexo II, se muestra el código de la Imagen base de Hadoop, usada para la implementación de este proyecto.

Tabla 2.6. Principales características de la Imagen Base de Hadoop.

Característica de la Imagen Base de Hadoop	Elemento especificado
Imagen Base	ubuntu:latest
Versión de Hadoop	3.3.1
Directorio de instalación	/opt/bd
Grupo creado	hadoop
Usuario creado	hdadmin

Para la creación de esta imagen primero se debe ingresar en la ruta Hadoop_cluster/Base/ y, dado que el sistema operativo donde se desea implementar el lago de datos es Ubuntu,

el comando para la creación de la imagen de nombre hadoop-base-image se muestra a continuación:

```
sudo docker build -t hadoop-base-image .
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `sudo`: Es un comando en sistemas basados en Unix/Linux que se utiliza para ejecutar otro comando con privilegios de superusuario.
- `docker`: Es el comando de la interfaz de línea de comandos de Docker.
- `build`: Es un subcomando de Docker que se utiliza para construir imágenes a partir de un archivo de configuración llamado Dockerfile.
- `-t hadoop-base-image`: El argumento -t se utiliza para etiquetar la imagen que se va a construir. En este caso, la imagen se etiquetará con el nombre hadoop-base-image.
- `.`: Es el contexto del build. Indica que Docker debe buscar el Dockerfile en el directorio actual (.).

Para comprobar que esta imagen se haya construido correctamente, en cualquier directorio, se puede ejecutar el siguiente comando:

```
sudo docker image ls
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `image`: Es un subcomando de Docker que se utiliza para administrar imágenes. En este caso, se está utilizando para listar imágenes existentes.
- `ls`: Es un argumento del subcomando docker image, que se utiliza para listar las imágenes de Docker.

La creación de la imagen base de hadoop se puede evidenciar en la figura 2.12.

```
ada@cuscungo:~/Hadoop_cluster/Base$ sudo docker image ls
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
hadoop-base-image             latest    5f6a84fdfc8d   2 minutes ago    3.07GB
jupyter/datascience-notebook  latest    0e9b2dc11288   16 hours ago     6.03GB
jupyterhub/jupyterhub         latest    3d7d518a3ae1   2 months ago     308MB
ada@cuscungo:~/Hadoop_cluster/Base$
```

Figura 2.12. Comprobación de la imagen hadoop-base-image.

Creación de la imagen para el NameNode

Crear la imagen del NameNode en Docker es de suma importancia para la correcta implementación del lago de datos. Al encapsular el NameNode en una imagen, se asegura la portabilidad y la consistencia del componente en todo el clúster de contenedores. Esto simplifica la configuración y el despliegue, ya que la imagen contiene todas las dependencias y configuraciones necesarias para que el NameNode funcione de manera óptima.

Además, al utilizar Docker, se garantiza un entorno aislado y seguro para el NameNode, evitando posibles conflictos con otras aplicaciones o servicios en el servidor. La imagen del NameNode permite una administración más eficiente y facilita la escalabilidad, permitiendo que el lago de datos crezca y se adapte a las necesidades del laboratorio ADA de manera ágil y controlada.

La imagen del NameNode al partir de la imagen base de hadoop, antes creada, también posee las características presentadas en la Tabla 2.6. Además, es aquí, donde se configuran el factor de replicación, el tamaño de los bloques, se crea el directorio donde se guardará la metada del NameNode. Otro elemento importante es la configuración de la dirección y el puerto base donde escuchará la interfaz de usuario web de NameNode. De la misma manera, se exponen los puertos fundamentales para el funcionamiento del lago de datos de Hadoop. Todos los datos antes mencionados se resumen a continuación en la Tabla 2.7. Además, en el Anexo III, se muestra el código de la Imagen del NameNode, usada para la implementación de este proyecto.

Tabla 2.7. Principales características de la Imagen del NameNode.

Característica de la Imagen del NameNode	**Elemento especificado**
Nombre de la imagen	namenode-image
Imagen Base	hadoop-base-image:latest
Directorio para la metadata	/var/data/hadoop/hdfs/nn
Factor de replicación	3
Tamaño de los bloques	128 MB
Dirección y puerto de interfaz web	localhost:9870
Puertos expuestos	8020 9820 9870 9871

Para la creación de esta imagen primero se debe ingresar en la ruta Hadoop_cluster/NameNode/ y, dado que el sistema operativo donde se desea implementar el lago de datos es Ubuntu, el comando para la creación de la imagen de nombre namenode -image es el siguiente:

```
sudo docker build -t namenode-image .
```

Para comprobar que esta imagen se haya construido correctamente, en cualquier directorio, se puede ejecutar el siguiente comando:

```
sudo docker image ls
```

La creación de la imagen del NameNode se puede evidenciar en la figura 2.13.

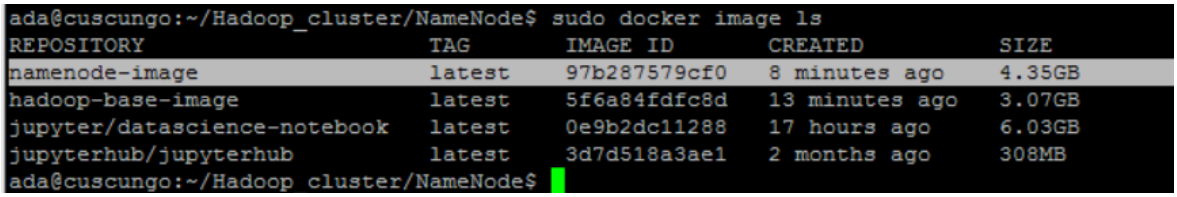

Figura 2.13. Comprobación de la creación de la imagen namenode-image.

Creación de la imagen para el ResourceManager

La imagen del ResourceManager en Docker juega un papel esencial en el funcionamiento armonioso del lago de datos con Hadoop. Al encapsular y estandarizar el ResourceManager en una imagen, se facilita enormemente la administración y despliegue del sistema distribuido. Esta imagen contiene todos los componentes necesarios para una óptima coordinación de recursos, lo que permite una asignación dinámica y eficiente de CPU y memoria en los nodos del clúster.

Con esta capacidad de asignación dinámica, el lago de datos puede aprovechar al máximo los recursos disponibles, optimizando el rendimiento y evitando el desperdicio de capacidades ociosas. Además, la imagen del ResourceManager asegura la escalabilidad del lago de datos, permitiendo que se adapte rápidamente a cambios en la carga de trabajo o en los requerimientos de procesamiento. Esta flexibilidad garantiza una operación fluida y sin interrupciones, incluso ante situaciones de crecimiento inesperado de datos o de solicitudes de recursos.

La imagen del ResourceManager parte de la imagen base de hadoop, antes creada, también posee las características presentadas en la Tabla 2.6. En esta imagen se abordan algunos elementos importantes para el ResourceManager, como es el caso del directorio

de almacenamiento temporal, donde este componente almacena la información acerca de la disponibilidad de recursos que tiene el lago de datos en determinado instante, se establece también la dirección y el puerto de la interfaz web de este componente.

Al igual que en el NameNode, en este caso se exponen los puertos necesarios para el funcionamiento adecuado de este componente. Todos los datos antes mencionados se resumen a continuación en la Tabla 2.8. Además, en el Anexo IV, se muestra el código de la Imagen del ResourceManager, usada para la implementación de este proyecto.

Tabla 2.8. Principales características de la imagen del ResourceManager

Característica de la Imagen del ResourceManager	**Elemento especificado**
Nombre de la imagen	resourcemanager-image
Imagen Base	hadoop-base-image:latest
Directorio de almacenamiento temporal	/var/tmp/hadoop-user
Dirección y puerto de interfaz web	localhost:8088
Puertos expuestos	8030 8031 8032 8033 8088 8090

Para la creación de esta imagen primero se debe ingresar en la ruta Hadoop_cluster/ResourceManager/ y, dado que el sistema operativo donde se desea implementar el lago de datos es Ubuntu, el comando para la creación de la imagen de nombre resourcemanager-image es el siguiente:

```
sudo docker build -t resourcemanager-image .
```

Para comprobar que esta imagen se haya construido correctamente, en cualquier directorio, se puede ejecutar el siguiente comando:

```
sudo docker image ls
```

La creación de la imagen del ResourceManager se puede evidenciar en la figura 2.14.

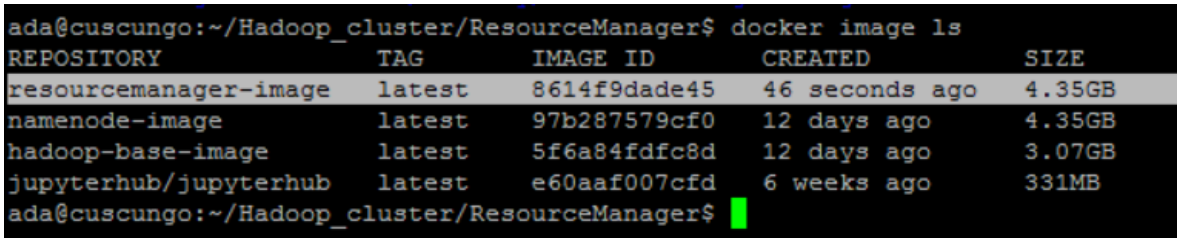

Figura 2.14. Comprobación de la imagen resourcemanager-image.

