Tipo de artículo: Artículo original

Análisis de factibilidad tecnológica: operativa de integración de switch SDN y servidor Temonet Edge para proyecto Temonet

Technological feasibility analysis: operation of integration of SDN switch and Temonet Edge server for Temonet project

Jenny Arizaga Gamboa[1*], **https://orcid.org/0000-0002-2098-9077**
Eduardo Alvarado Unamuno [2], **https://orcid.org/0000-0001-6145-7926**
Jorge Chicala Arroyave [3], **https://orcid.org/0000-0001-9630-2377**

[1] Universidad de Guayaquil Guayaquil, Ecuador. E-mail: ***jenny.arizagag@ug.edu.ec***
[2] Universidad de Guayaquil Guayaquil, Ecuador. E-mail: ***Eduardo. alvaradou@ug.edu.ec***
[3] Universidad de Guayaquil Guayaquil, Ecuador. E-mail: ***jorge.chicalaa@ug.edu.ec***

*** Autor para correspondencia:** ***jenny.arizagag@ug.edu.ec***

Resumen

El presente proyecto de tiene como objetivo general, analizar la factibilidad tecnológica/operativa de integración de Switch SDN y Servidor TEMONET EDGE para el proyecto TEMONET. La integración de sistemas ofrece numerosas ventajas como la interacción de todos los usuarios al poder acceder a datos en tiempo real desde cualquier dispositivo que mantenga conexión a Internet de forma estable. Al usar contenedores se brinda la ventaja del aislamiento, portabilidad, agilidad, escalabilidad y un mayor control a lo largo de todo el flujo de trabajo incluido en el proceso de integración expuesto en esta tesis y dando como resultado el correcto funcionamiento de las pruebas y conexiones de la integración realizada.

Palabras clave: Análsis de factibilidad; SDN; Temonet; Feasibility analysis; SDN; Temonet.

Abstract

The general objective of this project is to analyze the technological / operational feasibility of integration of Switch SDN and TEMONET EDGE Server for project TEMONET. System integration offers numerous advantages such as the interaction of all users by being able to access data in real time from any device that maintains a stable Internet connection. Using containers provides the advantage of isolation, portability, agility, scalability, and greater control throughout the entire workflow included in the integration process exposed in this thesis and resulting in the correct operation of tests and connections. of the integration carried out.

Keywords: Feasibility analysis; SDN; Temonet.

Recibido: 10/06/2021
Aceptado: 22/10/2021

Introducción

Uno de los problemas que hoy en día afecta al aprendizaje de los niños con dislexia es que necesitan que se les enseñe a leer de una manera específica y comprensible, debido a esto existen varios programas informáticos de lectura en internet diseñados para poder ayudarlos en su desarrollo educativo, estas instrucciones de educación especial se las realiza sin necesidad de asistir de forma presencial a un centro de estudio, como parte de la educación especial o como una intervención educativa especializada en problemas de dislexia (Chi et al., 2015).

El nombre oficial de este tipo de educación es Educación de Lenguaje Estructurado Multisensorial (MSLE), y muchos expertos consideran que es el estándar de oro para educar a niños con dislexia.

Debido a todas estas necesidades de comunicación se pone en consideración el tema Integración de Sistemas tecnológicos el cual tiene como propósito establecer la colaboración entre aplicaciones con alcances y funcionalidades específicas obteniendo un resultado eficiente en los procesos que normalmente se realizarían en forma manual/operativa o semiautomatizado por una persona.

La propuesta de integración entre sistemas tiene diversos alcances los cuales van desde el hecho de intercambiar información entre ellos como el de optimizar la infraestructura.

Materiales y métodos

Debido a que la dislexia es un problema del lenguaje, el problema a menudo comienza con la conciencia fonológica, habilidades del lenguaje que son esenciales para la lectura. Los profesionales que brindan estas formas de apoyo incluyen psicólogos, maestros y especialistas en lectura que se enfocan en problemas de aprendizaje, especialistas en aprendizaje y patólogos del habla y lenguaje.

