ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

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

EVALUACIÓN DE HIPERVISORES Y CONTENEDORES PARA LA GESTIÓN DE COMPUTACIÓN EN LA NUBE PARA EL LABORATORIO ADA

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

AARON ISMAEL BRAVO MENDOZA
aaron.bravo@epn.edu.ec

DIRECTORA: PhD. SILVIA DIANA MARTINEZ MOSQUERA
diana.martinez@epn.edu.ec

DMQ, AGOSTO 2023

CERTIFICACIONES

Yo, AARÓN ISMAEL BRAVO MENDOZA, 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.

AARON ISMAEL BRAVO MENDOZA

Certifico que el presente trabajo de integración curricular fue desarrollado por AARÓN ISMAEL BRAVO MENDOZA, bajo mi supervisión.

PhD. SILVIA DIANA MARTINEZ MOSQUERA
DIRECTORA DE PROYECTO

Certificamos que revisamos el presente trabajo de integración curricular.

Nombre1 Nombre2 Apellido1 Apellido2
REVISOR1 DEL TRABAJO
DE INTEGRACIÓN CURRICULAR

Nombre1 Nombre2 Apellido1 Apellido2
REVISOR2 DEL TRABAJO
DE INTEGRACIÓN CURRICULAR

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.

AARÓN ISMAEL BRAVO MENDOZA

PhD. SILVIA DIANA MARTINEZ MOSQUERA

HERNÁN ALEXIS CHUGÁ PORTILLA

ADRIAN OCTAVIO LAJE BARRAGÁN

DEDICATORIA

Dedico este logro a mi querida familia, que con su amor y apoyo incondicional han sido la base de este camino al éxito. Gracias por ser mi luz en la oscuridad y mi refugio en las alegrías. Este logro también es suyo.

AGRADECIMIENTOS

Quiero expresar un profundo agradecimiento a la Dra. SILVIA DIANA MARTINEZ MOSQUERA, quien ha sido una guía invaluable a lo largo de este proceso. Su orientación experta, paciencia y apoyo constante han sido fundamentales para el éxito de este trabajo.

También quiero agradecer de corazón a mi familia y amigos por estar a mi lado durante este viaje. Su aliento, comprensión y amor incondicional han sido mi fortaleza, Sin su apoyo, este logro no sería posible.

A todos ustedes, mi más profundo agradecimiento por creer en mí y por ser parte fundamental en este emocionante capítulo de mi vida.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este trabajo se realizó la evaluación de tecnologías de virtualización basadas en hipervisores y contenedores, llegando a la conclusión de que los contenedores son las herramientas más adecuadas para la implementación de computación en la nube para el laboratorio de análisis datos avanzados (ADA) de la Escuela Politécnica Nacional. Como resultado de esta evaluación se implementó sobre el servidor de altas prestaciones el software JupyterHub en un contenedores Docker con el fin de que investigadores y estudiantes de posgrado aprovechen el poder de esta tecnología al poder ejecutar sus algoritmos de ciencia de datos en lenguajes R y Python en un equipo de alta capacidad de cómputo. Además se logró la integración con Hadoop en donde los usuarios guardan y leen archivos en el sistema de archivos de Hadoop por medio de código que actúa como interfaz con este software de almacenamiento. El servidor cuenta con hardware de procesamiento gráfico y por ello fue necesario la instalación y configuración de software adicional para compatibilidad de contenedores con dicho dispositivo ya que por defecto los contenedores no tienen este soporte. Con la implementación de este entorno se satisfacen los requisitos identificados por los usuarios del laboratorio. La importancia de este proyecto radica en la necesidad de un entorno de computación de alto rendimiento y la necesidad de herramientas que gestionen los recursos del servidor en el laboratorio ADA. El diseño e implementación de este proyecto brinda a los usuarios el entorno necesario para sus tareas de análisis de grandes volúmenes de datos en este laboratorio.

PALABRAS CLAVE: Computación en la nube, Virtualización, Docker, SaaS, Unidad de procesamiento gráfico, Análisis de datos

ABSTRACT

In this work, the evaluation of virtualization technologies based on hypervisors and containers was carried out, reaching the conclusion that containers are the most appropriate tools for the implementation of cloud computing for the Advanced Data Analysis (ADA) laboratory of the National Polytechnic School. As a result of this evaluation, the JupyterHub software was implemented on a high performance server in a Docker container so that researchers and graduate students can take advantage of the power of this technology by being able to run their data science algorithms in R and Python languages on a computer with high computational capacity. In addition, integration with Hadoop was achieved where users save and read files in the Hadoop file system through code that acts as an interface with this storage software. The server has graphic processing hardware and therefore it was necessary to install and configure additional software for container compatibility with this device since by default the containers do not have this support. With the implementation of this environment, the requirements identified by the laboratory users are satisfied. The importance of this project lies in the need for a high performance computing environment and the need for tools to manage server resources in the ADA laboratory. The design and implementation of this project provides users with the necessary environment for their high-volume data analysis tasks in this laboratory.

KEY WORDS: Cloud computing, Virtualization, Docker, Saas, Graphics Processing Unit, Big Data

1 INTRODUCCIÓN

Según el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE), la computación en la nube es un modelo de procesamiento de información centralizado en el que los recursos computacionales son entregados como servicios en función de las necesidades de los usuarios.

Ante la necesidad de implementar una solución de computación en la nube basada en hipervisores o contenedores en el servidor de altas prestaciones del laboratorio de análisis de datos avanzados (ADA) que permita el desarrollo de trabajos de investigación de estudiantes de maestría y docentes, se requiere de un proyecto de trabajo de integración curricular que permita analizar, diseñar, e implementar el modelo más adecuado.

Con la solución implementada se permitirá gestionar el acceso a los recursos del servidor, facilitando el trabajo de investigación de análisis de datos de los docentes y estudiantes de posgrado de la Facultad de Ingeniería de Sistemas.

La solución se basa en el modelo SaaS (Software-as-a-Service) que utiliza contenedores de Jupyter y Hadoop con el fin de aprovechar la potencia que tienen los contenedores para proporcionar un entorno escalable y flexible para el análisis y procesamiento de los datos dentro del laboratorio ADA.

Jupyter ofrece una aplicación web que permite a docentes y estudiantes de posgrado crear y compartir archivos de código, ecuaciones, visualizaciones y texto descriptivo. Es compatible con varios lenguajes de programación como Python y R. Además, brinda un entorno interactivo y de colaboración para la exploración, el análisis y la visualización de los datos.

Por otro lado, Hadoop es un entorno que fue diseñado para procesar y almacenar grandes conjuntos de datos en un entorno informático distribuido, además proporciona capacidades informáticas distribuidas confiables y escalables, lo que lo hace óptimo para la gestión de las cargas de trabajo en el análisis y almacenamiento de los datos que se usan en el laboratorio.

En la solución implementada, los contenedores Docker se utilizaron para empaquetar y des-

plegar componentes Jupyter y Hadoop de tal manera que proporcionen un entorno aislado y portátil para los usuarios. Cada contenedor encapsula sus dependencias, bibliotecas y configuraciones necesarias de tal manera que se garantice la portabilidad y la coherencia en diferentes entornos de despliegue.

Los usuarios del laboratorio acceden a la solución por medio de una interfaz web proporcionada por Jupyter Notebook. Ellos pueden crear archivos que contengan código Python y R, los cuales con las librerías necesarias interactúan con el lago de datos proporcionado por Hadoop.

Hadoop proporciona la capacidad de almacenamiento distribuido con el componente Hadoop Distributed File System (HDFS). Los usuarios del laboratorio ADA pueden aprovechar estas herramientas para almacenar grandes volúmenes de datos y realizar algoritmos de análisis.

1.1 OBJETIVO GENERAL

Implementar una solución informática basada en hipervisores o contenedores para la gestión de computación en la nube en el servidor de altas prestaciones del laboratorio ADA que facilite la investigación a estudiantes de maestría y doctorado, docentes e investigadores en el campo de la analítica y ciencia de datos.

1.2 OBJETIVOS ESPECÍFICOS

- Realizar una revisión de literatura acerca de la implementación y uso de hipervisores o contenedores, con el fin de explorar el uso, las ventajas y los retos de estas tecnologías en la computación en la nube.

- Identificar los requerimientos de los usuarios del laboratorio ADA para el diseño e implementación de un entorno de computación en la nube basada en hipervisores o contenedores, con base a las necesidades levantadas.

- Diseñar un entorno de computación en la nube para el laboratorio ADA, empleando una arquitectura basada en hipervisores o contenedores, de tal forma que se mejore el rendimiento en el procesamiento actual de los datos de los investigadores del

laboratorio.

- Implementar en el servidor de altas prestaciones una solución basada en hipervisores o contenedores que permita la ejecución de los algoritmos de análisis de datos y la integración con un lago de datos.

- Evaluar los resultados en cuanto a rendimiento de CPU, GPU, memoria y almacenamiento luego de la implementación del entorno de computación en la nube con base a los requerimientos de los usuarios del laboratorio ADA.

1.3 ALCANCE

El trabajo de integración curricular consiste en la evaluación de herramientas de vistualización basada en hipervisores y contenedores en el servidor de altas prestaciones adquirido por la facultad de Ingeniería de Sistemas que permitirá a los docentes y estudiantes de posgrado acceder a un servicio de software con el fin de procesar datos en grandes volúmenes y con gran capacidad computacional de forma remota.

De acuerdo a los resultados obtenidos de la evaluación se instalará en el servidor las herramientas de Jupyter y Hadoop, que aprovechando los recursos computacionales del servidor mejorarán significativamente las capacidades del laboratorio y ayudarán en el procesamiento, análisis y la colaboración de datos. Jupyter brinda a los usuarios una interfaz web en donde se crean cuadernos interactivos para ejecutar código de los lenguajes Python y R, además a través de bibliotecas los usuarios son capaces de interactuar con el contenedor de Hadoop el cuál almacena grandes conjuntos de datos a través del sistema de archivos distribuidos (HDFS) que gestiona los datos de una manera tolerante a fallos y escalable.

Con la implementación de este entorno de computación en la nube los usuarios del laboratorio ADA tienen a su disponibilidad recursos superiores de CPU y GPU que facilitan el procesamiento de flujos de trabajo avanzados y entrenamiento de modelos. Además, que la gran capacidad en memora RAM permite la manipulación de grandes conjuntos de datos y el gran almacenamiento garantiza la persistencia de los datos.

El análisis, diseño, implementación y evaluación de la solución involucran algunas asignaturas presentes en la malla curricular de la carrera de Ingeniería en Ciencias de la Computación, y esto permite que el trabajo desarrollado cumpla con la integración curricular.

1.4 MARCO TEÓRICO

1.4.1 Computación en la nube

El National Insitute of Standards and Technology (NIST) define a la computación en la nube como un modelo que permite el acceso a la red bajo demanda, a un conjunto compartido de recursos computacionales configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser aprovisionados y liberados con un esfuerzo mínimo de gestión o interacción con el proveedor del servicio. La computación en la nube se caracteriza por cinco atributos clave que se los describe a continuación [1].

Autoservicio bajo demanda.- Hace que los usuarios provisionen y configuren recursos automáticamente según sus necesidades sin la intervención de personas por parte del proveedor del servicio.

Acceso amplio a la red.- Permite el acceso a los servicios y recursos por medio de la red a través de artefactos estándar, como Internet. Los usuarios acceden a la nube desde varios dispositivos y ubicaciones.

Agrupación de recursos.- Hace que los proveedores de servicios asignen y reasignen de una manera dinámica los recursos compartidos, como almacenamiento, capacidad de procesamiento, memoria, etc., con el fin de atender la demanda que presentan muchos clientes.

Elasticidad rápida.- Hace que los usuarios puedan hacer un escalamiento de recursos de forma vertical y horizontal de una manera ágil y automática, según los cambios en la demanda, garantizando un rendimiento óptimo y evitando costos innecesarios.

Servicio medido.- Implica que los recursos en la nube son optimizados y controlados de forma automática mediante monitoreo. Los usuarios y proveedores pueden medir y rastrear el uso de recursos de tal manera que se logra una facturación precisa y transparente.

Existen diferentes modelos de servicio en la computación en la nube, incluyendo infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS). Estos modelos ofrecen distintos niveles de abstracción y responsabilidades para los usuarios y se los detalla a continuación [2].

IaaS.- Abarca diversos servicios, desde servidores individuales hasta almacenamiento, re-

des y más. Los proveedores ofrecen máquinas físicas o virtuales, junto con recursos adicionales como imágenes, almacenamiento, cortafuegos y más. Estos recursos se proporcionan bajo demanda desde centros de datos. Los usuarios despliegan sus aplicaciones instalando sistemas operativos y software en la nube, encargándose de parches y mantenimiento. Los proveedores facturan según la cantidad de recursos utilizados, siguiendo el modelo de computación utilitaria.

PaaS.- Permite a los desarrolladores crear sus aplicaciones en una plataforma más extensible que SaaS, sacrificando características predefinidas. Ofrece flexibilidad en la seguridad al permitir agregar capas adicionales. Los proveedores proporcionan una plataforma con sistema operativo, entorno de programación, base de datos y servidor web. Las empresas de desarrollo de software utilizan PaaS para ejecutar y crear aplicaciones en la nube sin la necesidad de gestionar el hardware y software subyacente.

SaaS.- En el modelo SaaS, los proveedores gestionan y alojan el software de la aplicación en la nube, mientras que los usuarios acceden a él desde clientes en línea. No es necesario administrar la infraestructura ni la plataforma, lo que simplifica el mantenimiento. Estas aplicaciones se entregan bajo demanda, a menudo con pagos basados en el uso. No se requiere instalación local; solo un navegador y conexión a Internet.

En la figura 1.1 se muestra el resumen de los servicios ofrecidos en un servidor local y los servicios IaaS, PaaS y SaaS. Como se observa, en un servidor local la gestión de redes, almacenamiento, servidores virtualización, sistema operativo, *middleware*, tiempo de ejecución, datos y aplicaciones la gestiona el usuario.

Para los modelos de computación en la nube, en el modelo IaaS la gestión por parte del proveedor contempla desde la parte de redes hasta la virtualización, el usuario gestiona desde el sistema operativo hasta las aplicaciones, por otro lado en el modelo PaaS el proveedor del servicio gestiona desde la capa de redes hasta la capa de tiempo de ejecución, mientras que el usuario gestiona los datos y las aplicaciones, finalmente en el modelo SaaS toda la gestión pasa a manos del proveedor del servicio y el usuario solo hace uso de la aplicación.

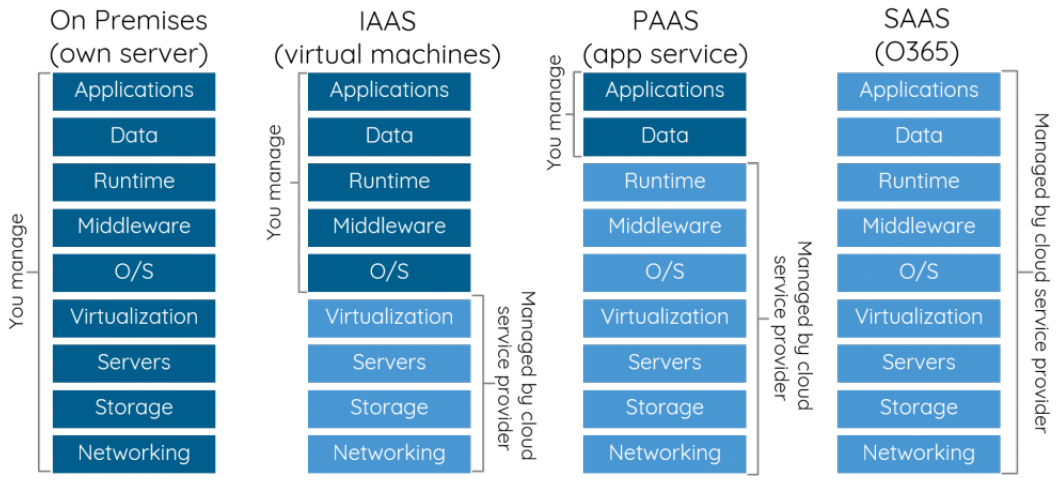

Figura 1.1: Servicios ofrecidos por un servidor local y servicios cloud [3]

Los modelos de despliegue de computación en la nube están relacionados con la ubicación y gestión de los sistemas. NIST reconoce cuatro modelos de despliegue [4].

Nube pública.- La infraestructura de la nube pública se pone a disposición del público en general, donde los recursos se proporcionan a través de Internet y cualquier usuario puede acceder desde la nube, que es propiedad del proveedor de la nube. La infraestructura de nube pública no es visible para el cliente donde se aloja la infraestructura

Nube privada.- La nube privada ofrece recursos exclusivos a una organización internamente, brindando seguridad y control total sobre los datos y la infraestructura. Solo usuarios internos tienen acceso, mientras que los externos no pueden. La protección es superior en comparación con la nube pública.

Nube comunitaria.- Se implementa infraestructura de nube pública para múltiples organizaciones, no para una en particular, sino para respaldar a una comunidad o grupo específico. Esto implica que organizaciones con objetivos y políticas afines o que forman parte de una comunidad, crean un centro de datos en la nube compartido para beneficio colectivo. La nube comunitaria se fundamenta en la confianza mutua entre sus miembros, que aprovechan sus ventajas conjuntas.

Nube híbrida.- Es una combinación de mas de un modelo de despliegue mencionados anteriormente, mantiene su identidad individual pero están interrelacionadas y permiten la portabilidad de datos y aplicaciones entre ellas

La computación en la nube brinda varios beneficios, como la escalabilidad flexible, la re-

ducción de costos, la disponibilidad global, la agilidad en el despliegue de servicios y la capacidad de centrarse en la innovación en lugar de la gestión de infraestructura [5].

Para finalizar este apartado, es importante destacar el papel que desempeña la virtualización en la eficiencia y flexibilidad de los entornos en la nube. La virtualización es la base sobre la que se construyen los servicios en la nube ya que permite la abstracción de hardware y la creación de recursos virtuales que se pueden gestionar dinámicamente.

1.4.2 Virtualización

La virtualización es una tecnología que brinda una forma de crear representaciones virtuales de servidores, almacenamiento, redes y otros recursos físicos. Al emplear software especializado, se simulan las funciones del hardware, lo que hace posible la ejecución simultánea de varias máquinas virtuales en un único servidor físico. Con esta estrategia las empresas aprovechan sus recursos de hardware de forma eficiente, maximizando el retorno de sus inversiones. Además, la virtualización jueva un papel importante cuando hay que impulsar los servicios de computación en la nube ya que proporciona a las organizaciones una gestión de infraestructura ágil y eficiente [6].

Un hipervisor es el intermediario entre las máquinas virtuales y el hardware de una máquina anfitrión. Su función principal es la de coordinar y gestionar las máquinas virtuales de tal forma que asegure que cada una tenga acceso adecuado a los recuros físicos que necesita para su correcta ejecución. Además el hipervisor desempeña un papel importante en la prevención de interferencias entre las máquinas virtuales, protegiendo así el espacio en memoria y los ciclos de cálculo lo que garantiza un entorno de ejecución seguro y eficiente [7].

1.4.2.1 Hipervisores Tipo 1

Un hipervisor Tipo I, que también se lo conoce como hipervisor *bare-metal*, se ejecuta de forma directa en el hardware físico de la computadora subyacente, esta interactúa directamente con su CPU, memoria y almacenamiento físico. A diferencia de los hipervisores de Tipo 2 que se instalan sobre un sistema operativo, el hipervisor de Tipo 1 ocupa el lugar del sistema operativo host, funcionando de manera independiente y optimizada para ofrecer un entorno eficiente y de alto rendimiento [8].

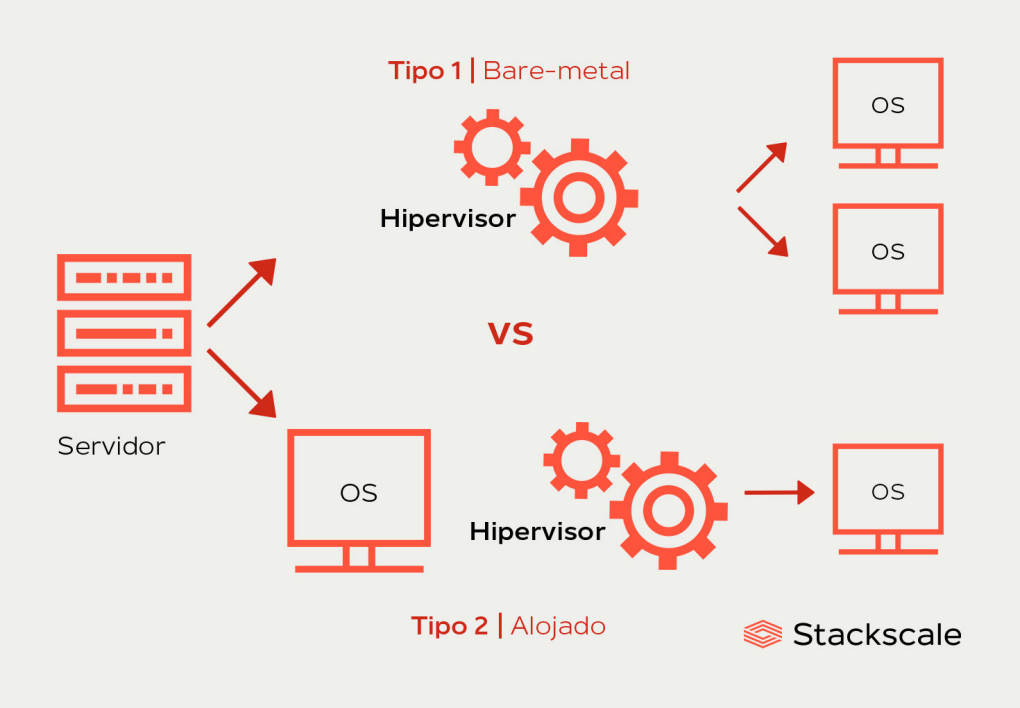

Figura 1.2: Tipos de hipervisores. [9]

1.4.2.2 Hipervisores Tipo 2

Un hipervisor de Tipo 2 opera de manera distinta a uno de Tipo 1 al funcionar como una aplicación dentro de un sistema operativo en lugar de ejecutarse directamente en el hardware. Aunque menos común en entornos basados en servidores, resulta adecuado para usuarios individuales de PC que requieren la ejecución de múltiples sistemas operativos en sus máquinas personales, como ingenieros, expertos en seguridad que analizan malware y usuarios comerciales que necesitan acceder a aplicaciones disponibles solo en otras plataformas. Estos hipervisores suelen incluir herramientas adicionales instaladas en el sistema operativo invitado, mejorando la conexión entre el sistema operativo host y la máquina invitada, permitiendo acciones como copiar y pegar información entre entornos o acceder a archivos y carpetas del sistema operativo host desde la máquina virtual invitada [8].

En la figura 1.2 se muestra un resumen de los tipos de hipervisores y se puede observar el esquema en donde los hipervisores de tipo I o Bare Metal no necesitan un S.O. anfitrión a diferencia de los hipervisores de tipo II que necesitan el S.O. anfitrión para gestionar los recursos.

La virtualización no se trata solamente de crear entornos virtuales de hardware, también se

puede virtualizar aplicaciones que usarán los recursos de la máquina de forma dinámica, esto se logra con la virtualización basada en contenedores.

1.4.3 Virtualización basada en contenedores

La virtualización basada en contenedores usa las capacidades del kernel con el fin de crear un entorno aislado para los procesos. A diferencia de los hipervisores, los contendores no cuentan con su propia capa virtualizada de hardware, mas bien aprovechan el hardware del sistema host como se observa en la figura 1.3. Como resultado, el software que se ejecuta dentro de un contenedor se comunica directamente con el kernel del host y se ejecuta en el sistema operativo y la arquitectura del sistema anfitrión [10].