Creación de la imagen del DataNode y el NodeManager

La creación de una sola imagen para el DataNode y NodeManager en Docker es de suma importancia para implementar el lago de datos con Hadoop de manera efectiva. Al combinar ambos servicios en una sola imagen, se logra una optimización de recursos y una mayor eficiencia en la implementación del lago de datos. Esto se realiza con el fin de que estos componentes coexistan en un solo contenedor, el DataNode y el NodeManager para que compartan recursos y operen en armonía, evitando la duplicación innecesaria de recursos y optimizando tanto el uso de la memoria como la capacidad de procesamiento.

Esta integración también facilita la administración y el despliegue del sistema distribuido, ya que solo es necesario gestionar una única imagen en lugar de dos. Además, la creación de una sola imagen para el DataNode y el NodeManager garantiza una mayor coherencia y compatibilidad entre ambos servicios, lo que reduce la posibilidad de conflictos y errores en el funcionamiento del lago de datos. Esta unificación simplifica el proceso de desarrollo y facilita la escalabilidad del clúster, ya que se puede replicar fácilmente este contenedor para agregar más nodos al sistema.

La imagen del DataNode y NodeManager al partir de la imagen base de hadoop, antes creada, también posee las características presentadas en la Tabla 2.6. En esta imagen se abordan algunos elementos importantes para estos dos componentes, como es el caso del directorio para los datos de HDFS del DataNode, donde este componente almacena los bloques de datos. Al igual que en las imágenes antes creadas, en este caso se exponen los puertos necesarios para el funcionamiento adecuado tanto del DataNode como del NodeManager.

Cabe mencionar que en este caso no existe la dirección y el puerto de la interfaz web de este componente, debido a que no debe permitirse el acceso a estos componentes de forma externa. Todos los datos antes mencionados se resumen a continuación en la Tabla

2.9. Además, en el Anexo V, se muestra el código de la Imagen del DataNode Y NodeManager, usada para la implementación de este proyecto.

Tabla 2.9. Principales características de la imagen del DataNode y NodeManager

Características de la Imagen del DataNode y NodeManager	Elemento especificado
Nombre de la imagen	dnnm-image
Imagen Base	hadoop-base-image:latest
Directorio para los datos HDFS	/var/data/hadoop/hdfs/dn
Puertos expuestos para los DataNodes	9864 9865 9866 9867
Puertos expuestos para los NodeManagers	8040 8042 8044 8048

Para la creación de esta imagen primero se debe ingresar en la ruta Hadoop_cluster/DataNode-NodeManager/ y, dado que el sistema operativo donde se desea implementar el lago de datos es Ubuntu, el comando para la creación de la imagen de nombre dnnm-image es el siguiente:

```
docker build -t dnnm-image .
```

Para comprobar que esta imagen se haya construido correctamente, en cualquier directorio, se puede ejecutar el siguiente comando:

```
sudo docker image ls
```

La creación de la imagen del DataNode y NodeManager se puede evidenciar en la figura 2.15.

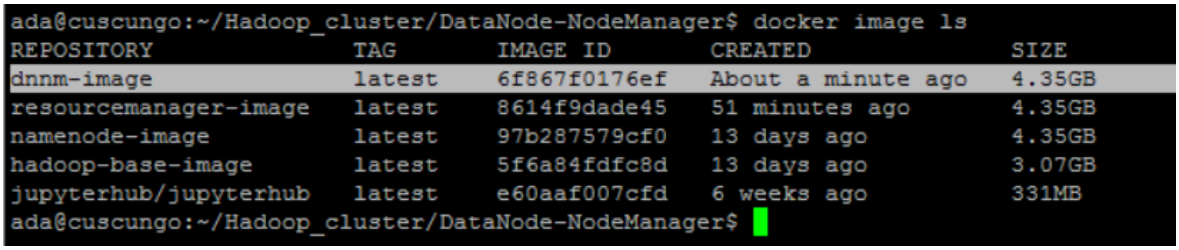

Figura 2.15. Comprobación de la imagen dnnm-image.

Creación de la red interna del lago de datos

La creación de una red interna en el lago de datos con Docker es fundamental para establecer una comunicación eficiente y segura entre los diferentes componentes del clúster. Esta red permite conectar el NameNode, el ResourceManager, el DataNode y el NodeManager, facilitando la coordinación y el intercambio de datos en el sistema

distribuido. La red interna proporciona un entorno aislado para los contenedores, lo que garantiza la integridad y la confidencialidad de los datos.

Además, permite establecer una comunicación directa y de alta velocidad entre los nodos del clúster, lo que mejora el rendimiento y minimiza la latencia en el procesamiento de datos. La conexión de los diferentes componentes del clúster a través de la red interna es esencial para lograr una arquitectura coherente y bien estructurada. Esto facilita la administración y el monitoreo del lago de datos, ya que todos los nodos pueden ser gestionados de manera centralizada.

El nombre con que se ha denominado a la red interna del lago de datos, es hadoop-net, y para su creación se usa el siguiente comando:

```
sudo docker network create hadoop-net
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `network`: Es un subcomando de Docker que se utiliza para administrar redes de contenedores.
- `create`: Este subcomando se utiliza para crear una nueva red.
- `hadoop-net`: En este caso, es el nombre que se ha seleccionado para identificar la red que conectará los diferentes componentes del lago de datos.

Con este comando Docker crea automáticamente una red virtual para los contenedores que formarán parte de esa red, es decir, que no existe ninguna otra intervención del usuario más que la ejecución de dicho comando. Sin embargo, Docker si permite la configuración de una red personalizada. Para la creación de una red se debe tener en cuenta los siguientes aspectos:

- **Driver de Red:** Docker ofrece varios drivers de red, como bridge, overlay, host y none. El driver bridge es el predeterminado y se utiliza para redes locales en un solo host. Si no se especifica un driver, se usará el driver bridge.
- **Asignación de IP:** Por defecto, Docker asigna automáticamente direcciones IP a los contenedores dentro de la red. Cada contenedor obtendrá una dirección IP dentro del rango de direcciones IP del driver bridge.
- **Máscara de Subred:** Para el driver bridge, Docker utiliza la máscara de subred 192.168.16.0/20 por defecto. Esto significa que los contenedores tendrán direcciones IP en el rango 192.168.16.1 hasta 192.168.31.254.

- **Puente de Red:** Docker crea un puente virtual en el host que está conectado a la interfaz de red del host y al que se conectan los contenedores dentro de esa red.
- **DNS:** Docker también configura automáticamente un servidor DNS para la red, permitiendo que los contenedores se refieran entre sí por nombre en lugar de dirección IP.

El próximo paso es comprobar que la red se ha creado correctamente, para esto se debe ejecutar el comando:

```
sudo docker network inspect hadoop-net
```

Los elementos de este comando ya han sido descritos anteriormente, con excepción de "inspect", que es un subcomando de Docker que se utiliza para visualizar información detallada sobre una red existente. A continuación, en la figura 2.16, se puede evidenciar la creación de la red antes descrita.

Figura 2.16. Comprobación de creación de la red hadoop-net.

Creación y ejecución de los contenedores

La creación y ejecución de los contenedores en Docker constituye el paso final y esencial en la implementación del lago de datos en el servidor de altas prestaciones del laboratorio ADA. Los contenedores permiten encapsular y desplegar de manera eficiente cada uno de

los componentes del clúster Hadoop, como el NameNode, ResourceManager, DataNode y NodeManager, garantizando una gestión más sencilla y modular del sistema. Al crear e implementar los contenedores, se asegura la independencia y la portabilidad de cada componente, lo que facilita la replicación del lago de datos en otros entornos y su escalabilidad en caso de requerir más nodos o recursos.

Además, la ejecución de los contenedores en una red interna específica asegura una comunicación confiable y segura entre ellos, fomentando un funcionamiento coherente y óptimo del lago de datos. Con la finalización de este paso crucial, la arquitectura del lago de datos con Hadoop estará completamente implementada y lista para operar, permitiendo el almacenamiento, procesamiento y análisis de grandes volúmenes de datos en el laboratorio ADA, con una infraestructura flexible, eficiente y de alto rendimiento.

A continuación, se presenta el comando, que se puede ejecutar en cualquier directorio, utilizado para crear y ejecutar el contenedor para cada componente del lago de datos. Además, este comando también permite al contenedor creado conectarse a la red interna del lago de datos hadoop-net

```
sudo docker container run --init --detach --name
[nombre_del_contenedor] --network= [nombre_de_red] --hostname
[nombre_host] -p [puerto_host:puerto_contenedor]
[nombre_de_imagen]
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `container`: Es un subcomando de docker que se utiliza para gestionar contenedores.
- `run`: Es un subcomando de docker container que se utiliza para crear y ejecutar un nuevo contenedor a partir de una imagen.
- `--init`: Es una opción que se utiliza para agregar un proceso de inicialización al contenedor. Esto es útil para que el contenedor se detenga correctamente cuando recibe una señal de cierre, lo que evita posibles problemas de apagado.
- `--detach o -d`: Es una opción que se utiliza para ejecutar el contenedor en segundo plano. Esto permite seguir utilizando la línea de comandos sin que el contenedor tome control de la terminal.
- `--name [nombre_del_contenedor]`: Es una opción que se utiliza para asignar un nombre al contenedor que se creará.

