ESCUELA SUPERIOR POLITÉCNICA AGROPECUARIA DE MANABÍ MANUEL FÉLIX LÓPEZ

CARRERA DE INFORMÁTICA

INFORME DE TRABAJO DE INTEGRACIÓN CURRICULAR PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN INFORMÁTICA

MECANISMO: SISTEMATIZACIÓN DE EXPERIENCIAS PRÁCTICAS DE INVESTIGACIÓN Y/O INTERVENCIÓN

TEMA:

ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES

AUTORES:

LIMBERT E. SANTANDER ALCÍVAR

GEMA M. ZAMBRANO PARRALES

TUTOR:

ING. ALFONSO T. LOOR VERA, MGTR.

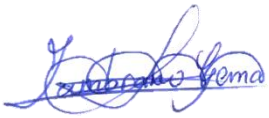
CALCETA, MARZO DE 2022

DECLARACIÓN DE AUTORÍA

Yo **LIMBERT ESNEIDER SANTANDER ALCÍVAR** con cédula de ciudadanía **1314555267**; y **GEMA MARÍA ZAMBRANO PARRALES**, con cédula de ciudadanía **1314189075**, declaramos bajo juramento que el Trabajo de Integración Curricular titulado: **ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES** es de nuestra 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.

A través de la presente declaración, concedo a favor de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos, conservando a mi favor todos los derechos patrimoniales de autor sobre la obra, en conformidad con el Artículo 114 del Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación.

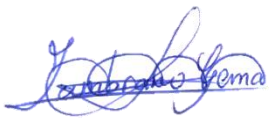
LIMBERT E. SANTANDER ALCÍVAR
CC: 1314555267



GEMA M. ZAMBRANO PARRALES
CC: 1314189075

AUTORIZACIÓN DE PUBLICACIÓN

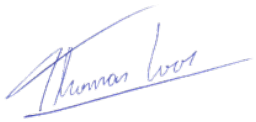
LIMBERT ESNEIDER SANTANDER ALCÍVAR, con cédula de ciudadanía **1314555267** y **GEMA MARÍA ZAMBRANO PARRALES**, con cédula de ciudadanía **1314189075**; autorizamos a la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, la publicación en la biblioteca de la institución del Trabajo de Integración Curricular titulado: **ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES**, cuyo contenido, ideas y criterios son de nuestra exclusiva responsabilidad y total autoría.



LIMBERT E. SANTANDER ALCÍVAR	**GEMA M. ZAMBRANO PARRALES**
CC: 1314555267	**CC: 1314189075**

CERTIFICACIÓN DEL TUTOR

ALFONSO TOMÁS LOOR VERA, certifica haber tutelado el Trabajo de Integración Curricular titulado: **ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES**, que ha sido desarrollado por **LIMBERT ESNEIDER SANTANDER ALCÍVAR** y **GEMA MARÍA ZAMBRANO PARRALES** previo a la obtención del título de Ingeniero en Informática, de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López**.**



Firmado digitalmente
porALFONSO
TOMAS LOOR VERA

_____

ING. ALFONSO T. LOOR VERA, MGTR.

CC: 1311655938

TUTOR

APROBACIÓN DEL TRIBUNAL

Los suscritos integrantes del Tribunal correspondiente, declaramos que hemos **APROBADO** el Trabajo de Integración Curricular titulado: **ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES**, que ha sido desarrollado por **LIMBERT ESNEIDER SANTANDER ALCÍVAR** y **GEMA MARÍA ZAMBRANO PARRALES**, previo a la obtención del título de Ingeniero en Informática, de acuerdo al **REGLAMENTO DE LA UNIDAD DE INTEGRACIÓN CURRICULAR DE CARRERAS DE GRADO** de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

Firmado digitalmente por DANIEL AGUSTIN MERA MARTINEZ

ING. DANIEL A. MERA MARTINEZ, MGTR.

CC:1301932156

PRESIDENTE DEL TRIBUNAL

Firmado digitalmente porRICARDO ANTONIO VELEZ VALAREZO

ING. RICARDO A. VÉLEZ VALAREZO, MGTR.

CC:1306391614

MIEMBRO DEL TRIBUNAL

Digitally signed by FERNANDO RODRIGO MOREIRA MOREIRA

ING. FERNANDO R. MOREIRA MOREIRA, MGTR.

CC:1311726689

MIEMBRO DEL TRIBUNAL

AGRADECIMIENTO

A la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López que nos dio la oportunidad de crecer como ser humano a través de una educación superior de calidad y en la cual hemos forjado nuestros conocimientos profesionales día a día;

A Dios por todas las bendiciones recibidas diariamente, y darnos la oportunidad de crecer en nuestras vidas personales y profesionales,

A nuestro tutor Ing. Alfonso Tomás Loor Vera, por habernos guiado con sus conocimientos, además de brindarnos su tiempo y apoyo incondicional en nuestro trabajo de integración curricular,

Al Ing. Javier Hernán López Zambrano, por darnos la oportunidad de participar en sus proyectos y desarrollar en la UDIV de Infraestructura nuestro trabajo de integración curricular,

A los docentes y amigos que nos brindaron su apoyo durante la formación profesional y personal, que gracias a su contribución y motivación se logró cumplir con las metas propuestas.

LOS AUTORES

DEDICATORIA

Dedico este trabajo de integración curricular a mis padres por haber sido un pilar fundamental en mi vida y apoyarme en todo momento de manera incondicional durante esta etapa.

A mis hermanos por su apoyo en todos los momentos de mi vida.

A las personas de las que siempre tendré un apoyo incondicional, que han reconocido mi esfuerzo, me han dado palabras de aliento en todo momento.

LIMBERT E. SANTANDER ALCIVAR

DEDICATORIA

A mis padres por ser el pilar fundamental, que con el sacrificio realizado me ayudan a conseguir mis objetivos personales y profesionales además del afecto y el cariño que me dan cada día.

A mis hermanos por ser inspiración y estar conmigo en todo momento, que con sus alegrías me motivan a continuar.

A las personas que me acompañaron e impulsaron a continuar en las adversidades que se me presentaron en este caminar, y juntos celebramos la alegría de mis logros alcanzados.

GEMA M. ZAMBRANO PARRALES

CONTENIDO GENERAL

CONTENIDO DE TABLAS

CONTENIDO DE FIGURAS

RESUMEN

El siguiente trabajo de integración curricular tuvo como objetivo desarrollar un estudio exploratorio de la tecnología de Docker como contenedor de aplicaciones y cuantificar el coste computacional. Para esto se aplicó la metodología de The Project Approach que cuenta con tres fases que son: comenzar, trabajo práctico, culminar e informar; en la primera fase se describió toda la información inicial que se obtuvo a través de las técnicas de investigación como fue la entrevista al encargado de la Unidades de Docencia, Investigación y Vinculación (UDIV) de Infraestructura y la encuesta a los estudiantes de la Carrera de Computación. Además, se aplicó la Revisión Sistemática de la Literatura, que permitió extraer los fundamentos teóricos y prácticos de la misma; en la siguiente fase se elaboró escenarios virtualizados con Docker a través de la fase del trabajo practico donde se experimentaron los diferentes escenarios virtualizados; por último se generó una guía de aplicación que servirá para realizar nuevos proyectos en la UDIV de Infraestructura, estos proyectos serán puestos en funcionamiento con los estudiantes de la Carrera de Computación.

PALABRAS CLAVE

Virtualización de aplicaciones, Docker, contenedores.

ABSTRACT

The following curricular integration work aimed to develop an exploratory study of Docker technology as a container for applications and to quantify the computational cost. For this, the methodology of The Project Approach was applied, which has three phases that are: start, practical work, culminate and report; in the first phase, all the initial information obtained through research techniques is described, such as the interview with the person in charge of the Unidades de Docencia, Investigación y Vinculación (UDIV) and the survey of the students of the Computing major. In addition, the Systematic Literature Review was applied, which will extract the theoretical and practical foundations of it; in the next phase, virtualized scenarios were developed with Docker through the practical work phase where the different virtualized scenarios were experienced; finally, an application guide was generated that will serve to carry out new projects in the Infrastructure UDIV, these projects will be put into operation with the students of the Computer Science Degree.

KEYWORDS

Application Virtualization, Docker, containers.

CAPÍTULO I. ANTECEDENTES

1.1 DESCRIPCIÓN DE LA INSTITUCIÓN

La ESPAM MFL, es una Institución de Educación Superior que fue creada el 29 de abril de 1999, a través de la Constitución Política del Estado, Ley de Educación Superior, su Estatuto Orgánico y Reglamentos, con el objetivo de preparar a la ciudadanía ecuatoriana y convertirla en profesionales, según lo establecen los recursos naturales de su entorno, esta tiene como finalidad brindar servicios de acceso y transición de información de orden científico, investigativo, cultural y de vinculación, aplicado con las tecnologías de última generación que satisfaga las necesidades de la Comunidad Politécnica y del país; esta sede universitaria ofrece carreras en Ingeniería y Licenciatura, también programas de maestrías relacionadas a las carreras que tiene y con esto aportan al desarrollo profesional (ESPAM MFL, 2021a; Universidades de Ecuador, 2021)

Dentro de la ESPAM MFL, se encuentra la Carrera de Computación que tiene como misión la “formación de profesionales íntegros que conjuguen ciencia, tecnología y valores en su accionar, comprometidos con la comunidad en el manejo adecuado de programas y herramientas computacionales de última generación” y como visión los estudiantes deben de “ser referentes en la formación de profesionales de prestigio en el desarrollo de aplicaciones informática y soluciones de hardware” (ESPAM MFL, 2021b). Para beneficio de los estudiantes, la Carrera cuenta con diferentes Unidades de Docencia, Investigación y Vinculación (UDIV); como son: Desarrollo Computacional, Ciencias Básicas e Infraestructura, cada una de estas Unidades permiten a los estudiantes realizar prácticas en las áreas del conocimiento que ofrece la Carrera (Carrera de Informática, 2019).

La UDIV de Infraestructura, tiene como misión mejorar la calidad del proceso de enseñanza - aprendizaje en las asignaturas integradoras desde el ámbito de infraestructuras y como visión apoyar las funciones de docencia, investigación y vinculación de la Carrera de Computación; con esto se encarga de fortalecer el nivel de aprendizaje con una formación sólida, teórica, metodológica y práctica

a los estudiantes, en el análisis de problemas relacionados con los procesos del computador, auditoria y redes de datos (UDIV de Infraestructura, 2020). En su mejoramiento continuo y tecnológico busca el estudio y análisis de nuevas tecnologías existentes en el mercado y de esta manera fomentar el estudio y la generación de nuevos conocimientos en los estudiantes de la Carrera, y así, se puedan afianzar en un mundo tan competitivo.

1.2 DESCRIPCIÓN DE LA INTERVENCIÓN

La virtualización hace referencia a las tecnologías diseñadas para proveer abstracción de recursos fundamentales y esto hace que las herramientas que involucran la virtualización emulen y aíslan recursos informáticos dentro de un host, de esta manera clasificarlas en función del nivel de abstracción que proporciona (Ghatrehsamani et al., 2020; Yu et al., 2018). Estas tecnologías han desempeñado un papel importante en la industria de TI (Tecnologías de la Información) así el fin de mejorar el rendimiento del hardware, esto incluye recursos como la CPU, GPU, RAM y otros (Abuabdo & Al-Sharif, 2019)

El uso de estas herramientas de virtualización permite a los usuarios gozar de costos mucho más bajos que en los laboratorios reales; también, forman parte del cloud computing en el soporte de servicios digitales de las organizaciones con la optimización de costos y recursos, ya que permite la creación de los mismos a través de los servidores, facilitando el uso (Perdigón & Ramírez, 2020; Zaldivar, 2019).

Las herramientas de virtualización han sido muy importante a través de las máquinas virtuales, que son ordenadores de software que proporcionan la misma funcionalidad que las máquinas físicas; estas se ejecutan de manera independiente en el computador ya que se encuentran separadas lógicamente; además, si existe alguna falla entre las máquinas instaladas estas no afectan a la otra; su funcionamiento es a través de un sistema operativo completo y aplicaciones (Wei et al., 2020; VMware, 2021). Sin embargo, en los últimos años, han aparecido nuevas tecnologías de virtualización en el mercado como Docker, que es una herramienta de código abierto, la cual provee una capa de abstracción para la implementación de servicios, esto lo convierte en una alternativa de virtualización liviana con base a contenedores de aplicaciones que ejecuten procesos de manera aislada; el trabajo con esta tecnología se centra en las necesidades empresariales, y ayuda en el tiempo de ejecución de las misma, además, es usado activamente por millones de desarrolladores en todo el mundo (Docker, 2021; De Benedictis & Lioy, 2019; Shu et al., 2017)

La importancia de Docker radica en la construcción de imágenes que usan los desarrolladores para las aplicaciones, y con esto da acceso a la continuidad de DevOps a través de configuraciones simplificadas al momento de la implementación de servicios y productos; el término DevOps permite que las organizaciones de software alcance sus objetivos, siendo un colaborador empresarial global (Docker, 2021b; Erich et al., 2017; Nish Anil, 2020). Además de integrar herramientas de desarrollo, empaquetado de aplicaciones, gestión de proyectos, aislamiento de aplicaciones, entre otras.

El desarrollo de este trabajo de integración curricular se basa en la continuidad de la virtualización a través de las nuevas tecnologías, por esta razón la UDIV de Infraestructura busca el aprendizaje continuo de las actuales tecnologías. En la actualidad, la virtualización se está enfocando en contenedores de aplicaciones; para esto han visto a Docker como una herramienta prometedora para ahorrar costos de implementación de sistemas, servicios web, entre otros. Por ello, se realiza un análisis de Docker como contenedor de aplicaciones, con sus características esenciales, prácticas de laboratorio, y evaluación de un coste computacional de los escenarios virtualizados; que tiene como resultado la entrega de una guía de aplicación para la UDIV de Infraestructura, que podrá ser usada por los estudiantes de la Carrera de Computación para la implementación de nuevos proyectos.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Desarrollar un estudio exploratorio de la tecnología Docker como contenedor de aplicaciones para elaborar una guía de aplicación en la UDIV de Infraestructura de la Carrera de Computación.

1.3.2 OBJETIVOS ESPECÍFICOS

- Levantar la línea base de la investigación de Docker.
- Diseñar escenarios de virtualización de Docker como contenedor de aplicaciones.
- Cuantificar el coste computacional de los escenarios virtualizados.
- Elaborar la guía de aplicación de Docker como contenedor de aplicaciones.

CAPÍTULO II. DESARROLLO METODOLÓGICO DE LA INTERVENCIÓN

El trabajo de integración curricular tuvo como objetivo desarrollar un estudio exploratorio de la tecnología de Docker como contenedor de aplicaciones para elaborar una guía de aplicación en la UDIV de Infraestructura. El estudio está basado en la metodología de The Project Approach que da una visión inicial a la investigación para la adquisición de nuevos conocimientos y de esta manera ponerlos en práctica con base a la construcción de modelos, Helm & Katz (2016) en los ejemplos de aplicación de la metodología detallan tres fases que son: Comenzar, Desarrollar y Concluir; así mismo, se contó con una Revisión Sistemática de la Literatura (RSL) para dar un cumplimiento efectivo a esta investigación. A continuación, se detallan como se logró alcanzar los objetivos específicos que se establecieron para la elaboración del trabajo de integración curricular con su respectiva descripción.

2.1 LEVANTAR LA LÍNEA BASE DE LA INVESTIGACIÓN DE DOCKER.

Para el inicio del trabajo de integración curricular se aplicó la etapa inicial de la metodología The Project Approach denominada comenzar, Illinois Early Learning (2016), menciona que en esta etapa se detallan las ideas principales de la investigación describiéndolas a través de los métodos y técnicas, estas tienen un conjunto de preguntas de interés hacia las personas que adquirirán los nuevos conocimientos. Para el cuestionamiento de las personas se optó por las técnicas de investigación de entrevista y encuesta.

Según Grande Estefan & Abascal Fernández (2017), describen tres tipos de clases de entrevista que son: entrevista estructurada, esta se basa en que el entrevistador realiza las preguntas que figuran en un guión exclusivamente; entrevista semiestructurada, manifiesta que el entrevistador tenga la libertad de introducir nuevas preguntas según el desarrollo de la entrevista con el entrevistado; y por último la entrevista en profundidad; que determina un guión general que no ciñe preguntas concretas. El tipo de entrevista que se utilizó fue

una entrevista semiestructurada y se aplicó al responsable encargado de la UDIV de Infraestructura.

Mientras que la encuesta es una de las técnicas de investigación social de más extendido uso en las diferentes áreas del conocimiento y ha trascendido al ámbito estricto de la investigación científica, esta se ha convertido en una actividad cotidiana en la mayoría de investigaciones con el fin generar informacion y debate social en diversos ámbitos (López-Roldán & Fachelli, 2016). Esta encuesta se aplicó a los estudiantes de los cursos superiores de la Carrera de Computación de la ESPAM MFL, mediante un cuestonario online realizado en Google Forms con la finalidad de conocer el nivel de conocimiento que se encuentran los estudiantes en los temas de virtualización, especificamente con Docker, que es el software de estudio en este proyecto.

Además, la RSL formó parte del levantamiento de información relacionada con docker, misma que sirvió para dar respuesta a la problemática planteada. La RSL tiene como objetivo identificar, evaluar y combinar la evidencia de estudios primarios con el uso de métodos rigurosos, estas revisiones se han vuelto una importante metodología en la educación y es muy utilizada en la actualidad (Carrizo & Moller, 2018).

Las fases de la RSL según Kitchenham et al. (2009) son las siguientes:

2.1.1 PREGUNTAS DE INVESTIGACIÓN

Lo primero que se llevó a cabo fue la identificación del tema y de esta manera la formulación de preguntas que dieron respuesta a la investigación, dando como punto de partida a la búsqueda de información referente a Docker y su manera de usarlo, ya que en la actualidad se está utilizando cada vez más por ser de código de abierto.

2.1.2 PROCESO DE BÚSQUEDA

Para el proceso de la búsqueda de información bibliográfica se basó en cadenas de texto relacionadas al tema de investigación, misma que fueron aplicadas a las bases de datos bibliográficas como: Science Direct, ACM, IEEE Xplore; con esto

se obtuvo una variedad de fuentes de información de revistas de divulgación o de investigación científica, que permitieron ampliar el campo de conocimiento.

2.1.3 CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN

López (2019), señala que esta etapa se inicia eliminando los artículos duplicados y aquellos que no cumplen con los criterios de inclusión y exclusión. Para esto se establecieron los criterios de inclusión o exclusión (ver tabla 3) que fueron aplicados a los títulos y resúmenes; posteriormente se realizó una lectura rápida de los documentos encontrados para descartar aquellos artículos que no eran de interés para el tema de investigación.

2.1.4 EVALUACIÓN DE LA CALIDAD

Se refiere a la valoración de validez interna y posibles sesgos (Manterola et al., 2013a). El objetivo se basa en evaluar y analizar mediante métricas de calidad si los resultados obtenidos fueron los correctos para el tema investigado, o si se dejó a un lado algún tema de mayor relevancia con base a las interrogantes de la investigación.

2.1.5 EXTRACCIÓN Y ANÁLISIS DE DATOS

Esta fase se enfoca a la extracción de datos de los artículos seleccionados, aquí el investigador extrae los datos encontrados y clasifica los artículos por título, autor y otros campos que sean pertinentes. De esta manera es más factible manipular la información obtenida ya que en este tipo de revisiones la cantidad de información suele ser bastante grande (López, 2019).

El proceso se realizó con lecturas detalladas a los artículos seleccionados, resaltando la información pertinente que servirá para dar cumplimiento al objetivo planteado en esta investigación. Los datos encontrados fueron tabulados en archivos de Excel, lo que permitió realizar un correcto análisis de la información para dar respuestas a las preguntas de investigación. Este proceso detallado, permitió interpretar los datos desde el punto de vista cualitativo y cuantitativo (Manterola et al., 2013b).

2.2 DISEÑAR ESCENARIOS DE VIRTUALIZACIÓN DE DOCKER COMO CONTENEDOR DE APLICACIONES.

Para la elaboración de los escenarios de virtualización de Docker, se aplicó la segunda etapa de la metodología The Project Approach denominada Desarrollar; que se encuentra directamente relacionada con la parte investigativa poniendo en práctica los conocimientos adquiridos en la revisión sistemática. En esta etapa se realizó la creación de los escenarios de Docker como contenedor de aplicaciones con base a la información encontrada en la RSL.

2.3 CUANTIFICAR EL COSTE COMPUTACIONAL DE LOS ESCENARIOS VIRTUALIZADOS.

El desarrollo del objetivo se basó en la segunda etapa de la metodología The Project Approach, para ello, se aplicaron una serie de comandos de Docker que brindan información detallada del consumo de recursos de los contenedores, facilitando revisar el coste computacional de una manera clara y visible para el usuario.

Con los resultados obtenidos se generaron gráficos que representan una mejor forma de visualizar la información obtenida en cada uno de los contenedores, y así visualizarlos claramente en el trabajo de integración curricular.

2.4 ELABORAR LA GUÍA DE APLICACIÓN DE DOCKER COMO CONTENEDOR DE APLICACIONES.

Este último objetivo, comprende la etapa final de la metodología The Project Approach denominada Concluir, que como toda investigación busca publicar los resultado obtenidos, aquí se elaboró una guía de aplicación de estudio de Docker como contenedor de aplicaciones, que contiene los datos fundamentales de la entrevista realizada al Coordinador de la UDIV de Infraestructura y la encuesta ejecutada a los estudiante de los niveles superiores de la Carrera de Computación; también, se consideró el material investigado de Docker con base a la RSL y los experimentos de los escenarios virtualizados.

Esta guía comprende, la instalación de Docker como contenedor de aplicaciones para Linux y Windows, lista de comandos principales para la creación de contenedores, ejemplos prácticos de los escenarios virtualizados, generalidades de Docker y un informe ejecutivo del coste computacional.

CAPÍTULO III. DESCRIPCIÓN DE LA EXPERIENCIA

3.1 LEVANTAR LA LÍNEA BASE DE LA INVESTIGACIÓN DE DOCKER

3.1.1 ENTREVISTA AL RESPONSABLE ENCARGADO DE LA UDIV DE INFRAESTRUCTURA

Para esta actividad, se realizó una entrevista mediante sesión virtual en la plataforma de Google Meet, que fue dirigida al Ing. Javier López, responsable encargado de la UDIV de Infraestructura, las preguntas de la entrevista fueron desarrolladas por los autores conjuntamente con el tutor (Anexo 1A); esta tuvo como finalidad conocer algunos lineamientos de la UDIV de Infraestructura y el aporte del Trabajo de Integración Curricular.

A continuación, se presenta una tabla resumen con datos informativos y el análisis de los datos más relevantes de la entrevista.

Tabla 1: Datos obtenidos de la entrevista.

