DOI: http://dx.doi.org/10.23857/dc.v7i5.2302

Ciencias técnicas y aplicadas
Artículo de investigación

Virtualización del despliegue del ambiente de desarrollo de OdontoApp de la Universidad Católica de Cuenca

Virtualization of the deployment of the OdontoApp development environment of the Catholic University of Cuenca

Virtualização da implantação do ambiente de desenvolvimento OdontoApp da Universidade Católica de Cuenca

Ronald Andrés Espinoza-Urrego [I]
respinozau@ucacue.edu.ec
https://orcid.org/0000-0002-1622-6915

Jorge Cristian Cárdenas-Matute [II]
jcardenasm@ucacue.edu.ec
https://orcid.org/0000-0003-1136-6107

Correspondencia: respinozau@ucacue.edu.ec

*Recibido: 25 de julio 2021 *Aceptado: 30 de agosto de 2021 * Publicado: 28 de septiembre de 2021

I. Ingeniero de Sistemas, Programador Senior. Universidad Católica de Cuenca, Cuenca, Ecuador.
II. Ingeniero de Sistemas, Docente de Facultad de Odontología, Universidad Católica de Cuenca, Cuenca, Ecuador.

Resumen

Este trabajo tiene como objetivo el proponer una solución para la virtualización del despliegue del ambiente de desarrollo de OdontoApp, de la Universidad Católica de Cuenca, utilizando los contenedores de Docker. Se ha analizado la madurez de la virtualización de servicios en ambientes de desarrollo, pruebas y producción; estudiando la virtualización de servicios a través de la herramienta Docker y comparándola con el proceso común de despliegue de ambientes de desarrollo, para determinar principalmente las diferencias en: cantidad de operaciones, tiempo de operación y cantidad de recursos de hardware. Finalmente, se detallan los resultados obtenidos que sustentan que la virtualización en contenedores, reduciría el tiempo necesario y automatizaría las tareas de aprovisionamiento que un desarrollador debe ejecutar para desplegar su ambiente de trabajo.

Palabras clave: Contenedores; docker; desarrollo de software; optimización; automatización.

Abstract

The objective of this work is to propose a solution for the virtualization of the deployment of the OdontoApp development environment, of the Catholic University of Cuenca, using Docker containers. The maturity of service virtualization in development, test and production environments has been analyzed; studying the virtualization of services through the Docker tool and comparing it with the common process of deployment of development environments, to determine mainly the differences in: number of operations, operation time and amount of hardware resources. Finally, the results obtained are detailed that sustain that virtualization in containers would reduce the necessary time and automate the provisioning tasks that a developer must execute to deploy their work environment.

Keywords: Containers; docker; software development; optimization; automation.

Resumo

O objetivo deste trabalho é propor uma solução para a virtualização da implantação do ambiente de desenvolvimento OdontoApp, da Universidade Católica de Cuenca, utilizando containers Docker. A maturidade da virtualização de serviço em ambientes de desenvolvimento, teste e produção foi analisada; estudar a virtualização de serviços através da ferramenta Docker e compará-la com o processo comum de implantação de ambientes de desenvolvimento, para determinar principalmente

as diferenças em: número de operações, tempo de operação e quantidade de recursos de hardware. Por fim, detalham-se os resultados obtidos que sustentam que a virtualização em containers reduziria o tempo necessário e automatizaria as tarefas de provisionamento que um desenvolvedor deve executar para implantar seu ambiente de trabalho.

Palavras-chave: Containers; docker; desenvolvimento de software; otimização; automação.

Introducción

Definiciones de diccionarios estándar, como el Diccionario Oxford de inglés, establece que el software es: "los programas y otra información operativa que utiliza una computadora". Para Sommerville, probablemente uno de los libros de texto más utilizados en la educación en ingeniería de software: “el software, además de programas de computadora, también es toda la documentación asociada y los datos de configuración que se requieren para que estos programas funcionen correctamente. Puede incluir documentación del sistema, documentación del usuario, y sitios web para que los usuarios descarguen información reciente del producto” (Pfeiffer, 2020).

Sin embargo, en una definición más específica y detallada, Pfeiffer nos dice que el software es: “la colección de código fuente, código fuente específico, código script, código binario, código de compilación, código de infraestructura, imagen, video, música, configuración, base de datos, fuente, archivo, marcado, documento, datos de la aplicación, prosa, jerga legal, entre otros componentes, que los humanos recopilan y almacenan, y que estos artefactos son más que solo código, datos o programas” (Pfeiffer, 2020).

Por lo tanto, el desarrollo de software requiere una estrecha cooperación y colaboración entre los miembros del equipo y todo tipo de actividades de desarrollo. Los miembros de proyectos de software en todos los roles deben comunicarse e interactuar continuamente con otros miembros del proyecto, así como con una variedad de partes interesadas (Mens et al., 2019).

Generalmente, el desarrollador de proyectos de software debe instalar y configurar en su computador un conjunto de programas y servicios necesarios para editar, ejecutar y probar el código del software. Por ejemplo, los componentes interrelacionados de un ambiente de desarrollo para una aplicación web, donde podemos encontrar: intérpretes y compiladores de código, gestores de base de datos y servidores web. Las versiones y configuraciones de estos programas y servicios deben ser idénticos

en todos los ambientes donde se ejecuta la aplicación web para garantizar condiciones de ejecución y resultados homogéneos.