En la Universidad de Guayaquil se está desarrollando el proyecto FCI TEMONET, el cual consta en la creación de una plataforma digital para obtener terapias médicas destinadas a tratar a pacientes con enfermedades de tipo cognitivo como la dislexia. Con la necesidad que ahora los padres desean llevar a sus hijos con un especialista para ayudarles a tratar y mejorar sus problemas de dislexia, el proyecto TEMONET está realizando los estudios necesarios para mejorar la infraestructura tecnológica y brindar un mejor servicio, por lo que se necesitaba tener el Switch SDN y el Servidor TEMONET EDGE integrados en un solo dispositivo de bajo costo como parte de la Plataforma TEMONET la cual ayuda a los infantes a crecer en su desarrollo integral todos los días (Firestone, 2017).

INTEGRACIÓN DE SISTEMAS

Integrar significa que un sistema o conjunto de elementos forme parte de otro y, en términos de gestión de los sistemas de información, significa que:

- Los sistemas están coordinados.
- Los distintos recursos de hardware no sean incompatibles entre los sistemas.
- Un mismo dato sea accesible por distintos sistemas.
- Los sistemas se entienden entre sí al nivel de software.
- Estén adecuadamente comunicados.
- Los distintos sistemas pueden crecer al unísono, sin perder cohesión ni consistencia. (Heredero, 2008)

LA INTEGRACIÓN DE DATOS

Es el proceso que permite combinar datos heterogéneos de muchas fuentes diferentes en la forma y estructura de una única aplicación.

Esto facilita que diferentes tipos de información, tales como matrices de datos, documentos y tablas, sean fusionados por usuarios, organizaciones y aplicaciones para un uso personal, de procesos de negocio o de funciones.

La integración soporta el procesamiento analítico de grandes conjuntos de datos alineando, combinando y presentando cada conjunto de informaciones de departamentos organizacionales y fuentes de datos remotas y externas, para cumplir con los objetivos del integrador.

VENTAJAS DE LA INTEGRACIÓN DE SISTEMAS

La integración ofrece las siguientes ventajas:

- Todos los usuarios pueden acceder a datos personales en tiempo real desde cualquier dispositivo con conexión a Internet.
- Se puede utilizar la misma información de inicio de sesión para todas las aplicaciones personales.
- El sistema pasa eficientemente los mensajes de control entre las aplicaciones.
- Se evita el uso de silos de datos, se mantiene la integridad de la información y se eliminan conflictos de datos que pueden surgir de la redundancia.
- La integración de datos ofrece escalabilidad para permitir una expansión futura en términos de número de usuarios y de aplicaciones.

• Reducir costes, entre otros, los derivados del mantenimiento interno de los sistemas y de su evaluación externa y certificación. (AEC, 2016)

Figura 1: Raspberry Pi

La Raspberry Pi es una computadora de bajo costo y con un tamaño compacto y puede ser conectada a un monitor de computador, y usarse con un mouse y teclado estándar. Es un pequeño computador que corre un sistema operativo Linux capaz de permitirles a las personas de todas las edades explorar la computación y aprender a programar lenguajes como Scratch y Python. Es capaz de hacer la mayoría de las tareas típicas de un computador de escritorio, desde navegar en internet, reproducir videos, manipular documentos de ofimática. (CL, s.f.)

Raspberry Pi Operating System es el sistema operativo recomendado para el uso normal en una Raspberry Pi.

Raspberry Pi OS es un sistema operativo gratuito basado en Debian, optimizado para el hardware Raspberry Pi. El sistema operativo Raspberry Pi viene con más de 35,000 paquetes: software precompilado incluido en un formato agradable para una fácil instalación en su Raspberry Pi.

Raspberry Pi OS es un proyecto comunitario en desarrollo activo, con énfasis en mejorar la estabilidad y el rendimiento de tantos paquetes Debian como sea posible.