DATOS PERSONALES	
NOMBRES Y APELLIDOS	Ing. Javier López Zambrano, Mgtr.
CARGO	Encargado de la UDIV de Infraestructura.
OBJETIVO DE LA ENTREVISTA	Obtener información para el desarrollo del trabajo de la Unidad de Integración Curricular.
ANÁLISIS	
Con esta entrevista, se conoció acerca de la UDIV de Infraestructura y se establecieron los requisitos principales del trabajo de integración curricular. Además, se pudo identificar la necesidad de elaborar una guía de estudio con los resultados obtenidos acerca de Docker que servirá de soporte para los estudiantes de la Carrera. Asimismo, se concluyó presentar los avances progresivos de la guía al Coordinador de la UDIV de Infraestructura.	

Fuente. Los autores

3.1.2 ENCUESTA A LOS ESTUDIANTES DE LA CARRERA DE COMPUTACIÓN

La encuesta se desarrolló para los estudiantes de la Carrera de Computación de la ESPAM MFL que cursan desde quinto a décimo nivel, ya que estos tienen los conocimientos más consolidados con respecto a las herramientas de virtualización. Con esto, se buscaba conocer el nivel de conocimiento que tienen los estudiantes en los temas de virtualización, especificamente con Docker.

Para la ejecución de esta actividad los autores se comunicaron con el Director de la Carrera de Computación, para socializar la encuesta con los estudiantes de quinto a décimo nivel; cabe mencionar, que debido a la emergencia sanitaria que vive el país por la pandemia de Covid-19, esta se realizó de manera online a través de Google Forms y no de manera presencial. La población total estimada para el envió de la encuesta fue de 74 estudiantes, de la cual se obtuvo 34 respuestas (Anexo 3), equivalente al 46% de la población.

La encuesta comprendió 4 secciones dependientes, con la finalidad de llegar al tema de estudio, se empezó con los conocimientos previos de la virtualización, seguido de su uso, luego preguntas relacionadas a Docker y por útimo si les gustaría que se enseñe el uso de esta herramienta en la Carrera de Computación.

La sección 1 obtuvo 34 respuestas y estuvo conformada por las siguentes preguntas:

- ¿Ha usado herramientas de virtualización?

En la figura 1 se observa el porcentaje de estudiantes que han usado herramientas de virtualización, de las 34 respuestas 26 fueron para la opción Si que equivale al 76.5% y 8 para la opción No con el 23.5%.

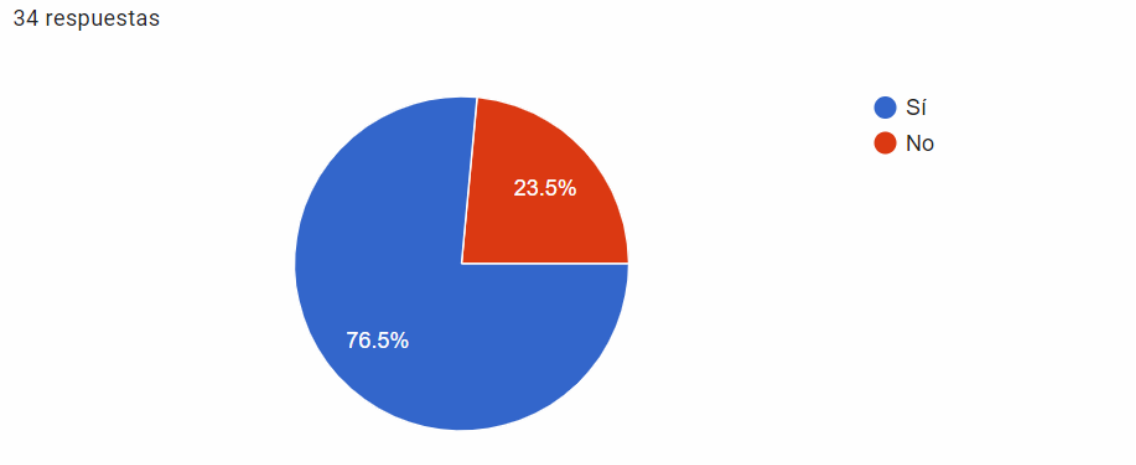


Figura 1: Porcentaje de estudiantes que han usado herramientas de virtualización.

Fuente. *Los autores*

- ¿Conoce usted la diferencia entre máquinas virtuales y contenedores de aplicaciones?

En la figura 2 se observa el porcentaje de estudiantes que conocen la diferencia entre máquinas virtuales y contenedores de aplicaciones, de las 34 respuestas 17 fueron para la opción Si que equivale al 50% y de la misma manera para la opción No.

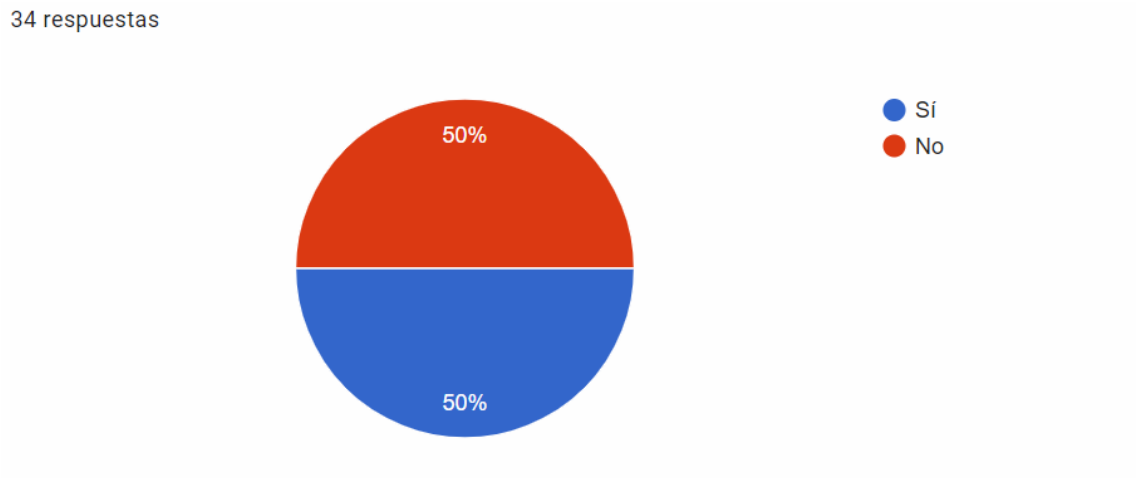


Figura 2: Porcentaje de estudiantes que conocen las diferencias entre máquinas virtuales y contenedor de aplicaciones.

Fuente. *Los autores*

La pregunta dependiente de esta sección fue la primera que contaba con las opciones de Si y No, si la respuesta era afirmativa pasaba a la sección 2, por lo contrario, esta se iba directamente a la cuarta sección en la cual se finalizaba la encuesta.

La sección 2 obtuvo 26 respuestas y estuvo conformada por las siguentes preguntas:

- ¿Para qué ha usado herramientas de virtualización?

En la figura 3 se observa el porcentaje de estudiantes que han usado herramientas de virtualización para alguna tarea específica, de las 26 respuestas 18 fueron para la opción Montar un Sistema Operativo que equivale al 69.2%, 6 fueron para la opción de servidor web con el 23.1%, 6 respuestas para la opción servidor de base de datos con el 23.1% y 7 para la opción virtualización de redes con el 26.9%.

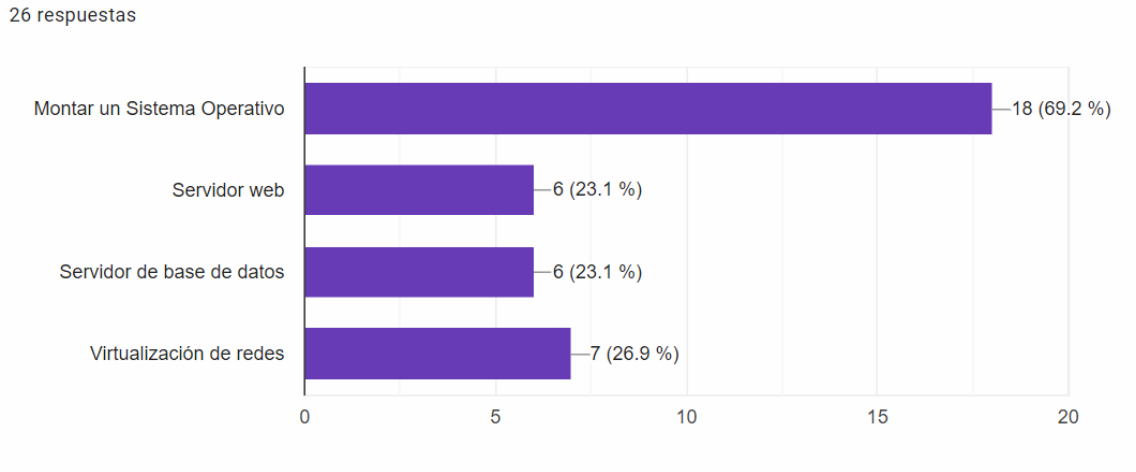


Figura 3: Porcentaje de estudiantes que han usado herramientas de virtualización para tareas específicas.

Fuente. *Los autores*

- ¿Conoce usted la tecnología de virtualización de aplicaciones llamada Docker?

En la figura 4 se observa el porcentaje de estudiantes que conocen Docker como tecnología de virtualización, de las 26 respuestas 21 fueron para la opción No que equivale al 80.8 % y 5 respuestas para la opción Si con el 19.2%.

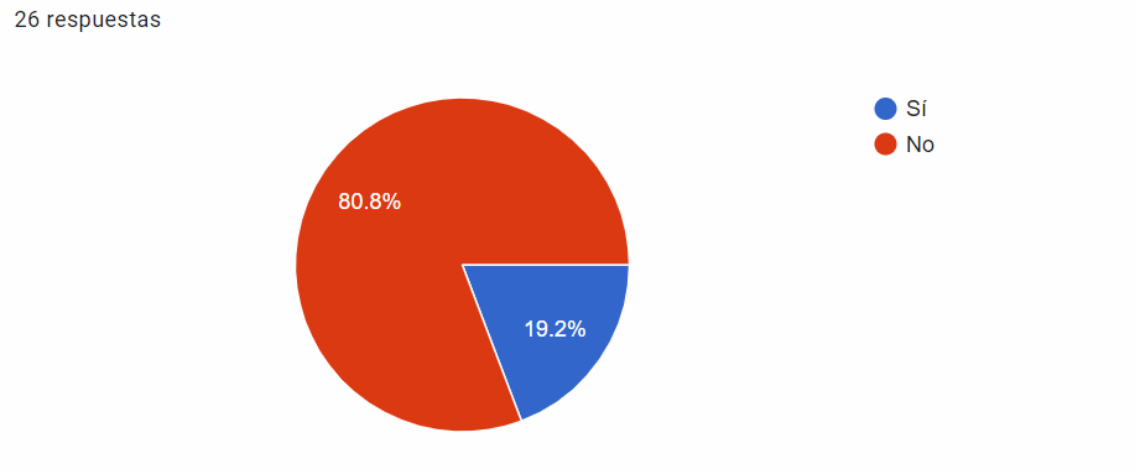


Figura 4: Porcentaje de estudiantes que conocen la tecnología de virtualización Docker.

Fuente. *Los autores*

La pregunta dependiente de esta sección fue la segunda que contaba con las opciones de Si y No, si la respuesta era afirmativa pasaba a la sección 3, por lo contrario, esta se iba directamente a la cuarta sección en la cual se finalizaba la encuesta.

La sección 3 obtuvo 5 respuestas y estuvo conformada por las siguentes preguntas:

- ¿Cómo conoció Docker?

En la figura 5 se observa el porcentaje de los lugares donde los estudiantes han conocido a Docker como tecnología de virtualización, de las 5 respuestas 2 fueron para la opción Universidad que equivale al 40%, 0 fueron para la opción congresos y trabajos con el 0%, 1 respuesta para la opción redes sociales, google e internet con el 20%.

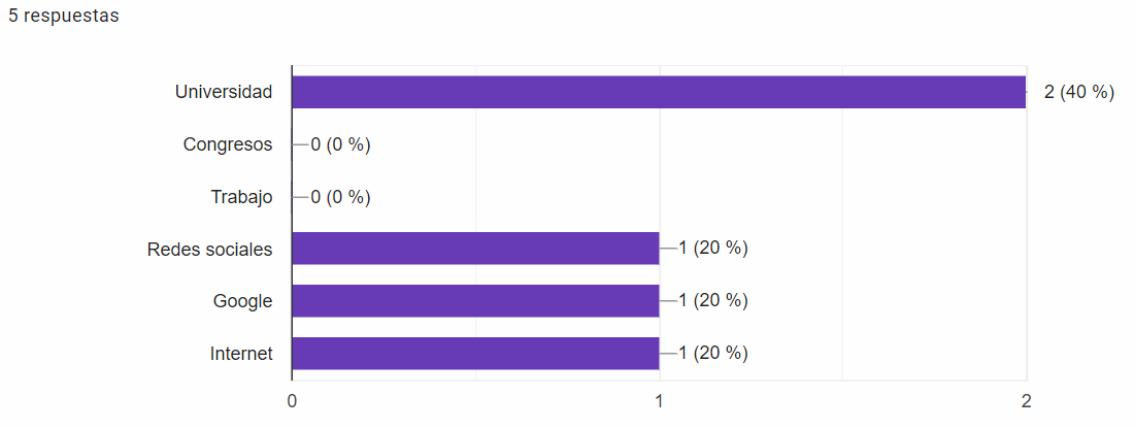


Figura 5: Porcentaje de lugares donde los estudiantes conocieron Docker.

Fuente. *Los autores*

- ¿Ha utilizado Docker?

En la figura 6 se observa el porcentaje de estudiantes que han utilizado Docker, de las 5 respuestas 3 fueron para la opción Si que equivale al 60% y 2 respuestas para la opción No con el 40%.

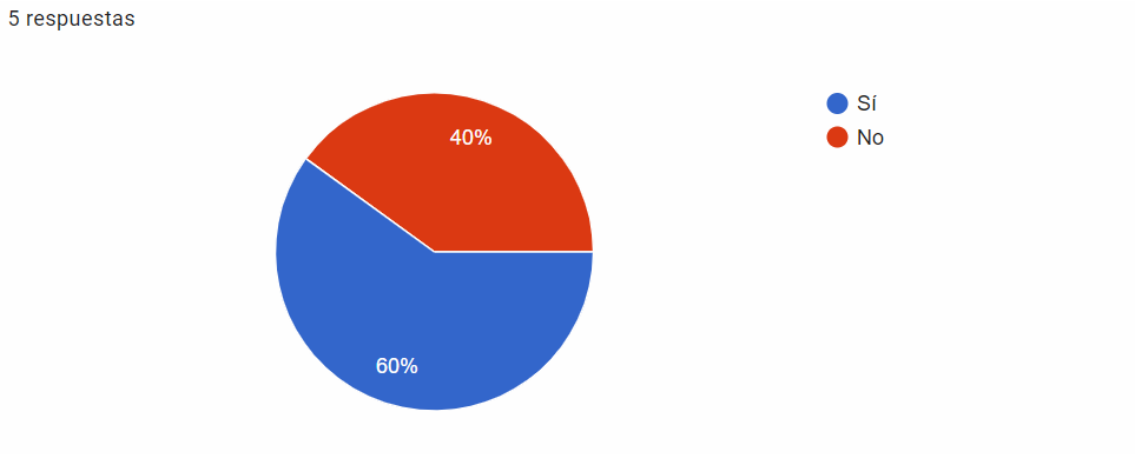


Figura 6*:* Porcentaje de estudiantes que han utilizado Docker.

Fuente. *Los autores*

- ¿Qué nivel de conocimiento tiene acerca de Docker?

En la figura 7 se observa el porcentaje del nivel de conocimiento de los estudiantes en Docker, las 5 respuestas fueron para la opción Bajo equivalente al 100%.

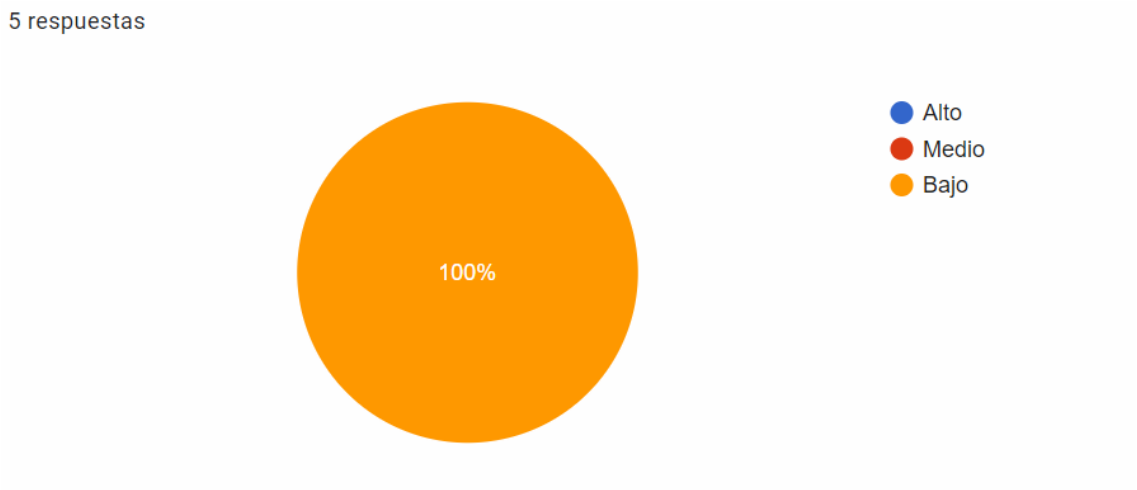


Figura 7: Porcentaje del nivel de conocimiento de Docker en los estudiantes.

Fuente. *Los autores*

- ¿Conoce la importancia de Docker para el tema de virtualización?

En la figura 8 se observa el porcentaje de estudiantes que conocen la importancia de Docker como tecnología de virtualización, de las 5 respuestas 3 fueron para la opción Si que equivale al 60% y 2 respuestas para la opción No con el 40%.

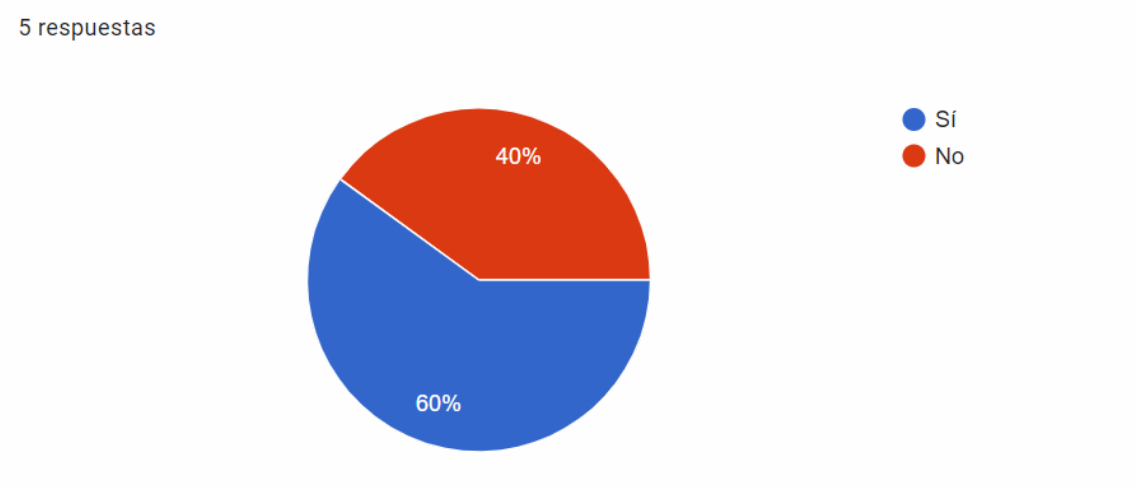


Figura 8*:* Porcentaje de estudiantes que conocen la importancia de Docker como tecnología de virtualización.

Fuente. *Los autores*

- ¿Conoce algún producto de contenedor de aplicaciones similar a Docker?

En la figura 9 se observa el porcentaje de estudiantes que conocen otro producto similar a Docker como contenedor de aplicaciones, de las 5 respuestas 3 fueron para la opción No que equivale al 60% y 2 respuestas para la opción Si con el 40%.

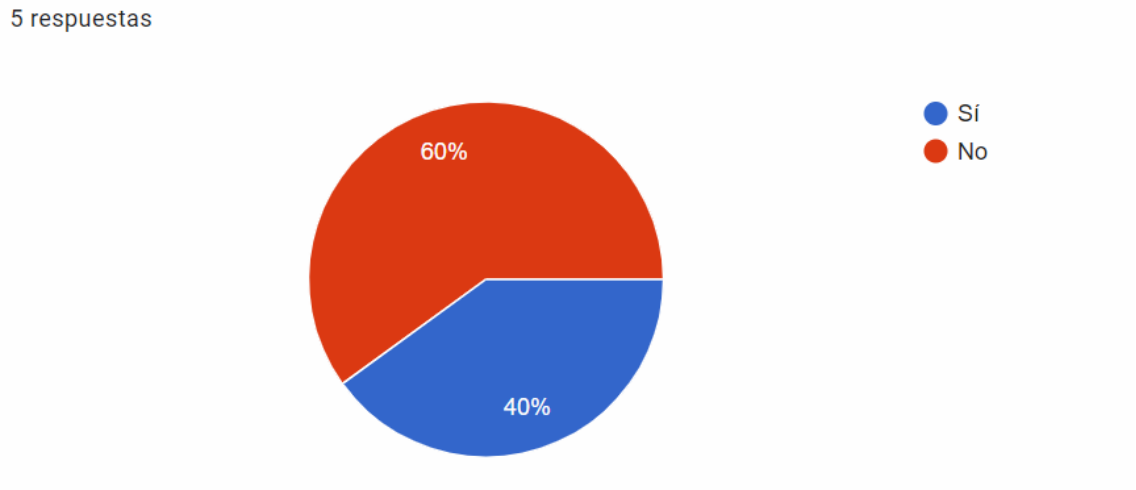


Figura 9: Porcentaje de estudiantes que conocen otro producto de contenedor de aplicaciones.*:*

Fuente. *Los autores*

En esta sección no había preguntas dependientes ya que aquí se encontraban las preguntas del nivel de conocimiento de la herramienta Docker por parte de los estudiantes, una vez terminada esta se iba directamente a la cuarta sección en la cual se finalizaba la encuesta.

La sección 4 obtuvo 34 respuestas, dando por finalizaba la encuesta y estuvo conformada por la siguente pregunta:

- ¿Le gustaría que la Carrera agregara Docker entre sus contenidos de aprendizaje?

En la figura 10 se observa el porcentaje de estudiantes que desean que Docker se agregue como contenido de aprendizaje en la Carrera de Computación, a la que 32 estudiantes respondieron si con un equivalente del 94.1% y 2 para la opción no con el 5.9%.