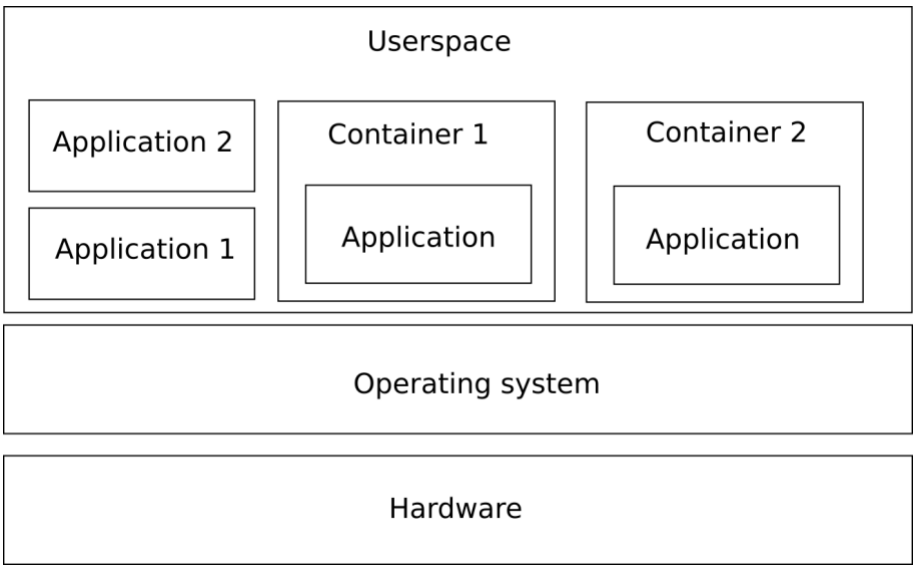

Figura 1.3: Esquema de virtualización basado en contenedores [10]

Este enfoque de virtualización prescinde de la emulación de hardware y la ejecución completa de un sistema operativo, lo que agiliza el inicio y mejora la eficiencia de los contenedores en comparación con las máquinas virtuales convencionales. Las imágenes de contenedor tienden a ser más compactas al evitar la inclusión de un conjunto completo de herramientas para ejecutar un sistema operativo, como controladores de dispositivos, kernel o sistemas init. Estas ventajas han impulsado un marcado aumento en la adopción de contenedores en los últimos años. Su uso optimizado permite un rendimiento superior a diferentes escalas y conlleva notables mejoras en seguridad. El paradigma de contenedores ha transformado la dinámica de desarrollo de software, liberando a los desarrolladores de

la necesidad de configurar manualmente sus entornos con todas las herramientas y ajustes específicos de cada proyecto. En términos de actualizaciones, los contenedores ofrecen la ventaja de que basta con reconstruir y distribuir una nueva imagen, simplificando la gestión de múltiples proyectos en la máquina de un desarrollador y evitando problemas de dependencias al separar proyectos de manera más sencilla. En resumen, la virtualización basada en contenedores ha proporcionado un enfoque ágil, eficiente y seguro para el desarrollo y despliegue de software [10].

1.4.3.1 Docker

Docker es una plataforma de código abierto que facilita el desarrollo, distribución y ejecución de aplicaciones, Su ventaja principal radica en la capacidad de separar las aplicaciones de la infraestructura subyacente, lo que permite un despliegue ágil de software. Con Docker, se puede administrar la infraestructura de manera similar a la gestión de las propias aplicaciones. Usando las metodologías de Docker para el envío, prueba y despliegue del código, se ha logrado reducir significativamente el tiempo entre el desarrollo y la puesta en producción. Esto optimiza todo el ciclo de desarrollo y entrega de software. Docker brinda la capacidad de empaquetar y ejecutar las aplicaciones en entornos aislados llamados contenedores. Este enfoque de aislamiento y seguridad permite la ejecución simultánea de varios contenedores en un mismo servidor. Los contenedores son ligeros y contienen todas las dependencias para ejecutar la aplicación, de esta forma se elimina la necesidad de depender de una configuración específica del sistema anfitrión. La facilidad de compartir contenedores es una de las ventajas de Docker, esto asegura que todas las personas con las que se comparte tengan el mismo entorno de ejecución consistente [11].

Docker utiliza una arquitectura de cliente-servidor como se observa en la figura 1.4, donde el cliente Docker se comunica con el demonio Docker para llevar a cabo múltiples tareas, como construir, ejecutar y distribuir los contenedores. El demonio Docker asume la responsabilidad de realizar las operaciones más complejas relacionadas con los contenedores [11].

El demonio Docker es el principal componente de la arquitectura la cual recibe instrucciones del cliente sobre la instrucción, ejecución y envío de los contenedores Docker. Brinda la posibilidad de ejecutar el cliente Docker y el demonio en un mismo dispositivo, o a su vez el cliente puede conectarse al demonio Docker de forma remota [12].

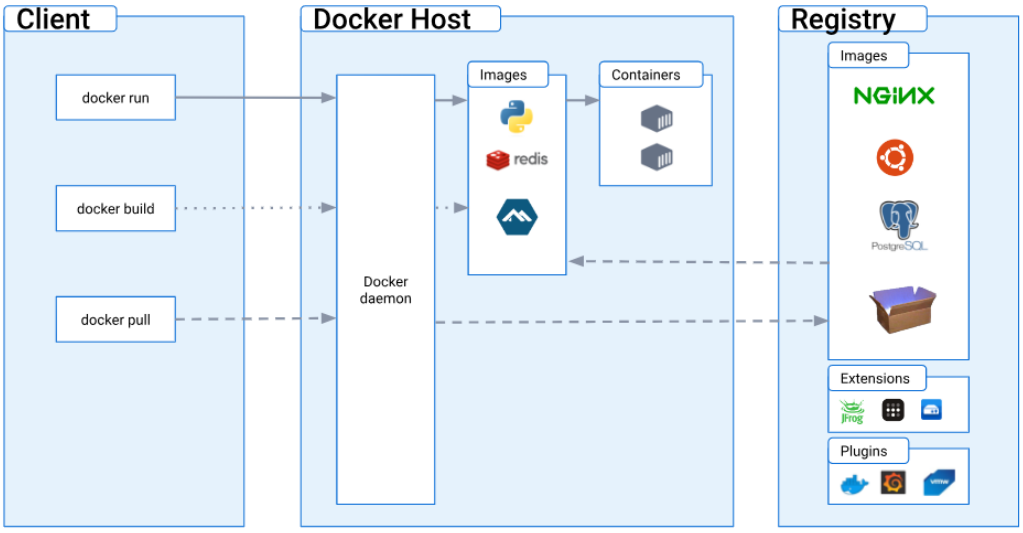

Figura 1.4: Arquitectura de Docker [11]

Demonio Docker.- Es el centro de las interacciones con Docker, espera las peticiones de la API de Docker y gestiona el estado de los objetos Docker.

Cliente Docker.- Es el componente más sencillo, ayuda a los usuarios a comunicarse con Docker. El cliente se llama cuando el usuario escribe comandos como docker run o docker pull. Su trabajo es de comunicarse con el demonio Docker mediante peticiones del Protocolo de Transferencia de Hipertexto (HTTP) que es un protocolo de comunicación que se utiliza en los servicios web para la transferencia de datos entre un cliente y un servidor web.

Registros Docker.- Los registros de Docker proporcionan una plataforma para almacenar imágenes de Docker. Docker Hub y Docker Cloud son dos registros públicos que pueden ser utilizados por cualquier imágenes, y por defecto Docker está diseñado para buscar imágenes en Docker Hub. Algunas empresas crean sus propios registros para almacenar y compartir sus propias imágenes internamente.

Imágenes Docker.- Para Docker todo se basa en imágenes, estas actúan como plantillas de sólo lectura para crear nuevos contenedores. La plataforma para crear una nueva imagen es una imagen base. El usuario puede crear y ejecutar su propia imagen de Docker definiendo los pasos en un archivo Dockerfile. Cada instrucción escrita en un Dockerfile crea una capa adicional en la imagen como se observa en la figura 1.5. Cuando el usuario hace un cambio en Dockerfile e intenta crear una imagen, sólo se recrean aquellas capas en las que los cambios han sido realizados por ese usuario en específico.

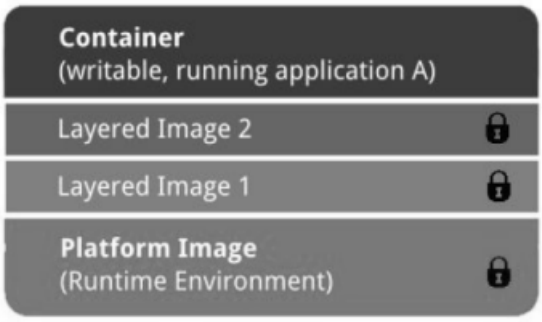

Figura 1.5: Imagen creada usando Dockerfile [12]

Contenedores Docker.- Se puede decir que el contenedor Docker es una forma ligera de máquina virtual. El contenedor Docker es creado usando imágenes o es una instancia de una imagen. Los contenedores tienen todo el conjunto de dependencias requeridas por una aplicación para ejecutarse de forma más compacta. Los contenedores Docker mantienen estados y estos cambian según las transiciones indicadas en la figura 1.6, los estados son representados por círculos y las transiciones por flechas.

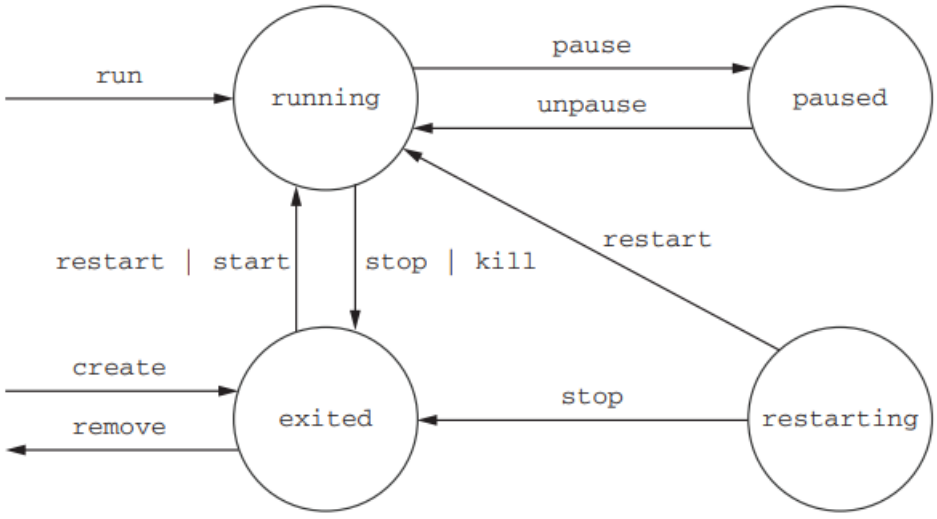

Figura 1.6: Diagrama de estados de contenedores Docker [13]

1.4.3.2 Kubernetes

Kubernetes es una plataforma portátil, extensible y de código abierto diseñada para la gestión de cargas de trabajo y servicios en contenedores, que facilita la configuración declarativa y la automatización. Su ecosistema extenso está en rápido crecimiento de tal manera que ofrece múltiples servicios, soporte y herramientas ampliamente disponibles. Los nodos trabajadores se encargan de alojar los Pods, que son los componentes de la carga de trabajo de una aplicación. Por otra parte, el plano de control es el encargado de gestionar los nodos trabajadores y los Pods dentro del clúster. En entornos de producción, es común que en el plano de control se ejecuten varias computadoras, y el clúster puede contar con varios nodos trabajadores, lo que brinda tolerancia a fallos y alta disponibilidad [14].

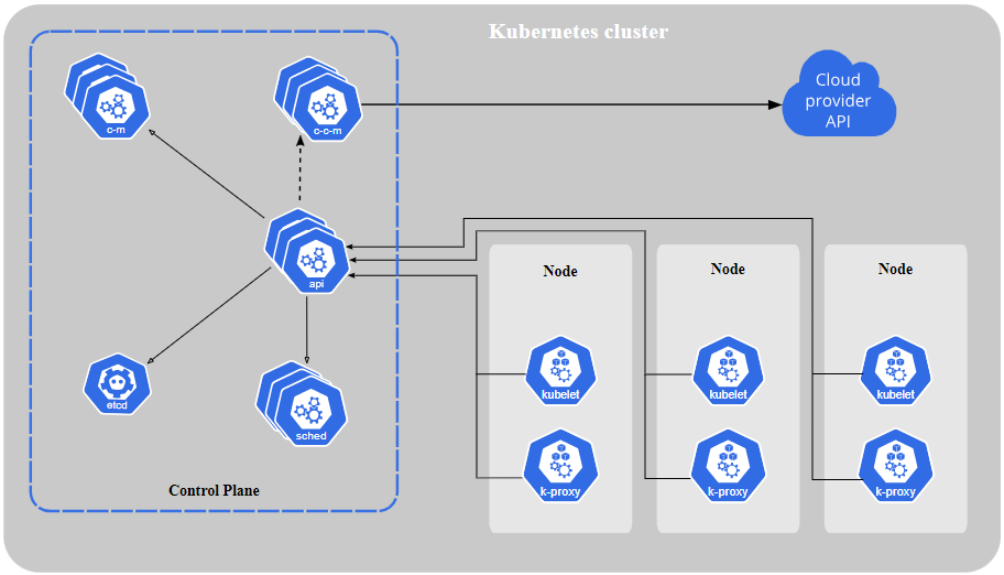

Figura 1.7: Componentes de un clúster de Kubernetes [14]

Como se ve en la figura 1.7, los componentes del plano de control son los responsables de tomar decisiones globales sobre el clúster, como la programación de tareas, así como detectar y responder a los eventos en clúster. Por ejemplo, pueden arrancar de forma automática un nuevo Pod cuando el número deseado de réplicas definidas en un despliegue no exitoso. Estos componentes del plano de control son capaces de ejecutarse en cualquier máquina del cluster. Sin embargo, por simplicidad, los scripts de configuración suelen iniciar todos los componentes del plano de control en la misma máquina, evitando la ejecución de contenedores de usuario en dicho nodo [14].

Los entornos de virtualización basada en contenedores, permite la encapsulación de aplicaciones y dependencias en un sólo componente, de esta manera los clientes solicitan servicios o recursos a servidores centralizados, por ello es importante tener claro como funciona la arquitectura cliente servidor.

1.4.4 Arquitectura cliente-servidor

En el mundo de la informática, el modelo cliente-servidor ha ganado una amplía popularidad debido a su extensa aplicación en la cotidianidad. Este modelo utiliza protocolos normalizados que permite la comunicación efectiva de los clientes y servidores. Algunos de estos protocolos comunes son el Protocolo de Transferencia de Archivos (FTP), el Protocolo Simple de Transferencia de Correo (SMTP) y el Protocolo de Transferencia de Hipertexto. El modelo cliente-servidor se compone de dos elementos escenciales: el cliente y el servidor. En este enfoque, los clientes son aquellos puntos finales que envían solicitudes, mientras que los servidores son máquinas encargadas de responder a dichas peticiones. Esta interacción permite una comunicación, donde los datos son intercambiados entre el cliente y el servidor, cada uno desempeñando funciones específicas. Se ha demostrado que el modelo cliente-servidor ofrece una solución eficiente con el fin de diseñar aplicaciones distribuidas, ya que proporciona ventajas clave, como escalabilidad y facilidad de mantenimiento. Al centralizar la lógica del negocio y los recursos en el servidor, las tareas de administración y actualización del sistema se facilita de manera modular, lo que resulta en una mayor seguridad y confiabilidad [15].

Un cliente en el contexto de la informática, puede referirse a una persona u organización que utiliza un servicio. Sin embargo, en el ámbito tecnológico, el término se refiere principalmente a un ordenador o dispositivo, también conocido como host, que en efecto utiliza un servicio o acepta información de un servidor. Estos dispositivos clientes abarcan un sin número de variedades de tecnologías, como ordenadores portátiles, estaciones de trabajo, dispositivos de Internet de las cosas (IoT) y otros dispositivos conectados a la red, como se muestra en la figura 1.8, varios dispositivos acceden por medio de Internet a servidores de propósito general realizando peticiones HTTP, estos manejan procesos internos y se conectan a bases de datos con el fin de procesar la petición y devolver una respuesta a los dispositivos clientes. En esencia, cualquier dispositivo que pueda conectarse a una red y solicitar o recibir datos de un servidor puede ser considerado un cliente. Un servidor es un

dispositivo remoto que ofrece acceso a datos y servicios. Los servidores a menudo son dispositivos físicos como servidores en un bastidor, aunque con el avance de la computación en la nube, también se han introducido los servidores virtuales en esta ecuación. Estos servidores se encargan de administrar una variedad de procesos, como el manejo del correo electrónico, el alojamiento de aplicaciones, las conexiones a Internet o la impresión, entre otras funciones. Su función es fundamental para facilitar la entrega de servicios a los clientes y usuarios [16].

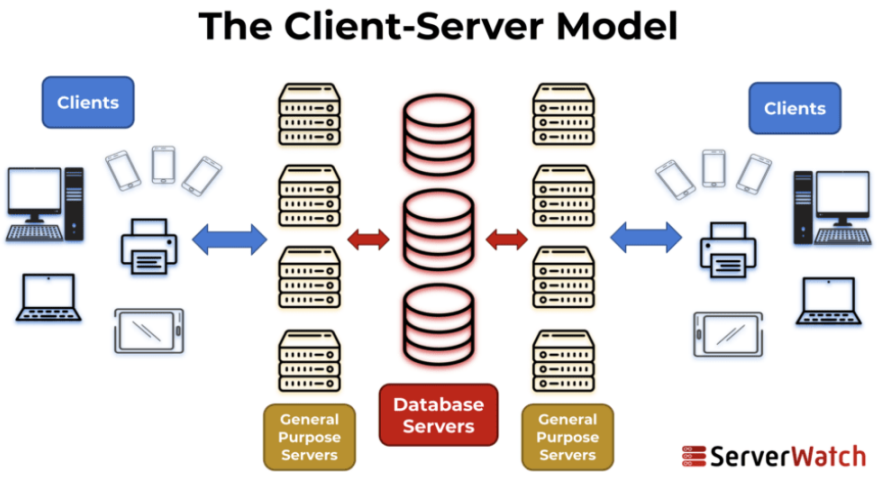

Figura 1.8: Componentes básicos de la arquitectura cliente-servidor [17]

Los distintos tipos de arquitecturas reflejan la evolución de como se han venido relacionando los clientes y los servidores con los avances tecnológicos. A continuación se describen los cuatro tipos [17].

Tier 1.- En esta arquitectura, todas las funciones del cliente-servidor, como la configuración, la interfaz de usuario, la lógica del negocio y la lógica de las bases de datos, se encuentran en los dispositivos de red. Esta estructura se utiliza principalmente en redes más pequeñas.

Tier 2.- La arquitectura de dos niveles añade un servidor a la ecuación y separa la capa de presentación del cliente, que hace solicitudes fuera de sus capacidades. Los clientes y servidores tienen más responsabilidades en la lógica del negocio y la lógica de la base de datos en diferentes niveles, lo que permite a los administradores controlar la distribución de tareas.

Tier 3.- Con el fin de mejorar y proteger aún más la arquitectura de dos niveles, se incorpora una capa de middleware entre el nivel del cliente (interfaz de usuario) y el nivel del servidor

(base de datos). Esta capa de aplicación proporciona un tercer nivel que permite una gestión más compleja de la lógica de la organización. El middleware, como los servidores de aplicaciones web, ofrece un balanceo de carga, mayor almacenamiento y seguridad.

Tier N.- En base a la arquitectura de tres niveles, la arquitectura multinivel describe el uso adicional del middleware para segmentar el tráfico y las funciones de red. Aunque esto puede aumentar la complejidad, las instituciones empresariales modernas necesitan la flexibilidad, escalabilidad y seguridad que proporcionan las arquitecturas multinivel.

Toda la teoría mencionada es necesaria para entender el funcionamiento de la computación en la nube, pero para un correcto despliegue de una solución se debe abordar trabajos relacionados sobre este tema.

1.5 TRABAJOS RELACIONADOS

En esta era de grandes cantidades de datos circulando a través de Internet y de avances tecnológicos, el campo de la investigación científica se ha visto altamente afectado por los crecientes volúmenes de datos generados y la complejidad de su análisis. En este contexto, los laboratorios de datos avanzados han surgido como base fundamental para los investigadores, dándoles la oportunidad de aprovechar el potencial de grandes conjuntos de información y de realizar nuevos descubrimientos en nuevas áreas.

En un laboratorio de datos avanzados, la virtualización ofrece una gran flexibilidad y escalabilidad. Al desvincular el sistema operativo y las aplicaciones del hardware, el administrador de la infraestructura puede crear y gestionar fácilmente varias aplicaciones en un único servidor físico. Estas aplicaciones actúan como componentes independientes, cada una con su propia configuración y dependencias. A medida que los proyectos de análisis evolucionan, la virtualización permite que el laboratorio adapte sin problemas sus recursos informáticos a las demandas de procesamiento cambiantes. Aumentar o reducir los recursos se transforma en una tarea fácil, que permite a los administradores asignar potencia de cálculo de forma dinámica según los requisitos específicos. Además, la tecnología de virtualización admite la creación de instantáneas y plantillas, lo que permite a los gestores de la infraestructura de los laboratorios replicar de forma rápida configuraciones complejas y probar con distintas configuraciones, de esta forma se agiliza el proceso de investigación y se maximiza la productividad.

En 2021, la Universidad Complutense de Madrid publicó el artículo ”Implementación de un laboratorio mediante contenedores” [18], el cual evalúa la puesta en funcionamiento de un laboratorio mediante contenedores utilizando Docker contra un entorno de virtualización con máquinas virtuales, los resultados arrojaron que los contenedores ofrecen mejores resultados en cuanto a almacenamiento, tiempos de lanzamiento y uso de memoria RAM.

De igual forma en 2020,en el arítculo ”Performance evaluation of Docker container and virtual Machine” [19], se realizaron comparaciones de las máquinas virtuales contra los contendedores y los resultados concluyeron que la entrada y salida de datos en disco, las pruebas de carga y la medición de la velocidad de operación tienen un mejor rendimiento en contenedores ya que la presencia de emuladores de procesadores hace que las máquinas virtuales sean menos eficientes en relación a los contenedores.

En el año 2018, un grupo de investigadores, en su trabajo ”Docker - based Implementation for an Astronomical Data Analysis Cloud Service ” [20], implementaron un servicio de computación en la nube basada en contenedores Docker con Jupyter, este trabajo destaca las ventajas de la implementación de Jupyter en Docker en un servidor de altas prestaciones, como quitarle responsabilidad de procesamiento al usuario y el aprovechamiento de la infraestructura de alto rendimiento, sin embargo también habla de los retos como la alta disponibilidad, la protección de los datos y la seguridad en la red que son puntos débiles.

En conclusión, esta revisión de trabajos relacionados ha servido de base para una aplicación local de un entorno de computación en la nube. Al abordar las características de las soluciones existentes, esta revisión además de enriquecer la comprensión del tema, también ha inspirado la creación de un enfoque específico para satisfacer las necesidades de los investigadores y estudiantes de posgrado del laboratorio ADA en donde se pretende ampliar los puntos fuertes de la bibliografía revisada contribuyendo al avance de los entornos de computación en la nube en el campo de la analítica avanzada de datos. Este trabajo resalta al implementar en un solo servidor las ventajas de los cuadernos de Jupyter y el almacenamiento con Hadoop, además que integra el uso de GPU con el fin de aprovechar sus capacidades de procesamiento.

El éxito de una implementación exitosa de una plataforma de computación en la nube depende de una metodología adecuada. Si el enfoque y las necesidades de los usuarios están definidos correctamente, la metodología aplicada permite alinear los objetivos del laboratorio con el uso eficiente de recursos.

2 METODOLOGÍA

La implementación de un entorno de computación en la nube en el servidor de altas prestaciones del laboratorio ADA de la Escuela Politécnica Nacional, hace un cambio al enfoque actual, en donde los docentes y estudiantes de posgrado hacen uso de sus recursos computacionales personales para realizar las tareas de análisis de datos. Se exploró y analizó este panorama y se propone una metodología con varios enfoques. Se usaron las metodologías cualitativa, experimental, descriptiva y design thinking.