Un Dockerfile se refiere a un archivo de texto el cual no tiene formato que contiene una serie de instrucciones las cuales son necesarias para crear una imagen que se convertirá en una sola aplicación utilizada para un propósito específico.

La generación de imágenes la realiza el demonio Docker. Es importante tener en cuenta que las compilaciones de Docker envían todos los contextos del directorio actual al demonio (Wang et al., 2017).

El punto importante es que cada comando se ejecuta con una nueva imagen.

Docker Image se refiere a plantillas que se usan para poder elaborar contenedores Docker y a su vez ejecutarlos . Los contenedores se inician ejecutando una imagen. Se puede definir a una imagen como una plantilla, una porción del estado de un contenedor.

Portainer es una herramienta web open-source la cual se ejecuta ella misma como un container, por tanto, deberemos tener Docker instalado. En esencia, Portainer ayuda a los desarrolladores a implementar aplicaciones nativas de la nube en contenedores de manera simple, rápida y segura.

Para usuarios avanzados, Portainer incorpora una API que le permite conectarse a herramientas CI / CD o paneles de control / herramientas de implementación de terceros.

La herramienta es compatible con el motor Docker y con Docker Swarm.

Open vSwitch es un switch virtual multicapa con licencia de código abierto Apache 2.0 . Además, está diseñado para admitir la distribución a través de múltiples servidores físicos, similar al conmutador virtual distribuido vNetwork de VMware o al Nexus 1000V de Cisco.

Open vSwitch está bien adaptado para funcionar como un conmutador virtual en entornos de VM. Además de exponer las interfaces de visibilidad y control estándar a la capa de red virtual, se diseñó para admitir la distribución en varios servidores físicos.

RYU es un controlador de redes definidas por software, provee componentes de software e interfaces API – application program interfaces- que facilitan el desarrollo de la administración y control de la SDN. El controlador Ryu es una infraestructura que proporciona componentes de software con una API bien definida que facilita a los desarrolladores la creación de nuevas aplicaciones de administración y control de redes. Esto facilita a los desarrolladores la creación de nuevas aplicaciones de gestión y control de red. Este controlador está desarrollado en Python.

Nginx, pronunciado como “engine-ex”, es un servidor web de código abierto que, desde su éxito inicial como servidor web, ahora también es usado como proxy inverso, cache de HTTP, y balanceador de carga. (Inc. K. , 2021)

Los contenedores son entornos ligeros de tiempo de ejecución que proporcionan a las aplicaciones los archivos, las variables y las bibliotecas que necesitan para ejecutarse, maximizando de esta forma su portabilidad.

Si bien las máquinas virtuales tradicionales permiten la virtualización de la infraestructura de computación, los contenedores habilitan la de las aplicaciones de software.

Dado que no incluyen sistemas operativos completos, los contenedores requieren recursos informáticos mínimos siendo rápidos y fáciles de instalar. Esta eficiencia permite que se implementen en clústeres, con contenedores individuales que encapsulan componentes únicos de aplicaciones complejas (Cello et al., 2017).

Separar los componentes de la aplicación en diferentes contenedores permite a los desarrolladores actualizar componentes individuales sin necesidad de rehacer la aplicación completa (LP, 2021).

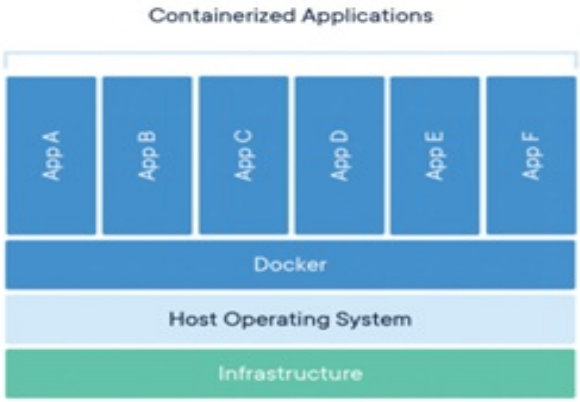


Figura 2: Esquema de aplicación de los contenedores

Docker es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará. (Docker, Docker Inc., 2021)

Docker le proporciona una manera estándar de ejecutar su código. Docker es un sistema operativo para contenedores. De manera similar a cómo virtualiza una máquina virtual, eliminando la necesidad de administrar directamente el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor. Docker se instala en cada servidor y proporciona comandos sencillos que puede utilizar para crear, iniciar o detener contenedores.