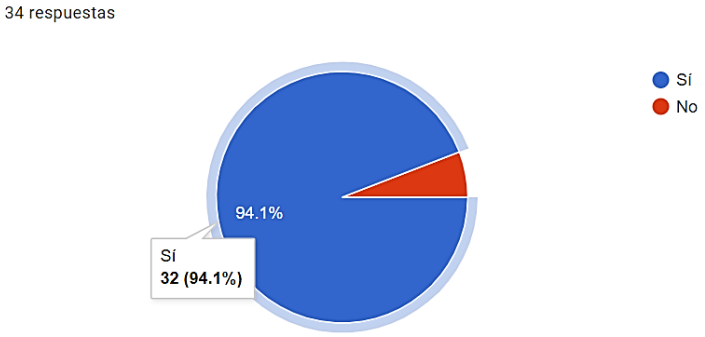


Figura 10: Porcentaje de estudiantes que agregarían Docker como contenido de aprendizaje en la Carrera de Computación.

3.1.3 REVISIÓN SISTEMÁTICA DE LA LITERATURA

3.1.3.1 PREGUNTAS DE INVESTIGACIÓN

Las preguntas de investigación que se llevaron a cabo en este estudio son:

- P1 ¿En qué escenarios es más utilizado Docker?
- P2 ¿Qué áreas se pueden beneficiar con el uso de docker?
- P3 ¿Cuáles son los beneficios de Docker como contenedor de aplicaciones?

Con base a P1, se visualizaron los escenarios que interactúa Docker como contenedor de aplicaciones; P2 analizaron la variedad de áreas en las que Docker es utilizado, por último, P3 detallaron los beneficios de Docker para que los estudiantes conozcan las principales características de esta potente tecnología como es Docker.

3.1.3.2 PROCESO DE BÚSQUEDA

Se realizó el proceso de búsqueda con base de datos científicas, con las respectivas cadenas de texto que hacen parte de la investigación, como “docker” "application container" or “contenedor de aplicaciones”, “virtualization” or “virtualización”; estas palabras fueron estipuladas haciendo referencia al tema del trabajo de integración curricular. A continuación, se detallan los motores de búsquedas en la que realizó la investigación.

- ACM
- IEEE Xplore
- Science Direct

Estos motores de búsqueda fueron seleccionados por su trayectoria y reconocimiento en la publicación de artículos científicos relacionados a las tecnologías con información de alta calidad, además de ser respaldadas por grandes instituciones.

En la tabla 2, se presenta un resumen de los artículos encontrados con base a las cadenas de textos mencionadas en el párrafo anterior, a su vez, se aplicó la

búsqueda avanzada que proporcionan los motores de búsquedas, con el filtro de los últimos 5 años de creación de los artículos que comprende desde el año 2016 hasta la actualidad, también la limitación de solo artículos científicos, descartando las revistas, estándares, libros, conferencias, al mismo tiempo se iba aplicando mediante los operadores lógicos las cadenas de texto para los títulos y resúmenes.

Tabla 2: Total de artículos encontrados en los motores de búsqueda.

Motores de búsqueda	Cantidad
ACM	151
IEEE Xplore	283
Science Direct	118
Total	552

Fuente. Los autores

En la figura 11, se visualiza el porcentaje de los artículos.

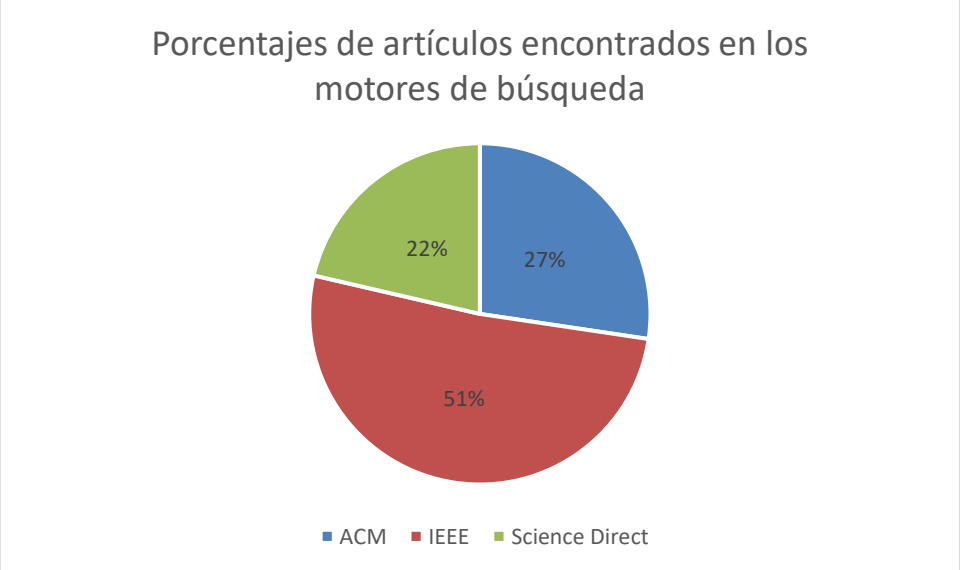


Figura 11: Porcentaje de artículos encontrados

Fuente. Los autores

3.1.3.3 CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN

Los criterios de inclusión y exclusión desarrollados se encuentran detallados en la tabla 3, este proceso inició eliminando los documentos duplicados de las diferentes bases de datos, posteriormente se aplicaron los criterios de inclusión y exclusión a los títulos y los resúmenes de los artículos encontrados. Además, para obtener unos resultados más significativos y relacionados al tema de investigación, se realizó una lectura rápida de los artículos seleccionados dejando los más incluyentes y que aportaban resultados relevantes al tema de interés. Cabe mencionar, que algunos criterios fueron aplicados directamente en el momento de realizar el proceso de búsqueda.

Tabla 3: Criterios de inclusión y exclusión

Criterios de inclusión	Criterios de exclusión
• Artículos de los últimos 5 años de antigüedad. • Títulos que contengan palabras como Docker, virtualización, contenedor de aplicaciones. • Resúmenes que contengan conceptualización, características principales, vulnerabilidades, arquitectura, aplicación en los ámbitos y análisis de Docker. • Idioma inglés y español.	• Artículos mayor a 5 años de antigüedad. • Artículos que contengan información similar a la ya consultada. • Información en repositorios de tesis y blogs.

Fuente. Los autores

Para la ejecución de esta tarea López (2019), señala que esta etapa se inicia eliminando los artículos duplicados; para lo cual se realizó una comparación de los 552 artículos encontrados en los tres motores de búsqueda mencionados y no se encontró artículos duplicados. De acuerdo con (F. Alfonso et al., 2005), los artículos científicos solo pueden publicarse si este no había sido publicado previamente.

En el análisis de los artículos encontrados se pudo observar que la información era única para cada motor de búsqueda, quedando la misma cantidad de artículos con la que se inició en cada motor de búsqueda.

Aplicando los criterios de inclusión y exclusión a los títulos a las diferentes bases de datos el resultado fue el siguiente, de los 151 artículos encontrados en ACM quedaron 19. En el caso de IEEE Xplore, se obtuvieron 283, y aplicando el mismo criterio anterior fueron seleccionados 32 artículos. Por último, se tiene el motor de búsqueda de Science direct con 118 artículos encontrados, de los cuales quedaron 27 artículos.

Los resultados obtenidos aplicando los criterios de inclusión y exclusión a los títulos de los artículos, se obtuvieron un total de 78 documentos, como se detalla en la tabla 4.

Tabla 4: Cantidad de artículos con los criterios de inclusión y exclusión de los títulos.

Motores de búsqueda	Cantidad
ACM	19
IEEE Xplore	32
Science Direct	27
Total	78

Fuente. Los autores

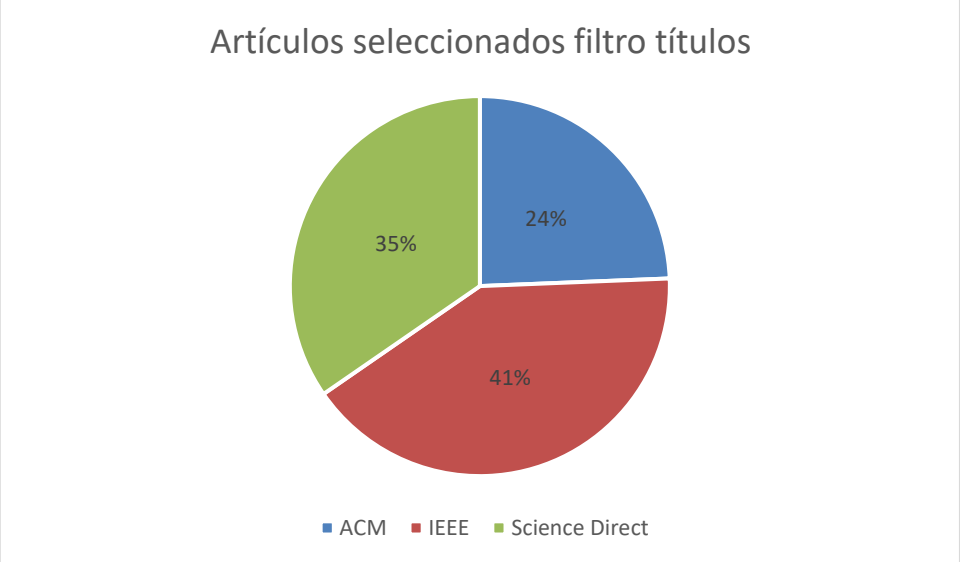


Figura 12: Porcentaje de artículos seleccionados según el título

Fuente. Los autores

Los resultados obtenidos aplicando los criterios inclusión y exclusión a los resúmenes, fueron aquellos que tenían más relevancia al trabajo de integración curricular, en donde se podían conocer las funcionalidades de Docker, como también su aplicación. A continuación, se presentan los resultados obtenidos que fueron 51 artículos que quedaron con base a estos criterios aplicados (ver Tabla 5).

Tabla 5 : Cantidad de artículos con los criterios de inclusión y exclusión de los resúmenes

Motores de búsqueda	Cantidad
ACM	11
IEEE Xplore	23
Science Direct	17
Total	51

Fuente. Los autores

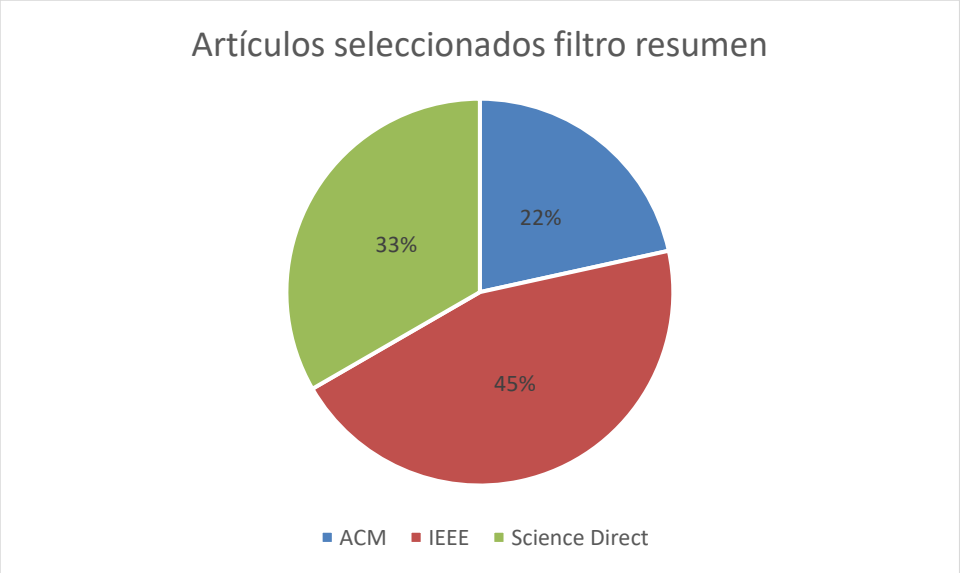


Figura 13: Porcentaje de los artículos seleccionado según el resumen.

Fuente. Los autores

Como aún se contaba con una gran cantidad de documentos se realizó una lectura rápida, considerando las preguntas de investigación propuesta en este documento, dejando los artículos con mayor relevancia al trabajo de integración curricular. A continuación, se presentan los resultados obtenidos que fueron un total 25 artículos como se detalla en la tabla 6.

Tabla 6: Resultado de la aplicación de los artículos con base a los resúmenes

Motores de búsqueda	Cantidad
ACM	4
IEEE Xplore	11
Science Direct	10
Total	25

Fuente. Los autores

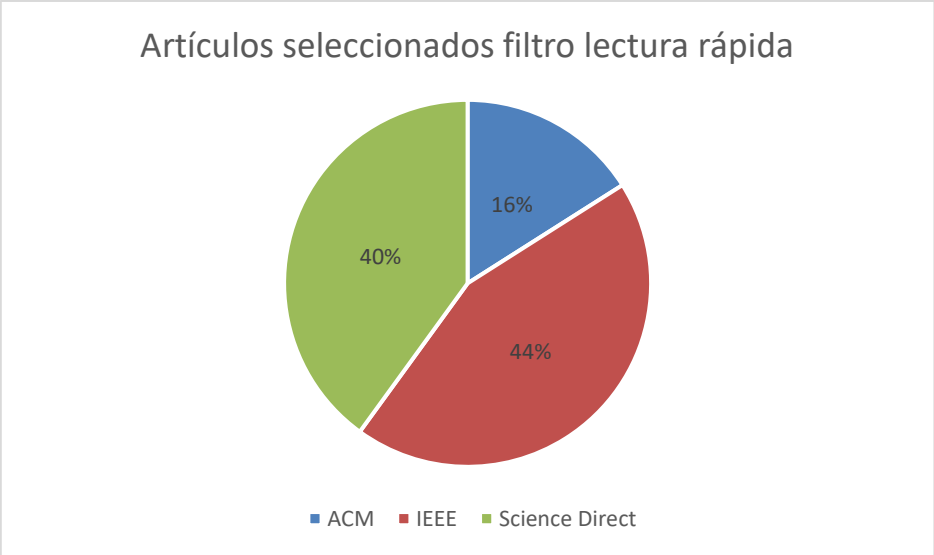


Figura 14: Porcentaje de los artículos filtrados lectura rápida.

Fuente. Los autores

Luego de aplicar los filtros pertinentes, se obtuvieron como resultado final 25 artículos que sirvieron para continuar con el proceso de esta investigación, mismo que están distribuidos de la siguiente manera: ACM 4, IEEE Xplore 11 y Science direct 10. A continuación, se detallan los artículos seleccionados ordenados por año.

Tabla 7: Artículos resultantes para la investigación

AÑO	TEMA	AUTORES	PAIS
2016	Analysis of a Network IO Bottleneck in Big Data Environments Based on Docker Containers	(China Venkanna Varma et al., 2016)	China
2017	Containers & Docker: Emerging Roles & Future of Cloud Technology	(Singh & Singh, 2017)	India
2017	Experimenting with Docker: Linux Container and BaseOS Attack Surfaces	(Mohallel et al., 2017)	Manchester
2017	Enabling Docker Containers for High-Performance and Many-Task Computing	(Azab, 2017)	Norway
2017	Optimizing the docker container usage based on load scheduling	(Sureshkumar & Rajesh, 2017)	India
2017	Emergency communication system with Docker Containers, OSM and Rsync	(Kumar, 2017)	India
2017	Improving Scalability of Apache Spark-based Scale-up Server through Docker Container-based Partitioning	(Kyong et al., 2017)	Thailand
2017	Use of Docker for deployment and testing of astronomy software	(Morris et al., 2017)	UK
2017	Container-based Virtual Elastic Clusters	(Alfonso et al., 2017)	España
2018	Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine	(Lingayat et al., 2018)	India
2018	Application deployment using Microservice and Docker containers: Framework and optimization	(Wan et al., 2018)	China
2018	Deploying Docker Swarm cluster on hybrid clouds using Occopus	(Kovács et al., 2018)	Hungría
2018	Container-based Architecture for Flexible Industrial Control Applications	(Goldschmidt et al., 2018)	Alemania
2019	Container based resource management for data processing on IoT Gateways	(Ahmed et al., 2019)	Canada
2019	Database Docker persistence Framework based on Swarm and Ceph	(Hong et al., 2019)	China
2019	Exploiting Docker container sover Grid computing for a comprehensive study of chromatin conformation in different cell types	(Merelli et al., 2019)	Italy

2019	Is it Safe to Dockerize my Database Benchmark?	(Grambow et al., 2019)	Chipre
2019	Security Analysis and Threats Detection Techniques on Docker Container	(Huang et al., 2019)	China
2019	SmartDBO: Smart Docker Benchmarking Orchestrator for Web-application	(Jha et al., 2019)	USA
2020	Application Research of Docker Based on Mesos Application Container Cluster	(Li et al., 2020)	China
2020	DIVDS: Docker Image Vulnerability Diagnostic System	(Kwon & Lee, 2020)	Corea del Sur
2020	Docker Container Log Collection and Analysis System Based on ELK	(Chen et al., 2020)	China
2020	Docker for Multi-containers Web Application	(Sharma et al., 2020)	india
2020	Performance Evaluation of Docker Container and Virtual Machine	(Ghatrehsamani et al., 2020)	India
2020	Docker based Intelligent Fall Detection using Edge-Fog Infrastructure	(Divya & Leena Sri, 2020)	India

Fuente. Los Autores

3.1.3.4 EVALUACIÓN DE LA CALIDAD

Este proceso se realizó de manera sistemática, leyendo cada uno de los artículos encontrados de manera detallada para así obtener los datos más relevantes. Además, la búsqueda de los artículos se realizó en bibliotecas digitales destacadas dando la certeza que los artículos publicados son de alto interés. Esta evaluación se realizó conjuntamente con los criterios de las preguntas de investigación que fueron:

- P1 ¿En qué escenarios es más utilizado Docker?
- P2 ¿Qué áreas se pueden beneficiar con el uso de docker?
- P3 ¿Cuáles son los beneficios de Docker como contenedor de aplicaciones?

Con la lectura y el análisis de cada uno de los artículos, se fue seleccionando según el contenido que respondían a las preguntas de investigación (ver tabla 8).

Tabla 8: Artículos analizados con base a las preguntas de investigación

AÑO	TEMA	P1	P2	P3
2016	Analysis of a Network IO Bottleneck in Big Data Environments Based on Docker Containers		X	X
2017	Containers & Docker: Emerging Roles & Future of Cloud Technology			X
2017	Experimenting with Docker: Linux Container and Base OS Attack Surfaces			X
2017	Enabling Docker Containers for High-Performance and Many-Task Computing		X	
2017	Optimizing the docker container usage based on load scheduling			X
2017	Emergency communication system with Docker Containers, OSM and Rsync			X
2017	Improving Scalability of Apache Spark-based Scale-up Server through Docker Container-based Partitioning	X		
2017	Use of Docker for deployment and testing of astronomy software		X	
2017	Container-based Virtual Elastic Clusters			X
2018	Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine	X	X	
2018	Application deployment using Microservice and Docker containers: Framework and optimization	X		X
2018	Deploying Docker Swarm cluster on hybrid clouds using Occopus		X	
2018	Container-based Architecture for Flexible Industrial Control Applications	X	X	
2019	Container based resource management for data processing on IoT Gateways		X	
2019	Database Docker persistence Framework based on Swarm and Ceph	X		
2019	Exploiting Docker container sover Grid computing for a comprehensive study of chromatin conformation in different cell types		X	
2019	Is it Safe to Dockerize my Database Benchmark?	X		
2019	Security Analysis and Threats Detection Techniques on Docker Container	X		

2019	SmartDBO: Smart Docker Benchmarking Orchestrator for Web-application	X		
2020	Application Research of Docker Based on Mesos Application Container Cluster		X	X
2020	DIVDS: Docker Image Vulnerability Diagnostic System			X
2020	Docker Container Log Collection and Analysis System Based on ELK			X
2020	Docker for Multi-containers Web Application	X		
2020	Performance Evaluation of Docker Container and Virtual Machine	X		
2020	Docker based Intelligent Fall Detection using Edge-Fog Infrastructure		X	
		10	10	10

Fuente. Los autores

Con respecto a la pregunta 1, se encontró que 10 de los artículos investigados mencionaban escenarios en los cuales se podía aplicar Docker utilizando contenedores en el desarrollo de las aplicaciones web, manejo de base de datos y elaboración de microservicios; para la pregunta 2, se encontró que 10 artículos indican que Docker se encuentra presente en el área empresarial y educativo; por último, la pregunta 3, relacionada a los beneficios que brinda Docker se encontraron también en 10 artículos, beneficios como la optimización del rendimiento del computador, minimización de costos, estandarización y productividad, y otros. A continuación, se presenta un resumen de la evaluación de los artículos de acuerdo a las preguntas de investigación.

Tabla 9: Cantidad de artículos que tienen contenidos relacionados a las preguntas de investigación

Preguntas de investigación	Referencia
P1	10
P2	10
P3	10

Fuente. Los autores

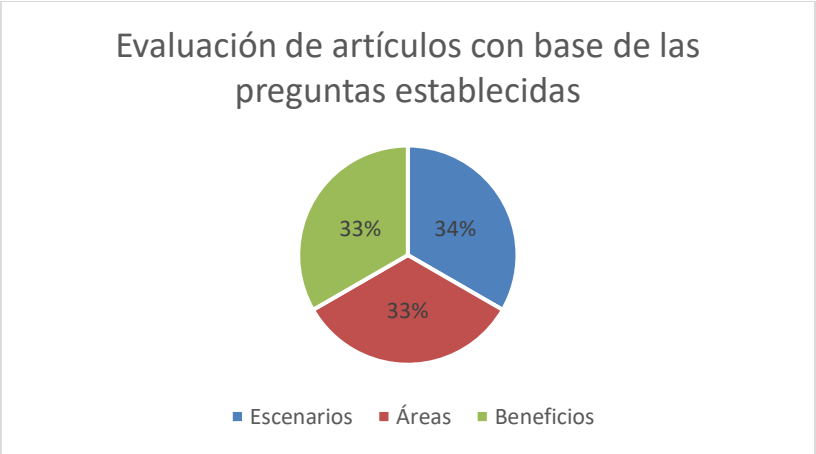


Figura 15: Evaluación de artículos

Fuente. Los autores

3.1.3.5 EXTRACCIÓN Y ANÁLISIS DE DATOS

Con base a la información encontrada para esta investigación; se contaba al inicio de la búsqueda con 552 artículos y aplicando los criterios de inclusión y exclusión quedaron 25 artículos, unos relacionado a una sola pregunta de investigación mientras que otros tenían hasta dos; en conclusión, estos artículos contienen temas de conceptualización, utilidades, y manera de usar la aplicación.

El análisis de datos se realizó con las lecturas detalladas de los artículos seleccionadas, permitiendo la identificación de datos relevantes para dar respuestas a las preguntas planteadas y dar cumplimiento a los objetivos. Para el análisis se realizaron matrices en Excel con los datos correspondientes de los artículos seleccionados que permitió a los desarrolladores del trabajo de integración curricular interpretar los datos cualitativos y cuantitativos. A continuación, se presenta el análisis de los artículos con base a las preguntas de la RSL.

- P1 ¿En qué escenarios es más utilizado Docker?