- `--network=[nombre_de_red]`: Es una opción que se utiliza para conectar el contenedor a una red Docker existente. Esta red permite la comunicación entre contenedores que forman parte de la misma red.
- `--hostname [nombre_host]`: Es una opción que se utiliza para establecer el nombre de host del contenedor. Esto asigna un nombre de host interno al contenedor.
- `-p [puerto_host:puerto_contenedor]`: Es una opción que se utiliza para mapear el puerto del contenedor al puerto del host. Esto permite acceder al servicio que se ejecuta en el contenedor a través de un determinado puerto en el host.
- `[nombre_de_imagen]`: Es el nombre de la imagen que se utilizará para crear el contenedor.

Una vez explicado cada elemento del comando usado para la creación y ejecución de los contenedores, se procede a presentar el comando utilizado para cada componente del lago de datos:

NameNode:

```
sudo docker container run --init --detach --name namenode --network=hadoop-net --hostname namenode -p 9870:9870 namenode-image
```

ResourceManager:

```
sudo docker container run --init --detach --name resourcemanager --network=hadoop-net --hostname resourcemanager -p 8088:8088 resourcemanager-image
```

DataNodes y NodeManagers:

```
docker container run --init --detach --name dnnm1 --network=hadoop-net --hostname dnnm1 dnnm-image
```

```
docker container run --init --detach --name dnnm2 --network=hadoop-net --hostname dnnm2 dnnm-image
```

```
docker container run --init --detach --name dnnm3 --network=hadoop-net --hostname dnnm3 dnnm-image
```

```
docker container run --init --detach --name dnnm4 --network=hadoop-net --hostname dnnm4 dnnm-image
```

Por último, en cualquier directorio, para comprobar que todos los contenedores se han creado correctamente se debe ejecutar el siguiente comando:

```
docker ps
```

A continuación, en la figura 2.17 se muestran los contenedores creados, de todos los componentes de la arquitectura del lago de datos.

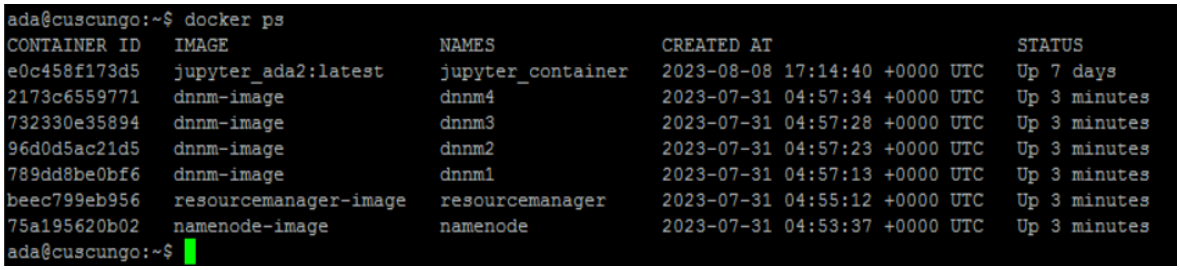

Figura 2.17. Contenedores Docker creados en el servidor del laboratorio ADA.

Por último, si se desea agregar uno o más DataNodes a la arquitectura del lago de datos, esto en caso de que aumenten las capacidades del servidor del laboratorio ADA, simplemente se debe ejecutar el comando para crear un DataNode, y simplemente se debe sustituir el nombre del DataNode. Por ejemplo, si se deseara implementar un quinto DataNode en la arquitectura antes descrita el comando sería el siguiente:

```
docker container run --init --detach --name dnnm4 --network=hadoop-net --hostname dnnm5 dnnm-image
```

Con esto el quinto DataNode se creará y automáticamente pasará a formar parte de la red interna del lago de datos. Este proceso se debe repetir para todos los DataNodes que se desean añadir a la arquitectura del lago de datos.

Autorización de usuarios para uso del lago de datos

En esta sección, se abordará el esencial proceso de autorización de usuarios para acceder al lago de datos. La seguridad y el control de acceso son fundamentales en cualquier sistema de almacenamiento y procesamiento de datos, y en el contexto del presente proyecto, es fundamental asegurar que solo los usuarios autorizados puedan acceder y manipular los datos almacenados en el lago es de suma importancia.

Para que un usuario pueda acceder al lago de datos, primero se debe crear un espacio específico para los datos de dicho usuario. Para esto primero se debe acceder al contenedore del NameNode, lo cual se realiza mediante la ejecución del siguiente comando en cualquier directorio:

```
docker container exec -ti namenode /bin/bash
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `docker container exec`: Este es el comando de Docker para ejecutar comandos en un contenedor en ejecución.
- `-ti`: Estos son indicadores que especifican opciones para la ejecución del comando. "t" se utiliza para asignar una interfaz TTY (Terminal) y "i" permite la interacción interactiva.
- `namenode`: Es el nombre o ID del contenedor en el que se ejecutará el comando. En este caso, el contenedor se llama "namenode".
- `/bin/bash`: Es el comando que se ejecutará dentro del contenedor. En este caso, se está ejecutando el *shell* interactivo de *Bash* en el contenedor "namenode", lo que te permite interactuar con el sistema de archivos y ejecutar comandos en el entorno del contenedor.

Una vez que se haya accedido al contenedor del NameNode lo siguiente es crear un directorio dentro de la ruta /user/ del lago de datos. Para una mejor administración de los archivos del lago de datos, se recomienda crear un directorio con el nombre del usuario en cuestión, por ejemplo, si el usuario es “myUser”, el directorio a crearse debería ser /user/myUser. Para realizar esto, en el contenedor del NameNode, se debe ejecutar el siguiente comando:

```
hdfs dfs -mkdir /user/myUser
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `hdfs dfs`: Es el comando para interactuar con el sistema de archivos distribuido HDFS.
- `-mkdir`: Es una opción del comando que indica que se va a crear un nuevo directorio.
- `/user/myUser`: Es la ruta del directorio que se creará en el sistema de archivos HDFS. En este caso, se crea un directorio llamado "myUser" dentro del directorio raíz "/user".

Por último, se debe asignar como propietario del directorio antes creado, al usuario al que se desea otorgar el acceso al lago de datos. Para lo cual se debe ejecutar, en el contenedor del NameNode, el siguiente comando:

```
hdfs dfs -chown myUser /user/myUser
```

A continuación, se describe la funcionalidad de cada elemento del comando:

- `-chown myUser`: Es el indicador que especifica que se va a cambiar el propietario del archivo o directorio. En este caso, se está cambiando el propietario al usuario "myUser".
- `/user/myUser`: Es la ruta del archivo o directorio al que se le va a cambiar el propietario. En este caso, se está cambiando el propietario del directorio "/user/myUser".

Con esto se garantiza que el usuario tiene acceso al lago de datos y que, además, tiene un espacio específico para el almacenamiento de sus datos. Es importante mencionar que cualquier aplicación que desee acceder al lago de datos, debe hacerlo mediante un usuario previamente configurado y autorizado en el lago de datos. Todo este proceso, se muestra a continuación en la figura 2.18.

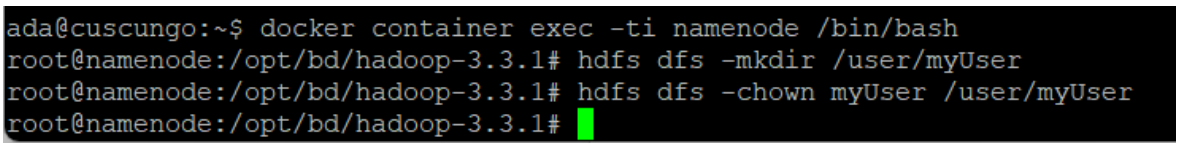

Figura 2.18. Comandos para la autorización y acceso de usuarios al lago de datos.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

La sección de resultados muestra el logro tangible de la implementación del lago de datos en el laboratorio ADA. Aquí, se presentarán los logros obtenidos, la efectividad de la arquitectura diseñada y la operatividad de los componentes clave. Los resultados no solo demostrarán la capacidad del lago de datos para gestionar grandes volúmenes de información, sino también el contraste entre el uso de un ordenador convencional y el uso de los beneficios del lago de datos, el cual se ha descrito en la sección de Evaluación del funcionamiento del lago de datos.

3.1 Resultados

Red interna del lago de datos

Una forma de comprobar que la red interna de Hadoop está funcionando correctamente es mediante la utilización del comando para inspeccionar una red Docker en el servidor donde se ha implementado el lago de datos:

```
sudo docker network inspect hadoop-net
```

Al ejecutar este comando, se puede obtener información detallada sobre todos los contenedores que forman parte de la red "hadoop-net" creada para el lago de datos.

Al inspeccionar esta red en el servidor, se puede verificar que todos los contenedores tienen sus respectivas direcciones IP asignadas dentro de la red interna. Esto indica que los contenedores están correctamente conectados y pueden comunicarse entre sí a través de la red interna de Hadoop. En la figura 3.1, se puede comprobar que en la red del lago de datos están conectados todos los contenedores creados y se les ha asignado direcciones IP para su correcta comunicación.