Algunas de las restricciones y conflictos que podemos encontrar con las versiones y configuraciones de estos programas en un ambiente de desarrollo son:

- Programas exclusivos: los programas se instalan a través de paquetes que son desarrollados para versiones específicas de los sistemas operativos, la disponibilidad y actualizaciones de estos paquetes está estrictamente limitada a la capacidad e interés del fabricante por ofrecer compatibilidad para los sistemas operativos y sus versiones disponibles en el mercado.
- Procesos únicos: los servicios se instancian sólo una vez, en un puerto específico o en un proceso, impidiendo ejecutar varias veces los mismos servicios en varios proyectos de desarrollo de software.
- Instalación: instalar y configurar los programas y servicios para desarrollar un proyecto de software requiere tener el conocimiento de los detalles de la instalación y configuración en cada sistema operativo para ajustarse a los requerimientos de cada proyecto.
- Gestión: iniciar, detener o reiniciar los servicios, requiere tener el conocimiento sobre como interactuar con los programas de administración de cada servicio.
- Programación: necesitamos conocer cómo interactuar con el sistema operativo y el servicio para programar su ejecución automática al iniciar el computador.
- Actualizaciones: las actualizaciones dependen mucho de cada sistema operativo, en el caso de que exista el paquete compatible para subir de versión a un programa, se requiere un arduo proceso de reinstalación que comúnmente es más complejo que la instalación, debido a que se reemplaza una versión del servicio por otra en el mismo sistema de archivos, lo que podría provocar conflictos de compatibilidad en sus configuraciones actuales.

Por lo tanto, desplegar ambientes de desarrollo de software para los miembros de un equipo puede conllevar mucho tiempo y esfuerzo por las particularidades de los diversos sistemas operativos y versiones que poseen los computadores personales de trabajo, sin embargo, existen algunas alternativas para tratar de aligerar este proceso para los desarrolladores. Una opción es construir una imagen a partir de un sistema operativo, que contenga todos los servicios necesarios para que el desarrollador lo copie en su disco duro y ejecute como su sistema base. Pero, los desarrolladores acostumbrados a usar un sistema operativo determinado y sus utilidades ofimáticas, difícilmente

pueden migrar a este nuevo sistema operativo que contiene dicha imagen. Otra alternativa es usar una máquina virtual (VM) para desplegar la imagen antes mencionada, pero podría disminuir el rendimiento del computador por la gran cantidad de recursos de hardware que necesita, sin contar con la dificultad para el usuario de utilizar dos sistemas operativos a la vez, además que la instalación a partir de la imagen quedará desactualizada y requerirá de intervención humana para actualizarse.

La industria del software tiene un alto nivel de competitividad por lo que optimizar los procesos es fundamental para lograr resultados rentables, un desarrollador de software generalmente está involucrado en varios proyectos y tareas a la vez, utilizando un gran número de programas y servicios, por lo que aprovisionar su computador para realizar su trabajo diario no debería repercutir en pérdidas de tiempo o reducción de productividad.

Han surgido nuevos grupos de herramientas para gestionar la complejidad de la actividad del desarrollo e implementación de software. Una de estas nuevas herramientas es la tecnología de contenedores, llamada también virtualización basada en contenedores, virtualización de SO o contenedorización (Watada et al., 2019). Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias. Las imágenes de contenedores son las plantillas que almacenan estáticamente las instrucciones necesarias para generar una instancia de contenedor, estos se ejecutan como un proceso aislado en el grupo de recursos del usuario. Cada máquina puede ejecutar varios contenedores, todos compartiendo el mismo núcleo del sistema operativo (Zhu & Bayley, 2018).

Los contenedores son una solución liviana que los desarrolladores pueden usar para implementar y administrar aplicaciones, a menudo se consideran una alternativa más liviana que las máquinas virtuales, las cuales incluyen el sistema operativo completo a diferencia de los contenedores, lo que permite que estos últimos proporcionen ventajas en el uso de recursos (Schmitt et al., 2019).

Las imágenes de contenedores se pueden enviar a través de una red más rápidamente e iniciarse casi instantáneamente como un proceso en un sistema operativo, mientras que se tarda mucho más transferir e iniciar una copia de máquina virtual. Los contenedores también brindan separación entre usuarios, logrando así las mismas ventajas de seguridad y privacidad que tienen las máquinas virtuales (Zhu & Bayley, 2018).

La utilidad de los contenedores no se limita a que sean una versión más ligera de las máquinas virtuales. Una característica interesante de los contenedores es que brindan portabilidad y, por lo

tanto, modularidad, lo que los hace adecuados para trabajar como componentes de software o como microservicios autónomos. Esto, nos permite mitigar 3 problemas cuando los sistemas de software crecen, los cuales se detallan a continuación (Schmitt et al., 2019):

1. Mantener el software se vuelve más difícil.
2. La adición de nuevas funciones al sistema se ralentiza.
3. Los requisitos de recursos para el software crecen.