En el análisis de los artículos científicos se encontró que Docker es una herramienta de virtualización muy usada en el desarrollo de aplicaciones web, creando contenedores que son distribuidos a los desarrolladores con toda la información que se necesita para llevar a cabo el trabajo especificado;

obteniendo un total de ocho referencias donde se menciona o utiliza Docker para contenedor de aplicaciones web. Además, también es utilizado para el almacenamiento de información, en las bases de datos a través de volúmenes, permitiendo compartir datos para que los desarrolladores al iniciar consten con una información predeterminada; alcanzando un total de 2 artículos para base de datos.

Tabla 10: Cantidad de referencias encontradas con base a la pregunta 1 de la RSL

Escenarios	Referencia
Aplicaciones Web	8
Base de Datos	2
Total	10

Fuente. Los autores

- P2 ¿Qué áreas se pueden beneficiar con el uso de docker?

El análisis de las áreas en las que se utiliza Docker se encontró que la aplicación se emplea en el ámbito educativo y empresarial; en el área educativa se puede detallar su aplicación en la investigación académica, astronomía y bioinformática; para el área empresarial Docker es utilizado en el Internet de las Cosas, tecnologías de la información, investigación y desarrollo de nuevas tecnologías, entornos de desarrollo y producción.

Tabla 11: Cantidad de referencias encontradas con base a la pregunta 2 de la RSL

Escenarios	Referencia
Empresarial	7
Educativo	3
Total	10

Fuente. Los autores

- P3 ¿Cuáles son los beneficios de Docker como contenedor de aplicaciones?

Con base a los artículos encontrados para el análisis de los beneficios que ofrece Docker se encuentra la optimización del rendimiento del computador y clúster de contenedores, minimización de costos, estandarización y productividad, facilidad en el despliegue de las aplicaciones, portabilidad, simplicidad en la configuración y creación de contenedores. Cada una de estos beneficios estaban distribuidos en los diferentes artículos.

3.2 DISEÑAR ESCENARIOS DE VIRTUALIZACIÓN CON DOCKER

Para el cumplimiento de este objetivo se establecieron tres actividades que corresponden a la descarga e instalación de las herramientas necesarias para la ejecución de Docker y la creación de los escenarios correspondientes con base a la información encontrada en la Revisión Sistemática de la Literatura, estos escenarios se desarrollaron con base a los resultados de la pregunta 1 de la RSL, los cuáles fueron dos, uno para aplicaciones web y otro para las base de datos.

3.2.1 DESCARGAR LAS HERRAMIENTAS NECESARIAS

Lo primero que se realizó fue la descarga de las herramientas para el funcionamiento de Docker, como VirtualBox que es un software de virtualización que sirve para la creación de máquinas virtuales; este se encuentra en la página oficial que es virtualbox.org y se ofrece como un software de código abierto ofreciendo varias funcionalidades para el uso profesional y empresarial.

También, se realizó la descarga de Ubuntu 20.04. que fue el sistema operativo donde se montó Docker y se realizó las pruebas de los escenarios propuestos.

Para la descarga de Docker en el sistema operativo se realizó mediante las líneas de comando, que se obtuvieron en la investigación realizada (Ver Anexo 3).

Figura 16: Logo de las herramientas descargadas

Fuente. Los autores

3.2.2 INSTALAR LAS HERRAMIENTAS

La instalación de VirtualBox se realizó de manera predeterminada, es decir, navegando a través de las instrucciones de instalación y no se realizó ninguna configuración especial. Una vez instalado VirtualBox se abrió el sistema para la instalación de Ubuntu 20.04 con la creación de una máquina virtual con sus respectivas configuraciones que contiene el nombre de la máquina, la selección del tipo del sistema operativo, la versión y la selección de la carpeta donde se encuentra la descarga de Ubuntu; Además de las especificaciones físicas del computador con 2GB de memoria RAM y 15GB de almacenamiento.

Para la instalación de Docker en Ubuntu 20.04 se realizó mediante las líneas de comando, que se encontraron en la investigación (Ver Anexo 3).

3.2.3 CREAR LABORATORIOS CON DOCKER

La creación de los escenarios se realizó con base a los resultados de las preguntas de investigación que detallaron un mayor uso en aplicaciones web y base de datos. Esto se hizo a través de las imágenes en Dockerfile con las instrucciones necesarias para su creación. A continuación, se adjuntan los resultados de los contenedores ejecutados en la máquina virtual. La figura 17 pertenece a una plantilla con Nginx y la figura 18 corresponde a un contenedor que ejecuta una instancia en MongoDB.

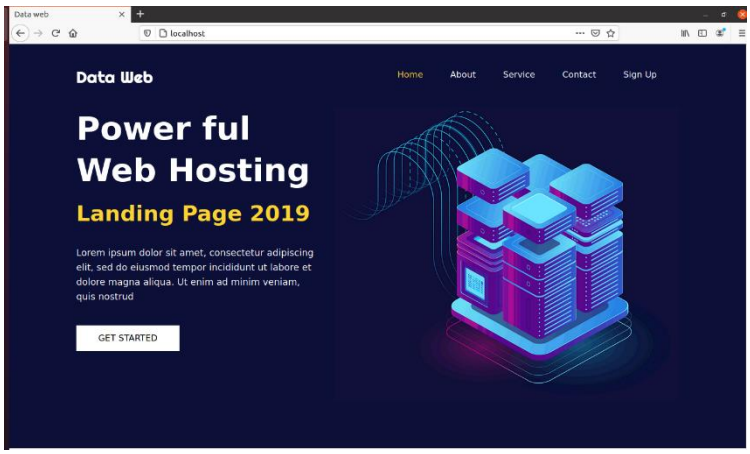


Figura 17: Plantilla de una aplicación web

Fuente. Los autores

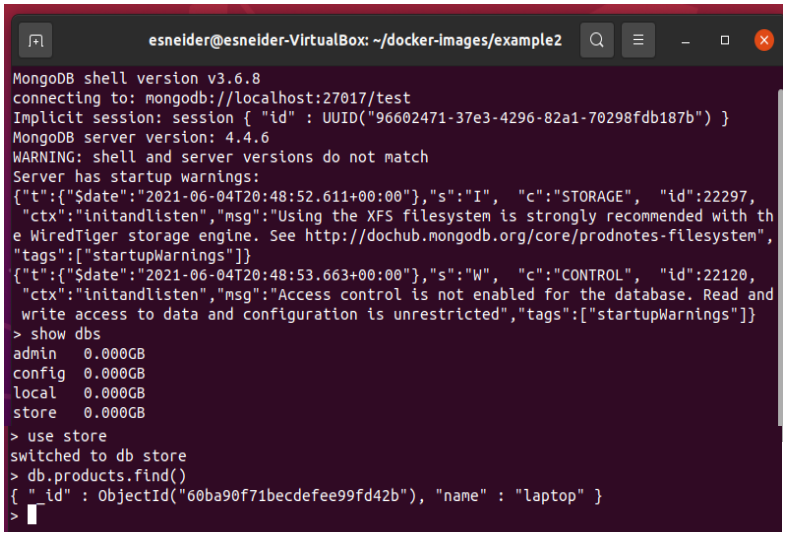


Figura 18: Instancia de Mongo DB en Docker

Fuente. Los autores

3.3 CUANTIFICAR EL COSTE COMPUTACIONAL DE LA APLICACIÓN DE DOCKER

3.3.1 REALIZAR EL ANÁLISIS DEL COSTE COMPUTACIONAL

Para la ejecución de esta actividad, se consideró los escenarios virtualizados con base a la revisión sistemática de la literatura. El coste computacional de los escenarios se realizó con el comando *docker stats.* Este comando permite consultar la información básica de los contenedores en ejecución, mostrando los siguientes valores:

- Container ID y Nombre, corresponde al ID y nombre del contenedor.
- CPU% y MEM%, muestra el porcentaje de la CPU y la memoria del host que está usando el contenedor.
- Men usage / Limit, detalla la memoria total que está usando el contenedor y la cantidad de memoria que se estableció para su uso.
- Net I/O, define la cantidad de datos enviados y recibidos que el contenedor ha generado en su interfaz de red.
- Block I/O, describe la cantidad de datos leídos y escritos del contenedor desde dispositivos de bloque en el host.
- PIDs, señala la cantidad de procesos o subprocesos que ha creado el contenedor.

Cabe mencionar, que la creación de la máquina virtual tuvo las siguientes características:

- CPU 1/8 de los que estaban disponibles para la creación de la máquina.
- 2GB de Memoria RAM.
- 15GB de Disco duro.
- Conexión a Red de tipo NAT(Network Address Translation).

Para esto en la Figura 19, se muestra los detalles del coste computacional del escenario virtualizado de una aplicación web con base a las características antes mencionadas.

Se pudo observar que el consumo de la CPU fue 0.00%, ya que las tareas que se estaban realizando en el contenedor no requerían de mucho procesamiento en la CPU; en el caso de la memoria se utilizó 2.996 MB lo que equivale al 0.15% de la memoria establecida en el host; y en la parte final se visualizó la cantidad de procesos o subprocesos que se ejecutaban en el contenedor que en este caso fueron 3.

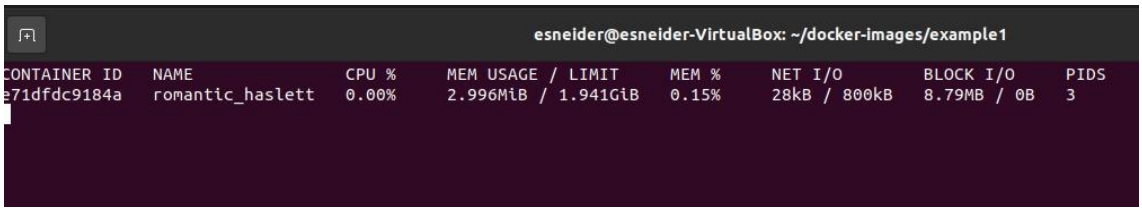


Figura 19: Detalles del coste computacional de la aplicación web del contenedor.

Fuente. Los autores

Para esto en la Figura 20, se muestra los detalles del coste computacional del escenario virtualizado de una base de datos en MongoDB con base a las características antes mencionadas.

En este se observó que el consumo de la CPU fue 0.48%; en el caso de la memoria se utilizó 168.7 MB lo que equivale al 8.49% de la memoria establecida en el host; y en la parte final se visualizó la cantidad de procesos o subprocesos que se ejecutaban en el contenedor que en este caso fueron 33.

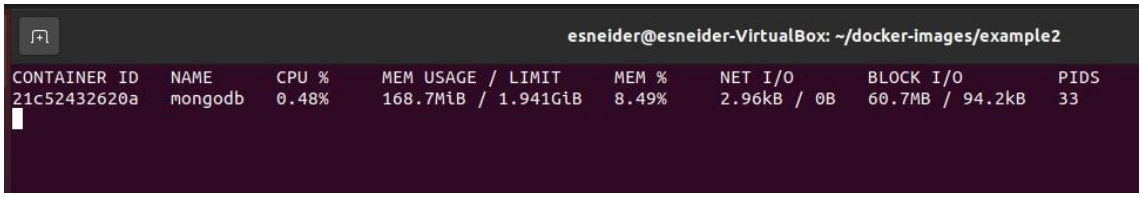


Figura 20: Detalles del coste computacional de la base de datos en MongoDB del contenedor.

Fuente. Los autores

A continuación, se presenta una tabla resumen de los datos obtenidos con su respectiva figura.

Tabla 12: Resumen del coste computacional.

Tipo	Porcentaje de CPU	Porcentaje de RAM
Aplicación Web	0,00%	0,15%
Base de Datos	0,48%	8,49%

Fuente. Los autores

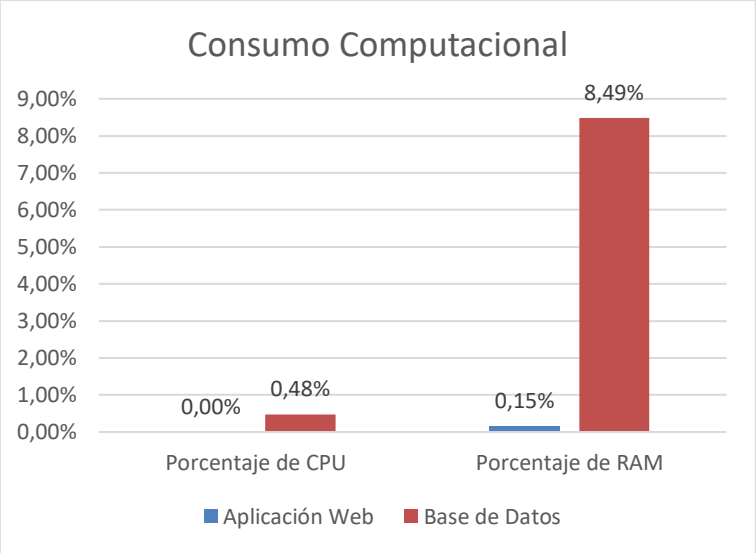


Figura 21: Consumo computacional.

Fuente. *Los autores*

3.4 ELABORAR UNA GUÍA DE LA APLICACIÓN DE DOCKER COMO CONTENEDOR DE APLICACIONES

3.4.1 GUÍA DE LA APLICACIÓN DE DOCKER

La redacción de la guía se realizó con base al levantamiento de información como fue la entrevista y la encuesta, además, de la obtención de información que se logró con la RSL. La guía contiene la información referente a la conceptualización general de Docker, la instalación para Linux y Windows, lista de comandos principales para la creación de contenedores, ejemplos prácticos de los escenarios virtualizados con su respectivo análisis del coste computacional (ver Anexo 3).

Cabe mencionar, que está guía servirá para que los estudiantes de la Carrera de Computación se guíen para futuros proyectos que nazcan desde la UDIV de infraestructura.

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Mediante la técnica de la entrevista y la encuesta se obtuvieron los requerimientos y el nivel de conocimiento de los estudiantes para la creación de la guía de aplicación de Docker; además la RSL permitió identificar y evaluar la información investigada con base a los artículos científicos que aportaron en el trabajo de integración curricular.
- Con el uso de las herramientas de virtualización se logró diseñar los escenarios de Docker como contenedor de aplicaciones encontrados con mayor relevancia en la RSL, obteniendo escenarios virtualizados con base a las aplicaciones web y bases de datos. Los escenarios ejecutados se crearon a través de los comandos que proporciona Docker.
- El análisis del coste computacional realizado a través del comando *docker stats* permitió a los autores del trabajo de integración curricular conocer el consumo de los escenarios virtualizados, con base a eso se observó que en las bases de datos hubo mayor consumo en la ejecución de las tareas, cabe mencionar, que en ambos escenarios el consumo de los contenedores es bastante optimizado.
- La creación de la guía de aplicación ofrece a los estudiantes de la Carrera de Computación, generalidades de Docker y prácticas básicas para elaborar futuros proyectos en esta herramienta de contenedor de aplicaciones.

4.2 RECOMENDACIONES

- Profundizar los contenidos de Docker de manera avanzada para el enriquecimiento académico, dado que es un campo amplio de estudio y conjunto con otras herramientas puede ejecutar más funcionalidades con mayores usos en el mercado laboral e investigativo.
- Investigar más escenarios de Docker, ya que existe una gran variedad de imágenes de las cuáles se pueden realizar combinaciones, generando nuevos escenarios para el uso de desarrolladores y/o administradores; ofreciendo agilizar la creación y ejecución aplicaciones distribuidas en cualquier escala.
- Buscar nuevas herramientas que permitan a los usuarios medir el coste computacional de los contenedores de Docker, puesto que el método utilizado en este trabajo de integración curricular proporciona información limitada; sin embargo, se visualizó que el rendimiento de Docker ha sido el esperado.
- Incluir más información de Docker a la guía de aplicación ya que es una herramienta con mucha demanda en el mercado laboral, extendiéndose con facilidad con otras herramientas y así los estudiantes de la Carrera de Computación puedan beneficiarse con las ventajas que ofrece al conocer esta herramienta.

BIBLIOGRAFÍA

Abuabdo, A., & Al-Sharif, Z. A. (2019). Virtualization vs. containerization: Towards a multithreaded performance evaluation approach. *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, *2019-Novem*, 1–6. https://doi.org/10.1109/AICCSA47632.2019.9035233

Ahmed, B., Seghir, B., Al-Osta, M., & Abdelouahed, G. (2019). Container based resource management for data processing on IoT gateways. *Procedia Computer Science*, *155*(2018), 234–241. https://doi.org/10.1016/j.procs.2019.08.034

Alfonso, F., Bermejo, J., & Segovia, J. (2005). Publicación duplicada o redundante: ¿podemos permitírnoslo? *Revista Española de Cardiología*, *58*(5), 601–604. https://doi.org/10.1157/13074852

Azab, A. (2017). Enabling docker containers for high-performance and many-task computing. *Proceedings - 2017 IEEE International Conference on Cloud Engineering, IC2E 2017*, 279–285. https://doi.org/10.1109/IC2E.2017.52

Carrera de Informática. (2019). *Carrera de Informática*. http://computacion.espam.edu.ec/recursos/archivos/archivos/2019112784954902.pdf

Carrizo, D., & Moller, C. (2018). *Estructuras metodológicas de revisiones sistemáticas de literatura en Ingeniería de Software: un estudio de mapeo sistemático Methodological structures of systematic literature review in software engineering: a systematic mapping study*.

Chen, L., Liu, J., Xian, M., & Wang, H. (2020). Docker container log collection and analysis system based on ELK. *Proceedings - 2020 International Conference on Computer Information and Big Data Applications, CIBDA 2020*, 317–320. https://doi.org/10.1109/CIBDA50819.2020.00078

China Venkanna Varma, P., Venkata Kalyan Chakravarthy, K., Valli Kumari, V., & Viswanadha Raju, S. (2016). Analysis of a Network IO Bottleneck in Big

Data Environments Based on Docker Containers. *Big Data Research*, *3*, 24–28. https://doi.org/10.1016/j.bdr.2015.12.002

de Alfonso, C., Calatrava, A., & Moltó, G. (2017). Container-based virtual elastic clusters. *Journal of Systems and Software*, *127*, 1–11. https://doi.org/10.1016/j.jss.2017.01.007

De Benedictis, M., & Lioy, A. (2019). Integrity verification of Docker containers for a lightweight cloud environment. *Future Generation Computer Systems*, *97*, 236–246. https://doi.org/10.1016/j.future.2019.02.026

Divya, V., & Leena Sri, R. (2020). Docker based intelligent fall detection using edge-fog infrastructure. *IFAC-PapersOnLine*, *53*(1), 45–50. https://doi.org/10.1016/j.ifacol.2020.06.008

Docker. (2021a). *Acerca de Docker: gestión e historial | Estibador*. https://www.docker.com/company

Docker. (2021b). *Potenciando el desarrollo de aplicaciones para desarrolladores | Estibador*. https://www.docker.com/

Erich, F. M. A., Amrit, C., & Daneva, M. (2017). A qualitative study of DevOps usage in practice. *Journal of Software: Evolution and Process*, *29*(6), 1–20. https://doi.org/10.1002/smr.1885

ESPAM MFL. (2021a). *ESPAM MFL*. http://www.espam.edu.ec/web/universidad/historia.aspx

ESPAM MFL. (2021b). *ESPAM MFL*. http://www.espam.edu.ec/web/oferta/grado/computacion.aspx

Ghatrehsamani, D., Denninnart, C., Bacik, J., & Amini Salehi, M. (2020). The Art of CPU-Pinning: Evaluating and Improving the Performance of Virtualization and Containerization Platforms. *ACM International Conference Proceeding Series*, 1–11. https://doi.org/10.1145/3404397.3404442

Goldschmidt, T., Hauck-Stattelmann, S., Malakuti, S., & Grüner, S. (2018). Container-based architecture for flexible industrial control applications.

Journal of Systems Architecture, *84*, 28–36. https://doi.org/10.1016/j.sysarc.2018.03.002

Grambow, M., Hasenburg, J., Pfandzelter, T., & Bermbach, D. (2019). Is it safe to dockerize my database benchmark? *Proceedings of the ACM Symposium on Applied Computing*, *Part F1477*, 341–344. https://doi.org/10.1145/3297280.3297545

Grande Estefan, I., & Abascal Fernández, E. (2017). *Fundamentos y técnicas de investigación comercial - Ildefonso Grande Esteban, Elena Abascal Fernández - Google Libros* (13° Edición). ESIC Editorial. https://books.google.com.ec/books?hl=es&lr=&id=zbaaDgAAQBAJ&oi=fnd&pg=PA19&dq=tecnicas+de+investigación+entrevistas&ots=U2QJ4NGMta&sig=ru7JR51-HW_8TTmVMNM7kW_vTUs&redir_esc=y#v=onepage&q&f=false

Helm, J. H., & Katz, L. G. (2016). *Young Investigators: The Project Approach in the Early Years* (Tercera). Teachers Collage Press. https://books.google.com.ec/books?id=5mQLDAAAQBAJ&printsec=frontcover&dq=young+investigators+the+project+approach+in+the+early+years&hl=es&sa=X&redir_esc=y#v=onepage&q&f=false

Hong, S., Li, D., & Huang, X. (2019). Database docker persistence framework based on swarm and ceph. *ACM International Conference Proceeding Series*, 249–253. https://doi.org/10.1145/3341069.3342985

Huang, D., Cui, H., Wen, S., & Huang, C. (2019). Security Analysis and Threats Detection Techniques on Docker Container. *2019 IEEE 5th International Conference on Computer and Communications, ICCC 2019*, 1214–1220. https://doi.org/10.1109/ICCC47050.2019.9064441

Illinois Early Learning. (2016). *El Método de Enseñanza por Proyectos. Fase 1: El inicio del proyecto | Illinois Early Learning Project*. https://illinoisearlylearning.org/es/tipsheets/projects-phase1-sp/

Jha, D. N., Nee, M., Wen, Z., Zomaya, A., & Ranjan, R. (2019). SmartDBO: Smart

Docker Benchmarking Orchestrator for Web-application. *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, 3555–3559. https://doi.org/10.1145/3308558.3314137

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., & Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. *Information and Software Technology*, *51*(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009

Kovács, J., Kacsuk, P., & Emődi, M. (2018). Deploying Docker Swarm cluster on hybrid clouds using Occopus. *Advances in Engineering Software*, *125*(September 2017), 136–145. https://doi.org/10.1016/j.advengsoft.2018.08.001

Kumar, S. (2017). *Emergency communication system with Docker Containers, OSM and Rsync*. 1064–1069.

Kwon, S., & Lee, J. H. (2020). DIVDS: Docker Image Vulnerability Diagnostic System. *IEEE Access*, *8*, 42666–42673. https://doi.org/10.1109/ACCESS.2020.2976874

Kyong, J., Jeon, J., & Lim, S. S. (2017). Improving scalability of Apache Spark-based scale-up server through Docker container-based partitioning. *ACM International Conference Proceeding Series*, 176–180. https://doi.org/10.1145/3056662.3056686

Li, X., Jiang, Y., Ding, Y., Wei, D., Ma, X., & Li, W. (2020). Application Research of Docker Based on Mesos Application Container Cluster. *Proceedings - 2020 International Conference on Computer Vision, Image and Deep Learning, CVIDL 2020*, *Cvidl*, 476–479. https://doi.org/10.1109/CVIDL51233.2020.00-47

Lingayat, A., Badre, R. R., & Gupta, A. K. (2018). Performance Evaluation for Deploying Docker Containers on Baremetal and Virtual Machine. *Proceedings of the 3rd International Conference on Communication and Electronics Systems, ICCES 2018*, *Icces*, 1019–1023.

https://doi.org/10.1109/CESYS.2018.8723998

López-Roldán, P., & Fachelli, S. (2016). *METODOLOGÍA DE LA INVESTIGACIÓN SOCIAL CUANTITATIVA*.