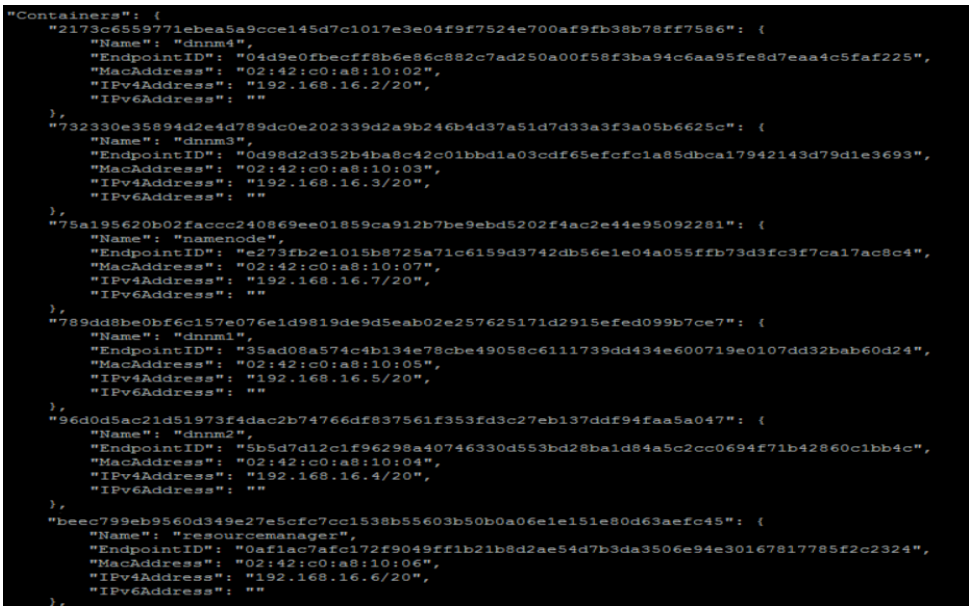

Figura 3.1. Resultado de la configuración de la red interna del lago de datos.

Para ejemplificar de mejor manera los resultados obtenidos con la configuración de la red interna del lago de datos, a continuación, en la figura 3.2 se muestra en un diagrama las direcciones IP asignadas a cada componente del lago de datos.

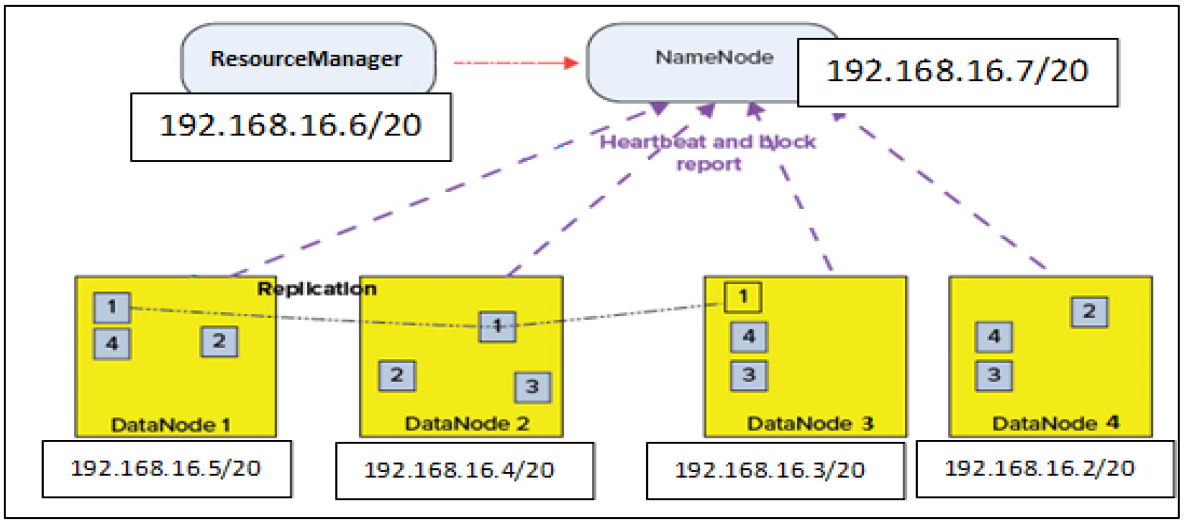

Figura 3.2. Diagrama de la red interna del lago de datos.

Funcionamiento del NameNode, ResourceManager, DataNodes y NodeManagers.

Comprobar el funcionamiento de todos los componentes de Hadoop es esencial para asegurar una arquitectura del lago de datos eficiente y confiable. Una manera de realizar esta verificación es asegurarse de que el NameNode y el ResourceManager puedan leer y comunicarse correctamente con los DataNodes. Esto implica validar que los DataNodes estén disponibles y accesibles para el NameNode y el ResourceManager, lo que garantiza una correcta asignación de tareas y recursos en el clúster.

Además, esta comprobación permite detectar posibles problemas de conectividad o configuración que puedan afectar el rendimiento general del lago de datos. Al asegurar que el NameNode y el ResourceManager puedan interactuar adecuadamente con los DataNodes, se asegura un funcionamiento fluido del sistema, una distribución efectiva de la carga de trabajo y la tolerancia a fallos.

Una manera efectiva de comprobar el funcionamiento del NameNode, y los DataNodes en el lago de datos con Hadoop es accediendo a su interfaz web a través de la dirección "http://localhost:9870". Al ingresar en esta dirección, se mostrará la interfaz web del NameNode, donde se puede verificar la disponibilidad y el estado de los DataNodes conectados al clúster.

En la interfaz web del NameNode, se puede observar la lista de DataNodes registrados en el sistema, lo que indica que la comunicación entre el NameNode y los DataNodes está

funcionando correctamente. Esta comprobación permite asegurarse de que los nodos están correctamente configurados y que la arquitectura del lago de datos está operativa y lista para procesar y almacenar datos. En la figura 3.3, se puede evidenciar el acceso a la interfaz web del NameNode, y se puede comprobar su correcta comunicación con los DataNodes.

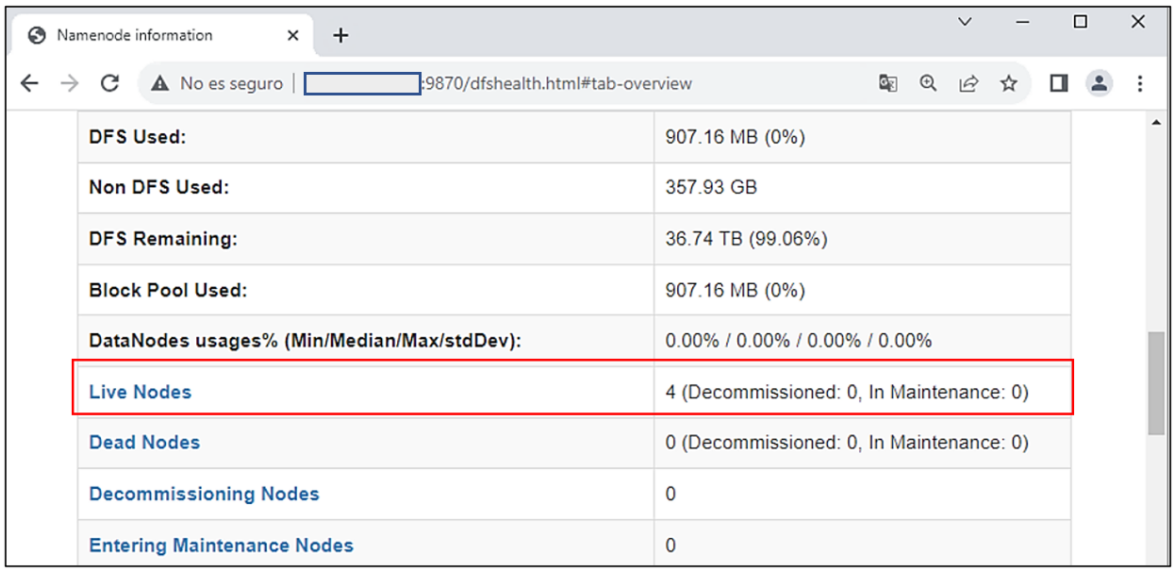

Figura 3.3. Comprobación de la detección de los DataNodes en la interfaz web del NameNode.

Por otro lado, una forma práctica de verificar el correcto funcionamiento del ResourceManager y la comunicación dentro de la red interna en el lago de datos con Hadoop es accediendo a su interfaz web a través de la dirección "http://localhost:8088". Al ingresar en esta dirección, se mostrará la interfaz web del ResourceManager, donde se puede comprobar el estado de los recursos y tareas gestionadas en el clúster. Además, al acceder a la interfaz web del ResourceManager, también es fundamental comprobar que los cuatro DataNodes se encuentren visibles en el clúster.

La presencia de los cuatro DataNodes en la interfaz web confirma que la red interna está funcionando adecuadamente y que los nodos de almacenamiento se han conectado correctamente al ResourceManager. Esta visualización de los DataNodes proporciona una verificación adicional de la correcta comunicación y coordinación entre los componentes del lago de datos con Hadoop, asegurando así que el sistema esté listo para manejar las operaciones de almacenamiento. El acceso a la interfaz del ResourceManager y la correcta detección de los 4 DataNodes se muestra en la figura 3.4.