La metodología cualitativa usa un enfoque de investigación centrado en comprender y entender fenómenos humanos desde una visión subjetiva y en profundidad. En contraste con la metodología cuantitativa que se basa en la obtención de números y análisis estadístico, la metodología cualitativa se centra en interpretar y comprender los significados, las experiencias y los contextos sociales [21].

La aplicación de una metodología cualitativa para el entorno de computación en la nube explora las experiencias y desafíos que tienen los usuarios del laboratorio ADA. Usando técnicas cualitativas como entrevistas, análisis de requisitos se puede entender de forma general las necesidades de los usuarios y así proporcionar un entorno de computación en la nube que satisfagan los requerimientos.

Una metodología experimental se basa en la definición y ejecución de experimentos diseñados para evaluar el sistema propuesto. Con esta metodología se establecen variables y se controlan para tener medidas del impacto de los cambios que se realicen en el entorno de computación en la nube. Esto nos da la capacidad de tener resultados empíricos y comparar diferentes opciones [22].

En el contexto de la implementación de un entorno de computación en la nube se involucran entornos de prueba basados en contenedores Docker donde se despliegan herramientas para ejecución de código Python y R y también para almacenamiento de grandes volúmenes de datos como Hadoop, este entorno permite apreciar el las difererencias del uso de un

equipo de altas prestaciones con el uso de equipos personales por parte de los docentes y estudiantes de posgrado.

La metodología descriptiva se utiliza para brindar una descripción detallada de los componentes y las configuraciones del sistema, así como para recopilar información sobre las características de un sistema. Esto ayuda a tener una mejor comprensión sobre las funcionalidades del sistema y qué aspectos pueden mejorarse [23].

Con esta metodología se ha logrado documentar el proceso de implementación, desde el análisis de requisitos del laboratorio hasta el despliegue del entorno con los respectivos manuales, tanto el manual de usuario destinado a los docentes y estudiantes de posgrado como el manual técnico destinado a los administradores de la infraestructura.

Por último, el enfoque de Design Thinking se utiliza para obtener el diseño conceptual del proyecto. Este enfoque nos brinda herramientas que nos ayudan a identificar problemas, definir objetivos y buscar soluciones con base a investigaciones previas y mejores práticas. Este enfoque ayuda a que el proyecto se base en un diseño adecuado y en la comprención de las necesidades del usuario [24].

Gracias al enfoque de Design Thinking, se involucran a las partes interesadas del laboratorio ADA para la implementación. Al adoptar esta metodología, la implementación del entorno de computación en la nube se ha convertido en un proceso colaborativo y centrado en la satisfacción de las necesidades de los usuarios de tal forma que sus tareas contribuyan al desarrollo del campo de análisis de datos.

2.1 ANÁLISIS DE REQUISITOS

El objetivo de este apartado es especificar los requisitos para el entorno que se desplegará en el servidor de altas prestaciones del laboratorio ADA. El entorno será el encargado de gestionar, procesar y analizar grandes volúmenes de datos creados por diversas fuentes de datos. Este análisis de requisitos se basa en las directrices proporcionadas por la NORMA IEEE 29148:2018 para la Ingeniería de Software y Sistemas - Ingeniería de Requisitos.

La solución atiende las necesidades específicas del laboratorio, manejará varias tareas de análisis de datos, permitiendo la colaboración entre los docentes y estudiantes de posgrado. La solución implementada se ejecuta en el hardware que se encuentra en las instalaciones de facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional.

Tabla 2.1: Partes interesadas

Partes interesadas	
1	Investigadores miembros del laboratorio
2	Docentes y estudiantes de posgrado
3	Coordinador del laboratorio

Tabla 2.2: Requisitos funcionales

Requisitos Funcionales				
Id	Requisito	Prioridad		
		Alta	Media	Baja
RF001	Creación de entornos separados para la ejecución de código y para almacenamiento de grandes volúmenes de datos	x		
RF002	Soporte para los lenguajes de programación Python y R, de tal forma que los usuarios ejecuten código en ambos lenguajes y accedan a las bibliotecas y paquetes relevantes para el análisis de datos	x		
RF003	La ejecución del código debe ser capaz de usar al máximo los recursos del servidor, tanto en procesamiento en CPU's como en GPU's	x		
RF004	Integración de entorno de ejecución de código con entorno de almacenamiento para permitir el acceso y manipulación de conjuntos de datos	x		
RF005	La solución debe ser capaz de escalar y garantizar un rendimiento óptimo de tal forma que maneje grandes volúmenes de datos y cargas de trabajo de alto rendimiento	x		
RF006	La solución debe incorporar medidas de seguridad tanto para la gestión del servidor como para la gestión de los usuarios		x	
RF007	La solución debe proporcionar herramientas de monitoreo para supervisar el estado de los entornos y el estado de los recursos físicos en el servidor.			x

La tabla 2.1 muestra las partes interesadas que se identificaron para el uso del entorno en el servidor.

Una vez que se identificaron las partes interesadas, se muestra en la tabla 2.2 los requisitos funcionales que son las características que el entorno debe brindar con el fin de ofrecen valor a los usuarios del laboratorio, estos fueron recopilados por medio de una entrevista con la directora del proyecto y el coordinador del laboratorio ADA.

La solución que se implementó en el servidor del laboratorio ADA juega un papel crucial en el procesamiento, almacenamiento y análisis de grandes cantidades de información. Por tanto, es importante abarcar varios requisitos no funcionales que respalden la calidad de la solución. Estos requisitos que se muestran en la tabla 2.3 toman en cuenta aspectos como la protección de datos, la alta disponibilidad, la capacidad de almacenamiento, la escalabilidad y la facilidad de uso. Además se consideran aspectos como rendimiento, tiempo de respuesta para consultas y procesamiento, y garantizar una óptima gestión de los accesos

por parte de los usuarios.

Tabla 2.3: Requisitos no funcionales

Requisitos no Funcionales				
Id	Requerimiento	Prioridad		
		Alta	Media	Baja
RNF01	Las herramientas a implementase deben ser open source	x		
RNF02	La solución debe garantizar la protección y confidencialidad de los datos sensibles	x		
RNF03	La solución sebe asegurar que el tiempo de actividad sea ininterrumpido de tal forma que los usuarios puedan realizar sus análisis en cualquier momento	x		
RNF04	La disponibilidad de los recursos es alta, sin embargo los usuarios utilizarán la plataforma en base a una política de asignación de recursos establecida por el coordinador del laboratorio	x		
RNF05	La solución debe adaptase a nuevos componentes de hardware que puedan añadirse en un futuro		x	
RNF06	La solución debe tener la capacidad de ser replicada en cualquier momento, de tal forma que el administrador pueda desplegar la solución aún si se adquiere nuevo servidor		x	
RNF07	La interfaz de los entornos de ejecución de código y de almacenamiento deben manejar peticiones recurrentes		x	
RNF08	El entorno de ejecución brindará accesos a los usuarios mediante un sistema de authenticacion con usuario y contraseña		x	
RNF09	Capacidad de recuperación en torno a fallos		x	

2.2 SELECCIÓN DE HERRAMIENTAS

Esta evaluación tiene como objetivo valorar las soluciones de virtualización y contenedorización para ser implementadas en el servidor de altas prestaciones del laboratorio de acuerdo al análisis de requisitos.

La solución debe tener la capacidad de ejecutar código Python y R por medio de una inter-

faz web, además debe implementarse un mecanismo de autenticación para que múltiples usuarios ejecuten algoritmos en sus propios espacios de trabajo sin que interfieran con otros usuarios. Este solución incorpora integración con Hadoop para almacenamiento de grandes volúmenes de datos.

2.2.1 Herramientas candidatas

En esta sección se describen algunas herramientas de virtualización basada en hipervisores y contenedores con el fin de incorporan una de ellas en el laboratorio ADA para la ejecución de algoritmos de analítica de datos.

2.2.1.1 VMware vSphere

VMware vSphere es una plataforma de virtualización completa y líder del sector diseñada para permitir a las organizaciones crear y gestionar infraestructuras virtualizadas. Desarrollado por VMware, vSphere ofrece un conjunto de tecnologías de virtualización que permiten a los usuarios ejecutar múltiples máquinas virtuales y aplicaciones en un único servidor físico, maximizando así la utilización del hardware y simplificando la gestión de TI [25].

2.2.1.2 Microsoft Hyper-V

Microsoft Hyper-V es el producto de virtualización de hardware de Microsoft. Permite a los usuarios crear y ejecutar versiones diferentes de software de un ordenador denominadas máquinas virtuales. Cada máquina virtual actúa como un ordenador comleto, ejecutando un sistema operativo y aplicaciones. Las máquinas virtuales se ejecutan en su propio espacio aislado de tal forma que se puede ejecutar más de una máquina virtual en el mismo harware de manera simultánea [26].

2.2.1.3 KVM (Kernel-based Virtual Machine)

KVM (Kernel-based Virtual Machine) es una solución completa de virtualización para Linux en hardware x86, que incorpora extensiones de virtualización como Intel VT o AMD-V. Brinda la infraestructura central para ejecutar múltiples máquinas virtuales con sistemas Linux

o Windows sin modificaciones. Cada máquina virtual posee su propio conjunto de hardware virtualizado, incluyendo red, disco y gráficos [27].

2.2.1.4 OpenStack

Es un sistema operativo de nube que controla grandes volúmenes de recursos informáticos, de almacenamiento y de red en un centro de datos, por medio de un panel de control se gestionan los recursos y ofrece a los administradores el control a tiempo para que los usuarios aprovisionen recursos en una interfaz web [28].

2.2.1.5 Xen

Xen es una tecnología de hipervisor de código abierto que proporciona capacidades de virtualización para ejecutar múltiples sistemas operativos invitados en un único host físico. Está desarrollado y mantenido por el Proyecto Xen, un esfuerzo de colaboración de varias organizaciones e individuos. Xen es un hipervisor de Tipo 1, que funciona directamente en el hardware sin necesidad de un sistema operativo anfitrión, lo que permite un alto rendimiento y una utilización eficiente de los recursos [29].

2.2.1.6 Proxmox Virtual Environment

Proxmox Virtual Environment (Proxmox VE) es una plataforma de gestión de servidores de código abierto para la virtualización empresarial. Integra el hiérvisor KVM y Linux Containers (LXC), almacenamiento definido por software y funcionalidad de red en una sola plataforma. La interfaz de usuario basada en web permite gestionar máquinas virtuales y contenedores [30].

2.2.1.7 Docker

Docker es una plataforma de código abierto diseñada para facilitar el desarrollo, distribución y ejecución de aplicaciones. Esta herramienta permite la separación eficiente entre las aplicaciones y la infraestructura subyacente, lo que agiliza el proceso de entrega de software. Mediante Docker, es posible administrar la infraestructura de manera similar a có-

mo se gestionan las propias aplicaciones. Al adoptar las metodologías de Docker para el empaquetado, prueba y despliegue de código, es viable reducir drásticamente el intervalo temporal entre la codificación y la puesta en marcha en entornos productivos [11].

2.2.1.8 Kubernetes

Kubernetes, a menudo abreviado como K8s, es una plataforma de orquestación de contenedores de código abierto que automatiza el despliegue, el escalado y la gestión de aplicaciones en contenedores. Desarrollado originalmente por Google y mantenido ahora por la Cloud Native Computing Foundation (CNCF), Kubernetes se ha convertido en el estándar del sector para gestionar cargas de trabajo en contenedores tanto en entornos locales como en la nube [14].

2.2.1.9 Podman

Podman es una herramienta de código abierto para Linux que simplifica la gestión de aplicaciones en contenedores y imágenes de la Iniciativa de Contenedores Abiertos (OCI). No requiere un demonio y ofrece una interfaz de línea de comandos similar a Docker, permitiendo a los usuarios familiarizados hacer la transición sin problemas. Al utilizar tiempo de ejecución de contenedores compatibles con OCI, como runc, crea y administra contenedores de manera eficiente, haciendo que sean prácticamente indistinguibles de otros motores de contenedores comunes [31].

2.2.1.10 LXC/LXD

LXC (Linux Containers) y LXD son tecnologías de contenedorización de código abierto que proporcionan virtualización a nivel de sistema operativo para ejecutar entornos Linux aislados. Desarrollada por Canonical, la empresa que está detrás de Ubuntu, LXD es una herramienta de gestión de alto nivel construida sobre LXC, que hace que la gestión de contenedores sea más fácil de usar y robusta [32].

2.2.1.11 OpenShift

OpenShift, es una plataforma de aplicaciones de nube híbrida respaldada por Kubernetes, fusiona servicios confiables y probados para simplificar el ciclo de desarrollo, modernización, implementación y gestión de aplicaciones. Con una experiencia uniforme en entornos locales, de red y nube, OpenShift ofrece opciones autogestionadas o gestionadas por terceros, permitiendo a los equipos enfocarse en sus tareas clave, sin importar la infraestructura subyacente [33].

2.2.1.12 Docker Swarm

Docker Swarm es una herramienta de orquestación y agrupación de contenedores que permite a los usuarios crear y gestionar un enjambre de nodos Docker, convirtiéndolos en un clúster escalable y tolerante a fallos. Desarrollado por Docker, Inc, Docker Swarm permite el despliegue y la gestión de aplicaciones en contenedores a través de múltiples hosts, proporcionando alta disponibilidad y balanceo de carga para cargas de trabajo distribuidas [34].

Luego de haber presentado algunas herramientas que permiten la virtualización basada en hipervisores y contenedores, se procedió a realizar una evaluación según los requisitos funcionales y no funcionales identificados anteriormente.

2.2.2 Criterios de evaluación

Durante este proceso se deben tomar en cuenta los requisitos funcionales y no funcionales que se identificaron anteriormente, en las tablas 2.4 y 2.5 se muestra el resumen de la evaluación.

Tabla 2.4: Evaluación según requisitos funcionales

HERRAMIENTA	RF							VALORACION TOTAL
	001	002	003	004	005	006	007	
VMware vSphere	1	0	0	1	1	1	2	6
Microsoft Hyper-V	1	0	0	1	1	1.5	1.5	6
KVM (Kernel-based Virtual Machine)	1.5	0	0	1	1	1	1.5	6
OpenStack	1	0	0	1	1.5	2	1.5	7
Xen	1	0	0	1	1.5	1	1.5	6
Proxmox Virtual Environment	1	0	0	1	1.5	1	1.5	6
Docker	2	2	2	2	2	1.5	1	12.5
Kubernetes	2	2	2	2	2	1.5	2	13.5
Podman	2	2	1.5	2	1.5	1	1	11
LXC/LXD	2	2	2	1.5	1	1	1	10.5
OpenShift	1	2	1	2	2	2	1.5	11.5
Docker Swarm	1.5	2	1	2	1.5	1	1	10

Tabla 2.5: Evaluación según requerimientos no funcionales

HERRAMIENTA	RNF									VALORACION TOTAL
	001	002	003	004	005	006	007	008	009	
VMware vSphere	0	1	2	1	2	2	2	2	2	14
Microsoft Hyper-V	0	1	2	1	2	2	2	2	2	14
KVM (Kernel-based Virtual Machine)	2	1.5	1.5	1	2	1.5	1.5	1	1	13
OpenStack	2	1.5	1.5	1.5	1	1.5	1.5	1	1.5	13
Xen	2	1.5	1.5	1	1.5	1.5	1.5	1	1.5	13
Proxmox Virtual Environment	0	1.5	1.5	1.5	1	1	1.5	1	1	10
Docker	2	2	2	2	2	2	2	1.5	1.5	17
Kubernetes	2	2	1,5	2	2	2	1.5	2	2	17
Podman	2	2	1.5	1.5	1.5	2	2	1	1	14.5
LXC/LXD	2	1.5	1.5	1	1	1	2	1	1	12
OpenShift	2	2	2	2	2	2	1	2	2	17
Docker Swarm	2	1.5	1.5	1	1	1	2	1.5	1	12.5

La evaluación asigna un peso de cumplimiento en escala del 1 al 2, donde 2 representa un excelente cumplimiento del requisito y 1 indica una capacidad adecuada pero limitada, además en ciertas herramientas se tiene un peso de 0 al ser que no tiene cumplimiento para cierto criterio de evaluación.

La evaluación ha mostrado a Docker, Kubernetes y OpenShift como las mejores soluciones y se ha seleccionado Docker debido a su simplicidad de uso y rapidez de implementación frente a las otras herramientas que aunque ofrecen grandes capacidades de virtualización, son más adecuadas para configuraciones más grandes y complejas con aplicaciones distribuidas, con Docker gracias a su alta demanda para despliegue de aplicaciones y una documentación extensa se logró la integración de las herramientas Jupyter, Tensorflow y Hadoop, estas herramientas en su conjunto brindan las funcionalidades necesarias para ejecutar las tareas de análisis de datos dentro del laboratorio, la amplia gama de imágenes en repositorios de Docker permitió que se incorpore además el uso de GPU en los entornos de ejecución de código de Jupyter, con esto se brinda total cumplimiento a los requerimientos identificados.

En resumen, la implementación de Docker en el laboratorio ADA permitió ya realizar pruebas de ejecución de código Python y R, el servicio de autenticación implementado ha permitido que una cantidad reducida de usuarios del laboratorio ejecuten sus primeras líneas de código en sus propios espacios de trabajo y se ha eliminado la dependencia de altas capacidades computacionales en computadores de personales.

2.3 DISEÑO DE LA ARQUITECTURA

La arquitectura a implementase con contenedores Docker en el laboratorio ADA está diseñada con el fin de proporcionar un entorno para el análisis y procesamiento de grandes volúmenes de datos. Esta arquitectura concentra la potencia que tienen los contenedores Docker para facilitar el despliegue y la gestión.

En base a los requerimientos detallados anteriormente se ha decidido implementar un entorno de computación en la nube adoptando un modelo de SaaS, el software a implementarse es JupyterHub que brinda cuadernos para que los usuarios ejecuten código en diferentes lenguajes de programación, los lenguajes implementados sobre Jupyter para este trabajo fueron Python y R.

Las aplicaciones de Jupyter y Hadoop son encapsuladas en contenedores Docker ya que con esto se evitan los conflictos de dependencias entre aplicaciones y de esa manera se facilita que este entorno pueda ser replicado en otros equipos de altas prestaciones.

Con el fin de que los usuarios hagan uso de todas las capacidades de procesamiento tanto en CPU y GPU se requiere de la herramienta NVIDIA Container Toolkit ya que es necesaria para que los análisis de datos que requieran el uso de GPU funcionen de forma consistente.

Una de sus principales ventajas es su capacidad de proporcionar aislamiento y gestión de recursos de GPU. Con este kit de herramientas, a la plataforma Jupyter se le asigna toda la potencia de procesamiento de GPU, lo que garantiza que los recursos no se sobrescriban y que varios contenedores puedan hacer uso las GPU de forma simultanea y sin interrupciones. En la figura 2.1 se observa un esquema de la arquitectura de NVIDIA Container Toolkit, en el nivel más bajo se tiene el componente físico de GPU y en el nivel más alto se tienen las aplicaciones encapsuladas en contenedores haciendo uso de las capacidades de GPU.

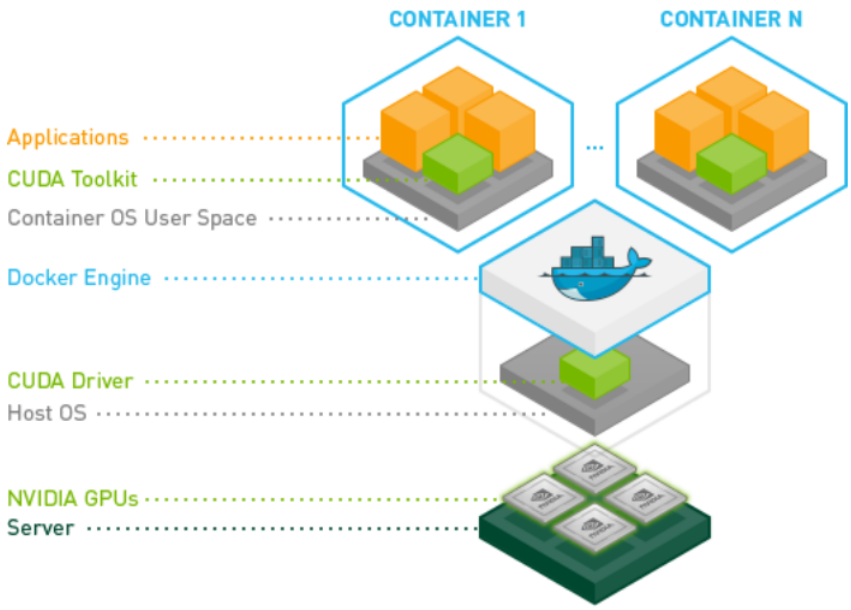

Figura 2.1: NVIDIA Container Toolkit. [35]

Al necesitar la ejecución de código Python y para hacer uso de los aceleradores gráficos disponibles en el servidor del laboratorio se ha escogido la imagen de Docker tensorflow / tensorflow:latest-gpu-Jupyter disponible en https://hub.docker.com/r/tensorflow/tensorflow ya que existen factores que atienden las necesidades del laboratorio para tareas avanzadas de análisis de datos, aprendizaje profundo y aprendizaje automático.

Esta imagen viene equipada con soporte para tarjetas gráficas, lo que permite a los docentes y estudiantes de posgrado aprovechar la potente aceleración que tienen las GPU para tareas de alto rendimiento. Las GPU son demasiado eficientes en el manejo de operaciones matemáticas complejas que intervienen en un análisis exhaustivo de datos, lo que acelera significativamente los tiempos de respuesta en comparación con entornos que se basan únicamente en CPU.

Esta imagen viene equipada con Tensorflow y Jupyter, que son las herramientas más usadas para el análisis de datos y la inteligencia artificial. Esta configuración hace posible que los usuarios puedan compartir cuadernos interactivos, lo que minimiza el desarrollo, la experimentación y la visualización de los resultados en un entorno de colaboración.

La plataforma Jupyter viene por defecto con el lenguaje de programación Python, al ser uno de los requerimientos la integración del lenguaje R, se ha implementado el lenguaje junto con su kernel para la ejecución de código en este lenguaje a través de los cuadernos de Jupyter.

Esta implementación para el análisis avanzado la hacen una herramienta importante en el conjunto de herramientas del investigador. Los cuadernos de Jupyter, con sus capacidades interactivas y de ejecución de código, proporcionan un entorno ideal para llevar a cabo análisis exploratorio de datos, modelado estadístico y visualización de datos. Al implementar R en Jupyter, los docentes y estudiantes de posgrado pueden hacer uso de toda la potencia de las bibliotecas estadísticas y de manipulación de datos que ofrece este lenguaje, al mismo tiempo que aprovechan la interactividad y facilidad de uso que ofrecen los cuadernos de Jupyter.

La plataforma Jupyter es más adecuada para uso individual de análisis de datos en una máquina local, por lo que se ha implementado JupyterHub que es una extensión de Jupyter. Una de las principales ventajas es su capacidad de admitir varios usuarios simultáneamente. Con Jupyter solo, los usuarios trabajan de forma aislada, pero gracias a JupyterHub, equipos de investigadores y estudiantes colaboran en proyectos en tiempo real. Esto fomenta un entorno más interactivo y dinámico, que permite a los usuarios compartir ideas, colaborar en código y explorar conjuntamente los datos.

Otra ventaja es que JupyterHub proporciona una gestión centralizada de los recursos del servidor. En lugar de asignar recursos dedicados a cada usuario. JupyterHub optimiza los recursos asignándolos dinámicamente según la demanda de los usuarios. Esto asegura un

uso eficiente de la capacidad del servidor y evita el desperdicio de los recursos, haciendo que el laboratorio ADA sea mas productivo.

Para JupyetHub, es importante la configuración de un proxy HTTP para brindar un acceso seguro y facilitar la comunicación entre los usuarios de JupyterHub y los distintos servidores de cuadernos Jupyter que están ejecutándose para diferentes usuarios del laboratorio. El proxy HTTP actúa como intermediario entre el navegador web del usuario y el servidor de JupyterHub, de esa forma se gestiona las peticiones y las reenvía a los destinos adecuados.