López, N. (2019). *HERRAMIENTA DE APOYO A REVISIONES SISTEMÁTICAS DE LA LITERATURA EN EL ÁREA DE LA COMPUTACIÓN*.

Manterola, C., Astudillo, P., Arias, E., & Claros, N. (2013a). Revisiones sistemáticas de la literatura. Qué se debe saber acerca de ellas. *Cirugia Espanola*, *91*(3), 149–155. https://doi.org/10.1016/j.ciresp.2011.07.009

Manterola, C., Astudillo, P., Arias, E., & Claros, N. (2013b). Revisiones sistemáticas de la literatura. Qué se debe saber acerca de ellas. *Cirugia Espanola*, *91*(3), 149–155. https://doi.org/10.1016/j.ciresp.2011.07.009

Merelli, I., Fornari, F., Tordini, F., D'Agostino, D., Aldinucci, M., & Cesini, D. (2019). Exploiting Docker containers over Grid computing for a comprehensive study of chromatin conformation in different cell types. *Journal of Parallel and Distributed Computing*, *134*, 116–127. https://doi.org/10.1016/j.jpdc.2019.08.002

Mohallel, A. A., Bass, J., M., & Dehghantaha, A. (2017). Experimenting with docker: Linux container and baseos attack surfaces. *International Conference on Information Society, i-Society 2016*, 17–21. https://doi.org/10.1109/i-Society.2016.7854163

Morris, D., Voutsinas, S., Hambly, N. C., & Mann, R. G. (2017). Use of Docker for deployment and testing of astronomy software. *Astronomy and Computing*, *20*, 105–119. https://doi.org/10.1016/j.ascom.2017.07.004

Nish Anil. (2020). *Contenedores como base para la colaboración de DevOps | Microsoft Docs*. https://docs.microsoft.com/es-es/dotnet/architecture/containerized-lifecycle/docker-application-lifecycle/containers-foundation-for-devops-collaboration

Perdigón, R., & Ramírez, R. (2020, January). *Plataformas de software libre para*

la virtualización de servidores en pequeñas y medianas empresas cubanas. Revista Cubana de Ciencias Informáticas. http://scielo.sld.cu/pdf/rcci/v14n1/2227-1899-rcci-14-01-40.pdf

Sharma, V., Saxena, H. K., & Singh, A. K. (2020). Docker for Multi-containers Web Application. *2nd International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2020 - Conference Proceedings*, *Icimia*, 589–592. https://doi.org/10.1109/ICIMIA48430.2020.9074925

Shu, R., Gu, X., & Enck, W. (2017). A study of security vulnerabilities on docker hub. *CODASPY 2017 - Proceedings of the 7th ACM Conference on Data and Application Security and Privacy*, 269–280. https://doi.org/10.1145/3029806.3029832

Singh, S., & Singh, N. (2017). Containers & Docker: Emerging roles & future of Cloud technology. *Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, ICATccT 2016*, 804–807. https://doi.org/10.1109/ICATCCT.2016.7912109

Sureshkumar, M., & Rajesh, P. (2017). Optimizing the docker container usage based on load scheduling. *Proceedings of the 2017 2nd International Conference on Computing and Communications Technologies, ICCCT 2017*, 165–168. https://doi.org/10.1109/ICCCT2.2017.7972269

UDIV de Infraestructura. (2020). *UNIDAD DE INFRAESTRUCTURA DE LA CARRERA DE COMPUTACION DE LA ESP20200911_13350524.pdf*.

Universidades de Ecuador. (2021). *Campus Calceta - ESPAM MFL*. https://www.universidades.com.ec/escuela-superior-politecnica-agropecuaria-de-manabi/campus-calceta

VMware. (2021). *¿En qué consisten la tecnología de virtualización y las máquinas virtuales? | VMware | LATAM*. https://www.vmware.com/latam/solutions/virtualization.html

Wan, X., Guan, X., Wang, T., Bai, G., & Choi, B. Y. (2018). Application deployment using Microservice and Docker containers: Framework and

optimization. *Journal of Network and Computer Applications*, *119*(December 2017), 97–109. https://doi.org/10.1016/j.jnca.2018.07.003

Wei, W., Wang, K., Wang, K., Gu, H., & Shen, H. (2020). Multi-resource balance optimization for virtual machine placement in cloud data centers. *Computers and Electrical Engineering*, *88*. https://doi.org/10.1016/j.compeleceng.2020.106866

Yu, F. R., Liu, J., He, Y., Si, P., & Zhang, Y. (2018). Virtualization for Distributed Ledger Technology (vDLT). *IEEE Access*, *6*(c), 25019–25028. https://doi.org/10.1109/ACCESS.2018.2829141

Zaldivar, A. (2019). *Laboratorios reales versus laboratorios virtuales en las carreras de ciencias de la computación*. México. http://www.scielo.org.mx/pdf/ierediech/v10n18/2448-8550-ierediech-10-18-9.pdf

ANEXOS

ANEXO 1. ENTREVISTA CON EL RESPONSABLE ENCARGADO DE LA UDIV DE INFRAESTRUCTURA

ANEXO 1A. RESPUESTAS DEL RESPONSABLE ENCARGADO DE LA UDIV DE INFRAESTRUCTURA

Formato de la entrevista dirigida al encargado de la UDIV de Infraestructura de la Carrera de Computación de la Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López.

Objetivo: Obtener información para el desarrollo del trabajo de la Unidad de Integración Curricular.

Entrevistado: Ing. Javier López Zambrano, Mgtr.

Fecha: **06/05/2021**

1) **¿Cuál es la misión y visión de la UDIV de Infraestructura?**

   Misión:

   Mejorar la calidad del proceso de enseñanza - aprendizaje en las asignaturas integradoras desde el ámbito de infraestructuras

   Visión:

   Ser una Unidad de apoyo a las funciones sustantivas de Docencia, Investigación y Vinculación de la Carrera de Computación

2) **¿Cuáles son los objetivos que tiene la UDIV de Infraestructura?**

   El objetivo principal es, Fortaleces el nivel de aprendizaje de los estudiantes de la Carrera de Computación de la ESPAM MFL dentro del proceso de enseñanza – aprendizaje, desde las funciones de Docencia, Investigación y Vinculación.

3) **¿Cuáles son las funciones que tienen actualmente en la UDIV de Infraestructura?**

Brindar un espacio donde los alumnos puedan fomentar sus conocimientos y experiencias en asignaturas integradoras desde el ámbito de infraestructuras, en el contexto de las funciones sustantivas: Docencia, Investigación y Vinculación.

Unidad estratégica de infraestructuras que sirve de apoyo a la Dirección de la Carrera de Computación.

4) **En la UDIV de Infraestructura ¿Se ha usado la tecnología de Docker?**

No, sin embargo, se conoce a breves rasgos sobre la potencialidad, y se requiere tener un análisis al respecto para que sirva de base para futuras implementaciones.

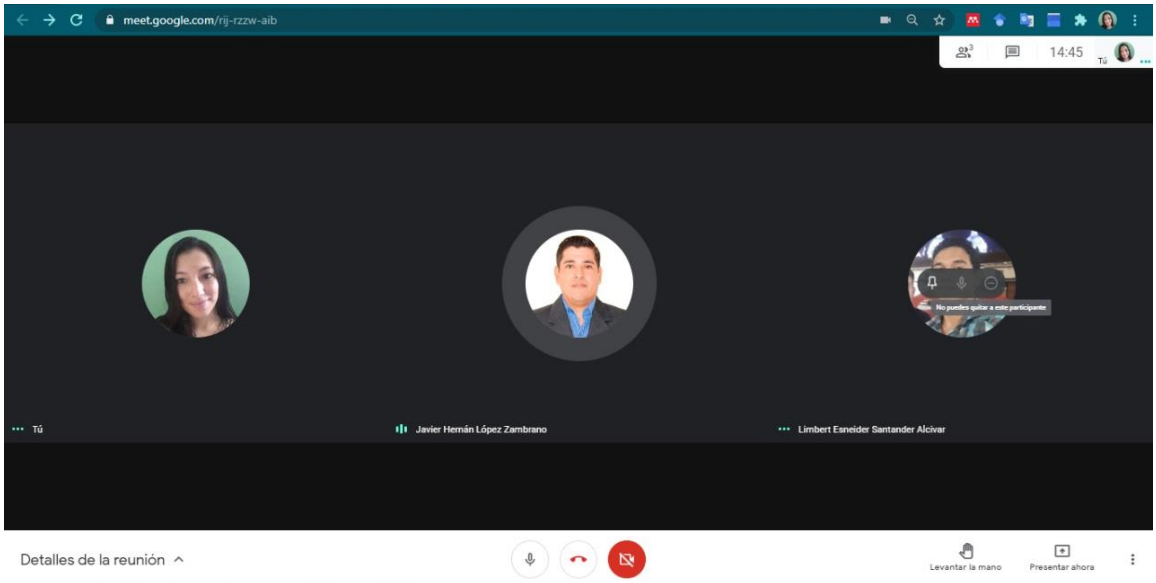
5) **Con base al Trabajo de Integración Curricular propuesto, ¿Qué objetivo cumplirá la información investigada para la UDIV de Infraestructura?**

Servir de documento base para futuras implementaciones de Sistemas

6) **¿Qué les llamó la atención para evaluar esta herramienta?**

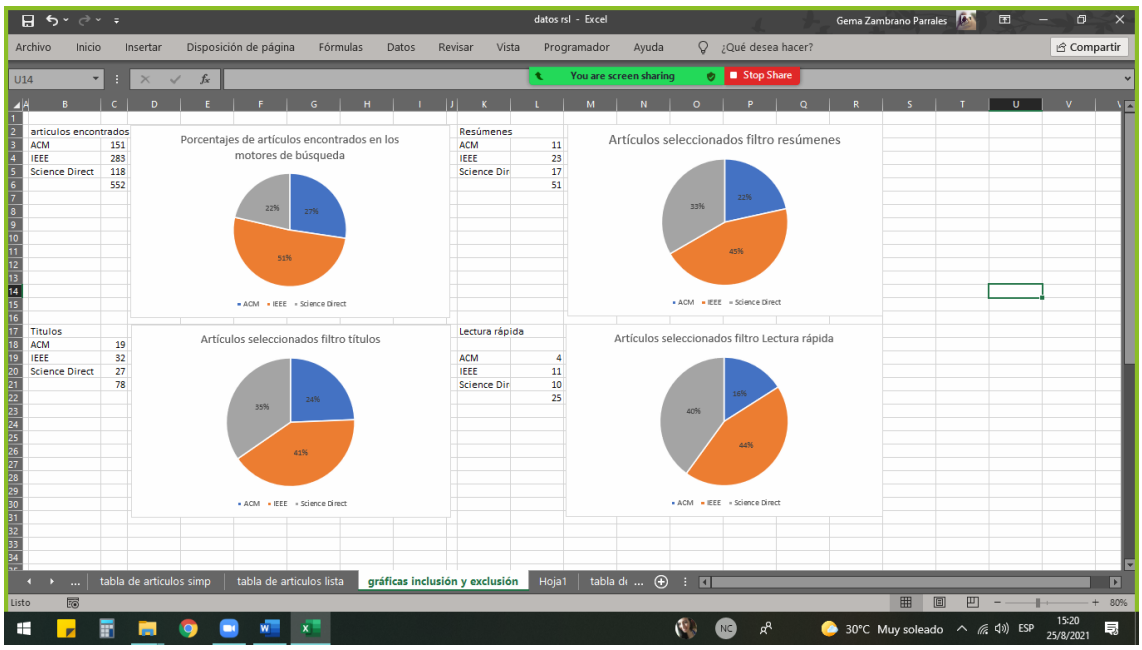
El hecho de que parece ser una herramienta prometedora para ahorrar costos de implementación de Sistemas, Servicios web, entre otros.

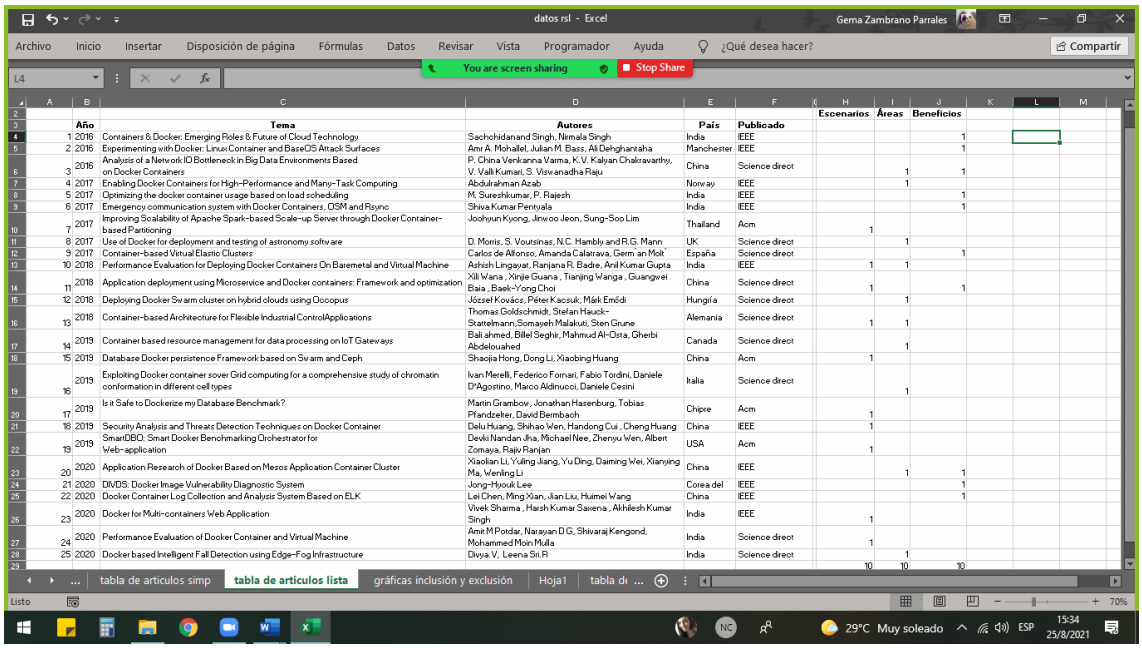
ANEXO 1B. CAPTURA DE PANTALLA DE LA SESIÓN DE LA ENTREVISTA REALIZADA



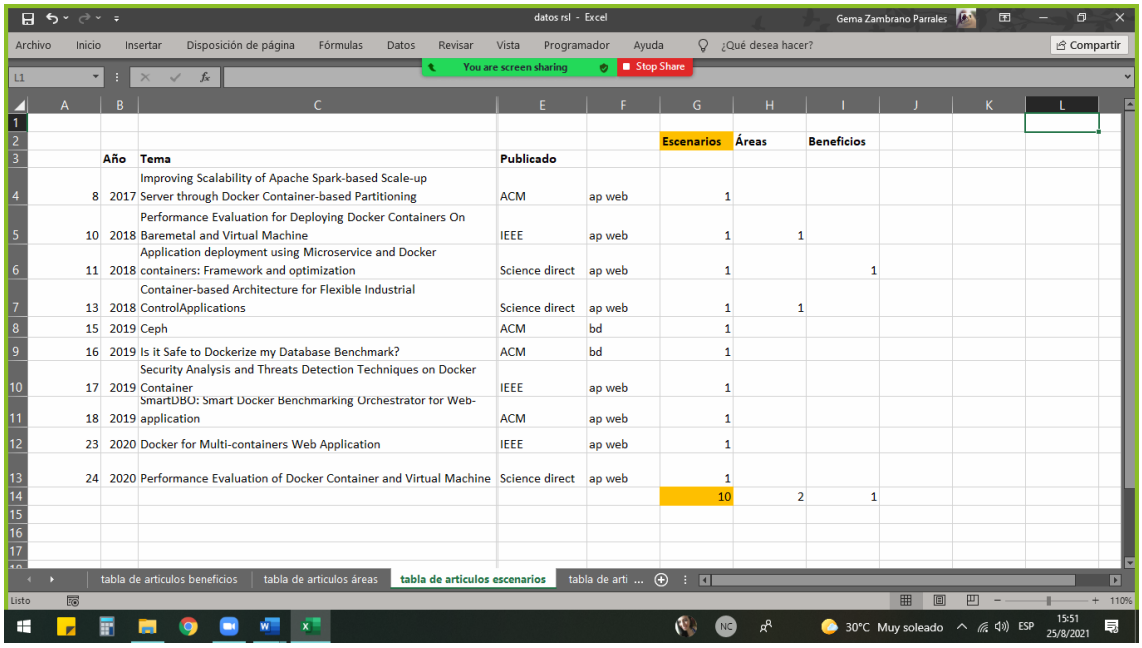
ANEXO 2. REVISIÓN SISTEMÁTICA DE LA LITERARURA

ANEXO 2A. TABULACIÓN DE DATOS DE LOS ARTÍCULOS SELECCIONADOS

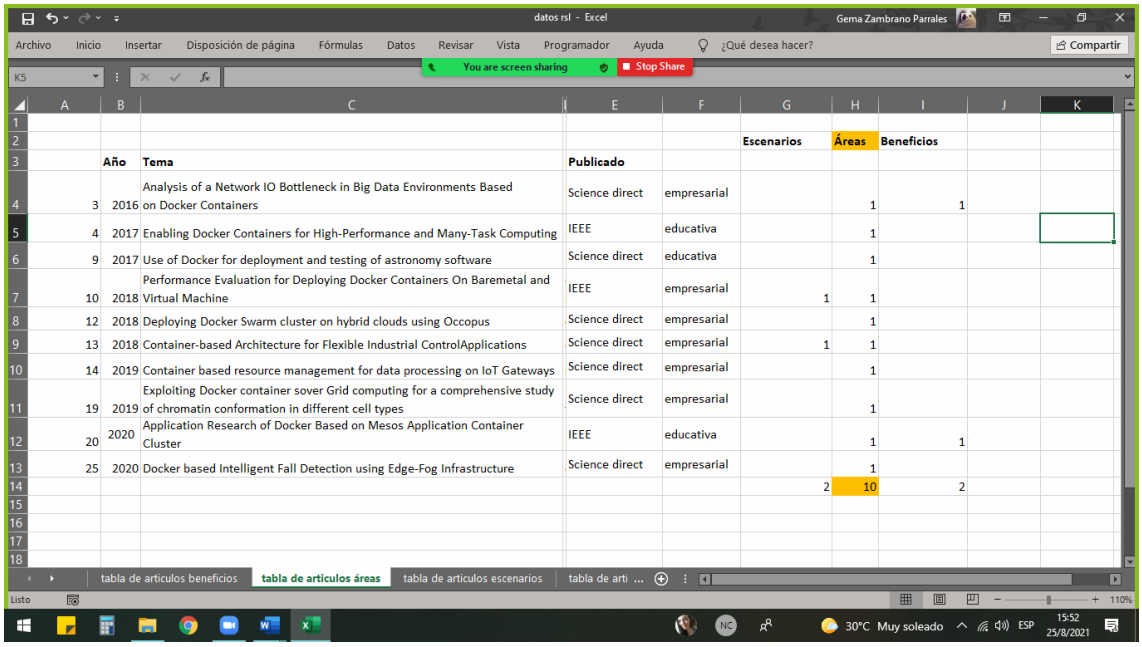




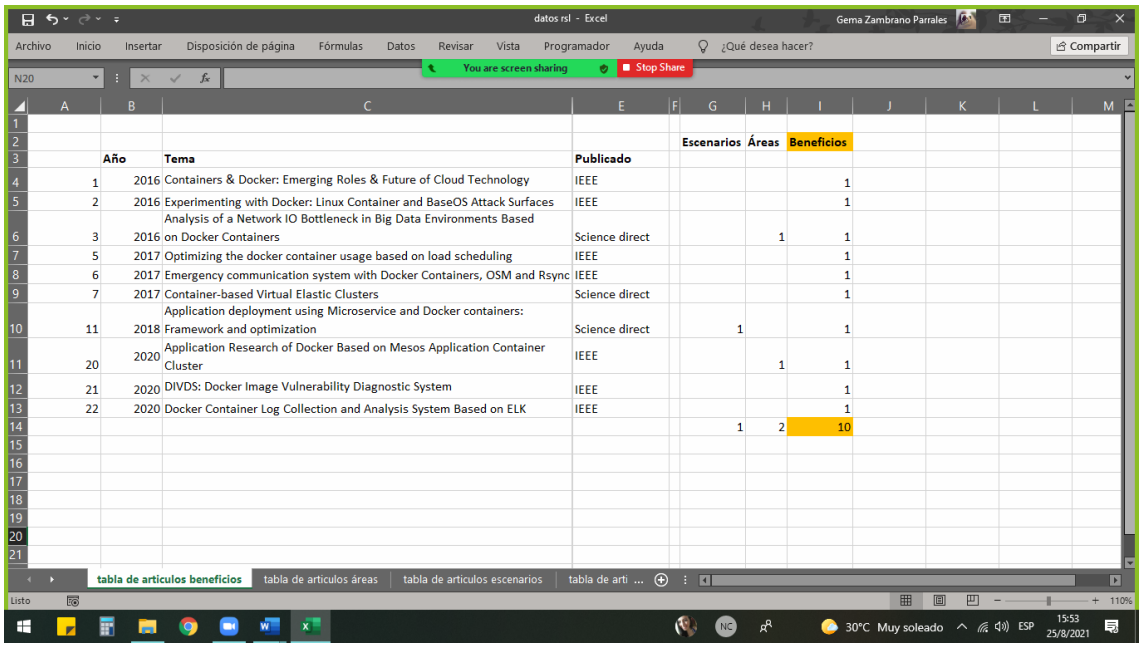
ANEXO 2B. EXTRACCIÓN Y ANÁLISIS DE DATOS DE LOS ARTÍCULOS SELECCIONADOS



	Año	Tema	Publicado		Escenarios	Áreas	Beneficios
8	2017	Improving Scalability of Apache Spark-based Scale-up Server through Docker Container-based Partitioning	ACM	ap web	1		
10	2018	Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine	IEEE	ap web	1	1	
11	2018	Application deployment using Microservice and Docker containers: Framework and optimization	Science direct	ap web	1		1
13	2018	Container-based Architecture for Flexible Industrial ControlApplications	Science direct	ap web	1	1	
15	2019	Ceph	ACM	bd	1		
16	2019	Is it Safe to Dockerize my Database Benchmark?	ACM	bd	1		
17	2019	Security Analysis and Threats Detection Techniques on Docker Container	IEEE	ap web	1		
18	2019	SmartDBO: Smart Docker Benchmarking Orchestrator for Web-application	ACM	ap web	1		
23	2020	Docker for Multi-containers Web Application	IEEE	ap web	1		
24	2020	Performance Evaluation of Docker Container and Virtual Machine	Science direct	ap web	1		
					10	2	1



	Año	Tema	Publicado		Escenarios	Áreas	Beneficios
3	2016	Analysis of a Network IO Bottleneck in Big Data Environments Based on Docker Containers	Science direct	empresarial		1	1
4	2017	Enabling Docker Containers for High-Performance and Many-Task Computing	IEEE	educativa		1	
9	2017	Use of Docker for deployment and testing of astronomy software	Science direct	educativa		1	
10	2018	Performance Evaluation for Deploying Docker Containers On Baremetal and Virtual Machine	IEEE	empresarial	1	1	
12	2018	Deploying Docker Swarm cluster on hybrid clouds using Occopus	Science direct	empresarial		1	
13	2018	Container-based Architecture for Flexible Industrial ControlApplications	Science direct	empresarial	1	1	
14	2019	Container based resource management for data processing on IoT Gateways	Science direct	empresarial		1	
19	2019	Exploiting Docker container sover Grid computing for a comprehensive study of chromatin conformation in different cell types	Science direct	empresarial		1	
20	2020	Application Research of Docker Based on Mesos Application Container Cluster	IEEE	educativa		1	1
25	2020	Docker based Intelligent Fall Detection using Edge-Fog Infrastructure	Science direct	empresarial		1	
					2	10	2



datos rsl - Excel — Gema Zambrano Parrales

Archivo Inicio Insertar Disposición de página Fórmulas Datos Revisar Vista Programador Ayuda ¿Qué desea hacer? Compartir

You are screen sharing — Stop Share

N20

	A	B	C	E	G	H	I
2					Escenarios	Áreas	Beneficios
3		Año	Tema	Publicado			
4	1	2016	Containers & Docker: Emerging Roles & Future of Cloud Technology	IEEE			1
5	2	2016	Experimenting with Docker: Linux Container and BaseOS Attack Surfaces	IEEE			1
6	3	2016	Analysis of a Network IO Bottleneck in Big Data Environments Based on Docker Containers	Science direct		1	1
7	5	2017	Optimizing the docker container usage based on load scheduling	IEEE			1
8	6	2017	Emergency communication system with Docker Containers, OSM and Rsync	IEEE			1
9	7	2017	Container-based Virtual Elastic Clusters	Science direct			1
10	11	2018	Application deployment using Microservice and Docker containers: Framework and optimization	Science direct	1		1
11	20	2020	Application Research of Docker Based on Mesos Application Container Cluster	IEEE		1	1
12	21	2020	DIVDS: Docker Image Vulnerability Diagnostic System	IEEE			1
13	22	2020	Docker Container Log Collection and Analysis System Based on ELK	IEEE			1
14					1	2	10

tabla de articulos beneficios | tabla de articulos áreas | tabla de articulos escenarios | tabla de arti ...