A pesar de sus bondades, el uso de contenedores para el despliegue de ambientes de desarrollo de software aún no es muy popular entre los desarrolladores en Ecuador, el caso de estudio de este documento se centra específicamente en la Facultad de Odontología de la Universidad Católica de Cuenca, en donde no existía un sistema informático para gestionar los procesos administrativos relacionados con las fichas odontológicas de los estudiantes, sus procesos se gestionaban con archivos físicos, lo que provocaba problemas al perderse, deteriorarse y ser difíciles de encontrar; gracias a esto, tratando de mejorar estos procesos se desarrolló la aplicación OdontoApp, sin embargo, una vez puesta en marcha se detalla que resulta dificultoso para nuevos desarrolladores, preparar el ambiente idóneo para desarrollar la aplicación, dado que se debe conocer como descargar, instalar, configurar y desplegar manualmente los programas adecuados en su computador personal.

Esta investigación estará orientada a proponer una solución para el despliegue del entorno de desarrollo de software de la aplicación OdontoApp basada en contenedores de software para analizar la madurez, ventajas, desventajas y diferencias de los contenedores frente a los mecanismos comunes de despliegue de ambientes de desarrollo en la industria, contribuyendo con una nueva alternativa para el equipo de desarrollo de OdontoApp para optimizar y automatizar el proceso de despliegue de su ambiente de trabajo.

Metodología

Proceso de despliegue de ambiente de desarrollo:

Se han propuesto muchos modelos, metodologías y marcos de trabajo involucrados al desarrollo y mantenimiento de productos de software, el objetivo primordial de estas herramientas es establecer un plan integral para el proceso de desarrollo de software. A pesar de los grandes esfuerzos en este campo, aún no existe una receta o fórmula perfecta para gestionar el desarrollo de software a gran escala o sus requisitos siempre cambiantes durante el proceso (Kaur & Sengupta, 2010). Por lo tanto,

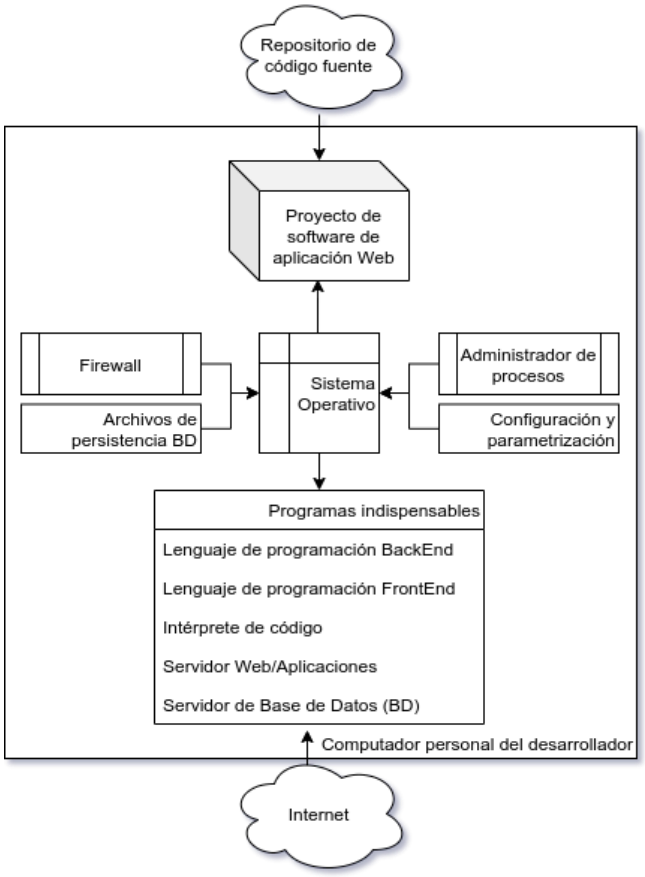
se ha popularizado entre los desarrolladores de software ambientes de ejecución desiguales, estructurados de acuerdo a preferencias, conocimientos, sistema operativo y tareas individuales de cada desarrollador. Dejando abierta la posibilidad al aparecimiento de brechas de seguridad, compatibilidad y mantenibilidad de un proyecto.

A pesar de las diferencias que pueden llegar a existir entre ambientes de desarrollo de un mismo proyecto de software, se ha procurado establecer componentes comunes de un ambiente de desarrollo de aplicación web, y posteriormente describir el proceso que tiene que realizar el desarrollador para lograr aprovisionar dicho ambiente en su equipo personal.

En la figura 1, se pueden observar los componentes de software básicos que necesita instalar un desarrollador en su equipo personal para poder ejecutar un proyecto de software. En la parte central se ubica el sistema operativo, el cual es la parte medular de este ambiente, debido a que, sobre su grupo de recursos como el almacenamiento, se instalarán todos los programas indispensables del ambiente.

El sistema operativo también será el encargado de gestionar las configuraciones del Firewall y el Administrador de procesos, los cuales nos permitirán abrir puertos como el 80 para ingresar a la aplicación web desde el navegador, y decidir cuándo iniciar o detener servicios como el servidor web o el servidor de base de datos. El sistema operativo también será el encargado de proveer las rutas para el almacenamiento de la persistencia de archivos del servidor de base de datos, archivos de configuración y variables de entorno.