JupyterHub ofrece su proxy ”configurable-http-proxy”. Es una herramienta importante en JupyterHub ya que permite el enrutamiento seguro de solicitudes entre los usuarios del laboratorio y los servidores individuales de Jupyter. Con esta herramienta el laboratorio tiene aislamiento y gestión centralizada de los componentes de Jupyter, lo que resulta importante en un entorno de colaboración en ciencia de datos. Según la figura 2.2 el usuario usa un navegador web para acceder a la interfaz de JupyterHub, el configurable-http-proxy dirige las solicitudes que entran desde el navegador hacia los servidores Jupyter individuales, los componentes Public facing y Inward facing se refieren a los aspectos externos e internos de Jupyter para comunicarse con las aplicaciones que se ejecutan en las sesiones de los usuarios.

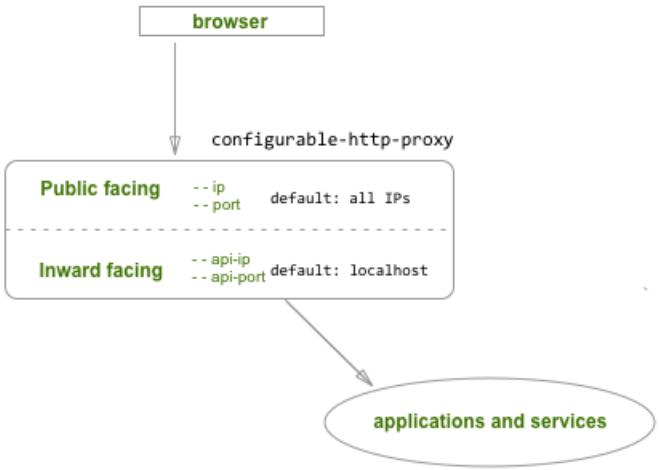

Figura 2.2: Funcionamiento de configurable-http-proxy. [36]

El sistema de autenticación por defecto de Jupyter hace referencia a tokens de un solo usuario, esto no es suficiente para el laboratorio ADA que maneja múltiples usuarios y datos sensibles, es por ello que se implementó Native Authenticator que permite el acceso con

usuario y contraseña basado en roles.

Con esta autenticación, el administrador del laboratorio puede definir funciones y permisos, lo que permite un control detallado sobre quién puede crear, modificar o acceder a los recursos de Jupyter.

El flujo de autenticación mostrado en la figura 2.3 empieza por uno o más usuarios administradores que se añaden en un archivo de configuración, los usuarios que requieren autenticación se dirigen a la ruta /signup, donde proporcionan un usuario y contraseña, luego el administrador autoriza el acceso al usuario y una vez autorizado, el usuario se dirige a la página de inicio de Jupyter e ingresa sus credenciales, teniendo una autenticación e inicio de sesión exitoso.

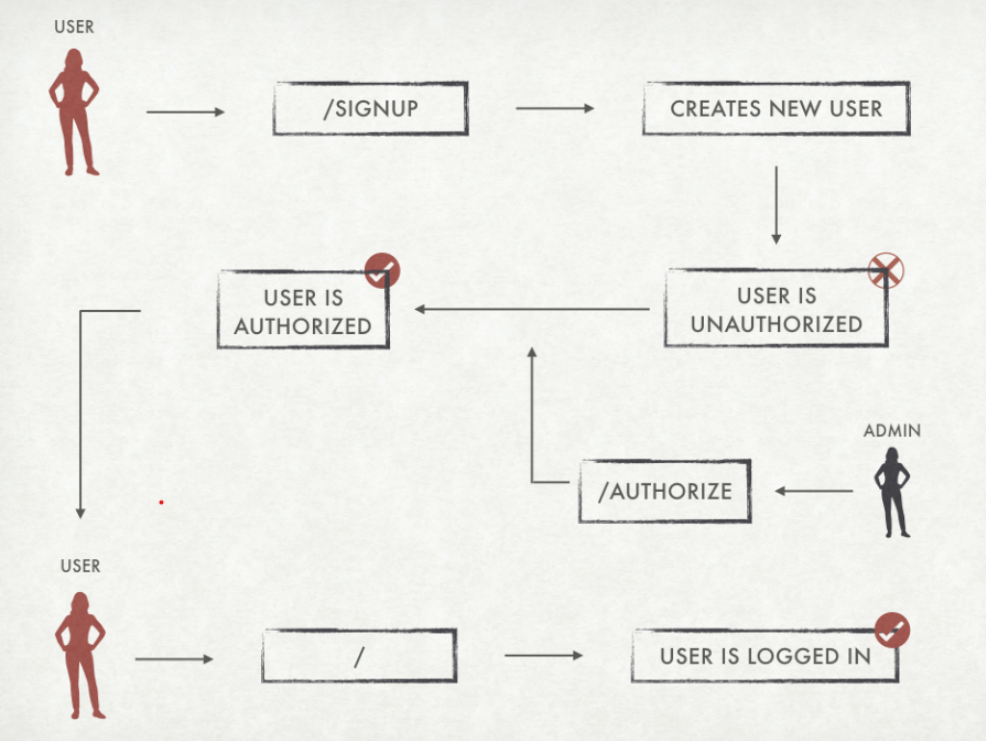

Figura 2.3: Flujo de Native Authenticator. [37]

En conclusión, la implementación de Native Authenticator en JupyterHub, brinda al administrador del laboratorio una forma simple de gestionar usuarios y recursos computacionales ya que agiliza la creación o eliminación de usuarios y mejora la experiencia. A medida que el laboratorio adquiera infraestructura o incorpore más personal, un sistema de autenticación fácil de usar es esencial para una gestión simplificada.

La implementación de contenedores Docker de Hadoop en el servidor del laboratorio ADA ofrece ventajas como proporcionar una solución ligera para desplegar clústeres de Haddop sin problemas. Esto asegura que el servidor pueda manejar fácilmente el almacenamiento de grandes volúmenes de datos. El uso de Docker para el despliegue de esta aplicación permite la fácil transferencia de todo el entorno Hadoop, añadiendo todas sus dependencias y configuraciones en contenedores independientes. Las capacidades de aislamiento de Hadoop con Docker evitan posibles conflictos con dependencias de Jupyter.

2.3.1 Arquitectura de red

Con el fin de mejorar la seguridad y privacidad, el acceso a la plataforma de análisis de datos está estrictamente limitada a dos métodos: a través de una Red Privada Vritual (VPN) que debe ser solicitada a la Dirección de Gestión de la Información y Procesos de la Escuela Politécnica Nacional (DIGIP) o una Infraestructura de Escritorio Virtual (VDI) con las credenciales institucionales. Al aplicar estos controles de acceso, se asegura que todas las conexiones de los usuarios estén cifradas y autenticadas, salvaguardando así los datos confidenciales y evitando accesos no autorizados. La VPN proporciona un túnel seguro para que los investigadores y estudiantes de posgrado se conecten al servidor de la aplicación, mientras que el VDI ofrece un entorno virtualizado que aísla la aplicación del sistema local del usuario, reduciendo el riesgo de posibles amenazas. Al imponer el uso de la VPN o VDI, se da prioridad a la protección de la plataforma implementada y se promueve una experiencia segura y fluida.

En la figura 2.4 se muestra un esquema de la arquitectura, en donde los usuarios acceden a través de Internet a las plataformas de VDI o VPN y luego acceden a la aplicación de Jupyter a través de un navegador web.

Los investigadores y estudiantes pueden hacer uso de cualquiera de los dos métodos, siguiendo los pasos que constan en el anexo 1 para la conexión por VDI y los pasos que constan en el anexo 2 para la conexión por VPN.

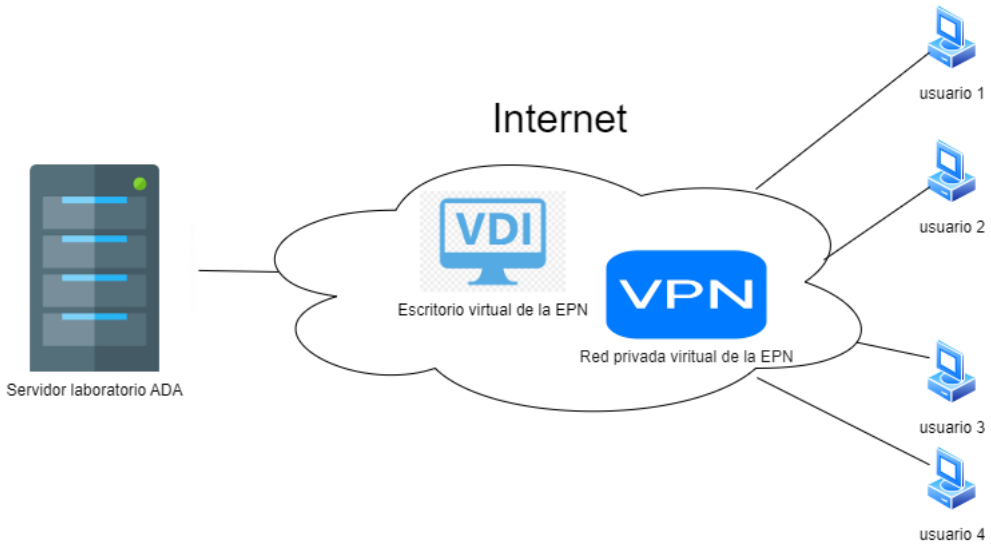

Figura 2.4: Arquitectura de red

2.3.2 Arquitectura de aplicaciones

La arquitectura a desplegarse mostrada en la figura 2.5 toma como base un servidor de altas prestaciones de la marca Dell, cuyas especificaciones técnicas se encuentran en el anexo 3, sobre este servidor se instala el sistema operativo Ubuntu 22.04 que es una distribución de Linux basada en Debian GNU/Linux, sobre este se ha integrado perfectamente los contenedores de Tensorflow y Hadoop. Sobre la imagen del contenedor tensorflow se logró implementar las herramientas de Jupyterhub, R, Python, configurable-http-proxy y nativeauthenticator que en su conjunto despliegan una plataforma de JupyterHub adecuada para el análisis de datos avanzados ya que integra los lenguajes de programación Python y R, hace uso de un proxy HTTP para un acceso seguro y que permita la comunicación entre los usuarios y los servidores de los cuadernos de Jupyter e implementa un mecanismo de autenticación que reduce el riesgo de accesos no autorizados.

La implementación de los clusters para el contenedor de Hadoop no está en el alcance de este documento, sin embargo se lo toma en cuenta para el despliegue del entorno en el servidor de altas prestaciones.

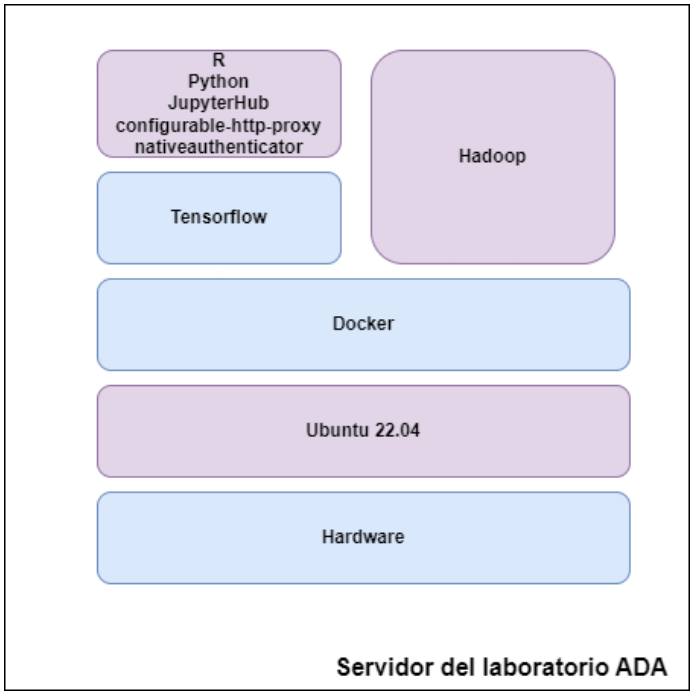

Figura 2.5: Arquitectura de aplicaciones

La arquitectura propuesta para el servidor del laboratorio ADA recopila algunos componentes que en su conjunto forman el entorno de computación en la nube requerida para que los docentes y estudiantes de posgrado ejecuten sus algoritmos de análisis de datos haciendo uso de las atlas capacidades de procesamiento que ofrece el servidor. La correcta implementación de las herramientas hacen que este trabajo aporte de manera significativa al desarrollo de las actividades del laboratorio.

2.4 IMPLEMENTACIÓN

En este apartado se consideran los aspectos técnicos para una correcta implementación del entorno de computación en la nube. Es aquí donde se debe realizar las implementaciones de manera ordenada y siguiendo la documentación necesaria para lograr que la implementación de la solución se desarrolle con éxito.

2.4.1 Sistema operativo

Para la implementación del entorno de análsis de datos, se toma como base el servidor de la marca Dell en el cuál se instaló el sistema operativo Ubuntu 22.04, en la figura 2.6 se observa la versión de la distribución instalada en la máquina.

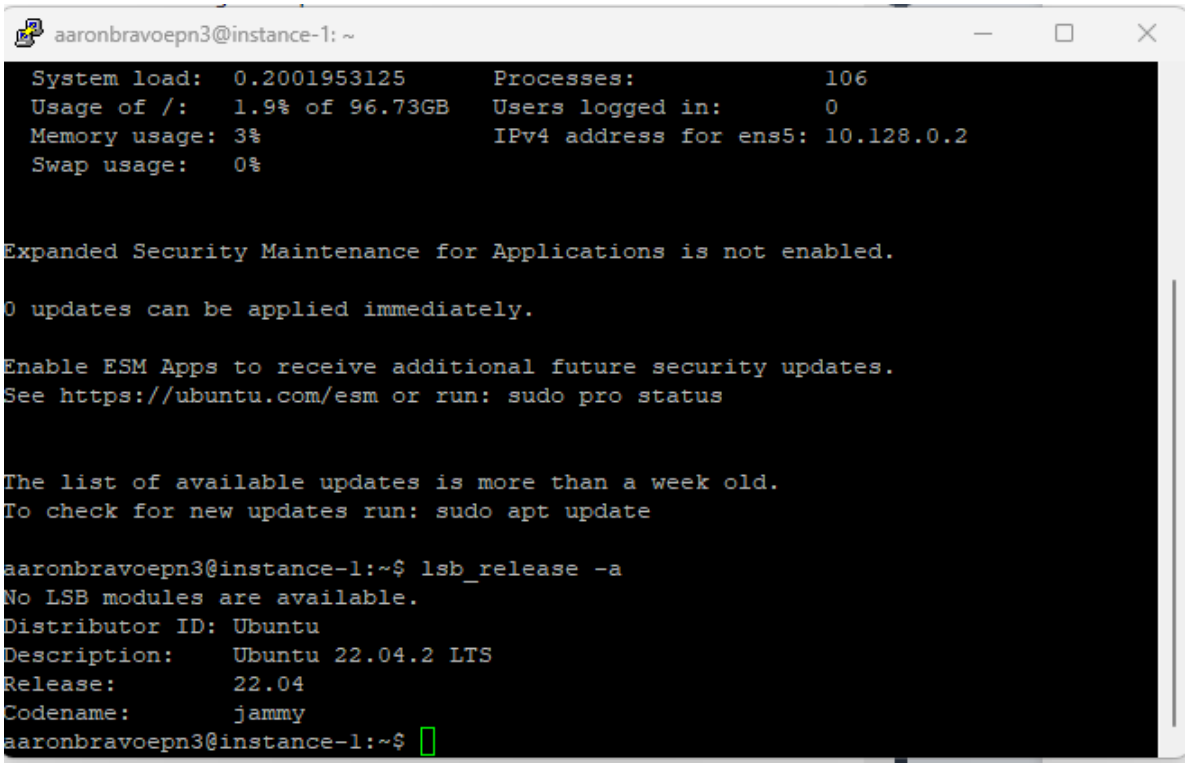

Figura 2.6: Versión del sistema operativo

Como primer paso se realizó una actualización de la lista de paquetes disponibles en los repositorios configurados en el sistema, como se observa en la figura 2.7, el sistema descargó la información más reciente de los paquetes, esto se lo realizó ejecutando el siguiente comando.

- - sudo apt update

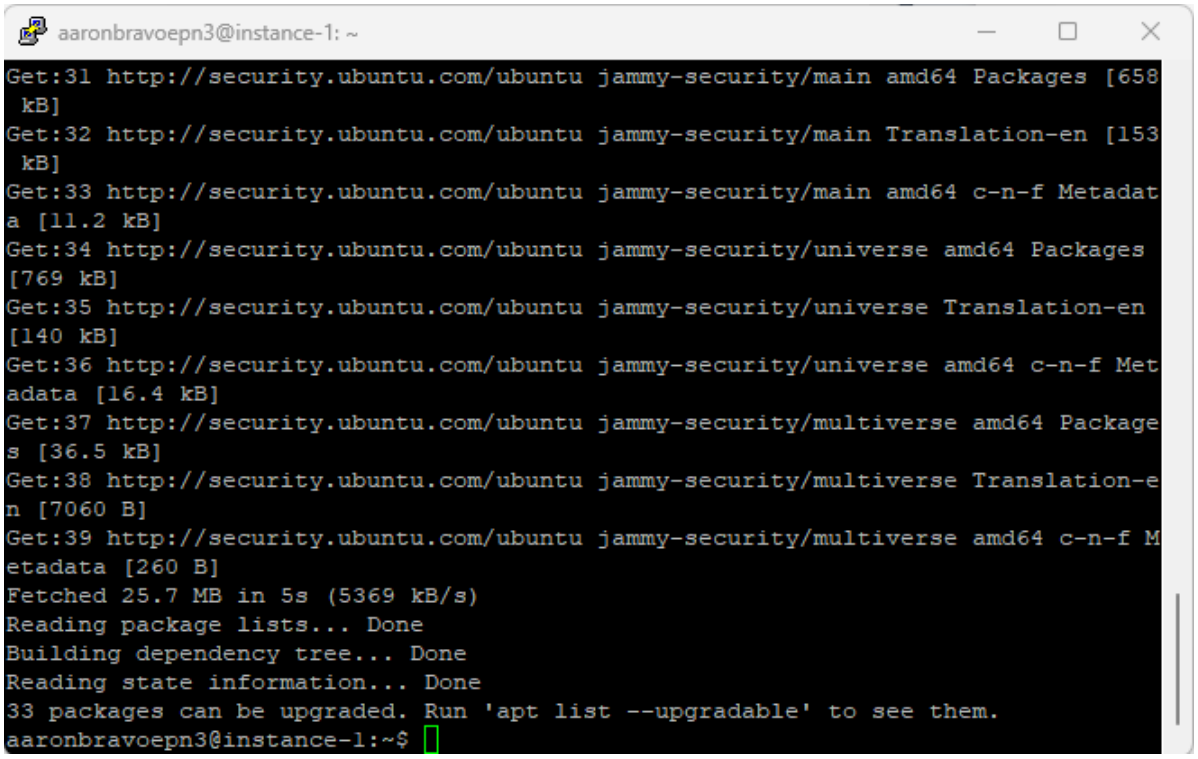

Figura 2.7: Actualización de lista de paquetes

2.4.2 Docker

Luego de la actualización de paquetes se instaló Docker Engine, siguiendo los pasos de la guía disponible en [38], luego de la instalación se agregó el usuario del sistema operativo al grupo Docker con el fin que este usuario pueda ejecutar los comandos y se hizo la verificación de la instalación de Docker, en la figura 2.8 se observa que se descarga y ejecuta una imagen de prueba de Docker.

Los comandos que se usaron para este paso se muestran a continuación.

- - sudo usermod -aG docker nuevo-usuario: Agregar usuario al grupo Docker.

- - docker run hello-world: Verificación de instalación de Docker.

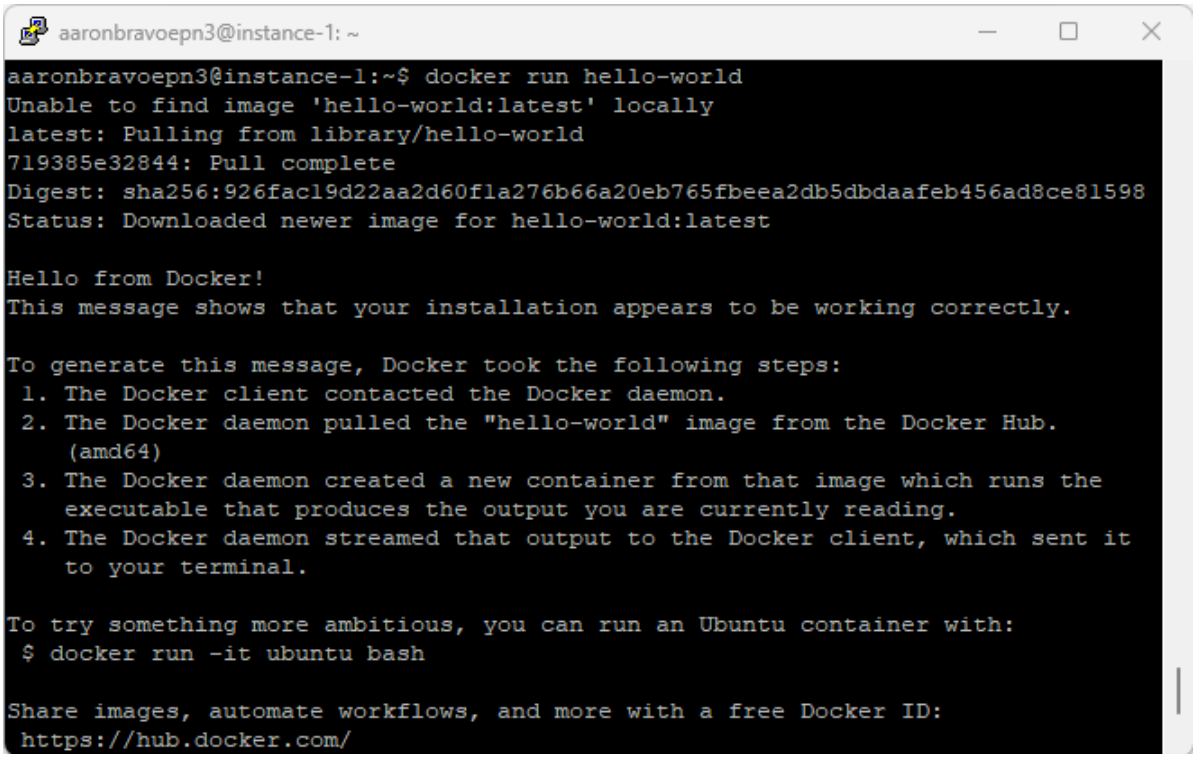

Figura 2.8: Verificación de instalación de Docker

2.4.3 Controladores GPU

Con el fin de implementar el controlador más adecuado, primero se muestra una lista de controladores recomendados y disponibles para el sistema, esta lista proporciona información sobre los controladores que Ubuntu recomienda para ciertos dispositivos, como tarjetas gráficas y otros componentes, como se observa en la figura 2.9, se despliega una lista para ciertos dispositivos y se procede a instalar el controlador que tiene la palabra ”recommended” al final. Esto hace que el driver instalado sea el adecuado para el hardware que se tiene en el servidor.