Al ejecutarse un contenedor, utiliza un sistema de archivos aislados personalizados los proporciona una imagen de contenedor. Dado que la imagen contiene el sistema de archivos del contenedor, también debe contener todo lo necesario para ejecutar una aplicación: dependencias, configuración, scripts, binarios, variables de entorno, un comando predeterminado para ejecutar, metadatos, etc. (Docker, Docker Inc., 2021)

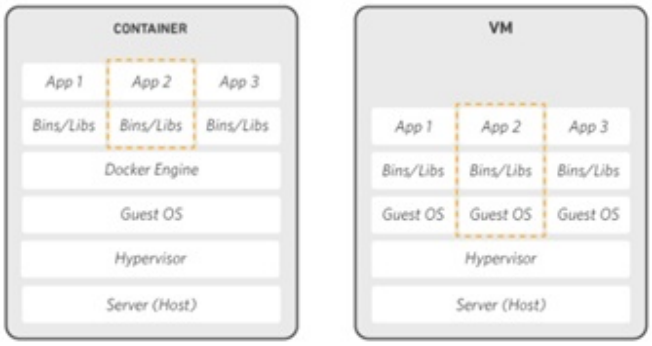


Figura 3: Docker

Docker le permite entregar código con mayor rapidez, estandarizar las operaciones de las aplicaciones, transferir el código con facilidad y ahorrar dinero al mejorar el uso de recursos. Con Docker, obtiene un solo objeto que se puede ejecutar de manera fiable en cualquier lugar. La sintaxis sencilla y simple de Docker le aporta un control absoluto. La amplia adopción significa que existe un gran ecosistema de herramientas y aplicaciones listas para su uso que puede utilizar con Docker. (Amazon Web Services, 2021)

Podemos empezar teniendo en cuenta que los contenedores, el hecho de que no es necesario de que tengan un sistema operativo completo, sino que estos reutilicen el implícito ayuda a la reducción de la carga que debe soportar la máquina física que lo aloja, el espacio de almacenamiento el cual se será utilizado y el tiempo que se necesitara para lanzar las aplicaciones. Es por lo que los contenedores son mucho más ligeros que las máquinas virtuales.

Por el lado de los contenedores esto se omite incluso no es necesario indicar qué recursos se va a necesitar, sino que es Docker Engine, en sus funciones de las necesidades de cada momento, el cual se encarga de asignar lo que sea necesario para que los contenedores tengan el correcto funcionamiento.

Mientras que una sola máquina virtual puede tardar un minuto o más en arrancar y esperar tener disponible nuestra aplicación, un contenedor Docker se levanta y responde en unos pocos segundos. El espacio ocupado en disco es muy inferior con Docker al no necesitar que sea necesario instalarle el sistema operativo completo para que pueda funcionar.

Cada tecnología tiene sus aplicaciones y sus ventajas según las necesidades y circunstancias de cada desarrollo.

Resultados y discusión

Un aspecto importante dentro del presente proyecto es la optimización del hardware dentro de la solución de borde donde se utiliza dos dispositivos un–Switch SDN y el Servidor TEMONET EDGE, por lo que se propone un análisis

de factibilidad y su implementación, dado el caso, utilizando elementos de virtualización o contenedores para poder integrar los dos dispositivos en uno solo utilizando un dispositivo de bajo costo.