Listo — 110%

30°C Muy soleado — ESP — 15:53 25/8/2021

ANEXO 3. CERTIFICACIÓN DE ENTREGA DE LA GUÍA PRÁCTICA DE LA APLICACIÓN DE DOCKER A LA UDIV DE INFRAESTUCTURA.

Carrera de COMPUTACIÓN

Calceta, 17 de diciembre de 2021

Ingeniero
Joffre Moreira Pico, MGS.
Director Carrera de Computación - ESPAM MFL
En su despacho. -

Asunto: CARTA AVAL

En mi calidad de encargado de la UDIV de Infraestructuras, comunico que una vez revisado el entregable de la tesis titulada: "**ANÁLISIS DE LA TECNOLOGÍA DOCKER COMO CONTENEDOR DE APLICACIONES**", doy constancia de la entera satisfacción al cumplimiento del trabajo de integración curricular y culminando de manera satisfactoria cada uno de los requerimientos solicitado por esta unidad, por tal motivo cumplo con el proceso de la emisión del aval correspondiente a los estudiantes: **ZAMBRANO PARRALES GEMA MARÍA** y **SANTANDER ALCÍVAR LIMBERT ESNEIDER**, destacando sus excelentes competencias y colaboración en el desarrollo de este proyecto de sistematización de experiencia.

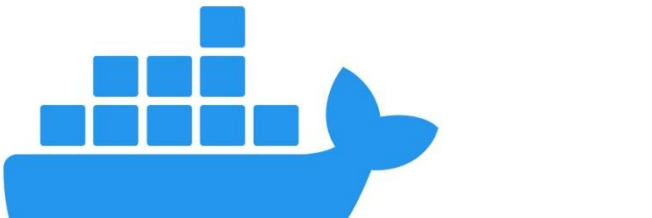
Para los fines legales pertinentes, me suscribo a usted.

Atentamente,

Firmado digitalmente por JAVIER HERNAN LOPEZ ZAMBRANO

Mgtr. Javier Hernán López Zambrano
Encargado de la UDIV de Infraestructuras
Carrera de Computación
ESPAM MFL
Correo: jlopez@espam.edu.ec

ANEXO 4. GUÍA PRÁCTICA DE LA APLICACIÓN DE DOCKER.



docker

UDIV DE INFRAESTRUCTURA

2021

Contenido

¿Qué es Docker?

Antes de conocer de Docker, es importante hablar sobre las herramientas de virtualización, y es que estas permiten alojar entornos aislados, añadiendo librerías y dependencias para cumplir con el servicio, a su vez la virtualización se puede realizar por máquinas virtuales y por contenedores (Todea, 2016).

Docker es una herramienta de virtualización en contenedores, que permite a los desarrolladores llevar a cabo sus ideas conquistando la complejidad del desarrollo de aplicaciones, este simplifica y agiliza el trabajo del desarrollo mediante la consolidación de los componentes de la aplicación.

Es una herramienta que permite desplegar aplicaciones en contenedores, de manera rápida y portable, lo cual significa, que Docker genera aplicaciones de bolsillo a través de imágenes y contenedores en donde se encontrará la información necesaria para el funcionamiento de la aplicación.

A continuación, se presenta la arquitectura de virtualización de la máquina virtual y de un contenedor.

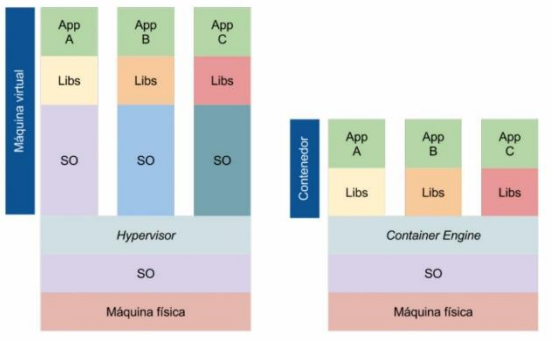


Imagen 1: Distinción entre MV y contenedores
Fuente: (Todea, 2016)

Estructura de Docker

Docker tiene como base Docker Hub, siendo el servidor de alojamiento; dentro de esto se encuentra el servicio propio de Docker llamado Docker Daemon, luego un API que es el canal de comunicación entre el cliente y servidor, y el cliente que en todo caso pueden ser contenedores, imágenes, volúmenes y redes.

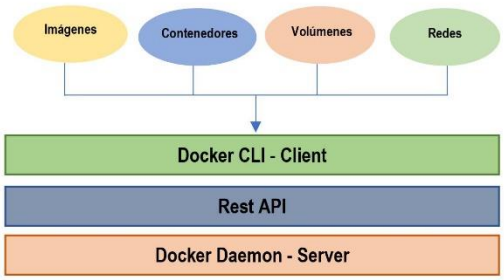


Imagen 2: Estructura de Docker

¿Qué es una imagen?

Una imagen de Docker es la instantánea de un contenedor, está compuesto por N capas que se usan para ejecutar el código de un contenedor Docker y en ella se encuentra un paquete que tiene toda la configuración necesaria para el funcionamiento del servicio. La imagen es la plantilla principal para ejecutar las aplicaciones siendo estos los nuevos contenedores.

Las imágenes de Docker son visibles en diferentes repositorios, ya sean públicos o privados, siendo el más conocido el Docker Hub, donde se encuentran un sinnúmero de imágenes para utilizar en cualquier proyecto.

¿Qué es una capa?

En lo anterior se menciona que las imágenes están hechas a partir de una capa, y esta es el conjunto de cambios en el sistema de archivos del contenedor; que pueden ser compartidas entre distintas imágenes.

¿Qué es un contenedor?

Un contenedor es una capa adicional que ejecuta en tiempo real todo lo que está dentro de las imágenes, esta es una capa temporal donde se puede crear, editar o eliminar mientras que las capas de imágenes son solo de lectura. Además, el contenedor se ejecuta sobre el mismo sistema operativo anfitrión de forma aislada y sin necesidad de uno propio ya que comparten el mismo kernel, lo cual los hacen mucho más ligeros.

¿Cómo instalar Docker?

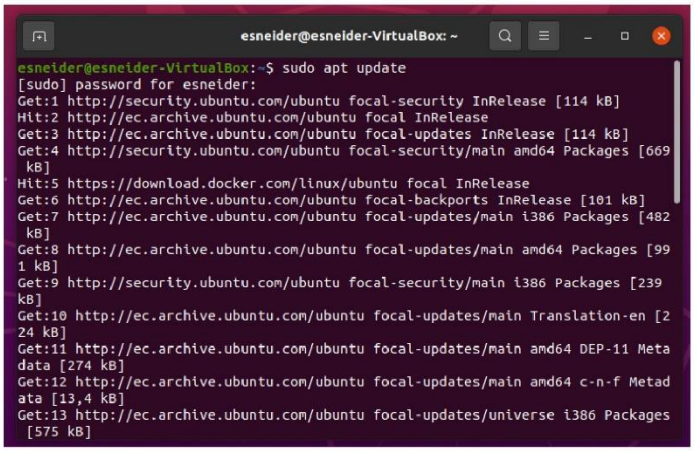
Instalación en Ubuntu

A continuación, se presentan los pasos de la instalación de Docker en Ubuntu 20.04.

Paso 1: Actualizar los paquetes existentes de Ubuntu y agregar una nueva fuente de paquetes de Docker con la clave GPG para garantizar las descargas, a continuación, se presenta un comentario de la actividad a hacer, el código y la captura de imagen.

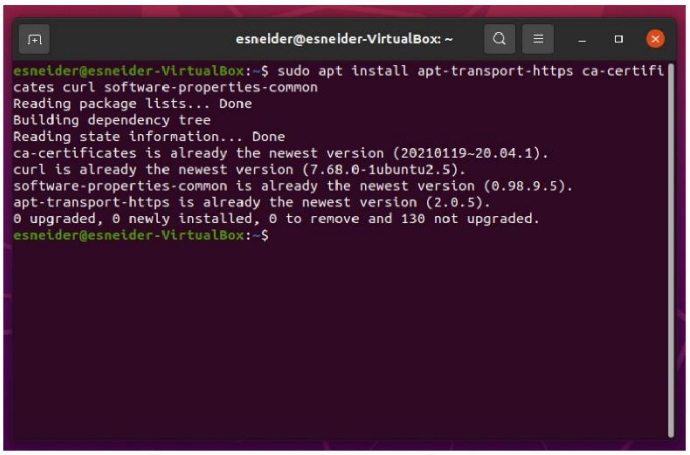
Actualizar los paquetes.

$ sudo apt update



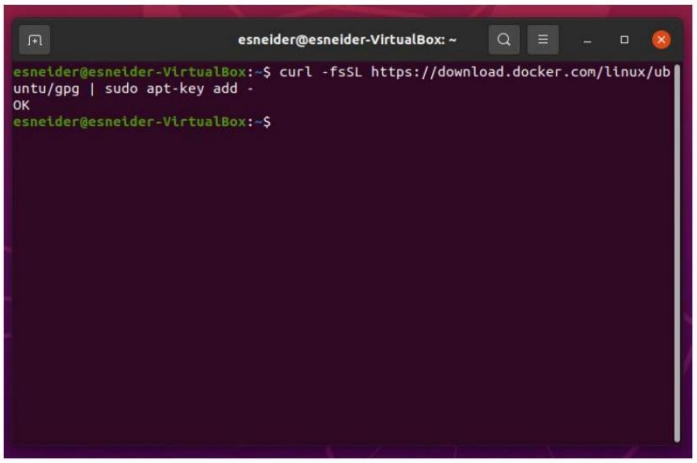
Instalar paquetes que permitan acceder a través de HTTPS.

$ sudo apt install apt-transport-https ca-certificates curl software-properties-common



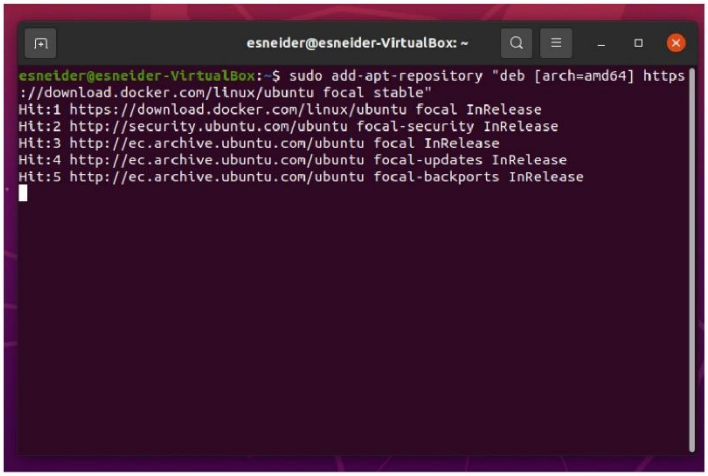
Agregar la clave GPG para el repositorio oficial de Docker.

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add –



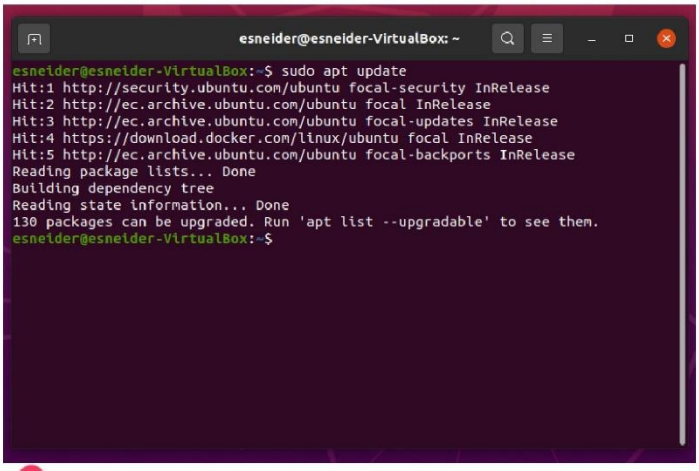
Agregar el repositorio de Docker a las fuentes de APT.

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"



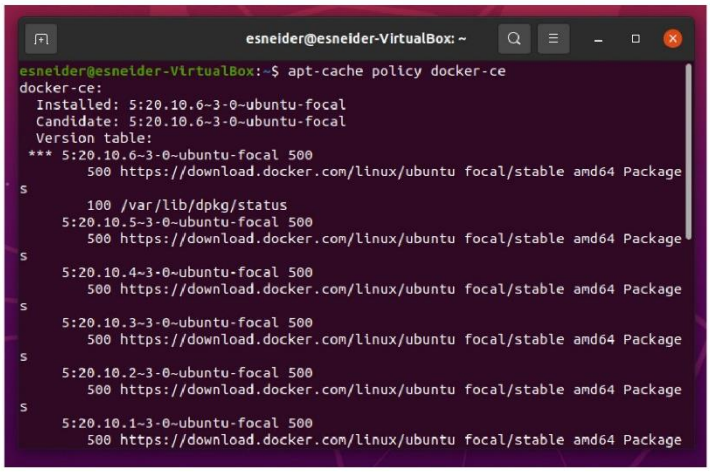
Actualizar los paquetes de base de datos con los nuevos paquetes instalados de Docker.

$ sudo apt update



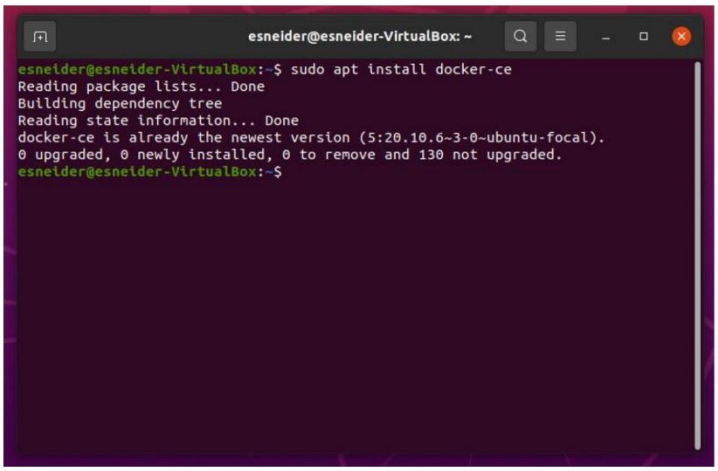
Ubicarse en el repositorio de Docker en lugar del repositorio de Ubuntu.

$ apt-cache policy docker-ce



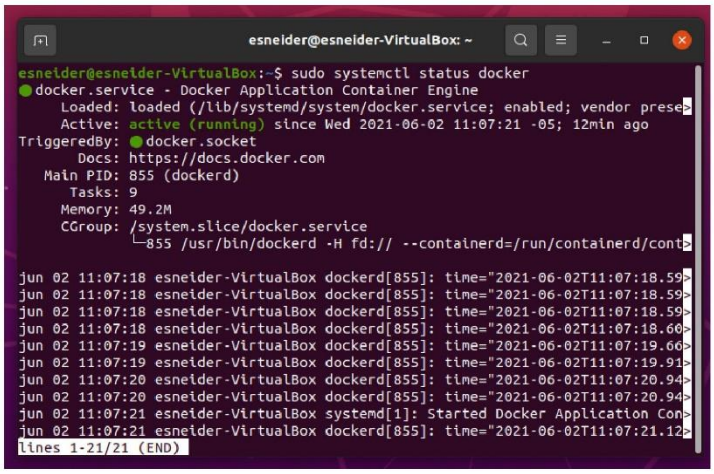
Instale Docker.

$ sudo apt install docker-ce



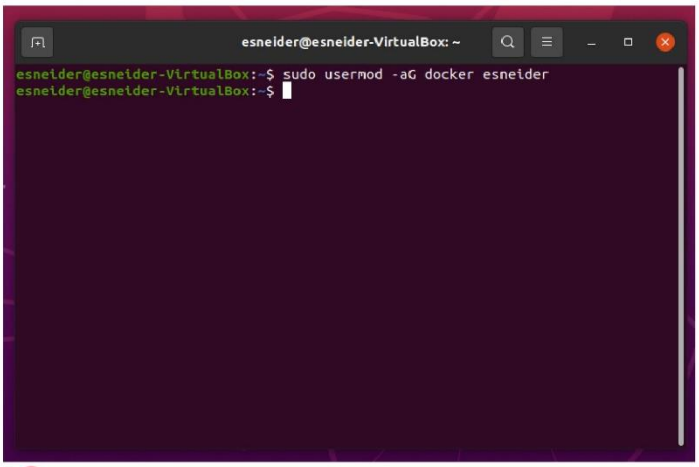
Habilitar la ejecución al inicio.

$ sudo systemctl status docker



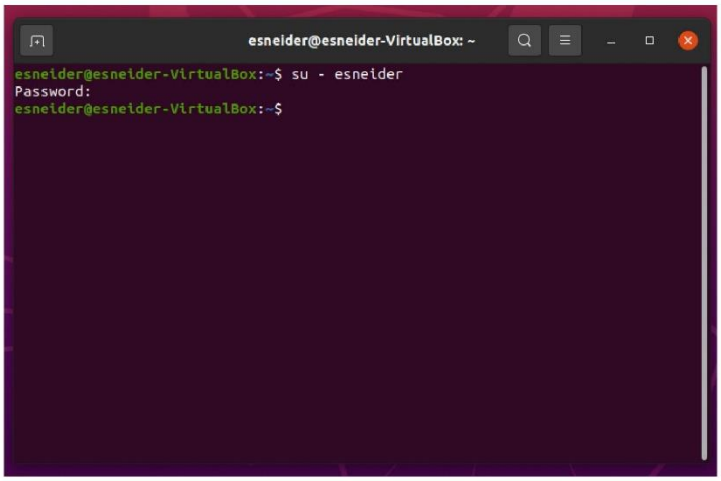
Paso 2: Ejecutar el comando Docker, sea este con usuario root o un usuario del grupo Docker, para esto agregar usuario en el grupo de Docker de la siguiente manera.

$ sudo usermod -aG docker ${USER}



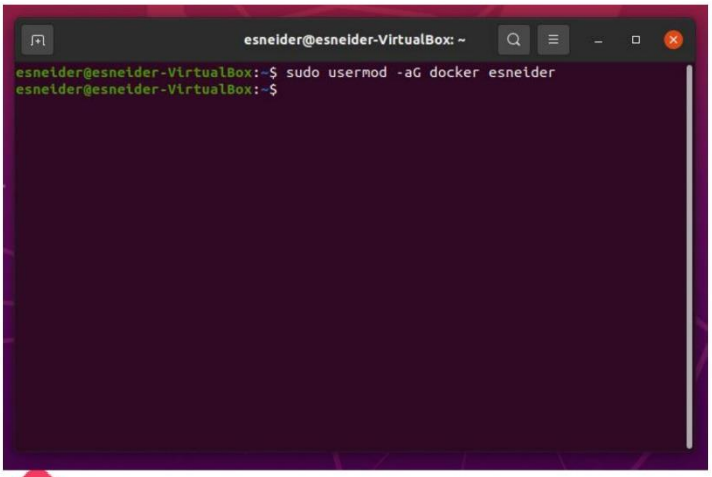
Aplicar la nueva membresía de grupo, y reinicie el inicio de sesión.

$ su - ${USER}

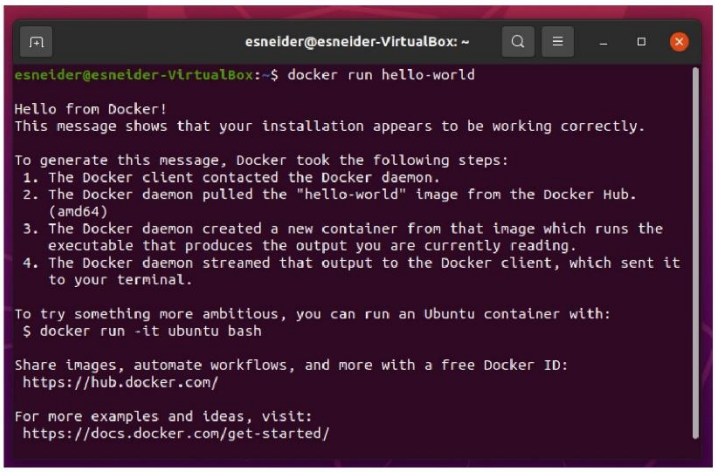


Después del proceso anterior, pedirá ingresar nuevamente la con contraseña de usuario, y se declara de forma explicita el nombre de usuario.