Figura 1: Componentes de un ambiente de desarrollo común de aplicación web

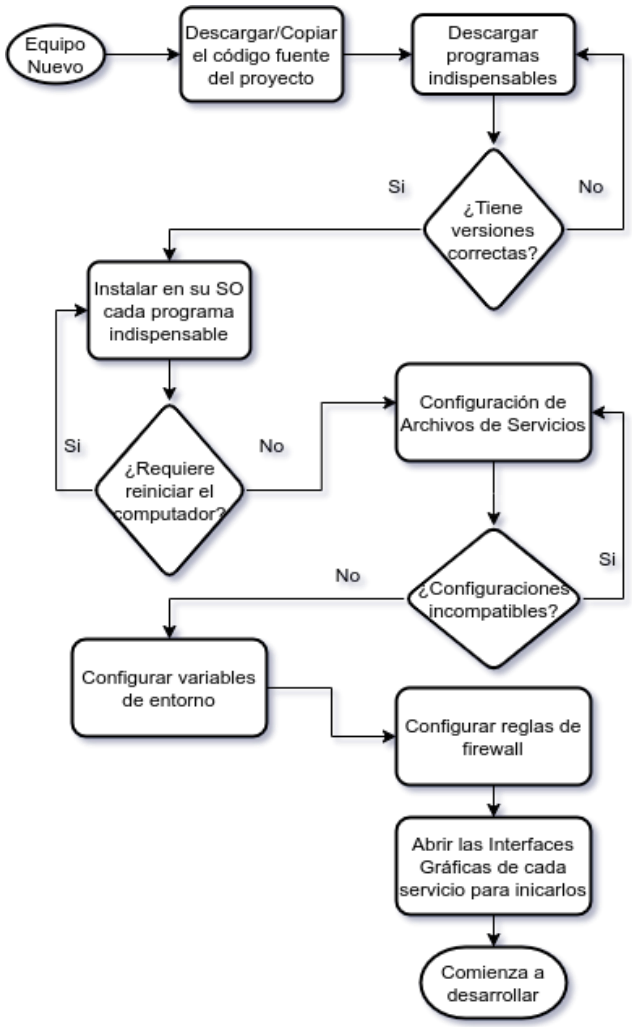


Fuente: Elaboración propia.

Para obtener el entorno parecido al propuesto, el desarrollador tiene que ejecutar un proceso muy parecido al que describe la figura 2, en donde comienza recibiendo un equipo personal nuevo y limpio, solo con el sistema operativo instalado, y termina con un equipo listo para comenzar a desarrollar y probar las aplicaciones del proyecto.

El proceso de instalación comienza por descargar desde el internet, el código fuente del proyecto y todos los programas indispensables para desplegar la aplicación web del proyecto. Por lo general, dentro de los proyectos de software se mantienen archivos como README.md o CHANGELOG.md, los cuales describen cambios y procedimientos técnicos para establecer configuraciones de los servicios y del propio proyecto que se está desarrollando.

Figura 2: Proceso de aprovisionamiento de un ambiente común para desarrollo web



Fuente: Elaboración propia.

Las configuraciones de servicios, se refiere a las modificaciones de los archivos que usan los servidores para establecer su comportamiento. Cada servidor web y servidor de base de datos, por ejemplo, tiene archivos de configuraciones que se crean en rutas preestablecidas al instalarse, estos archivos deben ser modificados de acuerdo a los requerimientos de cada proyecto de software. Las configuraciones propias del proyecto, puede requerir el establecimiento de variables de entorno a nivel del sistema operativo o también puede requerir copiar archivos con plantillas donde se establecen cadenas de conexión de base de datos, credenciales y secretos para autenticación con tokens, como parte de la parametrización para el despliegue de las aplicaciones.

El desarrollador de aplicaciones web generalmente tiene que realizar todo este proceso para cada proyecto de software para comenzar a desarrollar. Un procedimiento similar

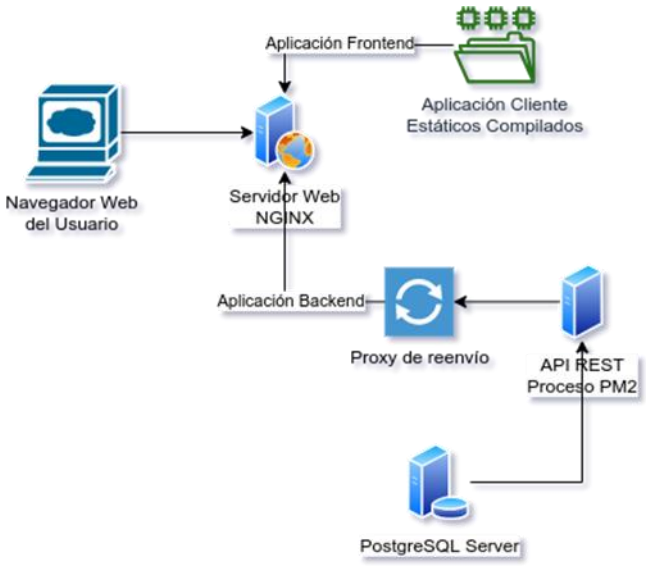
Debe realizar también cuando existen cambios de versiones y configuraciones de los servicios o del proyecto. Todo esto, con el fin de crear las condiciones necesarias para que el proyecto funcione correctamente, en algunos proyectos existen configuraciones muy precisas y relevantes, por lo que a los nuevos desarrolladores se les provee el computador personal listo, previamente aprovisionado por un equipo técnico que tiene manuales o imágenes preparadas, las cuales usan para replicar sobre un único sistema operativo las mismas condiciones de ejecución para cada computador del equipo de desarrollo.