- - ubuntu-drivers devices: Muestra una lista de controladores recomendados

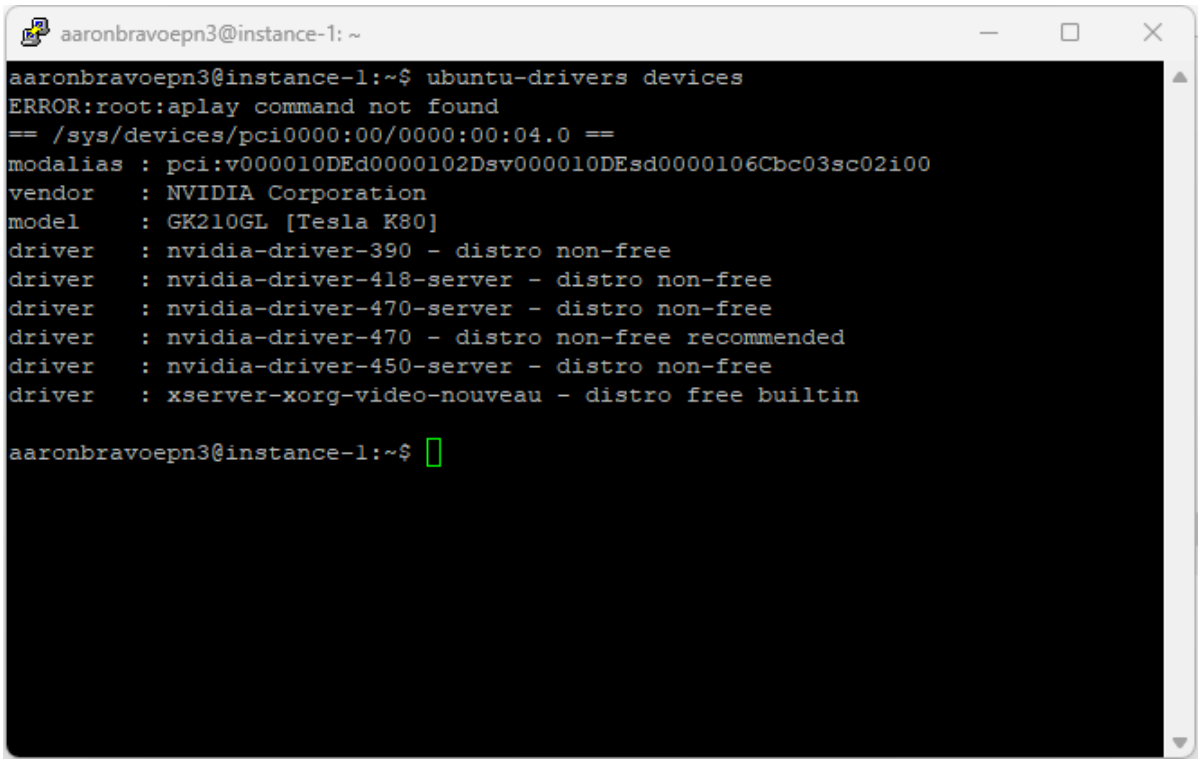

Figura 2.9: Controladores disponibles para la GPU en Ubuntu

Luego de instalar el controlador recomendado, se reinicia el equipo y se verifica que el sistema operativo reconozca el controlador instalado, en la figura 2.10, se muestra el estado actual de la GPU lo cual refleja que la instalación del controlador se realizó con éxito.

- - sudo apt install <controlador-recomendado>: Instala el controlador de GPU recomendado por el sistema operativo

- - sudo reboot: Realiza un reinicio del equipo

- - nvidia-smi: Muestra estado actual de la GPU

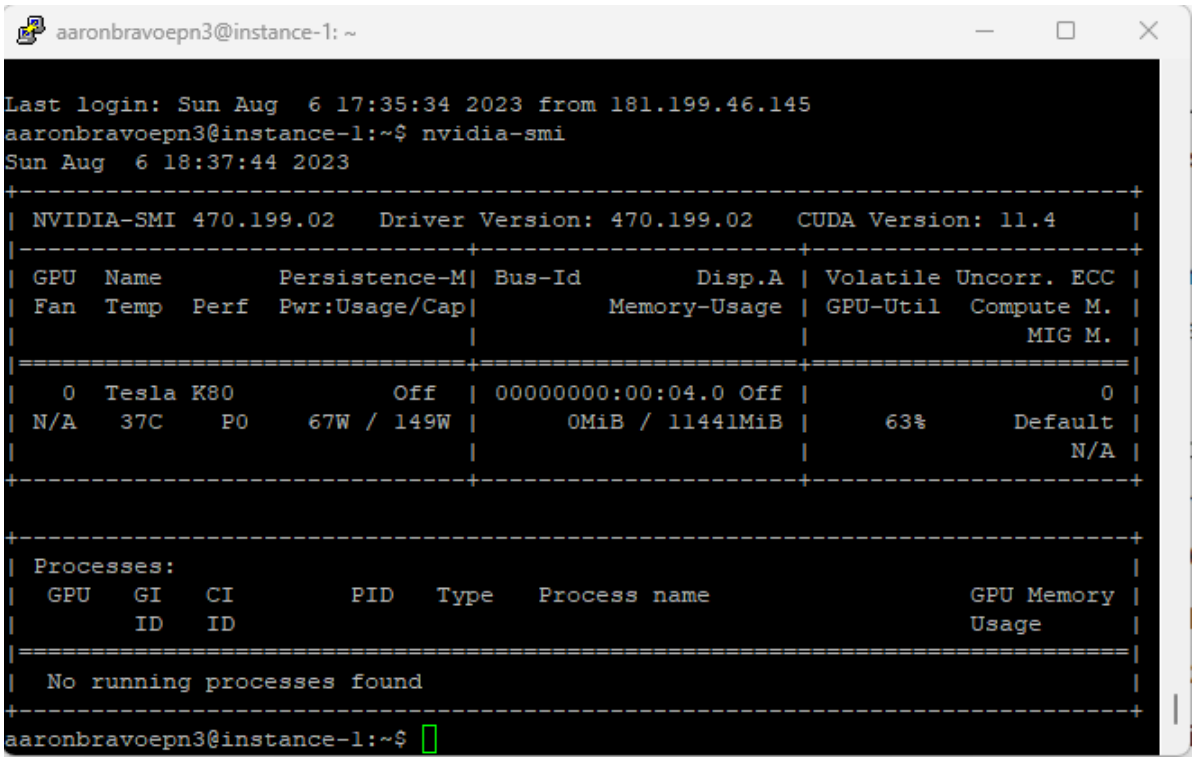

Figura 2.10: Estado de la GPU

2.4.4 NVIDIA CONTAINER TOOLKIT

Para la instalación de NVIDIA CONTAINER TOOLKIT, se siguen los pasos de su documentación oficial disponible en [39] en el apartado Docker Getting Started.

La documentación nos da pasos específicos para configurar Docker-CE utilizando el script oficial de Docker, configurar el repositorio de paquetes y la clave GPG, actualizar los repositorios locales y realizar la instalación.

Luego de la instalación se debe configurar el ”Docker daemon” para que reconozca ”NVIDIA Container Runtime”. Finalmente se realiza una prueba de la instalación para garantizar que los contendores creados tengan soporte para GPU. En la figura 2.11 se observa el estado de la GPU desde un contenedor Docker.

- - docker run –rm –runtime=nvidia –gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi: Prueba de instalación de NVIDIA CONTAINER TOOLKIT

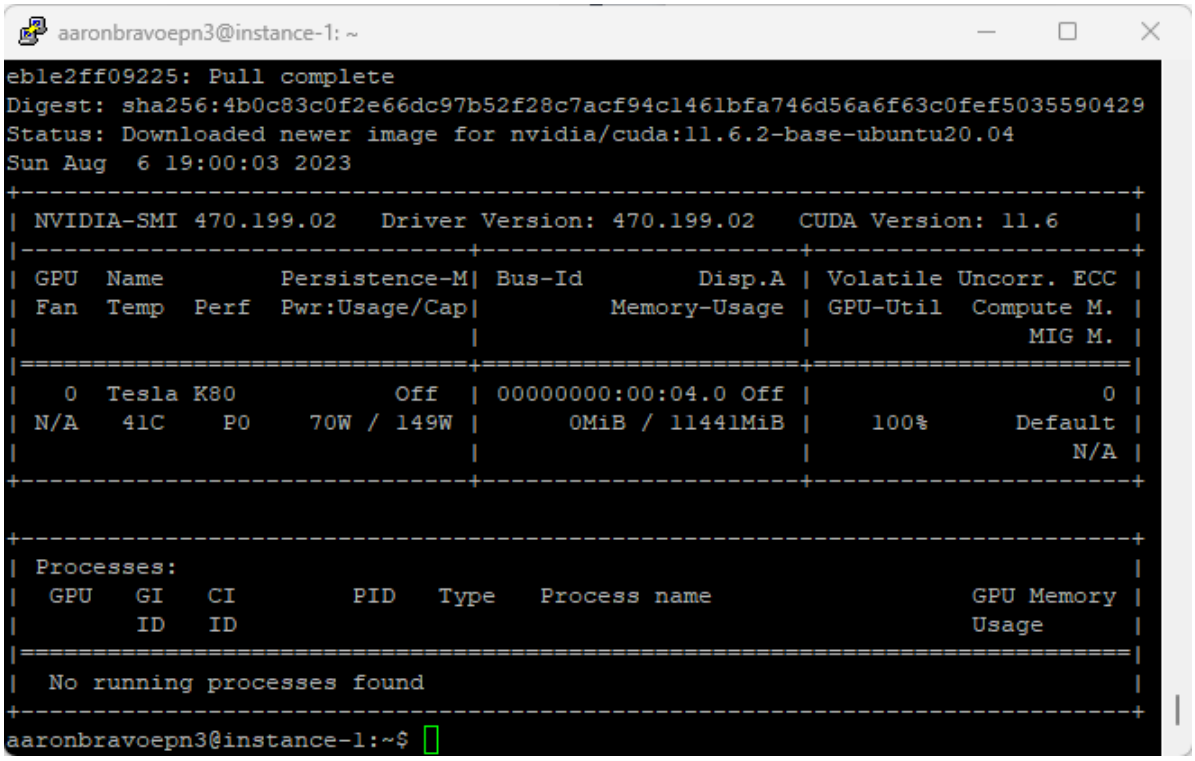

Figura 2.11: Prueba de NVIDIA CONTAINER TOOLKIT

2.4.5 Jupyter

Para la creación del contenedor de Jupyter se toma como base una imagen de tensorflow, estas imágenes aíslan la instalación de Tensorflow del resto del sistema. Los programas de Ternsorflow se ejecutan dentro del entorno virtual, que comparten recursos de la máquina anfitrión como directorios, GPU, y red. Según la documentación disponible en [40], docker es la forma más facil y adecuada para habilitar la compatibilidad de TendorFlow con GPU en Linux, porque solo requiere el controlador de GPU de NVIDIA y no es necesario la instalación del kit de herramientas CUDA de NVIDIA.

Las imágenes oficiales de TensorFlow para docker se encuentran en el repositorio de Docker Hub en [41].

Sobre esta imagen se procede a instalar los siguiente componentes.

- r-base
- r-base-dev

- npm
- nodejs
- python3
- python3-pip
- git
- nano

Luego de instalar los componentes mencionados se procede a instalar por medio del gestor de paquetes de Python pip las siguientes dependencias.

- jupyterhub
- notebook
- jupyterlab

Una vez instaladas estas dependencias se instala el paquete de configurable-http-proxy de forma global, esta instalación puede llegar a fallar para su integración con Jupyter por lo que también se procede a realizar una instalación por medio de su repositorio oficial disponible en [42]. Se clona el repositorio en un directorio local y dentro del directorio se ejecuta se instala por medio del gestor de paquetes de node (npm).

- - npm install -g configurable-http-proxy: Instalación de configurable-http-proxy por medio de npm

- - npm install: Instalación de configurable-http-proxy por medio del repositorio en github

Luego se procede a instalar el autenticador nativeauthenticator a través de su repositorio oficial de GitHub disponible en [43]. De igual forma se clona el repositorio en un directorio local y dentro del mismo se realiza la instalación.

- - pip3 install -e .: Instalación de nativeauthenticator por medio del repositorio en github

Luego se crea un archivo de configuración en un directorio específico cuyo contenido se encuentra en el anexo 4.

- - /etc/jupyterhub: Directorio de configuración de jupyterhub

- - jupyterhub_config.py: Archivo de configuración de jupyterhub

En el archivo se utiliza el modulo nativeauthenticator como autenticador, se establece como admin al usuario administrador, el cual tendrá privilegios adicionales en el servidor de Jupyter, que le permitirá gestionar y controlar el entorno de JupyterHub.

Además, se configura la URL predeterminada que redirigirá a los usuarios después de un inicio de sesión correcto. En este caso se establece que la URL predeterminada sea el árbol de archivos de Jupyter.

Luego se añade el entorno el kernel del lenguaje R para que se pueda ejecutar este lenguaje de programación dentro del entorno de Jupyter, los pasos para este proceso se los encuentra en la documentación oficial disponible en [44].

Finalmente, se lanza el servicio de jupyterhub con las configuraciones realizadas previamente, para ello se ejecuta el siguiente comando.

- - jupyterhub -f /etc/jupyterhub/jupyterhub_config.py: Ejecución de jupyterhub con sus respectivas configuraciones

Todos los pasos mencionados anteriormente se los añade a un archivo Dockerfile que consta en el anexo 5, este archivo sirve como base para construir una imagen Docker de tal forma que todas estas configuraciones no son necesarias de volverlas a realizar en caso de querer levantar nuevamente el entorno Jupyter desde un inicio.

Para copiar el archivo ”jupyterhub_config.py” al contenedor, se crea un directorio con los archivos Dockerfile y jupyterhub_config.py como se muestra en la figura 2.12.

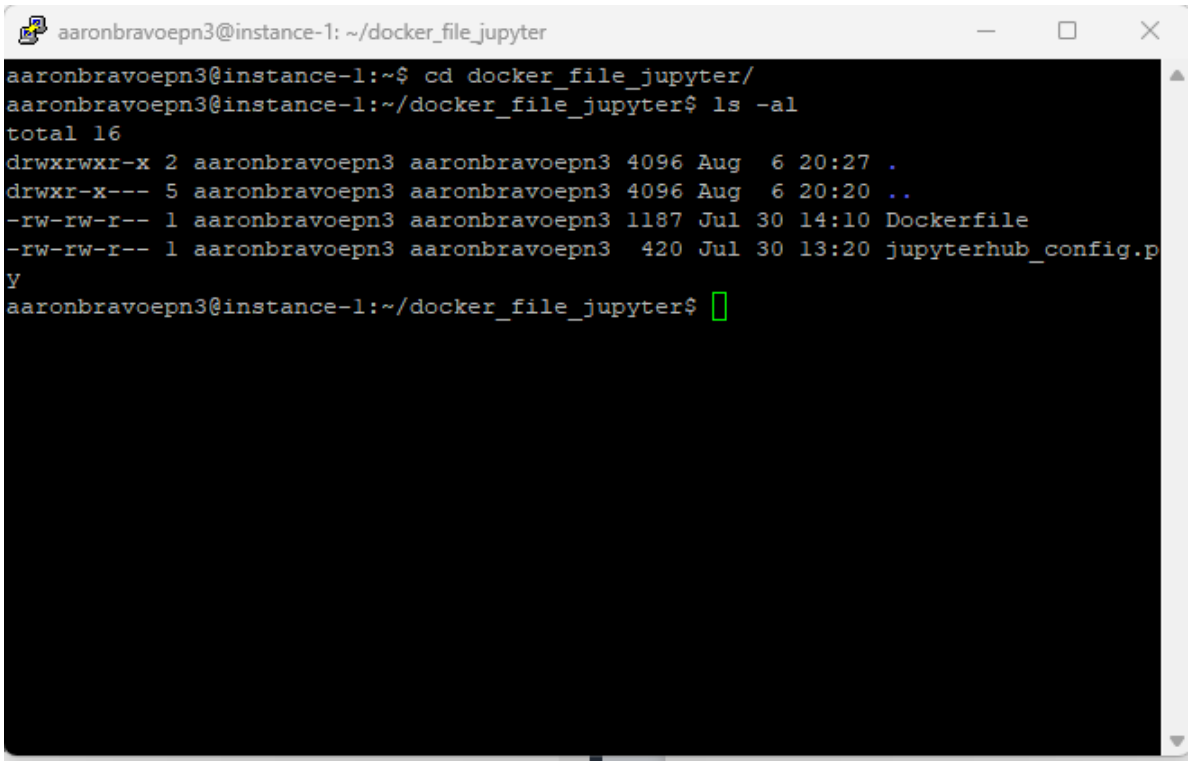

Figura 2.12: Directorio con archivos Dockerfile y jupyterhub_config.py

Con todos los pasos detallados previamente, se construyó una imagen que sirve de plantilla para desplegar el entorno de análisis con Jupyter.

- - docker build -t <nombre-imagen>.: Creación de imagen base para el despliegue de Jupyter.

En la figura 2.13, se puede observar que la creación de la imagen tuvo una duración de 1288.2 s, esto se debe a todas las dependencias y configuraciones necesarias para su construcción, además que se puede apreciar que la construcción de la imagen se la realizó con éxito.

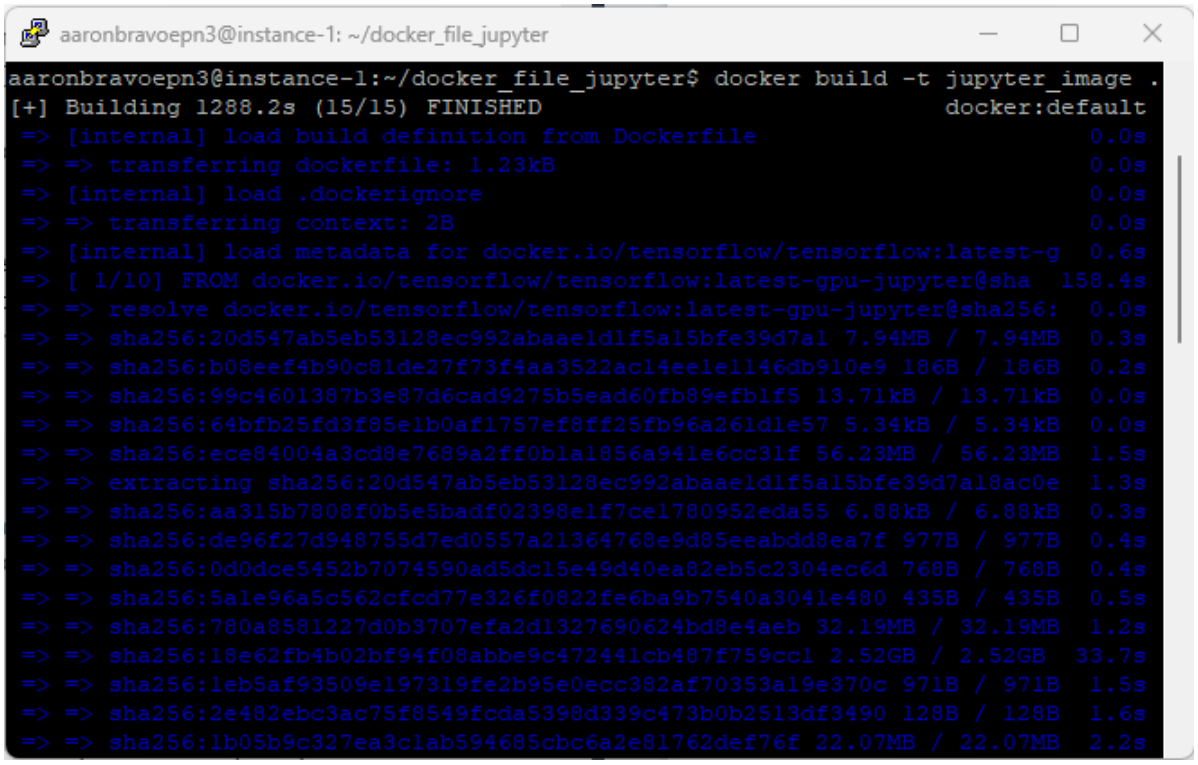

Figura 2.13: Construccion de la imagen Jupyter con archivo Dockerfile

Todas las imágenes construidas o almacenadas en el servidor pueden ser listadas con el fin de gestionarlas, las imágenes pueden ser almacenadas en el sistema operativo pero estas ocupan espacio de almacenamiento, por lo que se recomienda eliminar imágenes innecesarias, a continuación se listan algunos comandos de gestión de imágenes Docker.

- - docker pull nombre_imagen: Descarga una imagen desde el repositorio de Docker

- - docker images: Lista las imágenes disponibles en el sistema

- - docker rmi nombre_imagen: Elimina la imagen especificada

Para este caso, debido a que se descargó previamente imágenes para verificar instalación de Docker y de NVIDIA CONTAINER TOOLKIT, en la figura 2.14 se muestra la lista con estas imágenes y además la imagen construida para jupyterhub.

Con el fin de garantizar la persistencia de datos se crearon volúmenes en los contenedores, osea se crea un directorio en el sistema anfitrión para mapearlo a un directorio en el contenedor Docker, al directorio del sistema anfitrión se le deben otorgar permisos de escritura para que el contenedor de Jupyter guarde los archivos creados por los usuarios.

- - /home/ada/ws_jupyterhub: Directorio del sistema anfitrión

- - /home: Directorio del contenedor

- - sudo chmod o+w /home/ada/ws_jupyterhub: Agregar permisos de escritura al directorio del sistema anfitrión

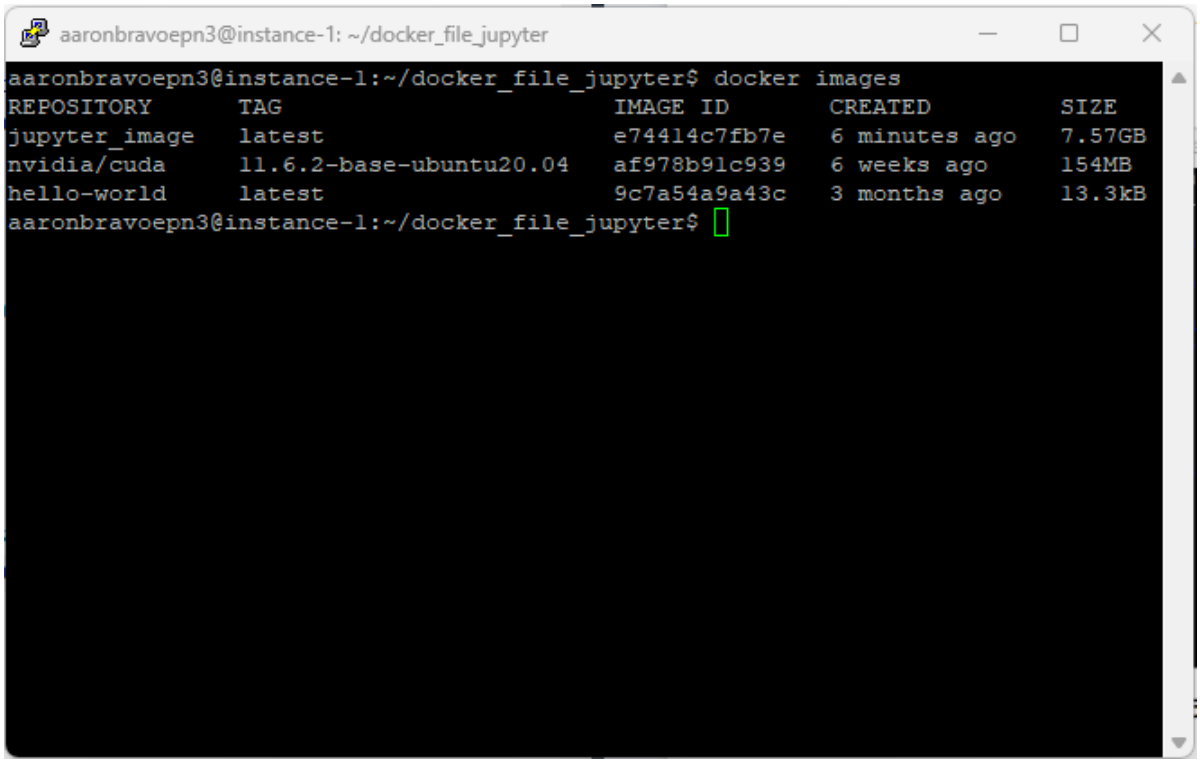

Figura 2.14: Imágenes disponibles

Un contenedor Docker toma una imagen como plantilla y se ejecuta según las especificaciones de arranque.

Las especificaciones de arranque son las siguientes.

- -d: Establece el modo detach, que ejecuta el contenedor en segundo plano.
- -it: Se combina con la opción -d y habilita interacción con la linea de comandos (CLI) del contenedor, aún cuando se ejecute en segundo plano.
- –gpus all: Indica que el contenedor haga uso de todas las GPU disponibles en el sistema anfitrión.
- -p 8000:8000: Mapea el puerto 8000 del contenedor al puerto 8000 del sistema anfitrión, de tal forma que se acceda a los servicios que se estén escuchando en el puerto 8000 del contenedor.