$ sudo usermod -aG docker username



Paso 3: Verificar que se pueda acceder a las imágenes y descargarlas de Docker Hub.

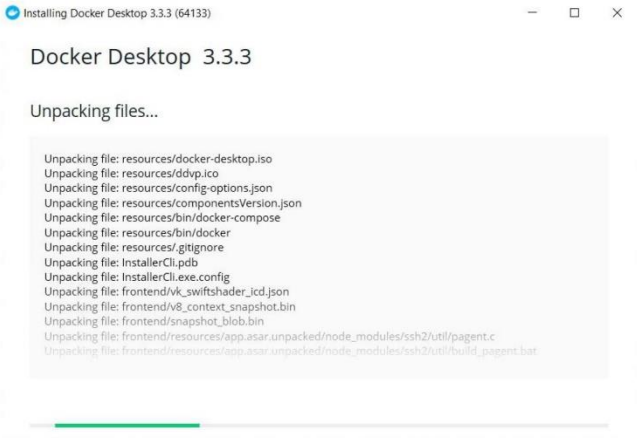


Instalación en Windows

A continuación, se presentan los pasos de la instalación de Docker en Windows

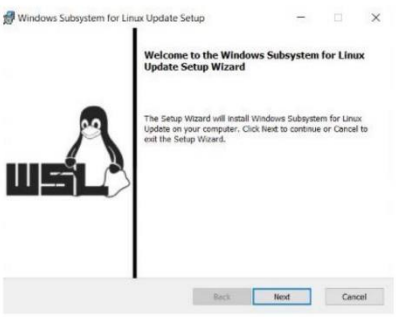
Paso 1: Para la instalación se visita la página oficial de Docker, y en el apartado de productos se encuentra Docker Desktop, se da clic en el botón de la descarga para obtener el ejecutable.

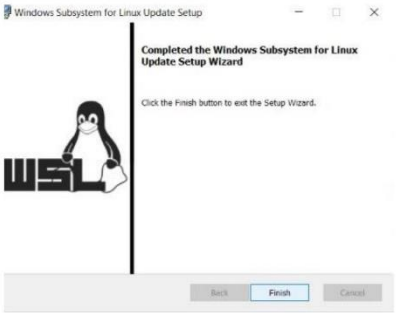
Paso 2: Continuar con la instalación del ejecutable descargado, para terminar la instalación solicita que Windows instale con el paquete WSL 2 y se reinicia la máquina.



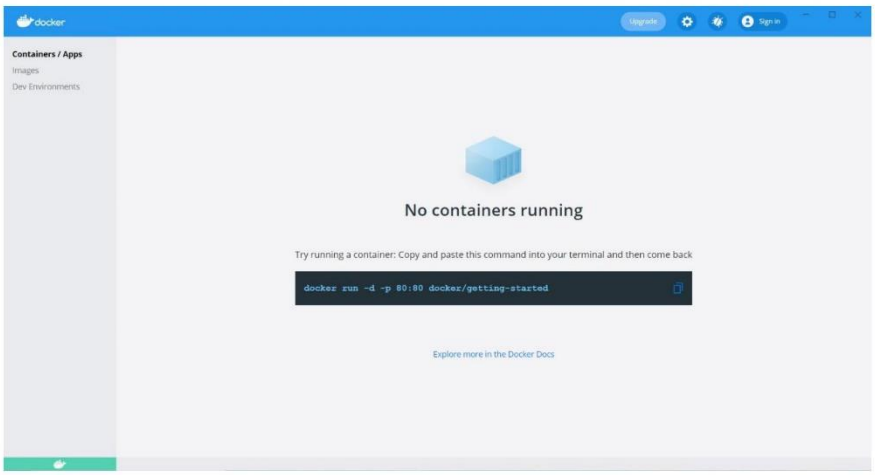
En caso de presentarse un error en la instalación de Docker Desktop como el siguiente:

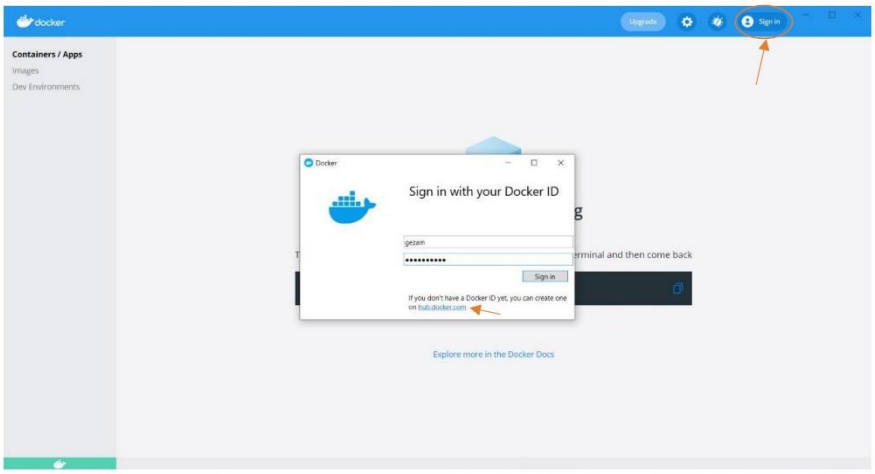
Ir al enlace de la advertencia, descargar e instalar el paquete WSL 2. Una vez instalado reiniciar la máquina.



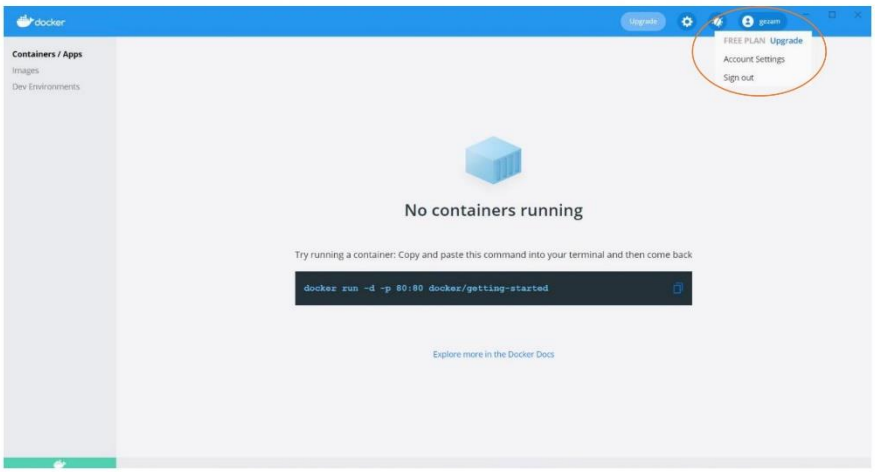


Paso 3: Una vez terminada toda la instalación se presenta la interfaz para el inicio de sesión del usuario en una cuenta de Docker Hub y para la creación de los contenedores.

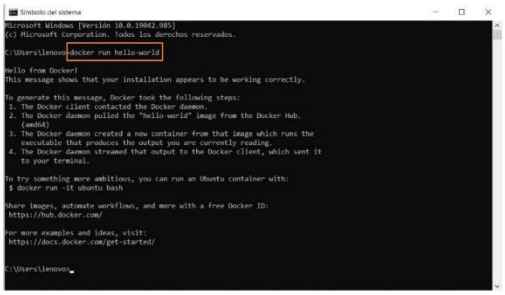


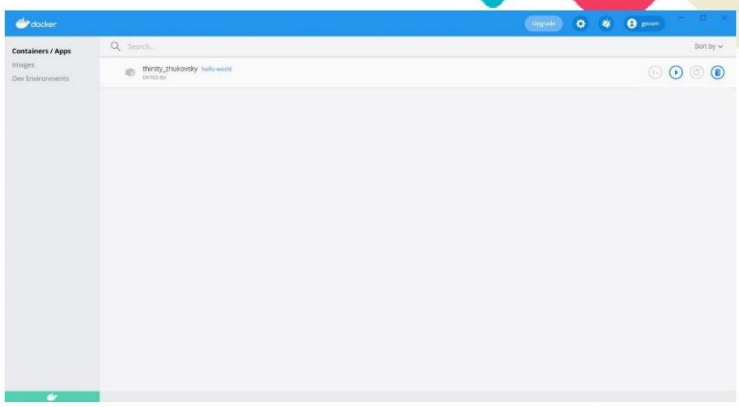


Una vez iniciada sesión, se comprueba el ingreso y ahí mismo se puede cerrar en caso que usted lo desee.



Paso 4: En el CMD verificar que se pueda acceder a las imágenes y descargarlas de Docker Hub.




docker
Upgrade
Containers / Apps
Images
Dev Environments
Search...
Sort by
thirsty_zhukovsky hello-world

¿Qué es Dockerfile?

Dockerfile es un archivo de texto plano que tiene una serie de instrucciones y permite la creación de las imágenes de Docker, luego estos archivos se convertirán en una sola aplicación utilizada para un determinado propósito.

Imagen 3: Representación del funcionamiento de Docker

Fuente: (Geekflare, 2019)

Las instrucciones que puede tener Dockerfile son las siguientes:

FROM, inicializa una nueva etapa de construcción y establece la imagen base para las instrucciones posteriores. Un Dockerfile válido debe comenzar con esta instrucción, esto se encuentra en cualquier imagen de Docker Hub extraídas de los repositorios públicos. A continuación, se presentan tres maneras de usarlo.

FROM [--platform=<platform>] <image> [AS <name>]

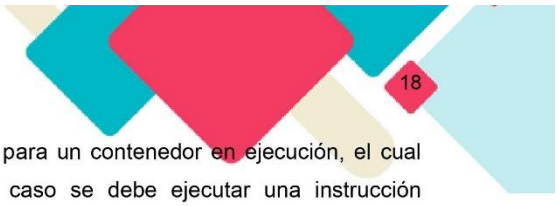
FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]

FROM [--platform=<platform>] <image>[@<digest>] [AS <name>]

RUN, ejecuta cualquier comando en una nueva capa por encima de las ya creadas, y emitirá los resultados commits, la imagen resultante se utilizará para el siguiente paso en el Dockerfile.

RUN <command>

RUN ["executable", "param1", "param2"]



CMD, proporciona valores predeterminados para un contenedor en ejecución, el cual puede incluir u omitir el ejecutable, cuyo caso se debe ejecutar una instrucción ENTRYPOINT, en el Dockerfile solo puede haber un CMD, en caso de haber más se aplicará el último.

CMD ["executable","param1","param2"](forma ejecutiva , esta es la forma preferida)

CMD ["param1","param2"](como parámetros predeterminados para ENTRYPOINT)

CMD command param1 param2 (forma de shell)

LABEL, agrega metadatos en una imagen, este es un par clave-valor, el label puede estar varias veces en una imagen. Para incluir espacios en el label use comillas y barras invertidas.

LABEL <key>=<value> <key>=<value> <key>=<value> ...

MAINTAINER, establece el campo autor de las imágenes generadas, sin embargo, la instrucción label es más flexible y debe ser usada en su lugar ya que permite configurar cualquier metadato que necesite y se puede visualizar fácilmente.

MAINTAINER <name>

EXPOSE, informa a Docker que el contenedor escucha en los puertos de redes especificados en tiempo de ejecución. Se puede especificar si el puerto escucha en TCP o UDP, en caso de no especificar el protocolo el valor predeterminado es TCP.

EXPOSE <port> [<port>/<protocol>...]

ENV, establece la variable de entorno con la estructura clave-valor, este valor estará en el entorno para todas las instrucciones posteriores en la etapa de compilación y también puede ser reemplazada; al igual que la línea de comando las comillas y las barras invertidas se pueden usar para incluir espacios dentro de los valores.

ENV <key>=<value> ...

ADD, copia nuevos archivos, directorios o URL de archivos remotos desde <src> y los agrega al sistema de archivos de la imagen en la ruta <dest>. La función –chown es compatible con Dockerfiles que se utilizan para crear contenedores de Linux y no funcionarán para contenedores de Windows.

```
ADD [--chown=<user>:<group>] <src>... <dest>

ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

COPY, esta instrucción copia nuevos archivos, directorios de <src> y los agrega al sistema de archivo del contenedor de la ruta <dest>; al igual que el comando anterior --chown solo funciona para contenedores de Linux.

```
COPY [--chown=<user>:<group>] <src>... <dest>

COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

ENTRYPOINT, permite configurar un contenedor que correrá como ejecutable.

```
ENTRYPOINT ["executable", "param1", "param2"]
```

VOLUME, crea un punto de montaje con el nombre especificado y la marca como que contiene volúmenes montados externamente desde el host nativo u otros contenedores.

```
VOLUME ["/data"]
```

En la instrucción de volúmenes se debe de tener en cuenta lo siguiente:

- En Windows: el destino de un volumen dentro del contenedor debe ser uno de los siguientes:
  - un directorio vacío o inexistente
  - una unidad que no sea C:
- Cambiar el volumen desde dentro del Dockerfile: si algún paso de compilación cambia los datos dentro del volumen después de que se haya declarado, esos cambios se descartarán.
- Formato JSON: la lista se analiza como una matriz JSON. Debe encerrar las palabras con comillas dobles (") en lugar de comillas simples (').

- El directorio del host se declara en tiempo de ejecución del contenedor: Preserva la portabilidad de la imagen, ya que no se puede garantizar que un directorio de host dado esté disponible en todos los hosts. Por esta razón, no puede montar un directorio de host desde dentro del Dockerfile. Esta instrucción no admite la especificación de un parámetro host-dir.

USER, establece el nombre de usuario y opcional el grupo de usuarios que se utilizará para la ejecución de la imagen, para cualquier instrucción RUN, CMD y ENTRYPOINT. Se debe de tener en cuenta que, al especificar un grupo de usuario, el usuario solo tendrá la membresía especificada. Para Windows, el usuario deber de crearse de primero con el comando net user.

USER <user>[:<group>]

USER <UID>[:<GID>]

WORKDIR, establece el directorio de trabajo para cualquier instrucción RUN, CMD, ENTRYPOINT, COPY y ADD que le siguen en el Dockerfile. Si el WORKDIR no existe, se creará incluso si no se utiliza en ninguna instrucción de Dockerfile.

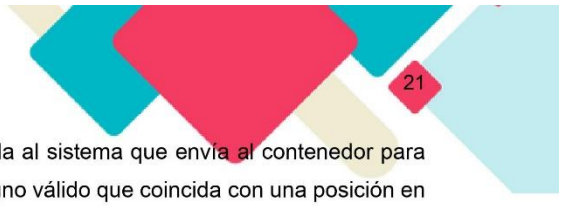
WORKDIR /path/to/workdir

ARG, define una variable que los usuarios pueden pasar en el momento de la construcción al constructor con el comando *docker build* , si un usuario especifica un argumento de compilación que no se define en el Dockerfile, este genera una advertencia. El archivo de Dockerfile puede tener uno o más argumentos.

ARG <name>[=<default value>]

ONBUILD, agrega a la imagen una instrucción de disparo a los metadatos que se ejecutará al momento posterior, cuando la imagen se use como base para otra construcción. El disparador se ejecutará en el contexto de la compilación descendente como que si se hubiera insertado inmediatamente después de la instrucción ONBUILD. Una vez ejecutados se borran los disparadores de la imagen final.

ONBUILD <INSTRUCTION>



STOPSIGNAL, establece la señal de llamada al sistema que envía al contenedor para salir. Esta señal puede ser un número sin signo válido que coincida con una posición en la tabla de llamadas al sistema del kernel.

STOPSIGNAL signal

HEALTHCHECK, verifica que un contenedor esté funcionando correctamente, si encuentra que un servidor web este atascado en el bucle infinito este lo puede detectar.

HEALTHCHECK [OPTIONS] CMD command (verifique el estado del contenedor)

SHELL, permite anular el núcleo predeterminado, también afecta las instrucciones anteriores y posteriores, se escribe en formato JSON en un Dockerfile. Para Linux se usa ["/bin/sh", "-c"] y en Windows ["cmd", "/S", "/C"]; particularmente es usada en Windows, donde existe dos Shell nativos *cmd y powershell*.

SHELL ["executable", "parameters"]

Creando imágenes con Docker

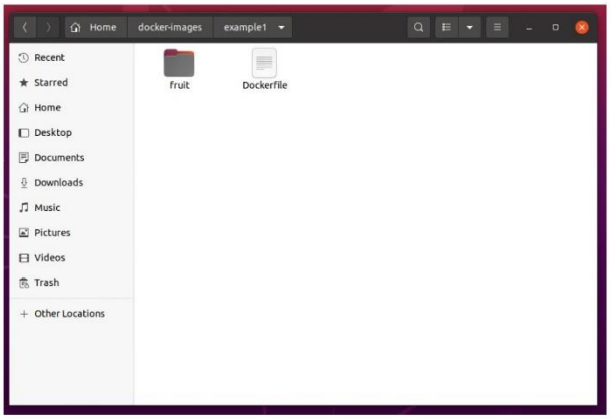
Para la creación de los escenarios, primero se realizó la configuración física de la máquina virtual y esta tuvo las siguientes características:

- CPU 1/8 de los que estaban disponibles para la creación de la máquina.
- 2GB de Memoria RAM.
- 15GB de Disco duro.
- Conexión a Red de tipo NAT(Network Address Translation).

Ejemplo 1: Creando imágenes de Aplicación Web

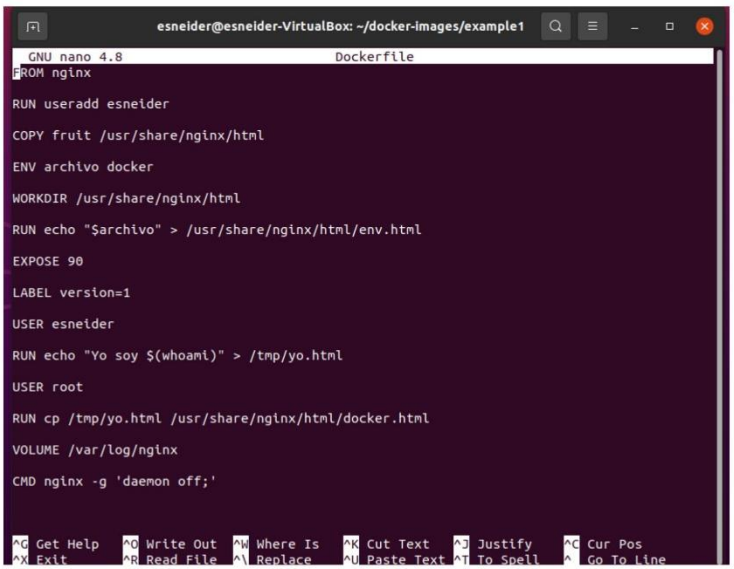
Crear imágenes en Dockerfile con algunas instrucciones, no siempre es necesario utilizar todas las instrucciones en un Dockerfile solo las necesarias.

En este caso se agregarán todas como práctica, *FROM nginx,* que actuará como servidor web; *RUN useradd <NombredeUsuario>*, aquí se agregará un usuario; *COPY fruit /usr/share/nginx/html,* copiará una plantilla de internet que estaba guardada en la carpeta fruit como la que se muestra a continuación.

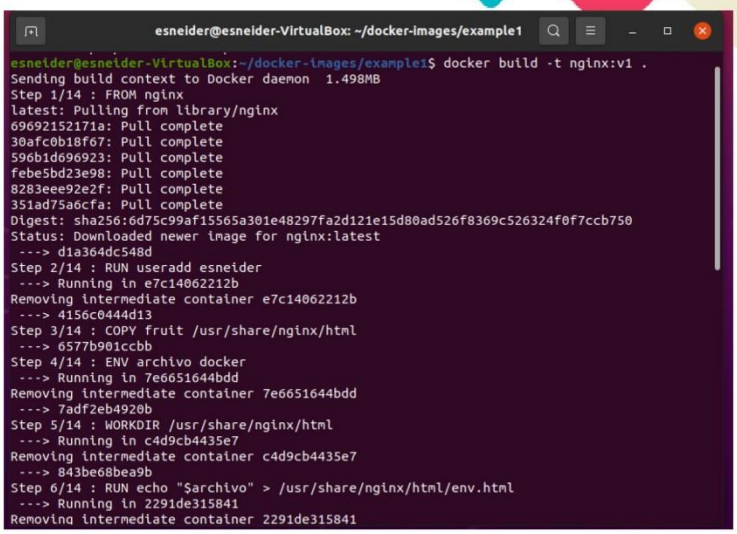


Después el comando ENV *archivo docker,* aquí se creará una variable de entorno y contendrá el texto “docker”; *WORKDIR </usr/share/nginx/html>,* se definirá el directorio de la carpeta raíz; *RUN echo <”$archivo” > /usr/share/nginx/html/env.html>,* enviará la

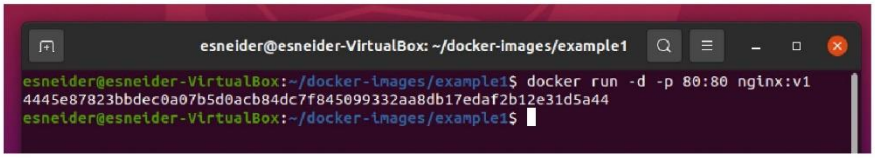
variable de entorno *archivo* al directorio; *EXPOSE 90,* expondrá el puerto 90; *LABEL version=1,* determina el nombre de una etiqueta; *USER <nombreusuario>,* indica que el usuario ejecutará una tarea con *RUN echo "Yo soy $(whoami)" > /tmp/yo.html,* define que usuario está en ejecución y lo muestra en *yo.html*; *USER root;* luego se utiliza el usuario root y se ejecuta una copia de la carpeta anterior y la lleva hasta la carpeta raíz *html* con el siguiente comando *RUN cp /tmp/yo.html /usr/share/nginx/html/docker.html; VOLUME /var/log/nginx,* guarda logs de nginx*; CMD nginx -g 'daemon odd;',* se utiliza para terminar la creación del Dockerfile, también se puede dejar el CMD por defecto.



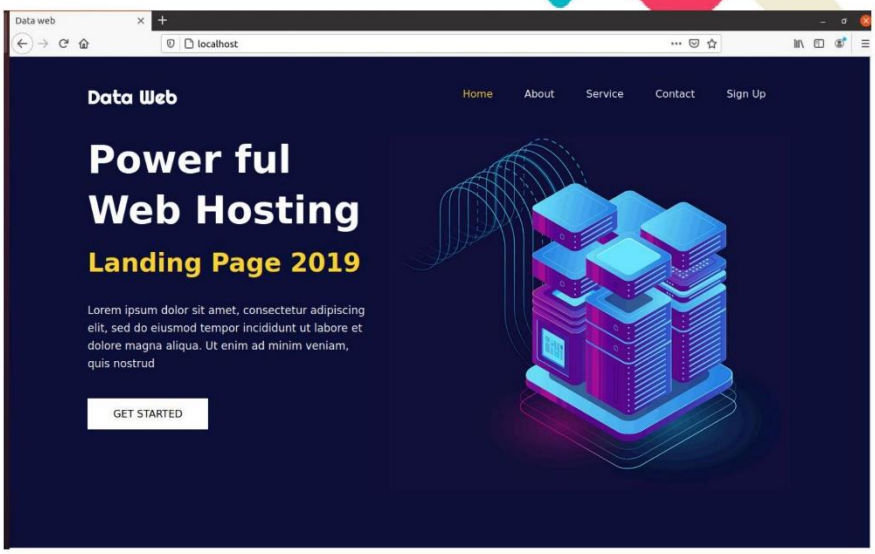
Luego se construye la imagen con con *docker build -t nginx:v1 .*; de esta manera copia todas las instrucciones que se detallaron con anterioridad.