En el caso de estudio, la aplicación web OdontoApp cuenta con varios componentes en su ambiente de producción como lo indica la figura 3, los cuales describiremos a continuación:

- Servidor de base de datos: El proyecto usa PostgreSQL versión 13 como motor de base de datos para almacenar toda la información persistente de la aplicación, como registros de usuarios, permisos, fichas odontológicas, entre otros.
- API REST: Esta aplicación es la encargada de ser el backend donde se realizarán todos los procesos del sistema y se conectará con el servidor de base de datos para almacenar la información. Está escrita en NodeJS versión 12 y usa el framework NestJS, para desplegarla en desarrollo se puede usar el servidor interno de NodeJS, pero en producción debe ser controlada por medio de un gestor de procesos como PM2 y servida en modo proxy a través de NINGX.
- Aplicación Frontend: Es la aplicación que se renderizará finalmente en el navegador del usuario, y se conectará con nuestra API para ejecutar todos los procesos. Está escrita en JavaScript usando NodeJS versión 14 y utiliza el framework VueJS, en desarrollo usa un servidor web propio del framework, pero para producción se debe compilar y comprimir en archivos estáticos los cuales se sirven a través de NGINX.
- Servidor Web NGINX: NGINX es un servidor web ampliamente usado en la industria del software, no necesariamente requerido en ambientes de desarrollo, pero imprescindible para el ambiente de producción de nuestro caso de estudio, por lo que un ambiente local, de pruebas o certificación que requiera simular el ambiente de producción, necesariamente debe considerarlo como el servicio frontal. Dentro del servidor web tenemos módulos que nos

permiten ciertas funcionalidades extras, como es el caso del Proxy de reenvío, el cual se utiliza para reenviar las peticiones de nuestro servidor web frontal hacia el servidor interno que desplegó PM2 para el API REST. De esta forma, nuestro servidor solo tiene expuesto hacia internet, los puertos 80 y 443 para comunicarse con el navegador web.

Figura 3: Componentes del ambiente de producción de la aplicación OdontoApp



Fuente: Elaboración propia.

Los desarrolladores de OdontoApp ejecutan todos los componentes del ambiente porque son programadores completos, es decir, realizan cambios en la Aplicación Cliente y también en el API REST. Los cambios en la base de datos se gestionan por medio de migraciones, soportadas por el framework que ejecuta la aplicación backend.

Por lo tanto, los programas indispensables (conforme la figura 1) que necesita tener instalado un desarrollador de OdontoApp en su equipo personal son:

- PostgreSQL Server.
- NodeJS versión 14.
- NodeJS versión 12.
- Manejador de Versiones de NodeJS (NVM).

El proceso para implementar todos los componentes del ambiente de desarrollo es similar al de la figura 2, con mínimas diferencias, como que no se requiere copiar archivos plantillas de configuraciones sino solamente establecer variables de entorno.

Tecnología de contenedores

Como se ha descrito antes, existen otras alternativas de despliegue de ambientes de desarrollo. Se ha escogido las herramientas de contenedorización para desarrollar el caso de estudio porque permiten empaquetar aplicaciones, sus dependencias y configuraciones en imágenes, que luego se ejecutan como contenedores sobre cualquier máquina que ejecute un demonio o un proceso Unix , logrando que la implementación de aplicaciones sea fácil debido a que la imagen es multiplataforma y se construye una sola vez, dándole agilidad extrema al proceso de aprovisionamiento y despliegue de ambientes de desarrollo de software (Kotouza et al., 2020). La contenedorización es la evolución de la virtualización convencional, uno de los conceptos pioneros en el inicio de nubes modernas, en donde ha contribuido con los siguientes beneficios (Watada et al., 2019):

- Optimizar los recursos de un centro de datos (hardware, software, energía y dinero) al consolidar muchas máquinas infrautilizadas en un solo sistema.
- Escalar la confiabilidad del proceso y la tolerancia a fallas mediante el aislamiento del desempeño.
- La migración de procesos y la replicación en ubicaciones geográficas.

Sin embargo, nuestro enfoque de virtualización va más allá de la virtualización convencional. La contenedorización nos ofrece muchas más ventajas en las actividades concernientes al despliegue de software, las cuales se detallan a continuación:

- Arquitectura de aplicaciones: La virtualización de servicios y aplicaciones en contenedores es más ligera en comparación con máquinas virtuales, esto permite la habilitación de nuevas de arquitecturas como la de microservicios con la creación de unidades de software autónomas (Piedade et al., 2020). Entre los beneficios que provee esta arquitectura está la evolución continua del software, integración de tecnología perfecta, rendimiento óptimo en tiempo de ejecución, escalabilidad horizontal y confiabilidad a través de la tolerancia a fallas. Todas las empresas de éxito de TI han adoptado un enfoque agresivo para implementarla. Ejemplos bien conocidos incluyen NetFlix, Amazon, EBay, Google y Microsoft (con Azure) (Zhu & Bayley, 2018).