- -v /home/ada/ws_jupyterhub:/home/: Mapea el directorio del sistema anfitrión al directorio del contenedor

- –name jupyter_container: Asigna el nombre de jupyter_container al contenedor

Con los parámetros mencionados anteriormente se ejecuta el contenedor. Si la ejecución del contenedor es exitosa, el sistema imprime en consola una cadena de caracteres como se muestra en la figura 2.15 que representa el identificador del contenedor Docker lanzado.

- - docker run -d -it –gpus all -p 8000:8000 -v /home/ada/ws_jupyterhub/:/home/ –name jupyter_container jupyter_image: Ejecución del contenedor desde la imagen construida previamente

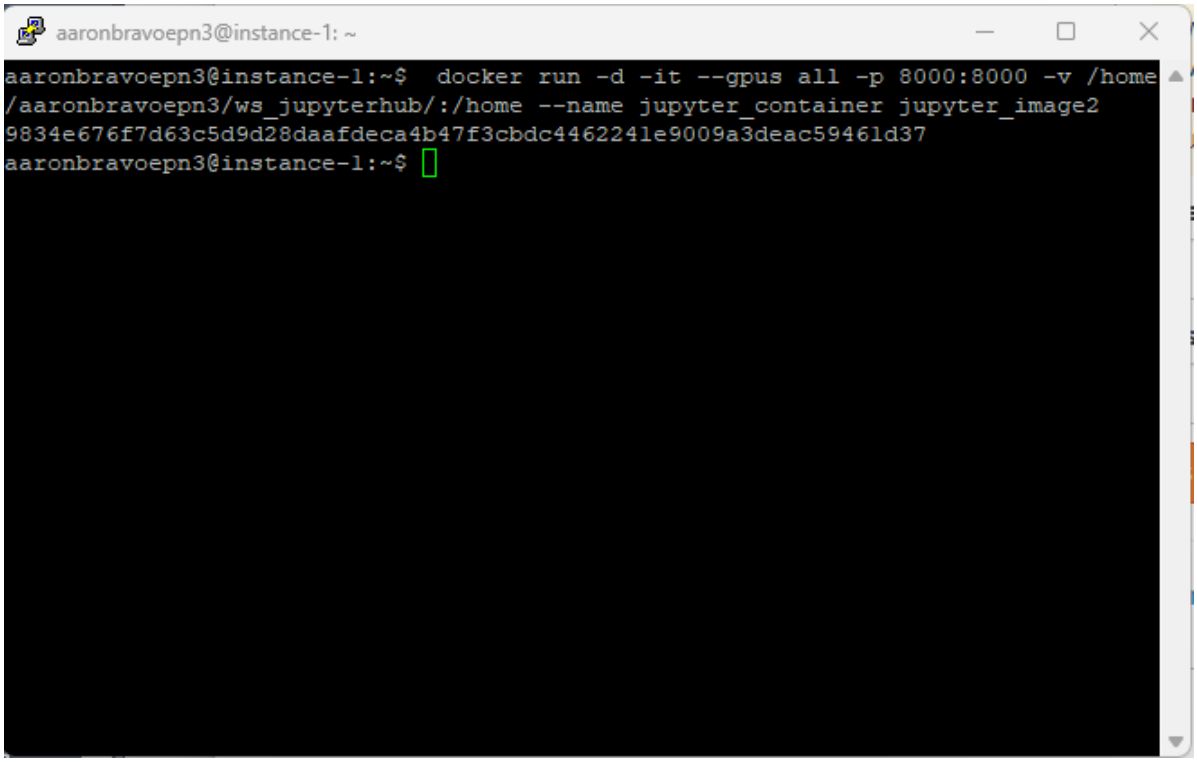

Figura 2.15: Ejecución del contenedor Jupyter

Luego que el contenedor se ha ejecutado con éxito se puede verificar su estado, como se observa en la figura 2.16, Docker presenta detalles de los contenedores, como su identificador, la imagen base, el servicio ejecutándose, el tiempo que tiene creado el contenedor, el status, los puertos mapeados y el nombre del contenedor.

- - docker ps -a: Muestra detalles de todos los contenedores del sistema

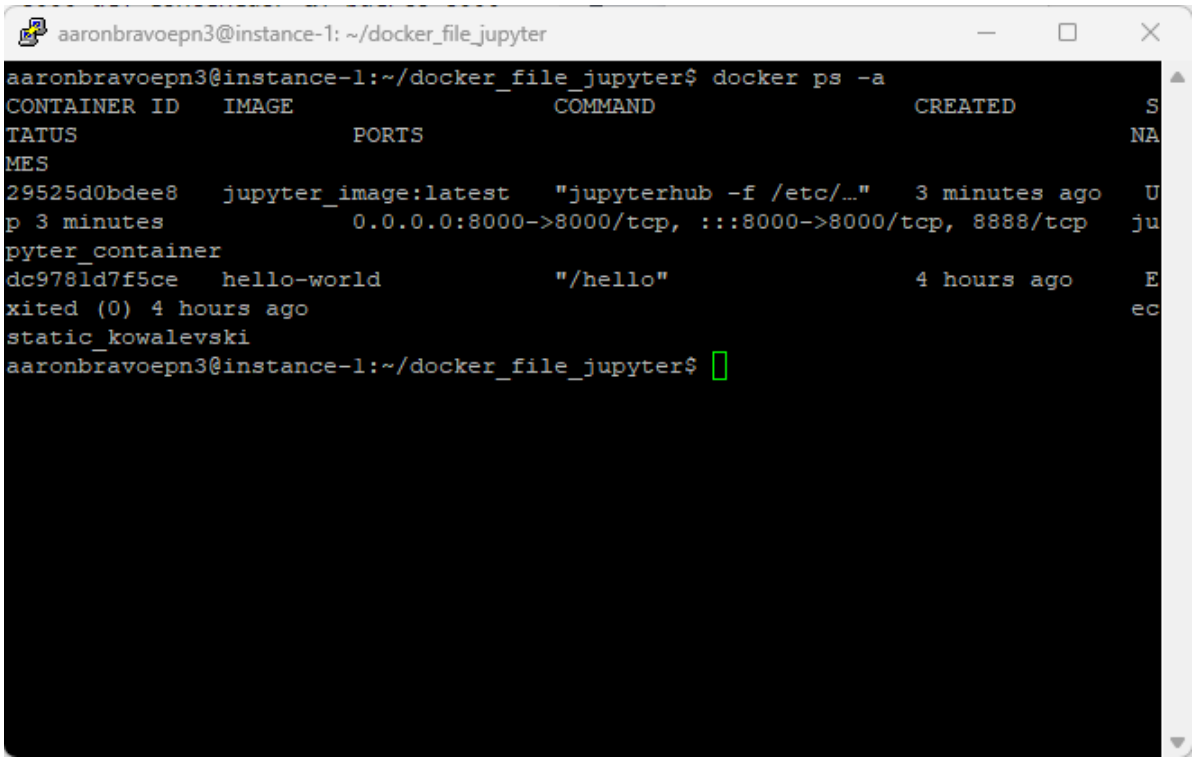

Figura 2.16: Estado de contenedores Docker

Hasta aqúi se han visto los pasos necesarios para el despliegue de la plataforma Jupyter, sin embargo para la integración de Jupyter con Hadoop requiere configuraciones adicionales para su comunicación. Sea un cluster de contenedores Hadoop ejecutándose en el servidor se procede a listar las redes configuradas en el sistema, en la figura 2.17 se muestra que existen las redes por defecto que son: bridge, host y none y adicionalmente se observa una red denominada hadoop-net donde están conectados los contenedores del cluster de Hadoop, a la red de Hadoop se conecta el contenedor donde se ejecuta Jupyter con el fin de que los usuarios del laboratorio sean capaces de escribir y leer archivos en Hadoop.

- - docker network connect hadoop-net jupyter_container: Coneta el contenedor de Jupyter a la red del cluster de Hadoop

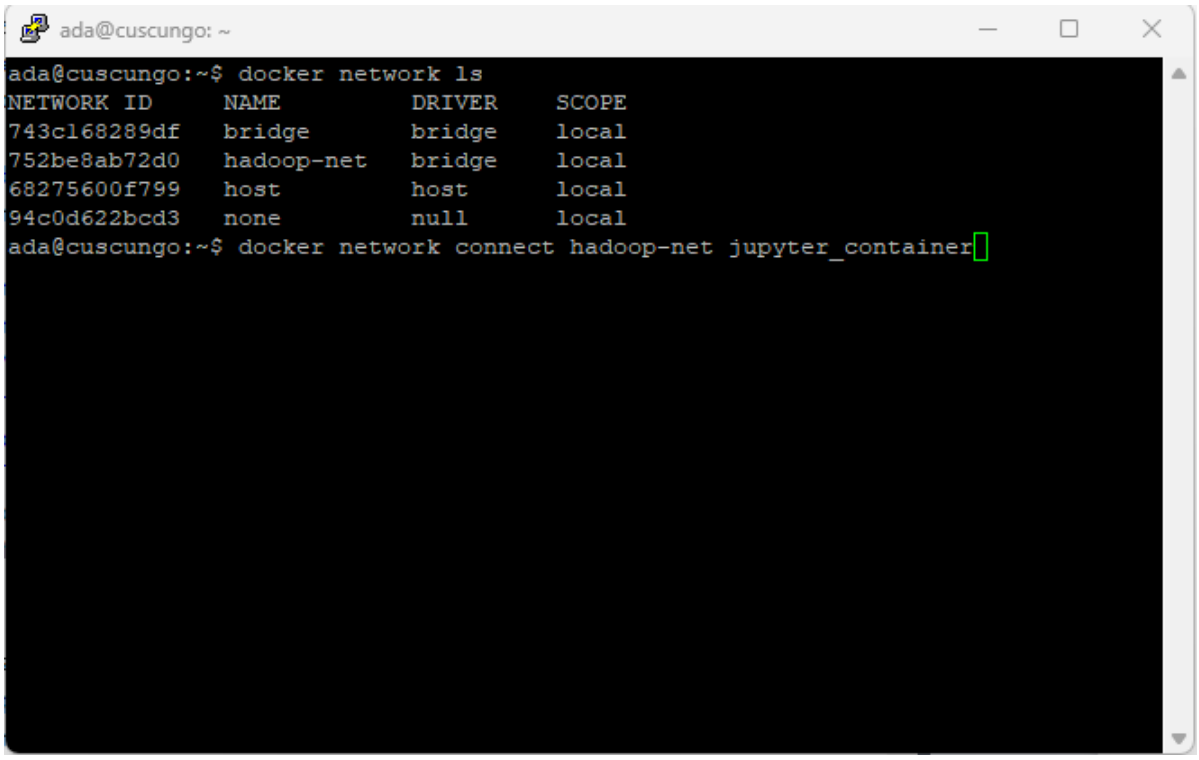

Figura 2.17: Redes disponibles en Docker

Los administradores de infraestructura pueden encontrar el manual de despliegue en el anexo 6.

Una vez que el contenedor se haya ejecutado, se procede a ingresar a la plataforma por medio de un navegador web, se coloca la ip en la url junto con el puerto 8000 de la siguiente manera: ”<dirección-ip>:8000” y se observa la interfaz de autenticación como en la figura 2.18

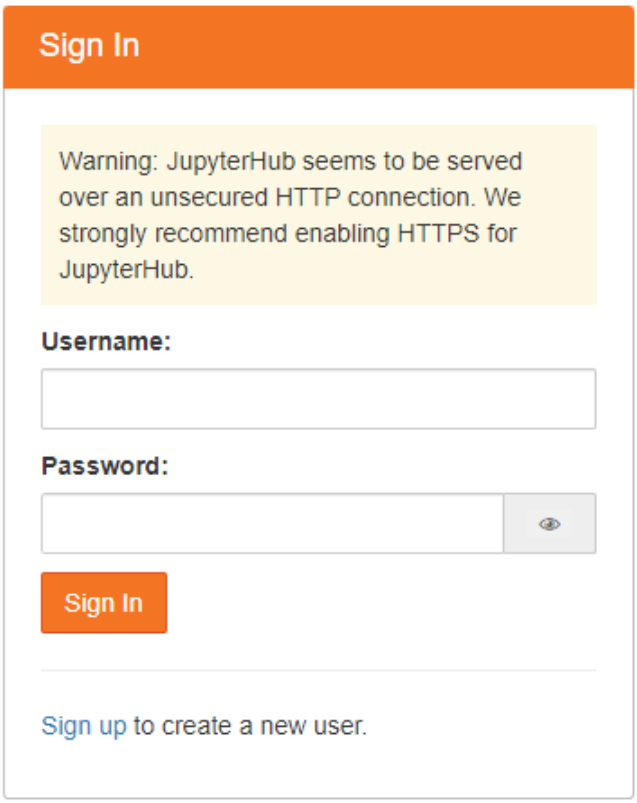

Figura 2.18: Interfaz de autenticación de Jupyter

Luego de la implementación, los usuarios siguen el manual disponible en el anexo 7 para su registro.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

Luego de la implementación de contenedores Docker en el servidor de altas prestaciones del laboratorio ADA se presentan en esta sección los resultados obtenidos de su despliegue. La tecnología de virtualización basada en contenedores Docker ha brindado un entorno simplificado para desplegar y gestionar las aplicaciones para las tareas del laboratorio, lo que implica mejoras significativas en la utilización de recursos y ha reducido la complejidad de la operativa. Con esta implementación, se ha logrado tiempos de despliegue de las aplicaciones mucho más rápidos, lo que permite a los investigadores y estudiantes de posgrado trabajar de la maneras más eficiente centrándose sólo en las tareas de análisis de datos dejando un lado la administración de las aplicaciones requeridas.

3.1.1 Funcionamiento de contenedor Jupyter

Se realizó la verificación de la aplicación Jupyter y los servicios requeridos que se identificaron en los requisitos. El contenedor para la plataforma de Jupyter de desplegó correctamente con soporte para los lenguajes Python y R y con soporte para GPU. Docker ofrece una herramienta para observar los registros de los contenedores en ejecución, la figura 3.1 muestra los registros en tiempo real del conrtenedor de Jupyter ahi se observan las actividades que realizan los usuarios al servicio de Jupyter.

- - docker logs -f nombre_contenedor: Registros en tiempo real de un contenedor Docker.

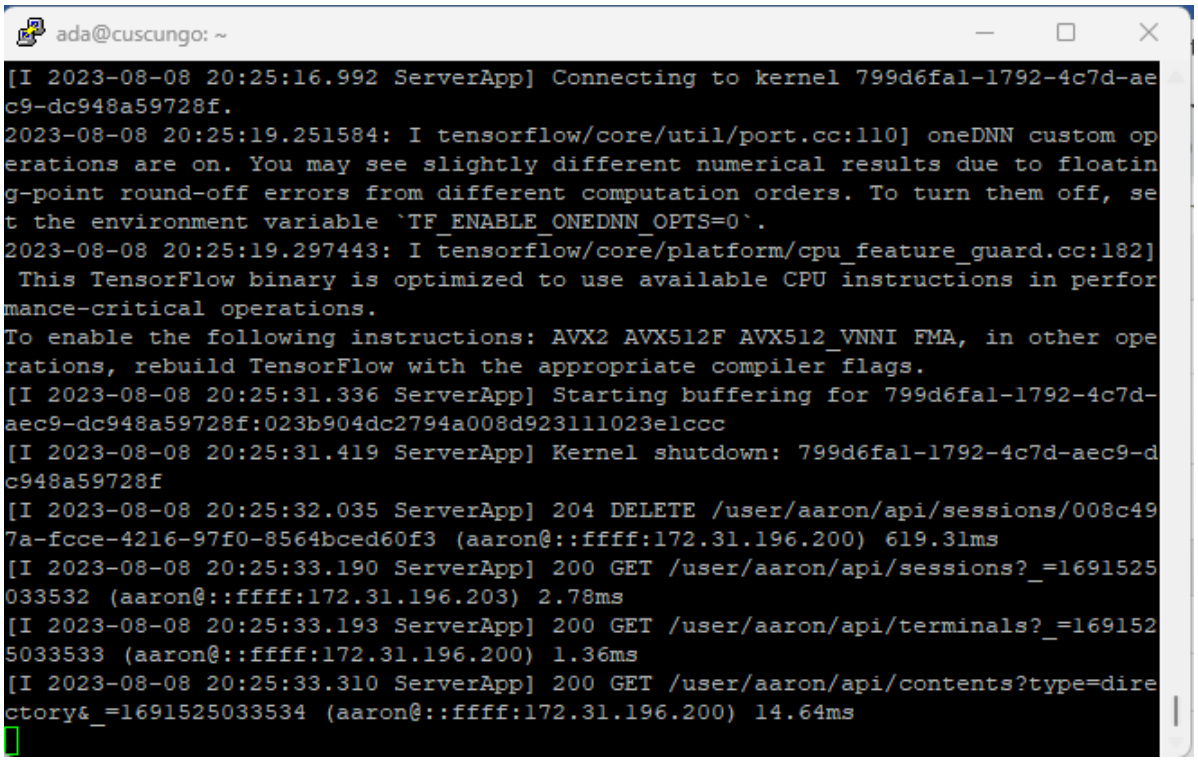

Figura 3.1: Registros en tiempo real del contenedor Jupyter

Para la verificación del servicio NativeAuthenticator, se procedió a crear el usuario ”admin”, que es el encargado de administrar los servicios de Jupyter.

En la figura 3.2, se muestra la interfaz de registro del usuario administrador, esta interfaz está disponible en la url http://172.31.124.130:8000/hub/signup, el usuario administrador proporciona el usuario admin y una contraseña segura, Jupyter identifica al usuario ”admin” como usuario administrador porque se lo definió en el archivo de configuración cuando se desplegó la aplicación, luego de su registro el usuario administrador ingresa con las credenciales proporcionadas y gestiona los usuarios en la http://172.31.124.130:8000/hub/authorize.

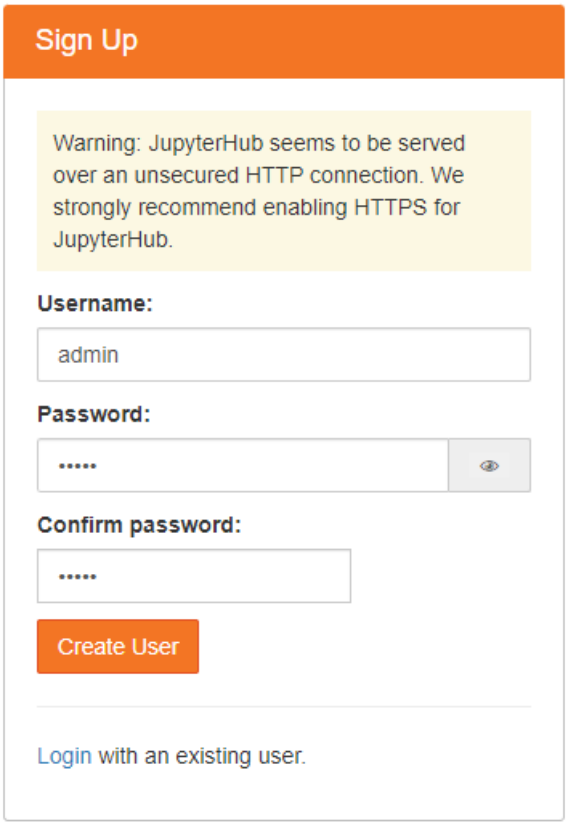

Figura 3.2: Creación de usuario administrador

En la figura 3.3 se muestra la interfaz de la gestión de usuarios, en esta interfaz el administrador permite o restringe el acceso , los nuevos usuarios registran sus credenciales en la url http://172.31.124.130:8000/hub/signup y notifican al administrador, el administrador presiona el botón ”Authorize” o ”Unauthorize”, luego que el administrador permite el acceso, los nuevos usuarios ingresan a la aplicación Jupyter y hacen uso de los servicios.

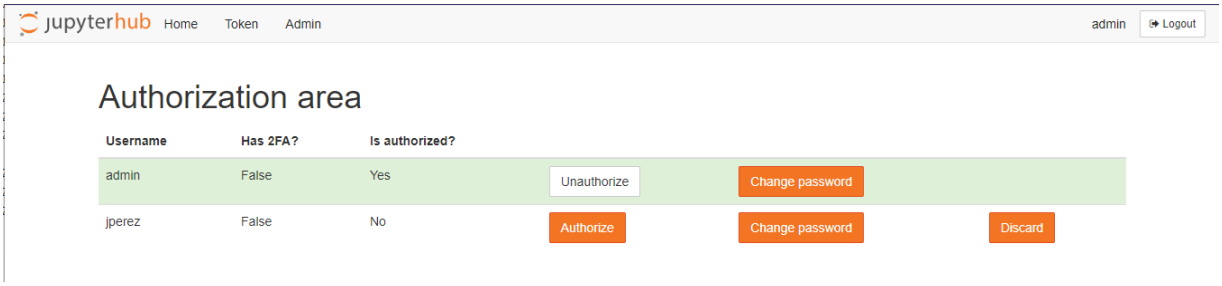

Figura 3.3: Creación de usuario administrador

Una ventaja de JupyterHub es que permite al usuario administrador gestionar los recursos, de tal forma que si un usuario hace mal uso de la plataforma o incumple con las políticas de uso definidas por el laboratorio ADA, el administrador está en la capacidad de restringir o limitar los servicios para usuarios en específico. En la figura 3.4, se muestra el panel de administración, en este ejemplo se muestra el usuario ”jperez”, al cuál previamente se le brindó los permisos de acceso e ingresó al sistema, si el administrador tiene la necesidad de limitarle o restringirle los servicios, presiona el botón ”Stop Server”, de esta forma los servicios de Juypyter se detendrían para el usuario ”jperez”.

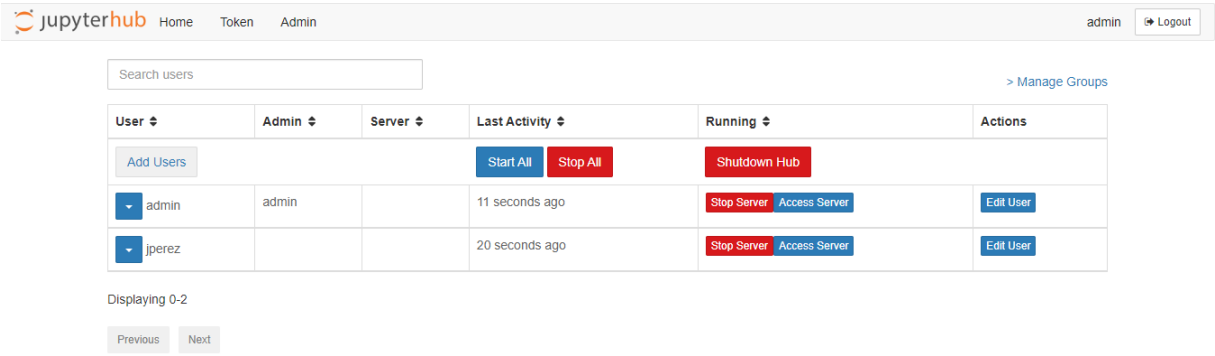

Figura 3.4: Panel de administración

El administrador también tiene la capacidad de cambiar algunos parámetros, presionando el botón ”Edit User” del panel de administración y como se observa en la figura 3.5, se puede cambiar el nombre de usuario, asignar como usuario administrador o eliminarlo definitivamente.

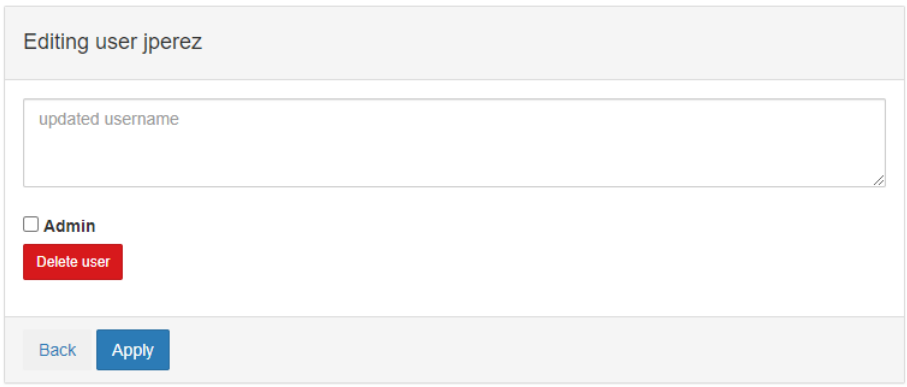

Figura 3.5: Creación de usuario administrador

Con el correcto despliegue de JupyterHub en el servidor de altas prestaciones y gracias a las operaciones de administración que ofrece esta plataforma, el administrador del laboratorio tiene el control necesario para que las capacidades computacionales del equipo se usen según políticas de uso adaptadas a las necesidades del laboratorio.