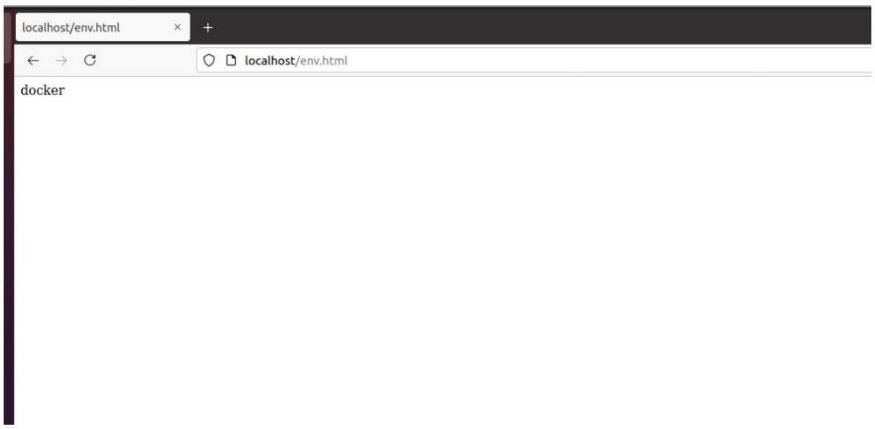
Después se crea un contenedor con el comando *docker run -d -p 80:80 nginx:v1* ; como resultado entrega el ID del contenedor.

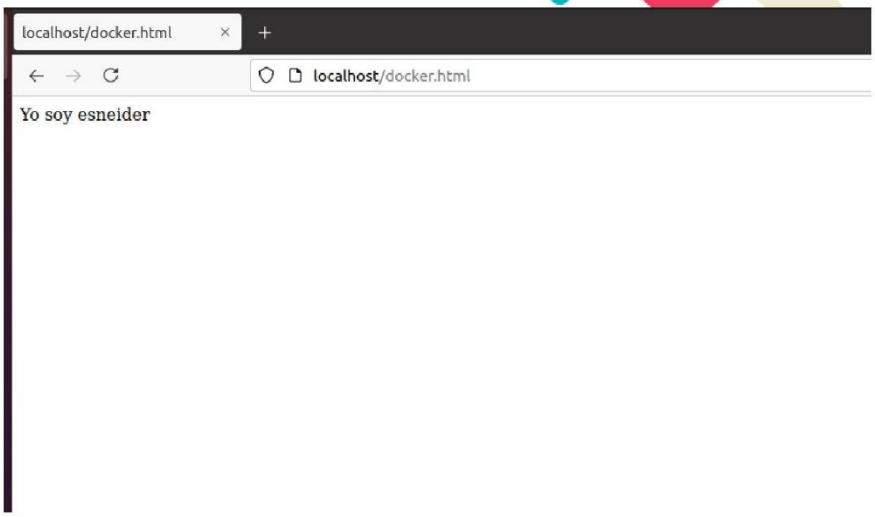


Luego en el navegador, se escribe localhost donde se encuentra copiada la plantilla copiada en los pasos anteriores.



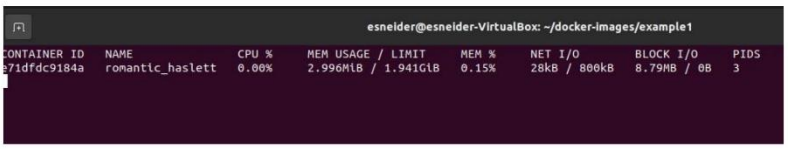
También se pueden comprobar las url creadas como lo fue el env.html donde se encuentra el texto docker y el docker.html donde se encuentra Yo soy <nombre de usuario>; con esta práctica se pudo comprobar cómo se tiene un servicio corriendo en un contenedor





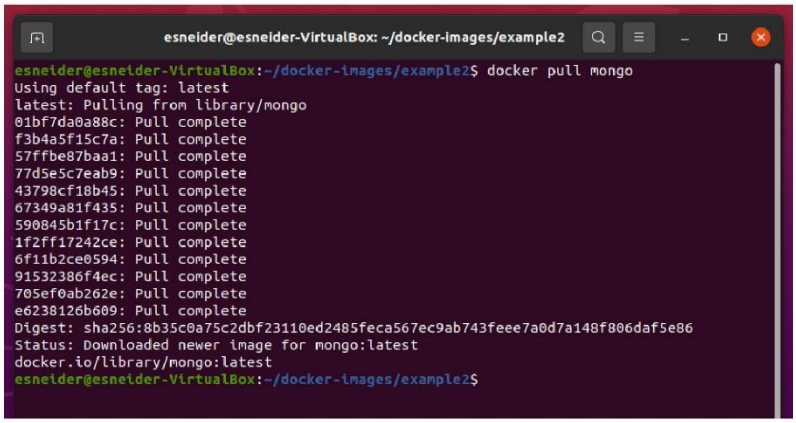
Coste Computacional del escenario de la aplicación web

El coste computacional en este escenario se pudo observar que el consumo de la CPU fue 0.00%, ya que las tareas que se estaban realizando en el contenedor no requerían de mucho procesamiento en la CPU; en el caso de la memoria se utilizó 2.996 MB lo que equivale al 0.15% de la memoria establecida en el host; y en la parte final se visualizó la cantidad de procesos o subprocesos que se ejecutaban en el contenedor que en este caso fueron 3.

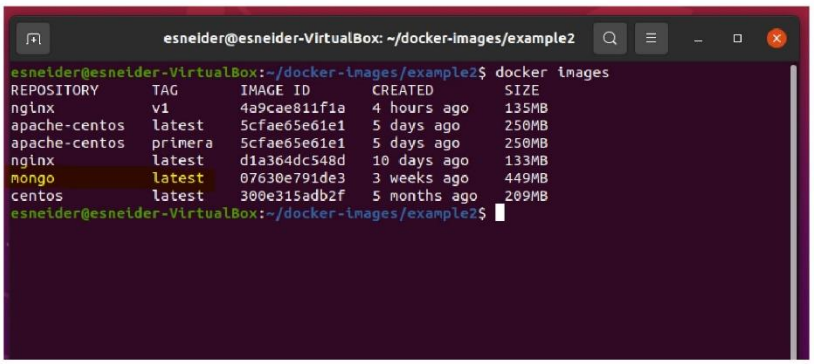


Ejemplo 2: Creando imágenes de Docker para Mongodb

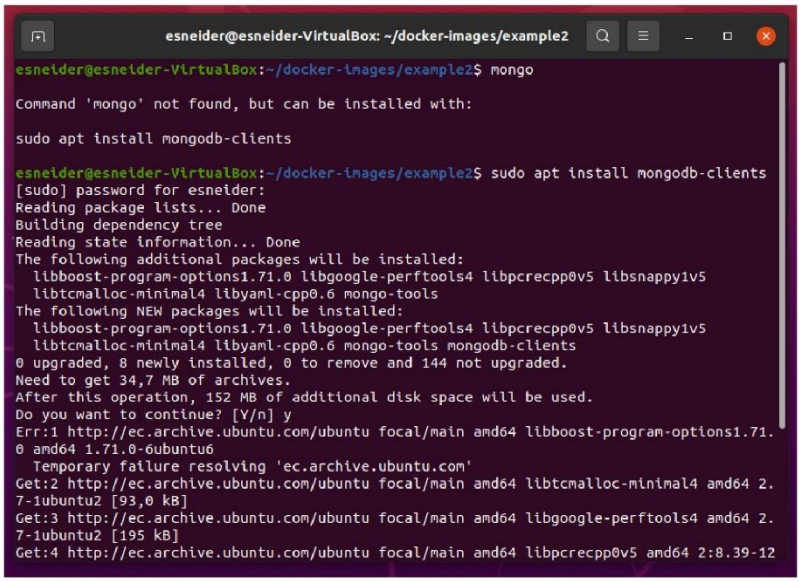
Lo primero que se debe hacer es abrir el terminal de Docker, una vez estando ahí se procede a descargar de internet la imagen de mongodb con el siguiente comando *docker pull mongo* una vez descargado se podrán generar múltiples instancias de una base de datos.



Luego de la descarga se comprueba la descarga y a través del comando *Docker images* enlista las imágenes que tiene en el computador, en la siguiente captura de pantalla se puede observar que existe una sola imagen llamada *mongo* con sus características.



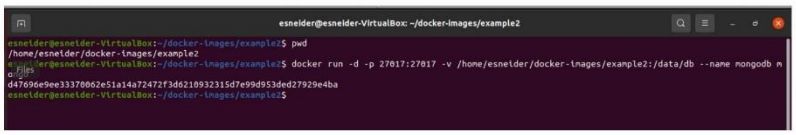
Luego, se instala un cliente, que le permita conectarse a la base de datos mongodb con el comando *sudo apt install mongodb-clients*, al encontrarse en modo root solicitará la contraseña de la máquina y una vez autentificada este comando se ejecutará. Esto descargará el shell de mongo, para conectarse al contenedor.



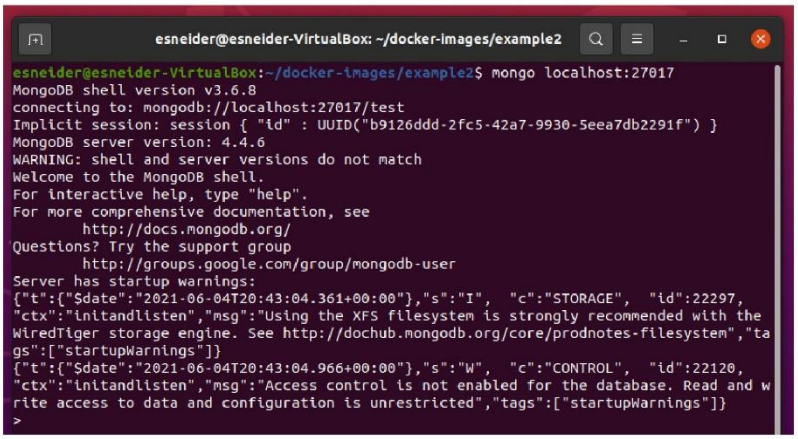
Una vez terminada la ejecución anterior, se procede a comprobar el funcionamiento del mismo con el comando *mongo,* y este imprimirá en pantalla las caracteristicas del cliente descargado.

Luego se verifica la ubicación de la carperta con el comando *pwd,* en esta carperta se guardarán los datos de la base de datos, ya que esta carpeta se conecterá a la carperta que está en el contenedor que tiene las base de datos .

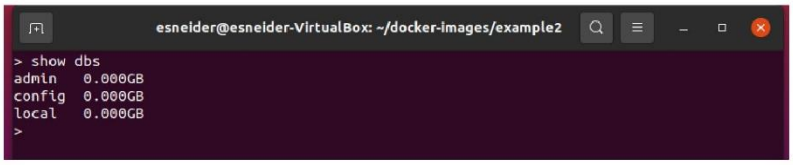
A través del comando *docker run -d -p 27017:27017 -v <ubicacióndelacaperta>:/data/db --name <nombre> mongo*, este permite conectarse al servidor mostrando el ID del proceso; este comando está combinado con varios comandos de acceso como el -d significa que se ejecutará en segundo plano; -p 27017:27017 permite establecer la conexión entre el puerto de la máquina y el contenedor; -v establece la conexión de las carpetas; --name indica el nombre que tendrá el contenedor.



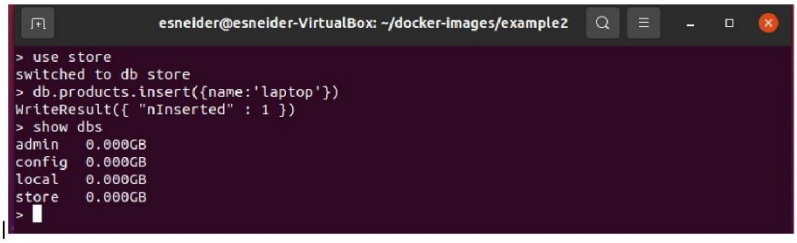
Haciendo uso del Mongo Clients, se accede al contenedor en el puerto 27017 el cual se detalló en el comando anterior referente al puerto expuesto, imprimiendo en pantalla las caracteristicas de la conexión del contenedor.



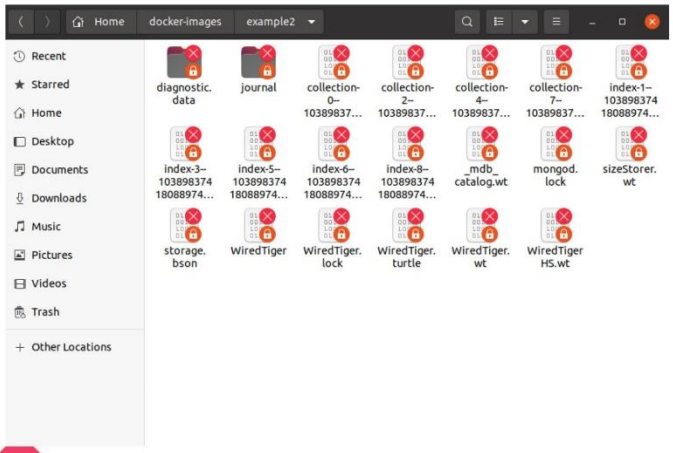
Con el comando *show dbs*, se muestra los nombres de la base de datos por defecto.



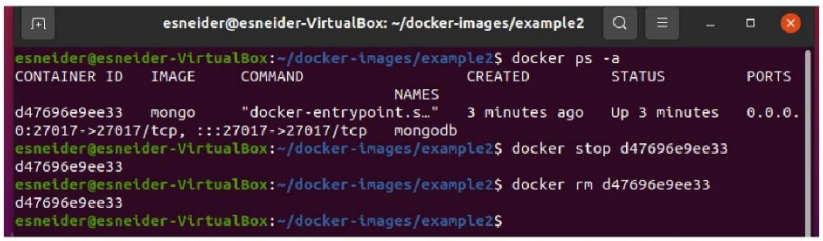
Con el comando *use store*, crea y usa la base de datos store, para luego crear una colección llamada pruducto e insertar un nuevo registro y como paso final se muestra nuevamente la base de datos y se comprueba que su creación se realizó con éxito.



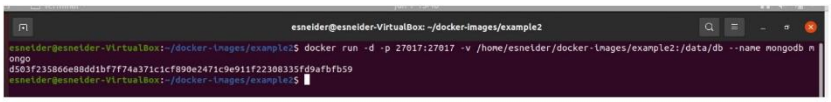
Cabe mencionar que si es paraliza el contenedor estos datos vuelven a cero, por eso, es importante crear una copia de los datos almacenados con el comando *-v* al correr el contenedor.



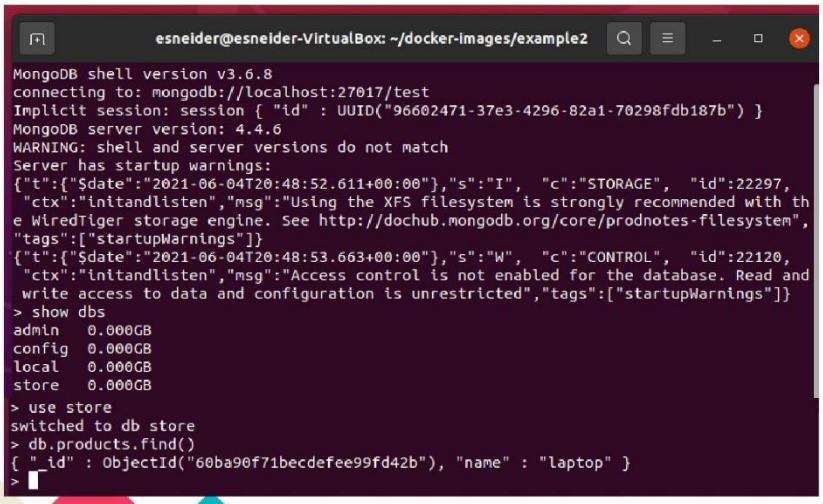
Una vez se haya verificado que se realizó la copia de los datos se puede detener el contenedor y eliminarlo con los comandos *docker stop* y *docker rm* seguido del ID del contenedor.



Para verificar que la información se ha guardado y se puede reutizar, se corre un nuevo contenedor con los comandos indicados anteriormente.

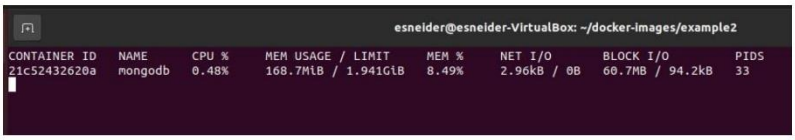


Para comprobar que la información ha persistido correctamente se ingresa al cliente de mongo y se puede mostrar la base de datos que se creó y el registro ingresado anteriormente.



Coste Computacional del escenario de mongodb

El coste computacional de este escenario virtualizado se observó que el consumo de la CPU fue 0.48%; en el caso de la memoria se utilizó 168.7 MB lo que equivale al 8.49% de la memoria establecida en el host; y en la parte final se visualizó la cantidad de procesos o subprocesos que se ejecutaban en el contenedor que en este caso fueron 33.



Ventajas de Docker

Docker como contenedor de aplicaciones es muy importante para el ámbito educativo y empresarial, y esté es aplicable a varias áreas del conocimiento; a continuación, se presentan algunas ventajas de esta herramienta de virtualización.

- Se logra la optimización del rendimiento del computador ya que varios contenedores con diferentes tareas se pueden ejecutar sobre un sistema operativo común, lo cual se ve reflejado en el mantenimiento y actualizaciones de los mismos; conjuntamente con el espacio de almacenamiento y las memorias ocupadas por cada sistema operativo. La minimización del coste computacional generado por la arquitectura que emplea Docker, permite que ese poder computacional se lo utilice en otras aplicaciones o contenedores para seguir expandiendo la estructura de los servicios a futuro. El coste computacional de Docker en los escenarios planteados se puede observar con mayor detalle en la página 26 y 32.
- Otra ventaja es el factor económico, ya que Docker al requerir menos poder computacional se puede implementar más cosas con computadoras no tan costosas, además de las licencias en caso de ser necesarias sólo se requerirían en un solo sistema operativo ya que este es compartido.
- Docker empaqueta software en unidades estandarizadas llamadas contenedores (ver “Estructura de Docker, página 4”), mismos que incluyen todo lo que se necesita para que el software desarrollado se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Proporciona una manera estándar de ejecutar su código estandarizando las operaciones de las aplicaciones lo cual facilita la implementación, la identificación de problemas y el retorno a una fase anterior para remediarlos.
- Docker permite la creación de una imagen de contenedor a través de la configuración del Dockerfile (ver el apartado de “*¿Qué es Dockerfile?*”, página 17) y usarla a lo largo de todo el proceso de despliegue, teniendo la capacidad de separar pasos no dependientes del proceso y ejecutarlos en paralelo. Como

resultado, el tiempo que toma desde la compilación a la producción se ve reducido.

- Los contenedores con su facilidad de despliegue ayudan a que los procesos no afecten a otros contenedores al momento de su ejecución. Además, con la configuración correcta de la imagen se puede tener resultados favorables en pocos segundos. Cabe mencionar, que cuando existan contenedores que no se necesiten, estos pueden eliminarse fácilmente.
- La portabilidad se realiza a través del modelo en imágenes, que facilita la compartición de servicios a través de múltiples entornos que para su ejecución tengan instalado Docker, para ello se puede revisar el apartado *"¿Qué es una imagen?"* de la página 4 de la Guía. Cabe mencionar, que los contenedores para la implementación sobre cualquier sistema, incorporan todos los archivos y dependencias de los servicios, evitando realizar de nuevo la configuración para su funcionamiento. Por su portabilidad Docker también es considerado como una Plataforma Multi-nube, donde grandes servidores del mercado proveen su disponibilidad.
- Desde su instalación Docker brinda la simplicidad de configuración a sus usuarios, y de la misma manera en la creación de contenedores; los desarrolladores pueden utilizar su propia configuración o códigos que finalmente son desplegados para su funcionamiento sin ningún problema (ver el apartado de "¿Cómo instalar Docker?", página 5).
- La escabilidad de Docker se realiza tanto vertical como horizontal, facilitando su funcionamiento en los diferentes entornos de producción; esto se realiza conjuntamente con la portabilidad. Esta ventaja aporta también en los DevOps ya que ofrece servidores de integración continua e infraestructuras escalables.

Desventajas de Docker

- Se requiere mínimo la versión de Kernel 3.10, debido que las versiones anteriores a la 3.10 no son compatibles con las nuevas características de Docker, ya que son implementadas con base a las actualizaciones del kernel de Linux.
- Inicialmente fue diseñada para entornos Linux, mientras que para clientes Windows aún está en desarrollo (ver apartado de "*Instalación en Windows*", página 11).
- Algunas versiones de Docker dan error debido a que se encuentran en constante desarrollo, esto se pudo observar a través de la instalación de Docker en Windows realizada por los autores de esta guía, en la que surgieron varios errores de WSL (Windows Subsystem for Linux) que se pudieron solventar. Por esta razón, se recomienda usar Docker en Linux.
- Solo soporta a sistemas operativos Linux de arquitectura de 64 bits.
- El kernel debe admitir cgroups y espacios de nombres.
- Tiene una curva de aprendizaje alta, debido que en las características encontradas en la investigación Docker abarca temáticas muy amplias para el aprendizaje, como la gestión de proyectos de desarrollo, control de versiones y depuración, la portabilidad y manejo de seguridad de los contenedores y otros.

Bibliografías

Geekflare. (2019, July 30). *¿Qué es Dockerfile y cómo crear una imagen de Docker?* https://geekflare.com/es/dockerfile-tutorial/

Todea, V. C. (2016). *Diseño e implementación de un sistema de entrega continua para aplicaciones web sobre contenedores Docker.* https://riunet.upv.es/bitstream/handle/10251/71386/TODEA - Diseño e implementación de un sistema de entrega continua para aplicaciones web sobre con....pdf?sequence=2&isAllowed=y

Geekflare. (2019, July 30). *¿Qué es Dockerfile y cómo crear una imagen de Docker?* https://geekflare.com/es/dockerfile-tutorial/

Todea, V. C. (2016). *Diseño e implementación de un sistema de entrega continua para aplicaciones web sobre contenedores Docker.* https://riunet.upv.es/bitstream/handle/10251/71386/TODEA - Diseño e implementación de un sistema de entrega continua para aplicaciones web sobre con....pdf?sequence=2&isAllowed=y