- Interfaz de administración: La gestión de contenedores se realiza explícitamente mediante un sistema distribuido, en lugar de simplemente dejar que el usuario invoque un contenedor desde dentro de cada tarea, existen orquestadores y administradores de clústeres que intercambian tareas entre grandes cantidades de contenedores (Watada et al., 2019). Con enfoque declarativo o con un lenguaje imperativo, las interfaces permiten describir los planos de control y los comandos que el usuario puede especificar para cambiar el estado deseado para un servicio. Esto nos ayuda, por ejemplo, a escalar la aplicación hacia arriba y hacia abajo con reglas preestablecidas (de Guzmán et al., 2018).
- Utilización de recursos: Los contenedores utilizan menos recursos de CPU, memoria y redes para proporcionar soluciones de gestión de recursos eficientes, escalables y rentables en infraestructuras de nube, que son las principales razones de su creciente uso (Watada et al., 2019). Permiten describir los detalles de cómo crear y coordinar los diferentes recursos, determinar el uso elevado de CPU o memoria, o un incremento en las solicitudes entrantes. Esto permite implementaciones muy flexibles que solo consumen recursos bajo demanda, sin la necesidad de adivinar la capacidad de la infraestructura por adelantado (de Guzmán et al., 2018).
- Integración y despliegue continuo: DevOps es una práctica que tiene como objetivo integrar el desarrollo de software y los procesos operativos. Se espera que los contenedores reduzcan los ciclos de implementación, mejore la calidad del software y acorte el tiempo para corregir errores. Esto ayuda a aumentar la frecuencia de las implementaciones, lo que ayuda a brindar un servicio más rápido a los clientes. Las organizaciones que practican DevOps pueden implementar cambios de software tan rápido como 500 veces al día (Mohan et al., 2018). Los contenedores aseguran la portabilidad y el balanceo de carga entre muchas instancias y permite su integración con herramientas DevOps como Ansible que gestiona configuración, aprovisiona y despliega servidores a través de SSH, y Kubernetes que ofrece una solución de gestión de cargas de trabajo y auto escalabilidad de servicios contenedorizados (Khalyly et al., 2020).
- Procesos de migración: Los operadores de centros de datos, deben lidiar con problemas de anchos de banda al momento de realizar migraciones. En comparación con el esquema de migración de máquinas virtuales, la técnica de migración basada en contenedores muestra una

mejora significativa para el proceso de migración y puede servir como un mecanismo de migración clave para mejorar el rendimiento de un centro de datos en la nube (Bhardwaj & Krishna, 2019).

Sistemas de contenedorización

En virtualización de contenedores existen varias herramientas disponibles en el mercado, con diferentes características y alternativas, a continuación, vamos a conocer cuáles son las principales:

1. En virtualización de contenedores una de las herramientas más populares entre los desarrolladores es Docker. Es una plataforma de software por etapas que le permite fabricar, probar y enviar aplicaciones rápidamente. Docker agrupa la programación en unidades fundamentales, consideradas el soporte que tienen todo lo que el producto necesita para ejecutarse, incluido el código, las bibliotecas, el tiempo de ejecución, el marco y los dispositivos. Al utilizar Docker, podemos usar y extender aplicaciones rápidamente a cualquier entorno y saber que su código se ejecutará (Marathe et al., 2019).
2. Rkt es un motor de contenedores de aplicaciones, se introdujo recientemente para superar las limitaciones existentes de los tiempos de ejecución del contenedor. Al ser una tecnología más reciente, se espera que el contenedor Rkt brinde un mejor soporte para las aplicaciones HPC . Las aplicaciones de la generación actual exigen capacidades de alto rendimiento, esencialmente se requiere mejorar el soporte disponible. Los investigadores han estado en la búsqueda de tecnologías que les permitan lograr un rendimiento que se parezca más a los escenarios dónde las aplicaciones se ejecutan directamente en el hardware físico (Yadav et al., 2018).
3. OpenVZ es una tecnología de virtualización a nivel de sistema operativo, que crea y administra múltiples contenedores Linux aislados, también llamados entornos virtuales (VEs), en un solo servidor físico. Cada VE funciona exactamente como un host independiente con su propio acceso raíz, usuarios y grupos, dirección IP y MAC, memoria, árbol de procesos, sistema de archivos, bibliotecas del sistema y archivos de configuración, pero comparte una sola instancia del sistema operativo Linux (Zheng et al., 2012).

Análisis comparativo

Realizamos un análisis comparativo basado en lo que se ha considerado como características fundamentales.

Tabla 1: Análisis comparativo de tecnologías de contenedores.

	Adopción[7]	**Comunidad**[8]	**Integración**[9]	**Soporte**[10]
Docker	Moderado	645 K	Aceptable	Estable
Rkt CoreOS	Moderado	864	Bajo	Incierto
OpenVZ	Bajo	448	Bajo	Incierto

Fuente: Elaboración propia.

i. Adopción: Esta medida se refiere a la acogida que recibe la herramienta de software por parte de los usuarios. Cuando un individuo no está seguro de la calidad de un producto de software, el comportamiento de adopción de los usuarios que proceden se convierte en una fuente clave de información. Un usuario puede inferir la información privada de otros en función de su observación de su comportamiento de adopción (Zhao et al., 2020).

ii. Comunidad: Esta métrica se refiere a la cantidad de repositorios de código públicos en GitHub que se relacionan con la herramienta. Donde principalmente es la comunidad que dirige el desarrollo y progreso de las herramientas donde colabora. Las empresas actuales alivian la presión de recursos de sus proyectos de software aprovechando la comunidad, incluidos los estudiantes del campus, los fanáticos de la programación, los aficionados al software, quienes pueden entregar software de alta calidad que varía desde pequeñas utilidades (como log4j, dom4j) hasta productos a gran escala (como Apache, Eclipse) (Zhou & Liu, 2010).