La presente propuesta tecnológica busca integrar el Switch SDN y el Servidor Temonet Edge en un dispositivo de bajo costo. El funcionamiento del modelo actual va de acuerdo con los siguientes diagramas:

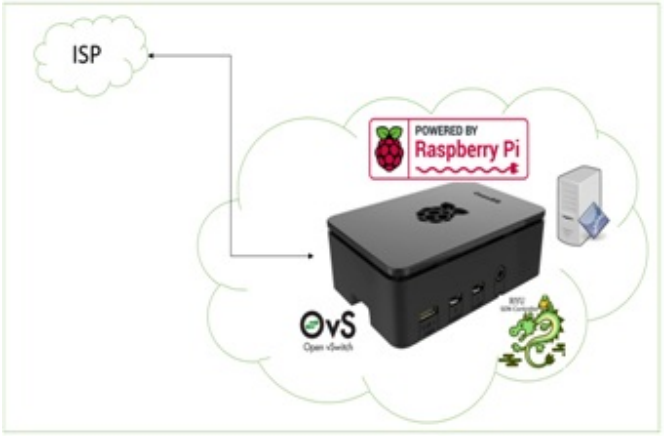


Figura 4: Arquitectura propuesta de la tecnología.

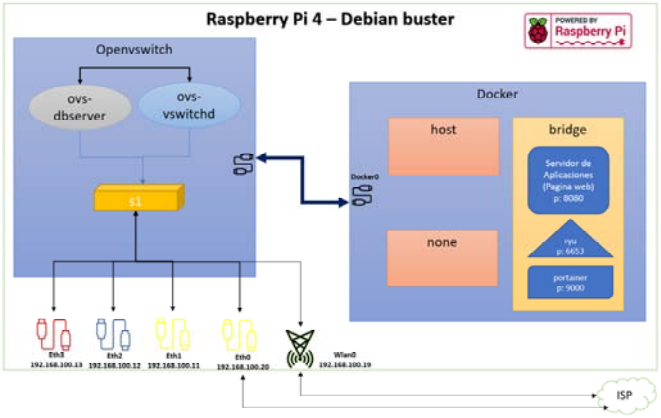


Ilustración 1 Esquema interno de la Raspberry Pi 4

Figura 5: Esquema interno de la Raspberry Pi4

FACTIBILIDAD TÉCNICA

El grafico anterior muestra como están conectados los componentes dentro del dispositivo. Elegimos al Raspberry Pi 4 que es un ordenador de placa reducida de bajo costo y cuyas funcionalidades descritas en el capítulo dos, indican que puede operar con sistemas operativos y programas open source sin problemas.

Los partes que componen el Switch SDN son el switch virtual OpenvSwitch y el controlador Ryu, y por el Servidor Temonet Edge realizamos una analogía de servidor web con una página simple que verifique la transaccionalidad del modelo prototipado.

El programa OpenvSwitch fue instalado directamente en el sistema operativo del Raspberry Pi 4 y los demás componentes los cuales son Controlador Ryu, Portainer y página web dentro de un servidor, fueron instalados dentro de contenedores de Docker (elementos dockerizados).

Todos los contenedores se comunican dentro de la red bridge de Docker y ésta se comunica con el OpenvSwitch del Raspberry, quien es el que permite o niega la comunicación hacia la red Docker desde las interfaces por la que accede el Cliente. El controlador Ryu, es también un contenedor dentro de la mencionada red Docker (Lazaris et al., 2014).