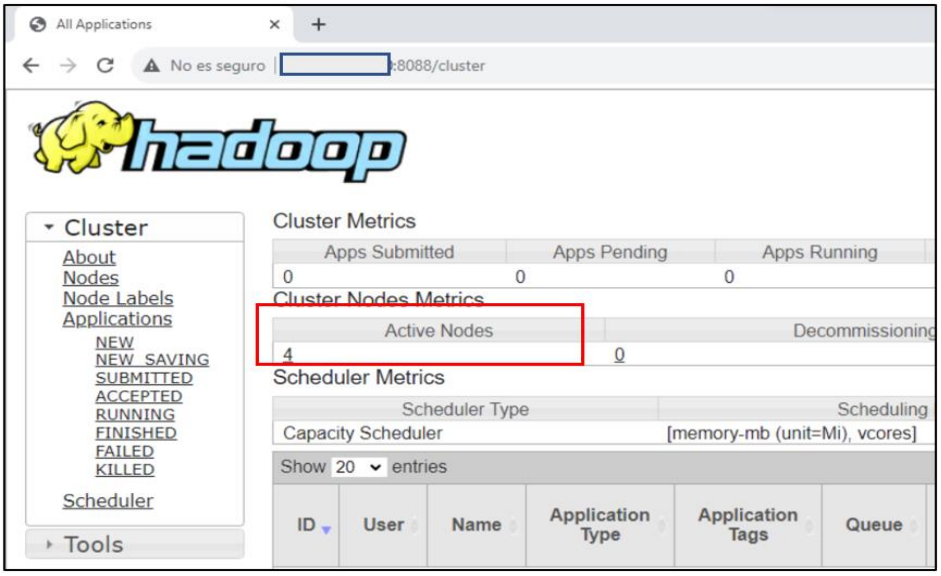

Figura 3.4. Comprobación de la detección de los DataNodes en la interfaz web del ResourceManager.

Subida de archivos al lago de datos

Para comprobar el funcionamiento general del lago de datos se ejecutó un script de análisis de datos, propiedad de uno de los usuarios del laboratorio ADA. La ejecución de este código se realizó en una interfaz web de un contenedor Docker creado a partir de la imagen de Jupyter Notebook, el cual está alojado en el mismo servidor donde se implementó el lago de datos.

Específicamente el papel del lago de datos es almacenar los datos que usará posteriormente el usuario para realizar sus respectivos análisis. En dicho código se carga al lago de datos el archivo abstract.csv que tiene un tamaño de 185 MB. En este punto es importante mencionar que el código usado para guardar datos desde la interfaz de Jupyter es el que se muestra en la figura 3.5.

Figura 3.5. Código usado para subir archivos al lago de datos.

Tamaño de bloques y factor de replicación

Para verificar los resultados de la configuración de estos aspectos en el lago de datos, se debe acceder a la interfaz web del NameNode y comprobar que el factor de replicación y el tamaño de los bloques de datos, sean los mismos que se configuraron en la parte inicial de la implementación.

Para el caso del factor de replicación, una vez accedido a la interfaz del NameNode, se debe dirigir al menú *Utilities*, para luego dirigirse al submenú *Browse the file System*, en la nueva pantalla que se presenta se debe ingresar en el directorio user/hdadmin/. En este directorio se debe mostrar el archivo subido al lago de datos por el usuario, con información adicional, como los permisos, el propietario, el grupo, el tamaño del archivo y la última modificación.

Sin embargo, los datos más importantes en esta evaluación es observar el factor de replicación (*Replication*) y el tamaño de los bloques (*Block Size*), que en este caso deben ser de 3, y de 128 MB, respectivamente, de ser así, se puede asegurar el correcto funcionamiento del lago de datos. La evidencia del resultado correcto de esta configuración se muestra a continuación en la figura 3.6.

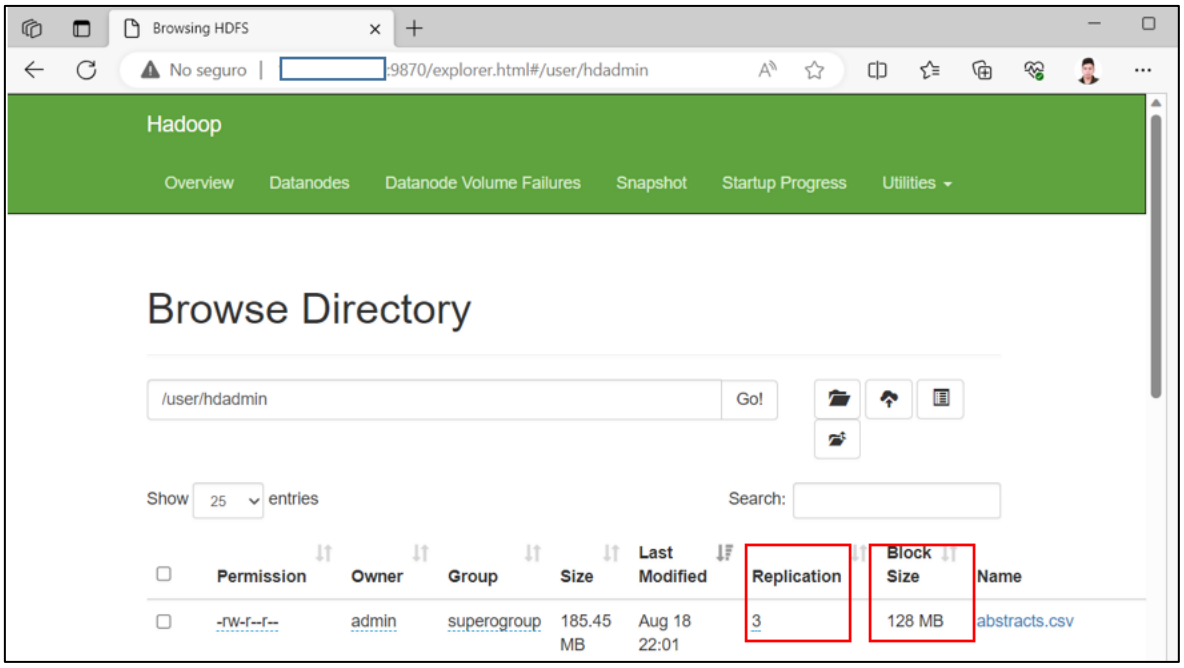

Figura 3.6. Resultados del factor de replicación y tamaño de bloques.

Otro aspecto importante a evaluar es que el archivo subido al lago de datos, se haya dividido en bloques y que se haya replicado en los diferentes DataNodes. Para esto, se debe ingresar en el nombre del archivo y aparecerá una nueva ventana llamada *File Information*, en la que se puede observar los bloques generados y en que DataNodes se ha replicada cada uno. Un ejemplo de lo antes mencionado se muestra a continuación en la figura 3.7.

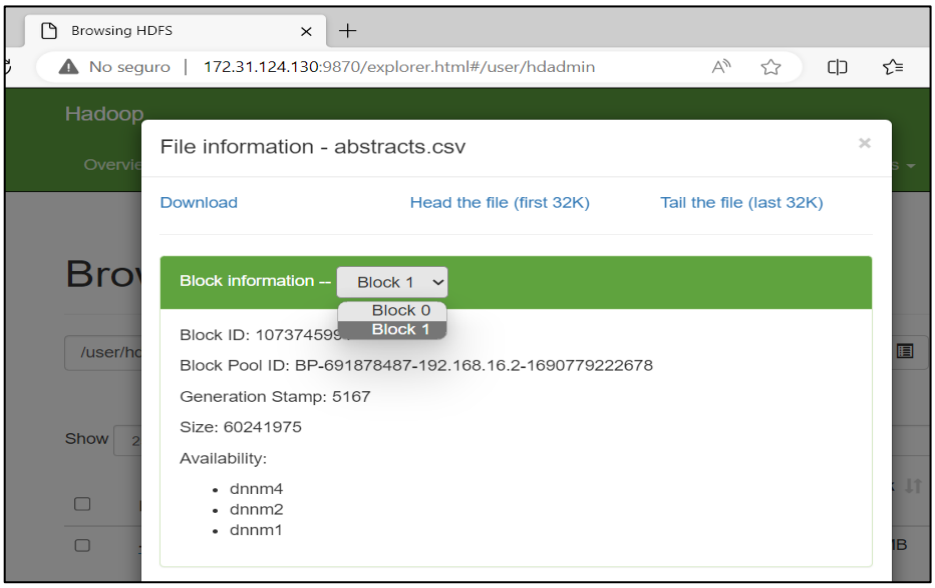

Figura 3.7. Resultados de la división y distribución de los bloques.

Como se puede observar en la figura 3.7, el archivo abstract.csv se ha dividido en dos bloques el Block 0 y Block 1, este resultado es el esperado teniendo en cuenta que el tamaño de bloque se estableció de 128 MB y el tamaño del archivo en cuestión tiene un tamaño de 185 MB. También se puede observar el resultado de la distribución de los bloques, como ejemplo se tiene al bloque 1 que se ha distribuido en los DataNodes 1, 2 y 4 con lo cual se puede verificar que el factor de replicación de 3 en el lago de datos está funcionando de forma correcta.

Lectura de archivos del lago de datos

Ahora en el caso de la lectura se usa el mismo código presentado en la figura 3.3, para permitir la interacción entre Python y el lago de datos. En este caso se realizó la lectura del mismo archivo abstract.csv que se subió anteriormente al lago de datos. El resultado de la ejecución de este código se muestra en la figura 3.8.