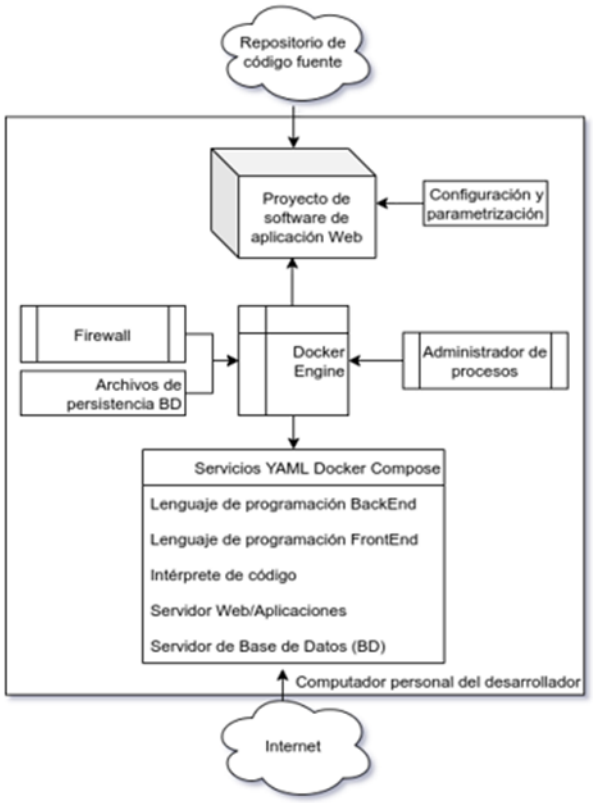
iii. Integración: Se refiere a la compatibilidad con herramientas de terceros. Es decir, que puede estar, funcionar o coexistir sin impedimento con otra. Por ejemplo, con orquestadores de contenedores y nubes públicas.

iv. Soporte: Esta métrica se refiere al soporte o respaldo comercial de empresas importantes del campo tecnológico para que el proyecto se mantenga y evolucione adecuadamente con el pasar del tiempo.

Docker como herramienta base

Por lo tanto, para los desarrolladores, Docker podría ser la mejor herramienta que le ayude a estandarizar sus programas y automatizar el despliegue de los mismos en el ambiente de desarrollo de su sistema operativo preferido.

Figura 4: Componentes del ambiente contenedorizado de una aplicación web.



Fuente: Elaboración propia.

Docker ofrece una utilidad llamada Docker Compose, la cual juega un papel importante al permitir orquestar múltiples servicios y definir configuraciones a través de archivos YAML, estas configuraciones especifican desde cómo se construyen las aplicaciones, las imágenes correspondientes y como se relacionan entre sí, hasta los volúmenes para la persistencia de datos y las redes para las conexiones entre servicios (Piedade et al., 2020).

En un ambiente virtualizado con Docker podemos delegarle la responsabilidad de buscar, descargar, instalar y desplegar por nosotros los programas (imágenes) en las versiones que deseemos. Así mismo, a través de Docker Compose, podemos definir las variables de entorno necesarias para la

ejecución de la aplicación, puertos de servicios internos y externos que serán abiertos en el firewall del computador y los volúmenes para la persistencia de datos. Como podemos observar en la figura 4, el sistema operativo no se encarga de todo como en el proceso común, sino que ahora es Docker que lo hará automáticamente. El desarrollador no debe preocuparse de instalar manualmente algún programa para desarrollar, solo debe tener instalado en su computador Docker Desktop para Windows y Mac o Docker y Docker Compose para Linux. La instalación se realiza una sola vez y los beneficios se replican cada vez que instale un nuevo proyecto.

Benchmarking entre el Proceso Común de Despliegue y Docker aplicado al caso de estudio

Se ha analizado las implicaciones de la ejecución del proceso común para el despliegue del ambiente de desarrollo de OdontoApp, en comparación al proceso de despliegue del mismo ambiente en contenedores Docker. A continuación, se muestran los parámetros que se considerarán para evaluar ambos procesos:

i. *Cantidad de operaciones que necesita realizar para el aprovisionamiento:* Se trata de la cantidad de procesos que realiza el desarrollador para instalar, configurar y desplegar un nuevo ambiente de trabajo.
ii. Tiempo de operación para el aprovisionamiento: Hace referencia al tiempo necesario para cumplir con todas las operaciones de la instalación de un nuevo ambiente de desarrollo; aquí se considerarán dos variables medidas en minutos, el tiempo de trabajo (TT) y el tiempo de espera (TE).
iii. Cantidad de recursos de hardware que consume el despliegue: Se refiere a la cantidad de uso de almacenamiento, porcentaje de CPU y de memoria RAM que utiliza el equipo para la ejecución del ambiente de desarrollo de OdontoApp.