En el prototipo se siguió el siguiente esquema para la operatividad:

1. Instalación del sistema operativo Raspberry pi OS Debian búster en la memoria SD.
2. Instalación de la paquetería Docker y configuración para el manejo de los comandos Docker sin necesidad de ser usuario root. Para observar la implementación ir al Anexo 2, Manual de Instalación de Docker.
3. Instalación de la imagen de Portainer para tener una interfaz gráfica, amigable para la administración de los servicios Docker. Para observar la implementación ir al Anexo 3, Manual de Instalación de Portainer.
4. Instalación, configuración del OpenvSwitch y creación del switch virtual y enlaces con las interfaces físicas eth1, eth2 y eth3. Para observar la implementación ir al Anexo 4, Manual de Instalación de OpenvSwitch en host Raspberry.
5. Creación de las imágenes para el Controlador Ryu. Para observar la implementación ir al, Manual de Instalación de Ryu en imagen Docker.

a. Creación de una imagen base en donde se instalan las paqueterías necesarias de Python y Ryu sobre una imagen de Ubuntu 18.04.

b. Creación de la imagen final de Ryu que proviene de la imagen base del punto anterior, en donde añade el archivo Python que ejecutará el controlador.

6. Creación de la imagen para la página html – css alojada en el servidor web. Para observar la implementación ir al Anexo 6, Manual de Instalación de Servidor Nginx y la página web.

RESULTADOS OBTENIDOS

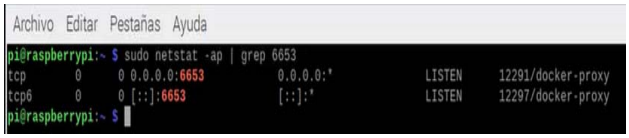


Figura 6: Observación de la escucha en el puerto del controlador

Ejecución del contenedor de ryu y del código Python con el comando ryu-manager bootryu2.py que controla el switch virtual

docker run -it –privileged –name ryu3 -p 6653:6653 luisps/ryu:911 /bin/bash

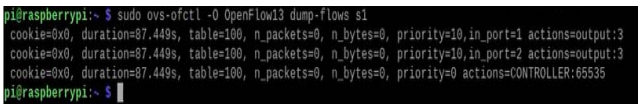
ryu-manager bootryu2.py

Figura 7: Ejecución del contenedor de ryu y del código Python

(comando sudo netstat -ap | grep 6653)

Figura 8: Observación de la comunicación entre el docker Ryu y el OpenvSwitch

(comando dump-flows y dump-groups)



(comando route -n)

```
pi@raspberrypi:~ $ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.100.1   0.0.0.0         UG    202    0        0 eth0
0.0.0.0         192.168.100.1   0.0.0.0         UG    306    0        0 wlan0
169.254.0.0     0.0.0.0         255.255.0.0     U     207    0        0 ovs-system
169.254.0.0     0.0.0.0         255.255.0.0     U     208    0        0 s1
169.254.0.0     0.0.0.0         255.255.0.0     U     211    0        0 veth85ee9a3
169.254.0.0     0.0.0.0         255.255.0.0     U     213    0        0 veth10753ce
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
192.168.100.0   0.0.0.0         255.255.255.0   U     202    0        0 eth0
192.168.100.0   0.0.0.0         255.255.255.0   U     203    0        0 eth1
192.168.100.0   0.0.0.0         255.255.255.0   U     306    0        0 wlan0
pi@raspberrypi:~ $
```