3.1.2 Funcionamiento de interfaz con lenguaje Python

El contenedor de Jupyter permite la ejecución de código en lenguaje Python para los análisis de datos, la figura 3.6 muestra el resultado de la ejecución de un código de prueba que crea un arreglo grande de números, los eleva al cuadrado y calcula su suma.

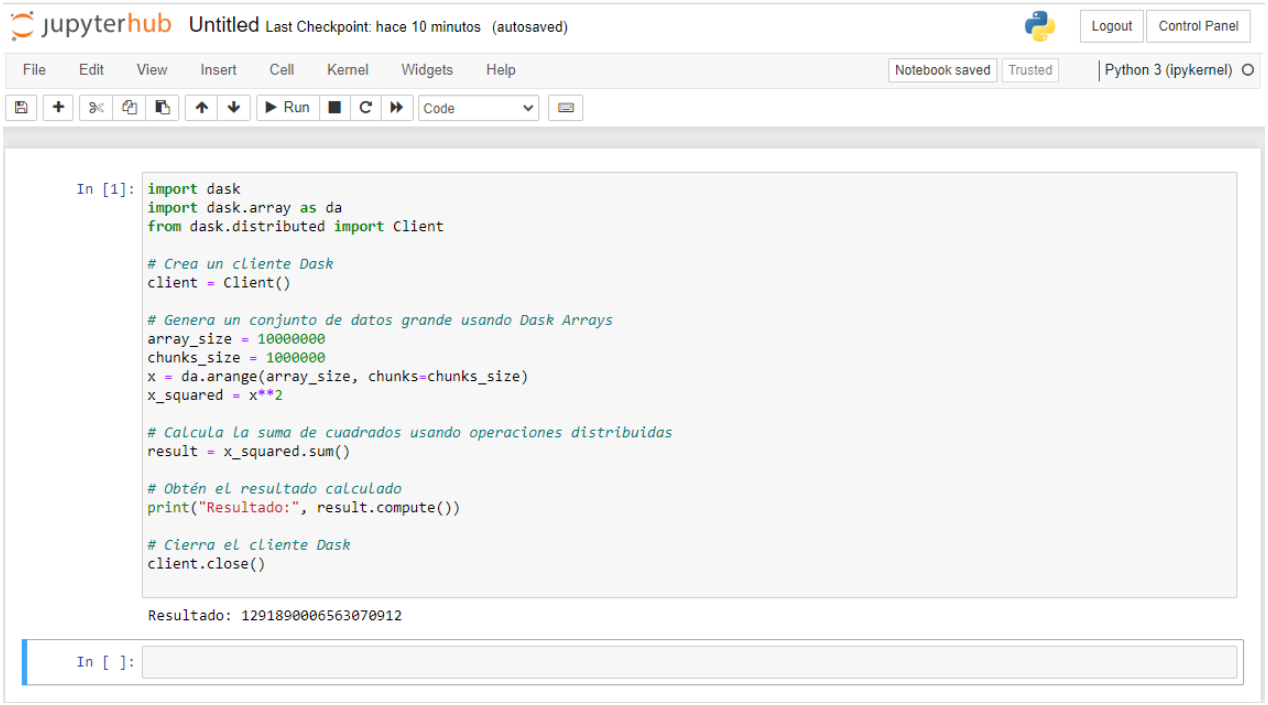

Figura 3.6: Prueba de ejecución de código Python

3.1.3 Funcionamiento de interfaz con lenguaje R

Los investigadores y estudiantes de posgrado del laboratorio ADA requieren ejecutar código de lenguaje R para su análisis de datos, en la figura 3.7 se observa el resultado de ejecutar un código que obtiene estadística de una matriz que resulta de la multiplicación de dos matrices con números aleatorios.

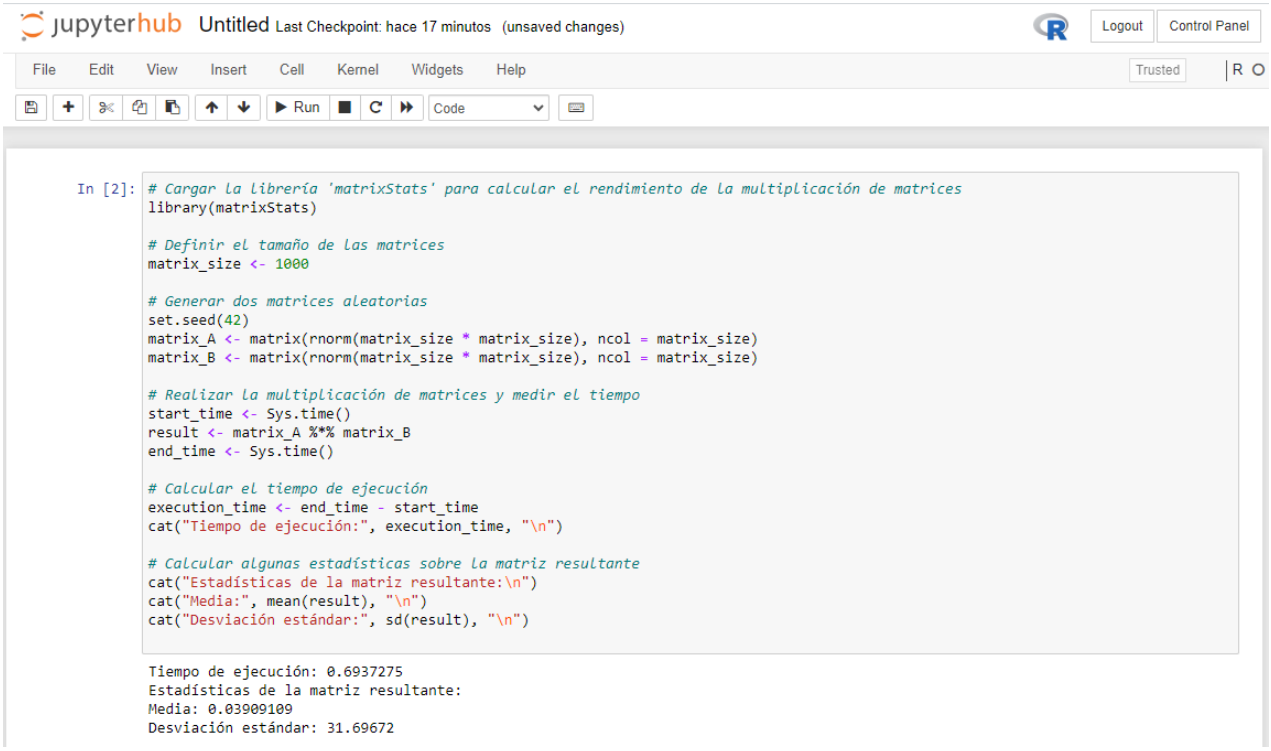

Figura 3.7: Prueba de ejecución de código R

3.1.4 Funcionamiento de interfaz con soporte para GPU

El soporte para GPU en Jupyter viene dado por la librería tensorflow de Python, en la figura 3.8 se observa la ejecución de un script que lista las GPU disponibles en el entorno.

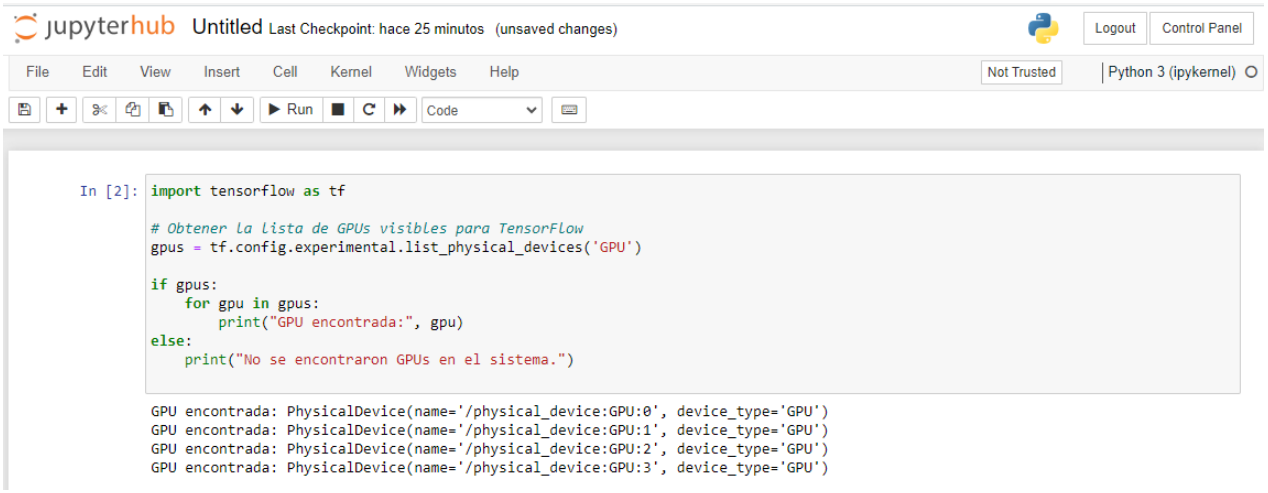

Figura 3.8: Test de GPU en Python

3.1.5 Funcionamiento de integración de Jupyter con Hadoop

Realizar operaciones de lectura y escritura al sistema de archivos de Hadoop desde el entorno de Jupyter implica un proceso sencillo con el cual se permite la integración de los cuadernos de Jupyter con el almacenamiento de grandes volúmenes de datos en Hadoop. En este apartado se muestran los resultados obtenidos y los pasos requeridos para llevar a cabo esta tarea. Para realizar esta acción se solicita previamente al administrador de la plataforma Hadoop permisos de escritura, para ello se debe proporcionar el nombre de usuario con el que se ingresa a Jupyter, en la figura 3.9 se muestra el código necesario para cargar un archivo en el sistema de archivos de Hadoop, para ello se crea una conexión con la dirección IP y el puerto del HDFS, luego se importa la clase InsecureClient del módulo hdfs, para establecer un cliente y cargar el archivo con la función ”upload”, dentro de la función se ingresan los parámetros de la ruta del HDFS donde se carga el archivo y la ubicación local del archivo.

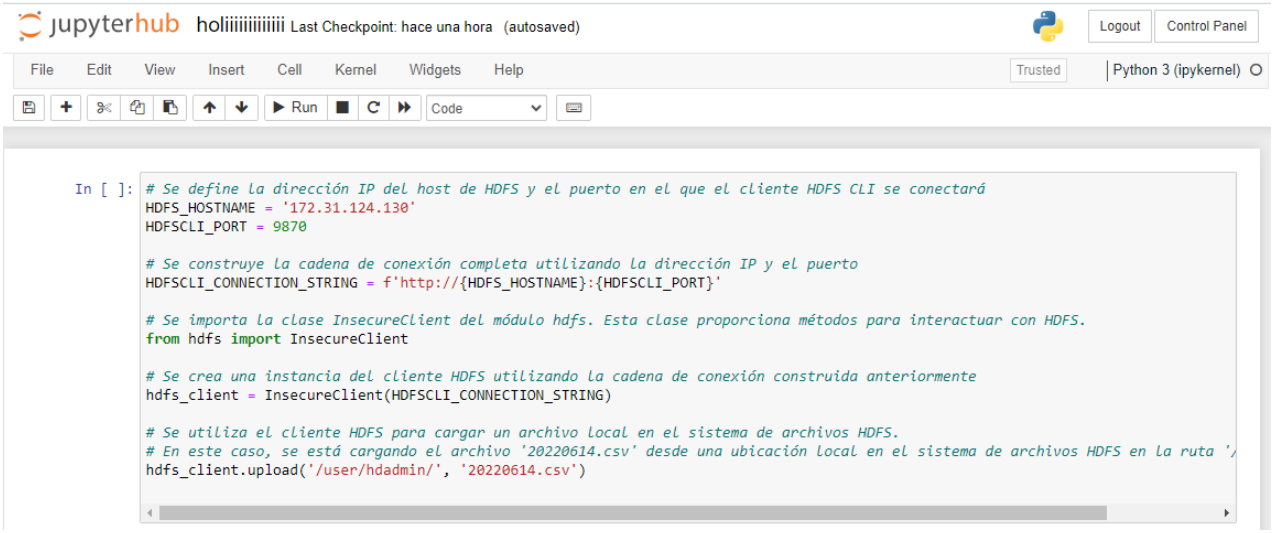

Figura 3.9: Código para cargar un archivo en el sistema de archivos HDFS

Hadoop proporciona una interfaz web accesible por medio de la url http://172.31.124.130:9870/explorer.html#/, en esta interfaz el navega en el sistema de archivos con el fin de verificar los archivos disponibles en el lago de datos. La figura 3.10 muestra los archivos cargados en en Hadoop y como se observa, el archivo ”20220614.csv”, se encuentra en la ruta ”/user/hdadmin” que se especificó en el paso anterior.

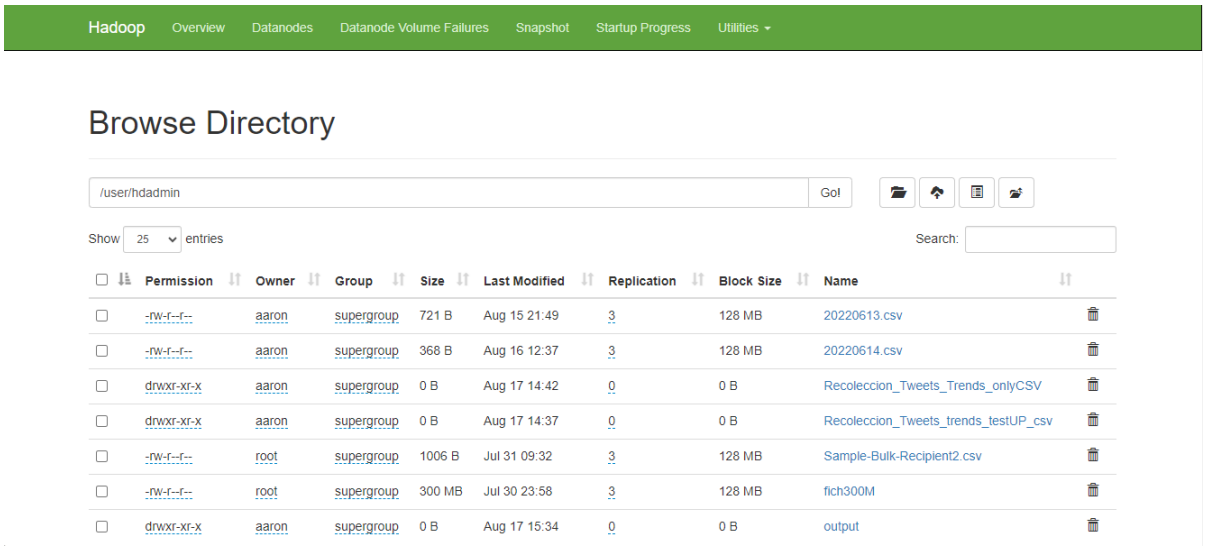

Figura 3.10: Verificación de carga de archivo a Hadoop en interfaz web

En el campo de la analítica de datos no es suficiente con el análisis de un sólo archivo por lo que los docentes y estudiantes de maestría van a requerir analizar un sin números de datos que la mayoría de veces los tienen disponibles en directorios locales de Jupyter. La figura 3.11 muestra el código que se usó para subir al sistema de archivos de Hadoop, una carpeta con varios archivos de tweets para su análisis. El procedimiento se parece al que se usó para subir un archivo con la diferencia de que como parámetro se especifica el directorio de toda la carpeta y no la de un archivo en específico. En la figura 3.10 mostrada previamente se observa que la carpeta ”Recoleccion_Tweets_trends_testUP_csv” se cargó exitosamente en el directorio ”/user/hdadmin/” que se especificó en el código de carga.

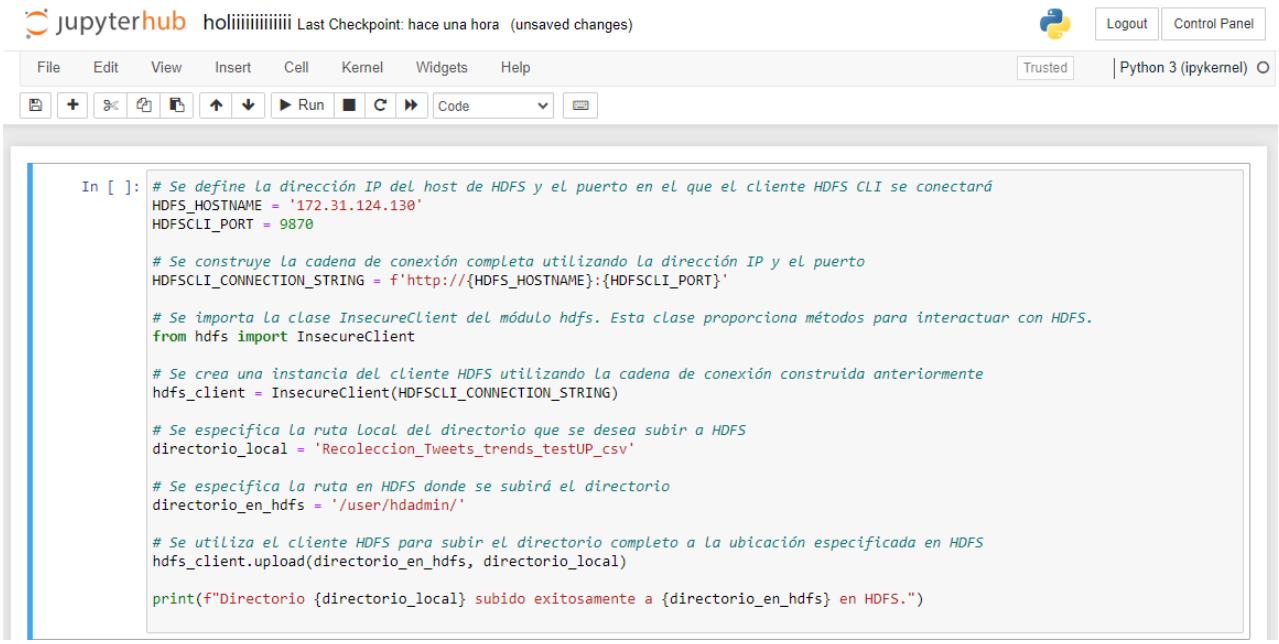

Figura 3.11: Código para cargar un directorio en el sistema de archivos HDFS

En Hadoop se guardan grandes volúmenes de datos, pero estos datos no sirven de nada si no se los procesa para generar conocimiento, por ellos los usuarios del laboratorio ADA, acceden a la información de los datos disponibles en el sistema de archivos. La figura 3.12 muestra un ejemplo de la lectura del archivo ”20220617.csv” en la ruta ”/user/hdadmin/output/”, al igual que la carga de archivos y directorios se establece una cadena de conexión, se instancia un cliente HDFS y se usa la función ”read”, para finalmente imprimir en consola el contenido del archivo.

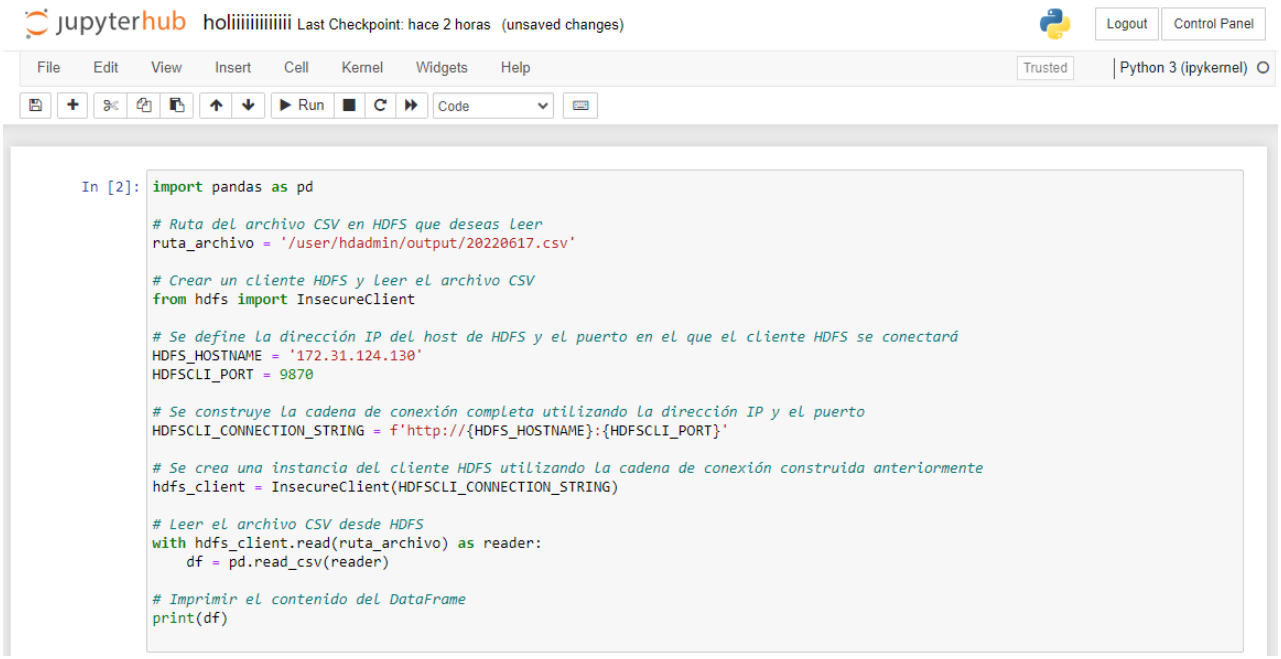

Figura 3.12: Código para leer un archivo en HDFS

3.1.6 Pruebas de rendimiento de contenedor Jupyter

Lo que se busca con la implementación de este entorno es que los docentes y estudiantes de posgrado trasladen la dependencia de altas características computacionales a un servicio en la nube, por ello es necesario observar el comportamiento del servidor de altas prestaciones contra los equipos personales con recursos estándar. Sea la figura 3.13, donde se muestra la ejecución de un algoritmo proporcionado por un usuario del laboratorio ADA, este algoritmo hace una limpieza de datos y al final muestra el tiempo de ejecución.