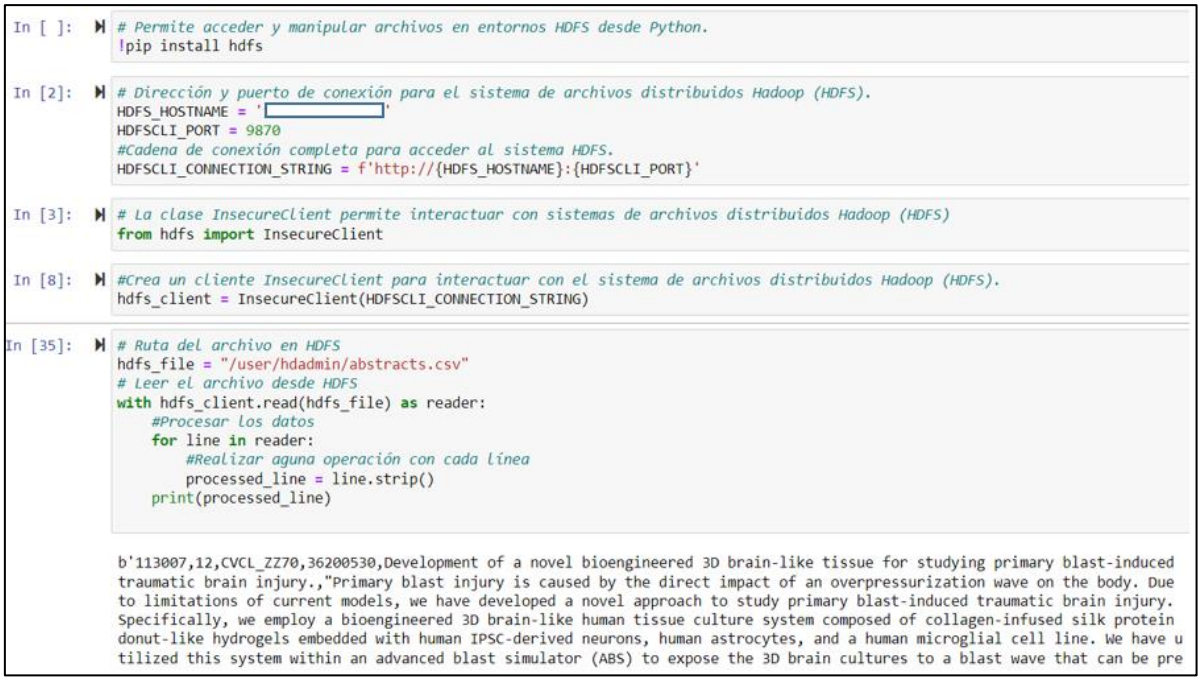

Figura 3.8. Código usado para leer archivos del lago de datos.

Como se puede observar en la figura 3.8, el usuario ha podido leer correctamente los datos del archivo que se encuentra en el lago de datos. Con este resultado se puede comprobar que el lago de datos funciona adecuadamente y permite tanto la subida de datos como su lectura para poder realizar posteriormente trabajos de análisis de datos.

Recepción del proyecto y manual técnico

En la validación del funcionamiento del lago de datos, uno de los usuarios finales del laboratorio ADA realizó con éxito su trabajo de análisis de datos. Estas pruebas confirmaron la efectividad y estabilidad de la implementación, ya que no se experimentaron problemas técnicos ni interrupciones durante la ejecución de sus tareas, validando así el rendimiento y la adecuación del lago de datos a las necesidades reales.

Asimismo, como parte de la culminación del proyecto, se ha desarrollado un manual técnico que detalla el proceso completo de creación y gestión del lago de datos. Este manual está dirigido tanto a los usuarios que realizarán análisis de datos en el lago como a los administradores encargados de mantener su operatividad. El manual abarca desde la configuración inicial del servidor con las herramientas necesarias hasta la creación de contenedores, el despliegue de componentes clave como el NameNode y el ResourceManager. Este recurso resulta fundamental para asegurar que tanto los usuarios como los administradores puedan aprovechar al máximo todas las capacidades del lago de datos. Este manual se puede evidenciar en el Anexo VI.

Además, se realizó la entrega oficial del proyecto al coordinador del laboratorio ADA. La aprobación se llevó a cabo durante una reunión vía Microsoft Teams el miércoles 16 de agosto del 2023. En esta reunión, tanto la directora de este trabajo como el coordinador evaluaron el proyecto, su implementación y funcionamiento. La aprobación final y la validación del coordinador de la calidad y el valor de este lago de datos en el contexto de los trabajos de análisis de datos del laboratorio ADA. La entrega oficial de este proyecto se puede evidenciar mediante un acta de la reunión mencionada, la cual se muestra en el Anexo VII.

3.2 Conclusiones

Se ha logrado obtener un conocimiento profundo del estado actual de las tecnologías usadas en los lagos de datos. La revisión de trabajos de investigación, estudios de casos y proyectos similares, reveló la flexibilidad y escalabilidad de los lagos de datos para integrar múltiples fuentes de información, permitiendo un acceso y análisis más ágil de los datos. Además, se destacó su capacidad para gestionar datos y respaldarlos mediante su replicación en un clúster de varios nodos. La adopción de tecnologías de código abierto y el enfoque en la seguridad y privacidad fueron aspectos sobresalientes en la investigación realizada. En conclusión, en este estudio se ha realizado una revisión de la documentación

existente en el campo de los lagos de datos que ha proporcionado una base sólida para comprender la tecnología de lagos de datos y su aplicación en el almacenamiento de grandes volúmenes de datos.

Se ha llevado a cabo un proceso de recopilación de requerimientos, a través de entrevistas y reuniones de trabajo con usuarios finales, administradores del laboratorio ADA y otros Stakeholders relevantes. Se logró identificar los requisitos clave para el diseño y funcionamiento del lago de datos. Se destacó la importancia de la escalabilidad, la seguridad, la interoperabilidad con sistemas existentes y la capacidad de análisis avanzados. Además, se consideraron restricciones técnicas y presupuestarias para garantizar la viabilidad del proyecto.

Se ha conseguido diseñar una arquitectura integral y detallada para el lago de datos, basada en los requerimientos identificados por las partes interesadas y las especificaciones técnicas del servidor de altas prestaciones del laboratorio ADA. Se seleccionaron tecnologías y herramientas apropiadas que garantizan la eficiencia y escalabilidad del sistema. La arquitectura integral incluye componentes para la ingestión, almacenamiento, procesamiento y análisis de datos, además de considerar mecanismos de seguridad y respaldo. En conclusión, la arquitectura del lago de datos presentada en este documento se diseñó alineando las soluciones técnicas con las necesidades y requisitos identificados previamente, asegurando así un sistema coherente y altamente funcional.

Se ha logrado implementar un lago de datos sobre un clúster compuesto por varios nodos, siguiendo los requerimientos técnicos y de rendimiento establecidos por el laboratorio ADA. Se llevó a cabo la configuración y el despliegue de todos los componentes previamente diseñados en la arquitectura del lago de datos. Se realizaron pruebas exhaustivas para comprobar la integridad y el correcto funcionamiento de cada componente y asegurar la coherencia del sistema. En conclusión, se alcanzó el objetivo de implementar un lago de datos con herramientas de código abierto, alineado con una arquitectura diseñada previamente, que cumple con los requerimientos de funcionalidad y rendimiento, y que, además, fue probada y validada por los usuarios finales del laboratorio ADA.

Se verificó el funcionamiento de todos los componentes del lago de datos mediante la ejecución de un script de análisis de datos por parte de un equipo de analistas del laboratorio ADA. Esta prueba consistió en ejecutar el script, antes mencionado, en una computadora personal, para luego, ejecutarlo haciendo uso de los beneficios del lago de datos de este proyecto, y posteriormente realizar una comparación. Los resultados de esta prueba comprobaron las ventajas que conlleva el uso de un lago de datos en el campo del

análisis de datos, ya que se pudo observar una mejora evidente en el tiempo de carga, descarga y procesamiento de datos en comparación a una computadora personal.

3.3 Recomendaciones

Se recomienda llevar a cabo una correcta configuración del servidor antes de iniciar la implementación del lago de datos. Esto se debe a que una partición de almacenamiento adecuada para todos los componentes es esencial para el éxito del sistema. Si no se proporciona suficiente espacio de almacenamiento, los contenedores Docker de los DataNodes podrían enfrentar limitaciones para replicar adecuadamente los bloques de datos, lo que afectaría la confiabilidad y el rendimiento del lago de datos en general. Una configuración cuidadosa y bien planificada asegura que el clúster cuente con la capacidad de almacenamiento necesaria para manejar grandes volúmenes de datos y permitir operaciones eficientes de lectura y escritura. Asimismo, se reducirían los riesgos de errores y problemas en la fase de implementación, asegurando un ambiente óptimo para el desarrollo y uso de los servicios del lago de datos.

Se recomienda realizar un análisis previo de los requerimientos y una detallada investigación sobre las demandas que enfrentará el lago de datos, especialmente en lo que respecta al tamaño de los datos que serán almacenados. Este análisis permitirá obtener una visión clara de la escala del sistema y las necesidades de almacenamiento, lo que resultará fundamental para tomar decisiones informadas al elegir el factor de replicación y el tamaño de bloques adecuados. Un factor de replicación bien ajustado garantizará la alta disponibilidad y tolerancia a fallos del lago de datos, mientras que un tamaño de bloques adecuado optimizará la eficiencia de lectura y escritura. Al conocer las demandas reales y anticipar el crecimiento futuro de los datos, se evitarán problemas de capacidad insuficiente y se asegurará un rendimiento óptimo y sostenible del lago de datos a lo largo del tiempo.