Figura 9: Tabla completa de enrutamiento del switch virtual

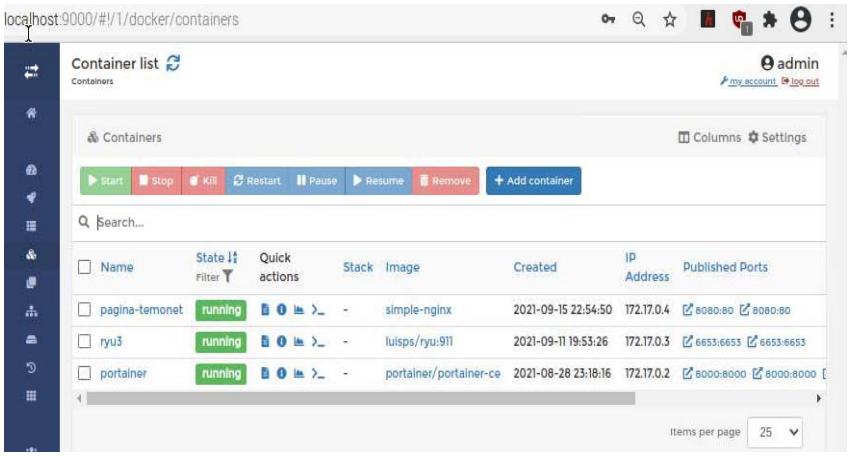


Figura 10: Verificación de los contenedores encendidos en Portainer

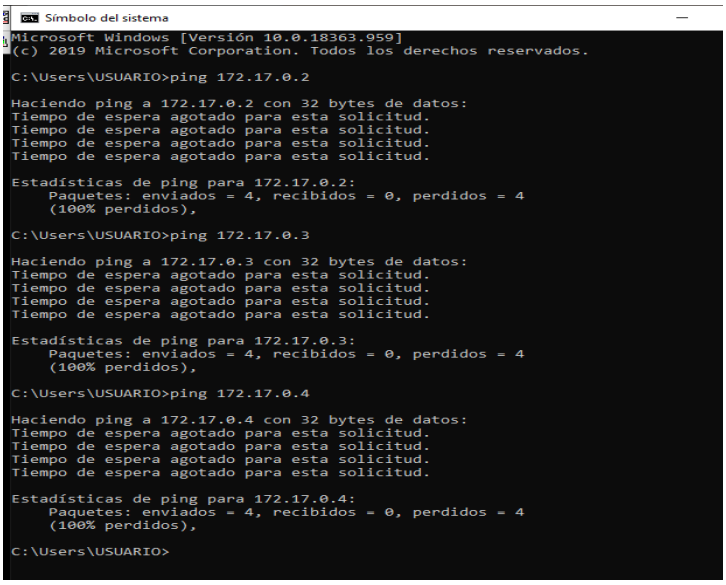


Figura 11: Prueba de comunicación con las direcciones ip de los contenedores demostrando que no son alcanzables desde el cliente

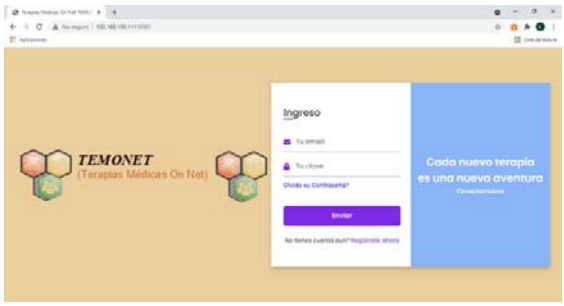


Figura 12: Ingreso a la página web del cliente conectado a la eth1

Figura 13: Prueba de velocidad de carga/descarga del enlace

ANÁLISIS DE FACTIBILIDAD

El propósito planteado en este proyecto de titulación se logró satisfactoriamente gracias al uso de herramientas de código abierto u open source y dispositivos de bajo costo por sus grandes beneficios que nos proporcionan, siendo alguno de estos:

- Raspberry PI: Es un computador de tamaño reducido el cual es capaz de ejecutar un sistema operativo Linux con la capacidad de permitirle a los usuarios sin importar la edad explorar la computación y tecnología.
- Raspbian: También conocido como Raspberry Pi OS se denomina un sistema operativo sin costo basado en Debian, optimizado para el funcionamiento de la Raspberry Pi.
- Dockers: Con el uso de este sistema se puede obtener un solo objeto que se puede ejecutar de manera confiable sin importar el lugar.
- Portainer: La aplicación nos permite la gestión de forma más fácil e intuitiva los contenedores Docker a través de una interfaz gráfica.
- Open vSwitch: Sistema diseñado para permitir una automatización masiva de la red a través de la extensión programable
- Ryu: Controlador de redes definidas por software.