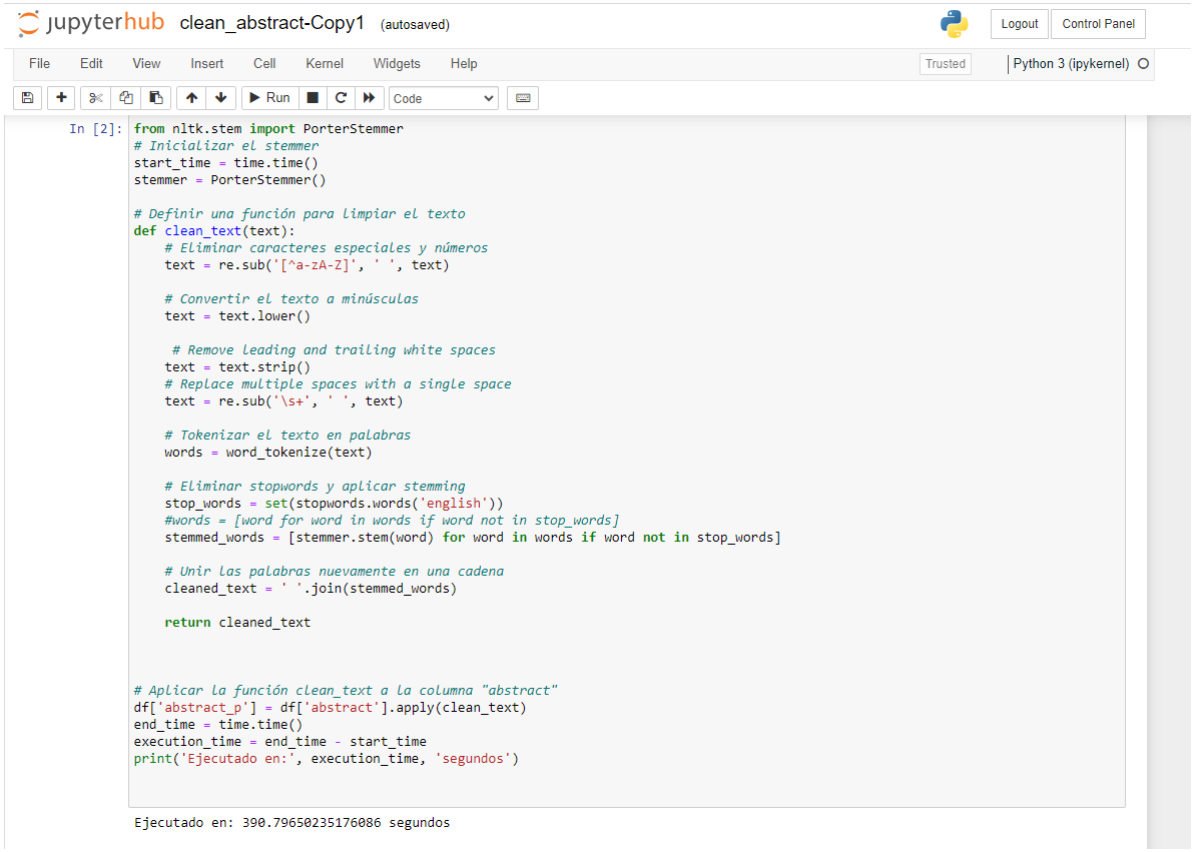

Figura 3.13: Prueba de rendimiento en el quipo de altas prestaciones del laboratorio ADA

El algoritmo mencionado previamente ejecutado en la plataforma implementada en el laboratorio se ejecuta en un tiempo promedio de 390 s, y como se observa en la figura 3.14, la ejecución del algoritmo no afecta de forma significativa a los 64 núcleos de procesador que tiene el servidor.

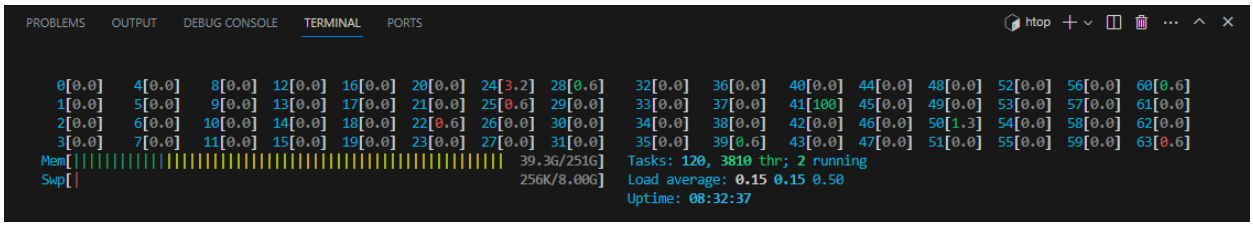

Figura 3.14: Prueba de rendimiento en servidor del laboratorio ADA.

El mismo algoritmo de limpieza de datos fue ejecutado en un equipo con 8GB de RAM, un procesador Intel(R) Xeon(R) CPU @ 2.30GHz con 2 núcleos y 100 GB de almacenamiento, la ejecución para este caso tomó un tiempo promedio de 446 s, esto nos indica que el uso del servidor si presenta ventaja al realizar algoritmos de análisis de datos. La figura 3.15,

muestra el uso de CPU al ejecutar el algoritmo y como se puede observar los 2 núcleos realizan un trabajo significativo en procesamiento.

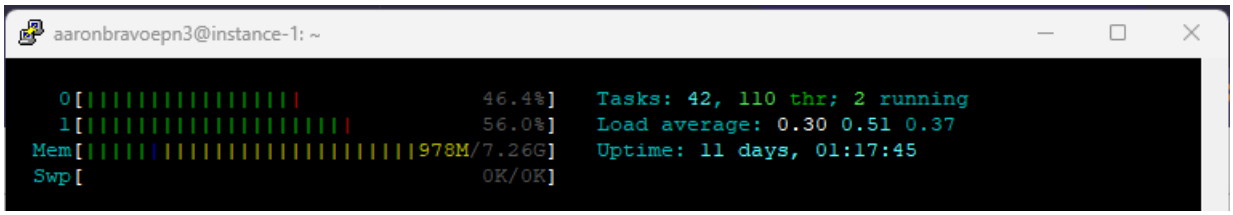

Figura 3.15: Prueba de rendimiento en equipo con prestaciones estándar.

Se realizó una prueba de rendimiento para comparar los tiempos de ejecución de un algoritmo proporcionado por un usuario del laboratorio, el algoritmo recopila y filtra Tweets relacionados con tendencias para un día en específico en el marco de las manifestaciones de Junio del 2021. La prueba consistió en ejecutar el algoritmo en el servidor de altas prestaciones y comparar los tiempos de ejecución con los tiempos proporcionados por el usuario. En la tabla 3.1 se muestran diferencias notables en los tiempos de ejecución. Para el caso del algoritmo ejecutado bajo el parámetro de las tendencias de Tweets con el día 13/06/2022 se observa que el usuario ejecutó el algoritmo en un tiempo de 31 minutos y 44 segundos y el mismo algoritmo ejecutado en el equipo con altas prestaciones tomó un tiempo de 14 minutos y 6 segundos, así mismo con el día 14/06/2022 el usuario realizó su ejecución en 1 hora 10 minutos y 12 segundos, mientras que en el equipo del laboratorio tomó un tiempo de 29 minutos y 24 segundos.

Tabla 3.1: Tiempos de ejecución del algoritmo de recopilación y filtración de Tweets.

Día	**Equipo personal**	**Equipo del laboratorio ADA**
	[s]	[s]
13/06/2022	1904.39	846.27
14/06/2022	4812.02	1764.64

Así mismo se procedió a realizar un análisis de los Tweets recopilados en 19 días en una sola ejecución, tomando un tiempo de 23 horas y 42 minutos, esto es importante recalcar ya que el usuario del laboratorio manifestó que no le fue posible ejecutar el análisis en una sola ejecución ya que se agotaron los recursos de su computador y tuvo que reiniciarlo.

Con estos resultados se observa que la utilización de un servidor de altas prestaciones su-

pone una ventaja en cuanto a rendimiento frente a equipos personales de los usuarios del laboratorio. Conforme se aumenta el volumen de los datos y la complejidad de los algoritmos, el uso de este servidor además de agilizar los cálculos y análisis, ofrece facilidad para adaptarse a las necesidades cambiantes de la analítica de datos

Finalmente se muestra el uso de la GPU en algoritmos que usan la biblioteca tensorflow. En la figura 3.16, se muestra un algoritmo que usa tensorflow para crear y entrenar un modelo de red neuronal.

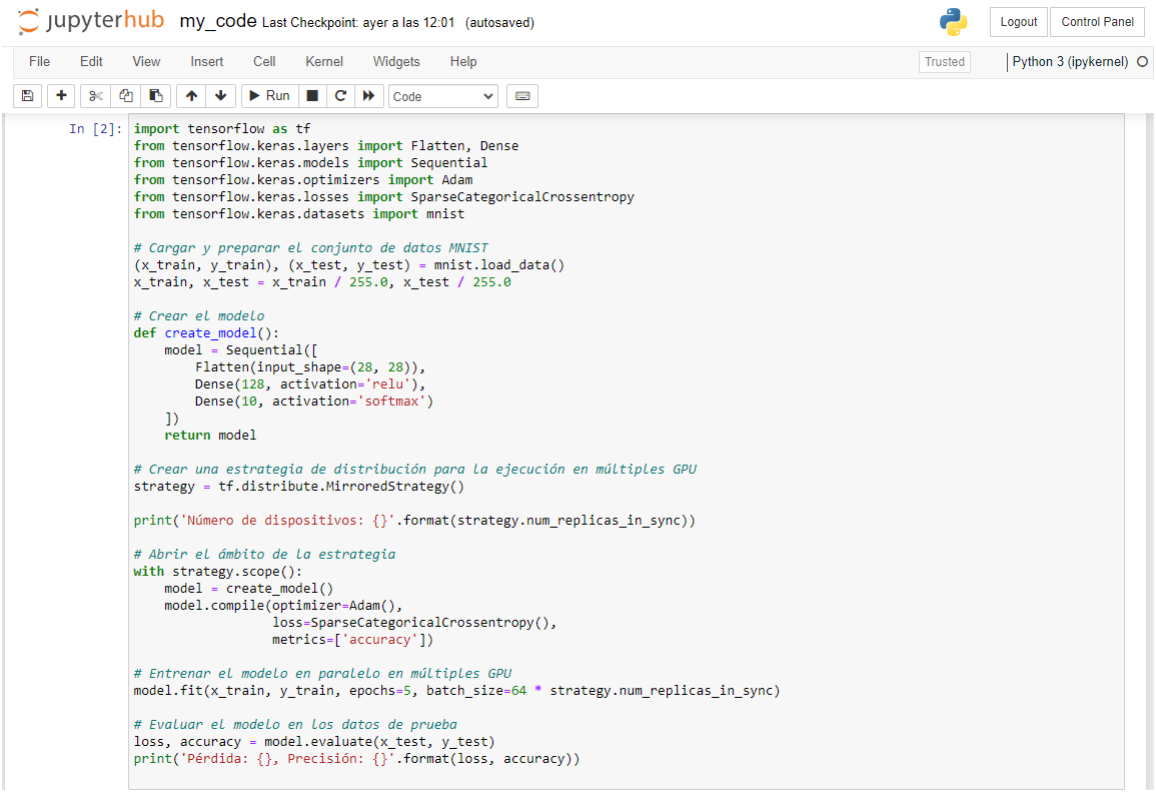

Figura 3.16: Algoritmo que usa la biblioteca tensorflow para uso de GPU.

Con la ejecución de este código se observa que la integración de tensorflow con soporte para GPU se ejecuta adecuadamente, la figura 3.17 muestra el estado de la GPU en tiempo real cuando se ejecuta el algoritmo, se observa que la tarjeta gráfica del servidor divide los procesos a todas las unidades disponibles, que para este caso son 4, como detalla el proveedor del equipo.

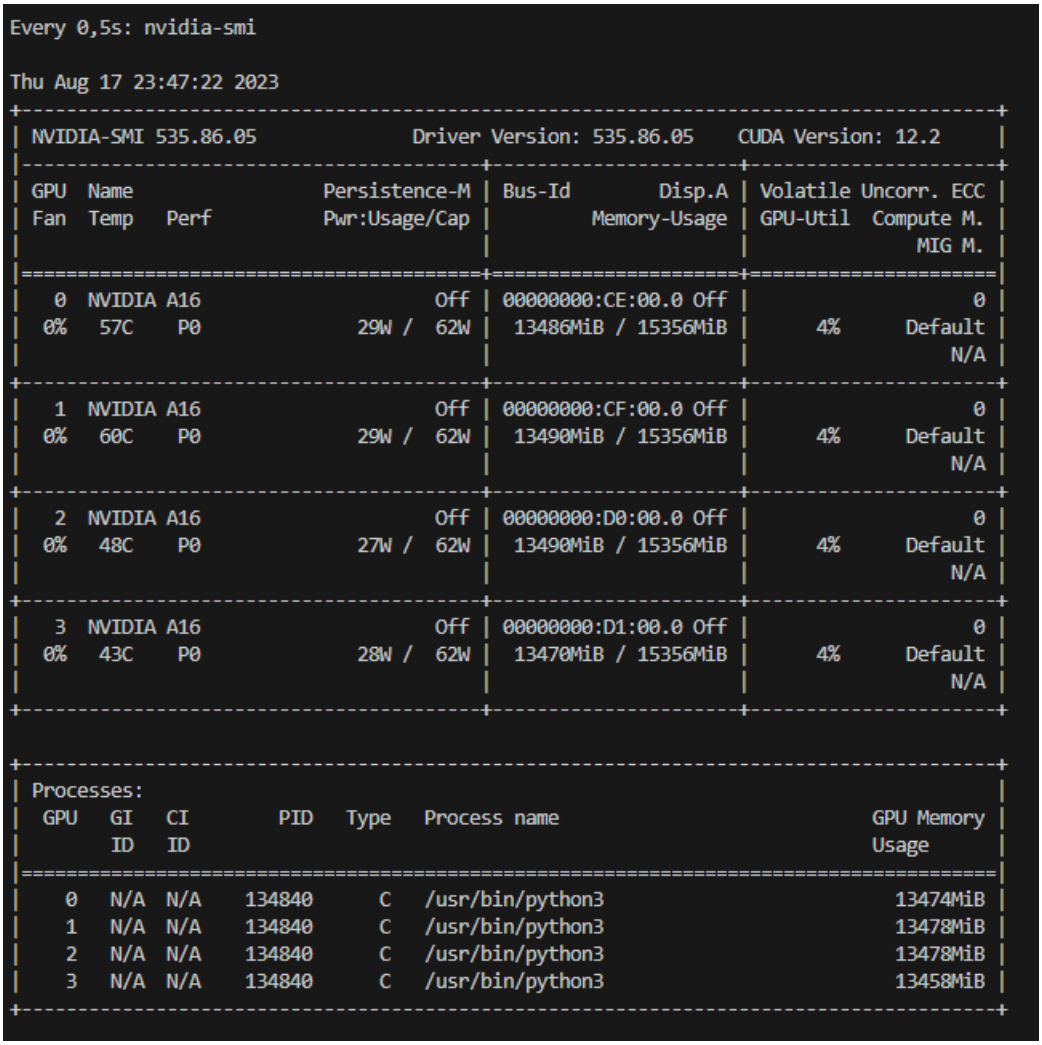

Figura 3.17: Monitoreo de uso de GPU al ejecutar un algoritmo.

3.2 CONCLUSIONES

La investigación acerca de hipervisores y contenedores llevada a cabo para explorar la implementación de estas tecnologías en laboratorios de datos avanzados ha proporcionado importantes conocimientos sobre las ventajas y retos. La implementación de contenedores en servidores de altas prestaciones presenta ligera eficiencia y un despliegue más rápido de las aplicaciones requeridas para los análisis de los datos, por otro lado los hipervisores pueden presentar problemas de seguridad y la posible sobrecarga de rendimiento.

La identificación de los requisitos de los usuarios del laboratorio ADA en este contexto de cloud computing ha aportado información importante para lograr que la infraestructura se adapte a las necesidades específicas de las tareas de análisis de datos. Este trabajo

subraya la importancia de diseñar una plataforma centrada en el usuario para optimizar la relación entre las tecnologías de cloud computing y el área de analítica de datos.

El diseño del entorno de computación en la nube se ha logrado con éxito mediante la implementación de una arquitectura basada en contenedores. Como resultado el laboratorio ahora cuenta con un adecuado entorno de computación en la nube que brinda las herramientas necesarias para las tareas de análisis de datos con los lenguajes Python y R y que además aprovecha el poder de GPU y tensorflow para el procesamiento. de grandes volúmenes de datos.

Los resultados obtenidos con la implementación del entorno de computación en la nube demuestran un impacto positivo. La integración de tecnologías como Docker, Jupyter y Hadoop han permitido ejecutar la primeras pruebas por parte de algunos de los miembros del laboratorio que han manifestado que esta implementación será de gran ayuda para el área de analítica de datos. Además la alineación de los requisitos de usuarios garantiza que el entorno satisface las necesidades específicas. En consecuencia el éxito de la implementación de este entorno abre las puertas a que se siga brindando de infraestructura de calidad a los laboratorios de la institución.

3.3 RECOMENDACIONES

El modelo SaaS ofrece una aplicación a través de un navegador, los usuarios de este servicio no gestionan sus datos ni sus recursos, debido a ello se recomienda que el administrador de la infraestructura realice monitoreo constante de los recursos del servidor y del estado de las aplicaciones.

El servidor del laboratorio ADA tiene recursos limitados por lo que se recomienda manejar políticas de uso para no sobrecargar las tareas en el servidor.

Para la implementación de este entorno se hace uso de todos los recursos de procesamiento como CPU y GPU por lo tanto es esencial verificar cuales son los requisitos físicos para el despliegue de software, además se debe revisar las especificaciones del hardware en sitios oficiales, el sistema operativo es importante ya que a veces las nuevas versiones no tienen compatibilidad con ciertos controladores de hardware.

JupyterHub es una plataforma que admite múltiples núcleos y lenguajes, en el mundo de la analítica de datos existen muchas más herramientas que Python y R por lo que se recomienda una reingeniería de requisitos con el fin de potenciar el laboratorio.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] P. Mell y T. Grance, «The NIST Definition of Cloud Computing,» National Institute of Standards y Technology (NIST), Special Publication 800-145, 2011.

[2] R. Bhoyar y N. Chopde, «Cloud computing: Service models, types, database and issues,» *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, n.º 3, 2013.

[3] Unknown. «Cloud Service Models - IaaS, PaaS, BaaS, SaaS.» [Imagen]. (), dirección: https://www.pngfind.com/mpng/ixJixbR_cloud-service-models-iaas-paas-baas-saas-hd/.

[4] M. O. Ahmad y R. Z. Khan, «The cloud computing: a systematic review,» *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 3, n.º 5, págs. 4060-4075, 2015.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg e I. Brandic, «Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,» *Future Generation Computer Systems*, vol. 25, n.º 6, págs. 599-616, 2009.

[6] *¿Qué es la virtualización?* https://aws.amazon.com/es/what-is/virtualization/, Sitio web de Amazon Web Services (AWS), Fecha de acceso: 25 de julio de 2023.

[7] *¿Qué es la virtualización?* https://www.ibm.com/es-es/topics/virtualization, Sitio web de IBM, Fecha de acceso: 25 de julio de 2023.

[8] *Hipervisores*, https://www.ibm.com/es-es/topics/hypervisors, Sitio web de IBM, Fecha de acceso: 25 de julio de 2023.

[9] Stackscale, *Hipervisores: Introducción y tipos*, Página web, Imagen extraída de: https://www.stackscale.com/es/blog/hipervisores/, jul. de 2023. dirección: https://www.stackscale.com/es/blog/hipervisores/.

[10] M. Eder, «Hypervisor-vs. container-based virtualization,» *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, vol. 1, 2016.

[11] Docker Documentation, *Docker Overview*, https://docs.docker.com/get-started/overview/, Fecha de acceso: 1 de agosto de 2023.

[12] G. Bhatia, A. Choudhary y V. Gupta, «The road to docker: a survey,» *International Journal of Advanced Research in Computer Science*, vol. 8, n.º 8, págs. 83-87, 2017.

[13] J. Nickoloff, *Docker in Action*, 1st. Shelter Island, New York: Manning Publications Co., 2016, pág. 28.

[14] Kubernetes, *Kubernetes Overview*, https://kubernetes.io/docs/concepts/overview/, Visitado el 25 de Julio 2023.

[15] H. S. Oluwatosin, «Client-server model,» *IOSR Journal of Computer Engineering*, vol. 16, n.º 1, págs. 67-71, 2014.

[16] J. Terra. «What is Client-Server Architecture? Everything You Should Know.» (feb. de 2023), dirección: https://www.simplilearn.com/what-is-client-server-architecture-article#terminology_basics.

[17] S. Ingalls. «What Is a Client-Server Model? A Guide to Client-Server Architecture.» (nov. de 2021), dirección: https://www.serverwatch.com/guides/client-server-model/.

[18] M. R. Dono, «Implementación de un laboratorio mediante contenedores,» *Universidad Complutense de Madrid*, 2021.

[19] A. M. Potdar, D. Narayan, S. Kengond y M. M. Mulla, «Performance evaluation of docker container and virtual machine,» *Procedia Computer Science*, vol. 171, págs. 1419-1428, 2020.

[20] M. Dıaz, M. Araya, C. Jauregui et al., «Docker-based implementation for an astronomical data analysis cloud service [M],» *XXVII Astronomical Data Analysis Software & Systems, ADASS*, págs. 1-4, 2018.

[21] S. B. Merriam, *Qualitative Research: A Guide to Design and Implementation*. San Francisco, CA: Jossey-Bass, 2009.

[22] W. R. Shadish, T. D. Cook y D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin, 2002.

[23] F. Morales, «Conozca 3 tipos de investigación: Descriptiva, Exploratoria y Explicativa,» *Recuperado el*, vol. 11, n.º 3, 2012.

[24] T. Brown, «Design Thinking,» *Harvard Business Review*, vol. 86, n.º 6, págs. 84-92, 2008.

[25] VMware India, *Server Virtualization with VMware vSphere*, https://www.vmware.com, Consulted on July 8, 2023.

[26] Microsoft, *Hyper-V Technology Overview*, https://learn.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview, Consultado el 22 de Julio de 2023, 2021.

[27] Linux-KVM, *Main Page*, https://www.linux-kvm.org/page/Main_Page, Consultado el 25 de julio de 2023, 2023.

[28] *OpenStack Documentation*, https://docs.openstack.org/2023.1/, Consultado el 30 de Julio de 2023, 2023.

[29] *Xen Project Documentation*, https://xenproject.org/help/documentation/, Consultado el 25 de Julio de 2023 de acceso, 2023.

[30] *Proxmox Virtual Environment Overview*, https://www.proxmox.com/en/proxmox-virtual-environment/overview, Consultado el 30 de Julio de 2023, 2023.

[31] *Podman Documentation*, https://docs.podman.io/en/latest/, Consultado el 27 de Julio de 2023, 2019.

[32] *Linux Containers*, https://linuxcontainers.org/, Consultado el 25 de Julio de 2023, 2023.

[33] *Red Hat OpenShift*, https://www.redhat.com/en/technologies/cloud-computing/openshift, Consultado el Día de Mes de Año de acceso, Año de acceso a la página.

[34] *Docker Engine - Swarm mode overview*, https://docs.docker.com/engine/swarm/, Consultado el 30 de Julio de 2023, 2023.

[35] *NVIDIA Container Toolkit*, https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/index.html, Acceso: Julio de 2023.

[36] JupyterHub Contributors, *JupyterHub Configurable HTTP Proxy*, GitHub repository, 2023. dirección: https://github.com/jupyterhub/configurable-http-proxy.

[37] Read the Docs. «Native Authenticator - Quickstart Guide,» Read the Docs. (), dirección: https://native-authenticator.readthedocs.io/en/latest/quickstart.html.

[38] Docker, *Get Docker Engine - Community for Ubuntu*, Docker Documentation, Julio 2023. dirección: https://docs.docker.com/engine/install/ubuntu/.

[39] *NVIDIA Container Toolkit Installation Guide*, https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/latest/install-guide.html, Acceso en línea: Julio 2023.

[40] TensorFlow, *Instalación de TensorFlow con Docker*, https://www.tensorflow.org/install/docker?hl=es-419, Acceso en línea: [Julio 2023].

[41] TensorFlow, *Perfil de TensorFlow en Docker Hub*, https://hub.docker.com/u/tensorflow, Acceso en línea: [Julio 2023].

[42] JupyterHub, *Configurable HTTP Proxy GitHub Repository*, https://github.com/jupyterhub/configurable-http-proxy, Acceso en línea: [Julio 2023].

[43] JupyterHub, *Native Authenticator GitHub Repository*, https://github.com/jupyterhub/nativeauthenticator, Acceso en línea: [Julio 2023].

[44] IRkernel contributors, *IRkernel Installation on Linux*, https://irkernel.github.io/installation/#linux-panel, Agosto de 2023.

5 ANEXOS

1. Manual de Usuario para el ingreso a Aplicaciones Virtualizadas VDI
2. Instructivo para la conexión VPN
3. Especificaciones técnicas del equipo
4. Archivo de configuración para JupyterHub
5. Archivo Dockerfile
6. Manual técnico para despliegue de JupyterHub
7. Manual de registro en JupyterHub