Se recomienda realizar pruebas con varias configuraciones de lago de datos, empezando con un DataNode y progresivamente aumentando el número, como 2, 4, etc. Este enfoque permitirá identificar la configuración óptima que cumpla con las necesidades de la organización, en este caso el laboratorio ADA. Al evaluar diferentes escenarios, se podrá determinar la cantidad adecuada de nodos necesarios para alcanzar el equilibrio entre rendimiento, escalabilidad y costo. Estas pruebas también brindarán información valiosa sobre el comportamiento del sistema en diferentes cargas de trabajo y el impacto en el rendimiento con el aumento de nodos. Al encontrar la configuración más adecuada, se podrá diseñar una arquitectura sólida y eficiente que garantice un almacenamiento y

análisis de datos óptimos, asegurando así una solución adaptada a sus necesidades y objetivos específicos.

4 REFERENCIAS BIBLIOGRÁFICAS

Aguilar, L. J. (2018). COMPUTACIÓN EN LA NUBE: Notas para una estrategia española en cloud computing. *Revista del Instituto Español de Estudios Estratégicos, 00*(0), 92-112.

Apache Software Foundation. (2023). *Arquitectura HDFS*. Obtenido de https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Replication

Aucancela, M., Naranjo, M., & Betún, J. (2018). Mapeo sistematico de literatura de un data lake. *mktDESCUBRE*, 50-66.

Balboa, M. (2022). Implementación de un Data Lake para la centralización de datos analíticos y transaccionales en la empresa Belcorp.

Balón de la Cruz, M. L. (2022). Propuesta de una guía de adopción en un Data Lake enmarcando los aspectos importantes de Cloud Computing. *Caso de estudio: Federación de Organizaciones Populares Peninsulares FOPOPE (Bachelor's thesis, La Libertad: Universidad Estatal Península de Santa Elena. 2022).*

Beltrán, V. (2022). Implementación de un Data Lake de los datos de uso del LMS Canvas de la Universidad Internacional SEK.

Carvajal, L. (2006). *Metodología de la Investgación Científica. Curso general y aplicado* (28 ed.). Santiago de Cali: U.S.C.

Cherradi, M., Bouhafer, F., & EL Haddadi, A. (2023). Data lake governance using IBM-Watson knowledge catalog. *Scientific African 21(18)*, e01854.

CloudDuggu. (2022). *HDFS Architecture*. Obtenido de Introduction to Apache Hadoop HDFS: https://www.cloudduggu.com/hadoop/hdfs/

Del Vecchio, J., Fabián, P., & Henríquez, C. (2015). Cloud computing: a model for the development of enterprises. *Prospect*, 81-87.

Díaz Palacios, j. (2020). Evaluación e implantación de la distribución Cloudera Distribution Hadoop para sistemas Big Data. *Tesis de Licenciatura*.

Fez, I., Arce, P., Belda, R., & Guerri, J. (2021). Uso de contenedores Docker en el entorno educativo . *In IN-RED 2021: VII Congreso de Innovación Edicativa y Docencia en Red*, 905-915.

Fonticiella Torre, M. Á. (2009). Diseño e implementación del servicio de almacenamiento S3 de Amazon.

Freeman, A. (2017). Docker Compose. *Essential Docker for ASP. NET Core MVC*, 97-117.

Freepng.es. (2023). *Sistema De Archivos Distribuido De Hadoop De Apache Hadoop Distributed File System*. Obtenido de https://www.freepng.es/png-aa9trj/

García, A., & Olmedo, J. (2021). Arquitectura distribuida de alta disponibilidad para la deteccion de fraude. *Revista Cubana de Ciencias Informáticas*, 199-224.

Giebler, C., Groger, C., Hoos, E., Eichler, R., Schwarz, H., & Mitschang, B. (2021). The Data Lake Architecture Framework. *Datenbanksysteme für Business, Technologie und Web (BTW)*, 351-370.

Guayas Espín, A. S. (2023). Análisis comparativo de las plataformas Amazon Cloud, Google Cloud, Azure Cloud. *Tesis de Licenciatura. Babahoyo: UTB-FAFI*.

Guevara, G., Verdesoto, A., & Castro, N. (2020). Metodologías de investigación educativa (descriptivas, experimentales, participativas, y de investigación-acción). *Recimundo, 4(3)*, 163-173.

Hvivani. (19 de 07 de 2014). *Arquitectura HDFS*. Obtenido de https://vivani.net/2014/07/19/arquitectura-hdfs/

Karuna, T., & Praseetha, N. (2020). Data Lake: A centralized repository. *International Research Journal of Engineering and Technology (IRJET)*, 2978-2981.

Kgautam. (08 de 30 de 2018). *Understanding basics of HDFS and YARN*. Obtenido de https://community.cloudera.com/t5/Community-Articles/Understanding-basics-of-HDFS-and-YARN/ta-p/248860

Khine, P. P., & Wang, Z. S. (2018). Data lake: a new ideology in big data era. *ITM web of conferences EDP Sciences*, 03025.

Leopoldo, P. (2017). Consideraciones acerca del Design Thinking y Procesos. *Gestão & Tecnologia, 17(3)*, 312-332.

Llanque, A. (2021). Estudio comparativo de metodologías centradas en el usuario para la definición de requisitos de software desde la perspectiva de la ISO/IEC/IEEE 29148: 2018.

Madera, C., & Laurent, A. (2016). The next information architecture evolution: the data lake wave. *Proceedings of the 8th international conference on management of digital ecosystems*, 174-180.

Mathis, C. (2017). Data Lakes. *Datenbank-Spektrum*, 289-293.

Megdiche, I., Ravat, F., & Zhao, Y. (2020). A use case of data lake metadata management. *Data Lakes*, 97-122.

Mehmood, H., Gilman, E., Cortes, M., Kostakos, P., Byrne, A., Valta, K., . . . Riekki, J. (2019). Implementing big data lake for heterogeneous data sources. *2019 ieee 35th international conference on data engineering workshops (icdew)*, 37-44.

Mejía, O. (2011). Computación en la nube. *ContactoS*, 45-52.

Munshi, A. A., & Mohamed, Y. A.-R. (2018). Data lake lambda architecture for smart grids big data analytics. *IEEE Access*, 40463-40471.

Murillo, J., Alonso, A., García, L., León, I., García, E., Gil, B., & Ríos, L. (2011). *Métodos de investigación de enfoque experimental.* Obtenido de Posgrado UNE: https://www.postgradoune.edu.pe/pdf/documentos-academicos/ciencias-de-la-educacion/10.pdf

Ojeda, M. (2020). Implementación de data lake en la nube para análisis de movilidad en la Ciudad de México y la ZMG.

Ponsico, M. (2017). Tecnología de Contenedores Docker. *Tesis de Licenciatura. Universitat Politècnica de Catalunya.*

Rad, B., Bhatti, H., & Agmadi, M. (2017). An introduction to docker and analysis of its performance. *International Journal of Computer Science and Network Security (IJCSNS)*, 228.

Raju, R., Mital, R., & Finkelsztein, D. (2018). Data Lake Architecture for Air Traffic Management. *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, 1-6.

Recuero, P. (27 de 07 de 2021). *Hadoop por dentro (II): HDFS y MapReduce*. Obtenido de https://palomarecuero.wordpress.com/2021/07/27/hadoop-por-dentro-ii-hdfs-y-mapreduce/

Rosero Abad, G. E. (2018). Funcionalidad y eficiencia de hadoop en clúster de raspberry pi (Bachelor's thesis, Quito).

Sánchez, M. (2015). Herramientas para Big Data: Entorno Hadoop.

Sawadogo, P., & Darmont, J. (2021). On data lake architectures and metadata management. *Journal of Intelligent Information Systems*, 97-120.

Shih, W.-C., Yang, C.-T. R., & Chiang, C.-I. (2021). Implementation and evaluation of a container management platform on Docker: Hadoop deployment as an example. *Cluster Comput 24*, 3421–3430.

Suárez, P., Gutierrez, J., Riva, C., & Tuya, J. (2017). Mantenimiento de la Consistencia Lógica en Cassandra. *JISBD 2017: XXII Jornadas de Ingeniería del Software y Bases de Datos*, 1-4.

Subramaniam, A. (22 de noviembre de 2022). *Hadoop YARN Tutorial – Learn the Fundamentals of YARN Architecture*. Obtenido de https://www.edureka.co/blog/hadoop-yarn-tutorial/

Trujillo, D., & García, S. (2019). Implementación del algoritmo JRIP en Apache Spark.

Vega, G., Ávila, J., Vega, A., Camacho, N., Becerril, A., & Amador, G. (2014). PARADIGMAS EN LA INVESTIGACIÓN. ENFOQUE CUANTITATIVO Y CUALITATIVO . *European Scientific Journal, 10(15).*, 524-528.