Conclusiones

La integración de aplicaciones, entornos o unidades independientes si es posible llevar a cabo dentro de un dispositivo de bajo costo, para lo cual el uso de contenedores, son necesarios porque logran menor consumo de recursos a nivel del kernel, autonomía entre cada contenedor y a su vez una integración y comunicación entre ellos.

El uso de contenedores brinda la ventaja del aislamiento, portabilidad, agilidad, escalabilidad y un mayor control a lo largo de todo el flujo de trabajo incluido en el proceso de integración. La ventaja más importante que se obtuvo es el aislamiento del entorno que se proporciona entre el desarrollo y las operaciones del proyecto.

La comunicación entre los partes integrantes de este proyecto se volvió más fluida dado que la ubicación de estas se encuentra en un mismo dispositivo.

El desarrollo con contenedores es ideal para un enfoque basado en microservicios para el diseño de aplicaciones. Bajo este modelo, las aplicaciones complejas se dividen en unidades más discretas y pequeñas. Las instancias de Docker son más ligeras, lo cual facilita el uso de hardware de bajos recursos como los Raspberry Pi.

Conflictos de intereses

Los autores no poseen conflicto de intereses.

Contribución de los autores

1. Conceptualización: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno, Jorge Chicala Arroyave.
2. Curación de datos: Eduardo Alvarado Unamuno, Jorge Chicala Arroyave.
3. Investigación: Jorge Chicala Arroyave.
4. Metodología: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno.
5. Administración del proyecto: Eduardo Alvarado Unamuno, Jorge Chicala Arroyave.
6. Software: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno.
7. Supervisión: Jorge Chicala Arroyave.
8. Validación: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno
9. Visualización: Jorge Chicala Arroyave.
10. Redacción – borrador original: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno, Jorge Chicala Arroyave.
11. Redacción – revisión y edición: Jenny Arizaga Gamboa, Eduardo Alvarado Unamuno, Jorge Chicala Arroyave.

Financiamiento

La investigación ha sido financiada por los autores.

Referencias

(AEC), A. E. (2016). PRINCIPALES VENTAJAS Y DIFICULTADES DE LA INTEGRACIÓN DE SISTEMAS. AEC, 29.

Alarcón, J. M. (14 de 06 de 2018). Krasis Consulting S.L.U. Obtenido de https://www.campusmvp.es/recursos/post/que-diferencia-hay-entre-docker-contenedores-y-maquinas-virtuales.aspx

Amazon Web Services, I. (2021). Amazon Web Services. Obtenido de https://aws.amazon.com/es/docker/

B., G. (2021). Hostinger.es. Obtenido de https://www.hostinger.es/

CL, R. (s.f.). Raspberrypi CL. Obtenido de https://raspberrypi.cl

DANIELA, G. M. (2017). dspace.espoch. Obtenido de http://dspace.espoch.edu.ec

Docker. (2021). Docker Inc. Obtenido de https://www.docker.com/

Docker. (s.f.). Docker. Obtenido de https://www.docker.com/resources/what-container

Cello, M., Xu, Y., Walid, A., Wilfong, G., Chao, H. J., & Marchese, M. (2017). BalCon: A distributed elastic SDN control via efficient switch migration. 2017 IEEE International Conference on Cloud Engineering (IC2E),

Chi, P.-W., Kuo, C.-T., Guo, J.-W., & Lei, C.-L. (2015). How to detect a compromised SDN switch. Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft),

Firestone, D. (2017). {VFP}: A Virtual Switch Platform for Host {SDN} in the Public Cloud. 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17),

Lazaris, A., Tahara, D., Huang, X., Li, E., Voellmy, A., Yang, Y. R., & Yu, M. (2014). Tango: Simplifying SDN control with automatic switch property inference, abstraction, and optimization. Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies,

Wang, C. a., Hu, B., Chen, S., Li, D., & Liu, B. (2017). A switch migration-based decision-making scheme for balancing load in SDN. IEEE Access, 5, 4537-4544. https://ieeexplore.ieee.org/iel7/6287639/6514899/07883881.pdf