5 ANEXOS

5.1 ANEXO I. Recursos del lago de datos.

Enlace a los recursos para la creación del lago de datos:

https://epnecuador.sharepoint.com/:f:/s/TICComputacinenlanube/Ei3HEEVDdfNIp28TdJGKuKQBlUNMSvIwFXjIUFyi29v4vQ?e=cWb4zM

5.2 ANEXO II. Código del Dockerfile de la imagen base de Hadoop.

```
FROM ubuntu:latest

# Switch to root user

USER root

ENV HADOOP_VERSION 3.3.1

ENV BASE_DIR /opt/bd

ENV LOG_TAG "[BASE Hadoop_${HADOOP_VERSION}]:"

ENV REPOSITORY https://dlcdn.apache.org/hadoop/common

# STEP 1: Instala el software necesario

RUN echo "$LOG_TAG Actualizando e instalando paquetes básicos" && \

    apt-get update && \

    apt-get install -y --no-install-recommends openjdk-8-jre python3 curl locales iputils-ping && \

    apt-get clean && \

    rm -rf /var/lib/apt/lists

# Genera locales

RUN echo "$LOG_TAG Creando locales" && \

    locale-gen es_ES.UTF-8 && update-locale LANG=es_ES.UTF-8

# Crea un directorio para la instalación en /opt:

RUN mkdir -p ${BASE_DIR}

# Change to /opt/bd directory

WORKDIR ${BASE_DIR}

# Descarga la última versión 3.1.1 Hadoop en /opt/bd

RUN echo "$LOG_TAG Descargando Hadoop" && \

    curl -fLO "${REPOSITORY}/hadoop-${HADOOP_VERSION}/hadoop-${HADOOP_VERSION}.tar.gz"  && \

    tar xzvf hadoop-${HADOOP_VERSION}.tar.gz && \

    ln -s hadoop-${HADOOP_VERSION} hadoop && \

    rm hadoop-${HADOOP_VERSION}.tar.gz

# Crea un grupo hadoop y un usuario hdadmin para ejecutar los diferentes demonios de Hadoop (HDFS y YARN). Cambia el propietario del directorio de hadoop
```

```
RUN groupadd -r hadoop && \
    useradd -r -g hadoop -d ${BASE_DIR} -s /bin/bash hdadmin
# Crea directorio para los ficheros de log
RUN mkdir ${HADOOP_HOME}/logs
# Establece permisos
RUN chown -R hdadmin:hadoop ${BASE_DIR}
```

5.3 ANEXO III. Código del Dockerfile de la imagen del NameNode.

```
FROM hadoop-base-image:latest
# Dockerfile para el NameNode
# Switch to root user
USER root
# Define valores de entorno
ENV HADOOP_VERSION 3.3.1
ENV LOG_TAG "[NameNode Hadoop_${HADOOP_VERSION}]:"
ENV BASE_DIR /opt/bd
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
ENV DATA_DIR /var/data/hadoop/hdfs
# Crea directorios para los datos de HDFS del NameNode y haz que sean propiedad del usuario hdadmin.
# En un sistema real, deberíamos crear varios directorios en particiones separadas con suficiente espacio libre.
RUN echo "$LOG_TAG Crea directorios para los datos de HDFS del NameNode y haz que sean propiedad del usuario hdadmin" && \
    mkdir -p ${DATA_DIR}/nn && chown -R hdadmin:hadoop ${DATA_DIR}
# Crea directorio para los ficheros de log
RUN echo "$LOG_TAG Crea directorio para los ficheros de log" && \
    mkdir ${HADOOP_HOME}/logs
# Copia los ficheros de configuracion
RUN echo "$LOG_TAG Copia los ficheros de configuracion y el script de inicio"
COPY Config-files/core-site.xml ${HADOOP_CONF_DIR}/core-site.xml
COPY Config-files/hdfs-site-namenode.xml ${HADOOP_CONF_DIR}/hdfs-site.xml
# Script de inicio
COPY Config-files/start-daemons-namenode.sh ${BASE_DIR}/start-daemons.sh
# Establece permisos
RUN chmod +x ${BASE_DIR}/start-daemons.sh && \
    chown -R hdadmin:hadoop ${BASE_DIR}
# Expose ports
```

```
EXPOSE 8020 9820 9870 9871
# Define valores de entorno
ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64/
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
ENV PATH ${PATH}:${HADOOP_HOME}/bin/:${HADOOP_HOME}/sbin/
WORKDIR ${HADOOP_HOME}
RUN echo "$LOG_TAG Iniciando demonios"
CMD ["/opt/bd/start-daemons.sh"]
```

5.4 ANEXO IV. Código del Dockerfile de la imagen del ResourceManager.

```
FROM hadoop-base-image:latest
# Dockerfile para el ResourceManager
# Switch to root user
USER root
# Define valores de entorno
ENV HADOOP_VERSION 3.3.1
ENV LOG_TAG "[Resourcemanager Hadoop_${HADOOP_VERSION}]:"
ENV BASE_DIR /opt/bd
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
# Crea directorio para los ficheros de log
RUN echo "$LOG_TAG Crea directorio para los ficheros de log" && \
    mkdir ${HADOOP_HOME}/logs
# Copia los ficheros de configuracion
RUN echo "$LOG_TAG Copia los ficheros de configuracion y el script de inicio"
COPY Config-files/core-site.xml ${HADOOP_CONF_DIR}/core-site.xml
COPY Config-files/yarn-site-resourcemanager.xml ${HADOOP_CONF_DIR}/yarn-site.xml
COPY Config-files/mapred-site-resourcemanager.xml ${HADOOP_CONF_DIR}/mapred-site.xml
# Script de inicio
COPY Config-files/start-daemons-resourcemanager.sh ${BASE_DIR}/start-daemons.sh
# Establece permisos
RUN chmod +x ${BASE_DIR}/start-daemons.sh && \
    chown -R hdadmin:hadoop ${BASE_DIR}
# EXPOSE PORTS
# ResourceManager ports
EXPOSE 8030 8031 8032 8033 8088 8090
```

```
# Crea el usuario no privilegiado
RUN useradd -u 2000 luser
# Cambia al usuario hdadmin
USER hdadmin
# Define valores de entorno
ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64/
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
ENV PATH ${PATH}:${HADOOP_HOME}/bin/:${HADOOP_HOME}/sbin/
WORKDIR ${HADOOP_HOME}
RUN echo "$LOG_TAG Iniciando demonios"
CMD ["/opt/bd/start-daemons.sh"]
```

5.5 ANEXO V. Código del Dockerfile de la imagen del DataNode y NodeManager.

```
FROM hadoop-base-image:latest
# Dockerfile para los DataNodes/NodeManagers
# Switch to root user
USER root
# Define valores de entorno
ENV HADOOP_VERSION 3.3.1
ENV LOG_TAG "[DNNM Hadoop_${HADOOP_VERSION}]:"
ENV BASE_DIR /opt/bd
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
ENV DATA_DIR /var/data/hadoop/hdfs
# Crea directorios para los datos de HDFS del DataNode y haz que sean propiedad del usuario
hdadmin.
RUN echo "$LOG_TAG Crea directorios para los datos de HDFS del DataNode" && \
    mkdir -p ${DATA_DIR}/dn && chown -R hdadmin:hadoop ${DATA_DIR}
# Crea directorio para los ficheros de log
RUN echo "$LOG_TAG Crea directorio para los ficheros de log" && \
    mkdir ${HADOOP_HOME}/logs
# Copia los ficheros de configuracion
RUN echo "$LOG_TAG Copia los ficheros de configuracion y el script de inicio"
COPY Config-files/core-site.xml ${HADOOP_CONF_DIR}/core-site.xml
COPY Config-files/hdfs-site-datanode.xml ${HADOOP_CONF_DIR}/hdfs-site.xml
COPY Config-files/yarn-site-nodemanager.xml ${HADOOP_CONF_DIR}/yarn-site.xml
COPY Config-files/mapred-site-nodemanager.xml ${HADOOP_CONF_DIR}/mapred-site.xml
```

```
# Script de inicio
COPY Config-files/start-daemons-dnnm.sh ${BASE_DIR}/start-daemons.sh
# Establece permisos
RUN chmod +x ${BASE_DIR}/start-daemons.sh && \
    chown -R hdadmin:hadoop ${BASE_DIR}
# EXPOSE PORTS
# Datanode ports
EXPOSE 9864 9865 9866 9867
# NodeManager PORTS
EXPOSE 8040 8042 8044 8048
# MapReduce ports
EXPOSE 50000-50050 50100-50200
# Cambia al usuario hdadmin
USER hdadmin
# Define valores de entorno
ENV JAVA_HOME /usr/lib/jvm/java-8-openjdk-amd64/
ENV HADOOP_HOME ${BASE_DIR}/hadoop/
ENV HADOOP_CONF_DIR ${HADOOP_HOME}/etc/hadoop/
ENV PATH ${PATH}:${HADOOP_HOME}/bin/:${HADOOP_HOME}/sbin/
WORKDIR ${HADOOP_HOME}
RUN echo "$LOG_TAG Iniciando demonios"
CMD ["/opt/bd/start-daemons.sh"]
```

5.6 ANEXO VI. Enlace al manual técnico del lago de datos.

El enlace al manual técnico es el siguiente:

https://epnecuador.sharepoint.com/:f:/s/TICComputacinenlanube/Er-yNhqg6XxCiYnWCDkTxEABOP2kvbYWiPvolJJFFxLPpg?e=JBzf7y

5.7 ANEXO VII. Enlace al acta de recepción del presente proyecto.

El enlace al acta de recepción del proyecto es el siguiente:

https://epnecuador.sharepoint.com/:b:/s/TICComputacinenlanube/EcHxiL1JRSBCr4B7SxHTxbwB75H5f9o7wmOTZdTK9dwbzw?e=KjsSfz