Cantidad de operaciones

La siguiente tabla comparativa expone las fases de la instalación vista en la figura 2. Se ha establecido un puntaje basado en la ponderación de acuerdo a la cantidad de servicios o programas que intervengan en cada fase. La ponderación propone asignar 0 al no requerimiento de intervención humana y 1 por cada tarea necesaria en un programa o servicio.

El puntaje total equivale a la suma de los puntajes, siendo el mejor el de menor puntaje debido a que requerirá de menor intervención humana para su ejecución.

Tabla 2: Comparación de cantidad de operaciones para desplegar OdontoApp.

Proceso	**Proceso Común**	**Proceso con Docker**
Descargar OdontoApp	1	1
Descargar Programas	3	0
Instalación	3	0
Configuración Servicios	1	1
Configuración Variables Entorno	1	0
Configuración Firewall	1	0
Ejecución	3	1
Puntaje Total	13	3

Fuente: Elaboración propia.

Tiempo de operación

Esta comparativa se basa en los datos obtenidos de un experimento propio, en el que se ejecutaron las operaciones para desplegar el ambiente de desarrollo de OdontoApp conforme al proceso común y paralelamente conforme al proceso con Docker. El computador utilizado para este experimento tiene características promedio en el entorno de desarrollo local. En cuanto a sus recursos tenemos los siguientes datos relevantes:

i. Hardware: 8 CPU Core i7 3.5 GHz, Disco de estado sólido de 256 GB y 12 GB de memoria RAM.
ii. Sistema Operativo: Windows 10 Pro 21H1.
iii. Conectividad: Descarga de internet a 26 Mbps, subida a 27 Mbps y latencia promedio de 83 ms.

En cuanto a las variables de tiempo consideradas, hacen referencia al tiempo de trabajo o tiempo utilizado en las actividades que se realizan de forma manual en la estación de trabajo para cumplir con la tarea y procesos necesarios, denominada TT, y el tiempo de espera que es el tiempo en el cual el computador realiza actividades autónomas como descargas o instalaciones, en el cual el operario solo puede espera hasta que estas actividades terminen, denominado TE.

Tabla 3: Comparación de tiempos de operación necesarios para desplegar OdontoApp.

Proceso	Proceso común		Proceso con Docker	
	TT	TE	TT	TE
Descargar OdontoApp	15	5	15	5
Descargar Programas	22	12	0	7
Instalación	12	8	0	5
Configuración Servicios	18	0	0	0
Configuración Variables Entorno	15	0	5	0
Configuración Firewall	15	0	0	0
Ejecución	5	3	1	3
Total (minutos)	102	28	21	20

Fuente: Elaboración propia.

Cantidad de recursos de hardware

En este experimento, se ha tomado cierta cantidad de recursos como referencia para la comparativa. Los recursos destinados al ambiente de desarrollo fueron los siguientes: En cuando a la CPU toda la capacidad existente, de la memoria RAM el 50% y del almacenamiento 2 Gigabytes (GB). Con base en esto se obtienen los siguientes resultados.

Almacenamiento: El almacenamiento se trata de la cantidad de espacio con la que cuenta el disco duro del computador del desarrollador para almacenar archivos. En sistemas de producción los discos duros pueden llegar a ocupar Terabytes de información, en el caso de ambientes de desarrollo el espacio en disco es limitado y compartido entre proyectos e información personal del desarrollador. Por lo tanto, es fundamental que los ambientes sean óptimos para que solo contengan los archivos necesarios para desarrollar. Las métricas a continuación están en megabytes, y se refiere a los programas o imágenes que se necesita descargar e instalar para desplegar el ambiente de desarrollo.

Tabla 4: Comparación de consumo de almacenamiento entre procesos de despliegue de OdontoApp.

Servicio / Programa	Proceso común	Proceso con Docker
Base de datos	277	109
Node 12	18	28
Node 14	30	40
Total	325	177

Fuente: Elaboración propia.

Memoria RAM: El computador del desarrollador comparte su espacio total de memoria RAM entre los procesos del sistema operativo y los procesos de las aplicaciones que ejecuta el usuario, entre las cuales pueden estar navegadores Web, editores de código, terminales, programas utilitarios, entre otros. Por tal motivo, se ha considerado solamente el 50% de la memoria RAM total (6 GB) para que el desarrollador ejecute el ambiente de desarrollo de OdontoApp. El ambiente necesita de 3 servicios para el desarrollo, las métricas del consumo de cada uno se detallan a continuación en megabytes (MB):

Tabla 5: Comparación de consumo de memoria RAM entre procesos de despliegue de OdontoApp.

Servicio	**Proceso común**	**Proceso con Docker**
Aplicación Frontend	1024	865
Aplicación Backend	25	11
Base de datos	12	13
Total	1061	889

Fuente: Elaboración propia.

Procesamiento: La última comparativa de recursos, pero no menos importante, es la de los porcentajes de uso de CPU de los procesos de despliegue común y con Docker. El proceso común instala los programas directamente en el sistema operativo, lo que permite darle acceso directo sin límite a los recursos de hardware como CPU. Esto no solo pone en riesgo la estabilidad del sistema operativo completo al poder tomar indefinidamente toda la capacidad de procesamiento, sino que también pone en riesgo el computador por algún virus debido a que los procesos no están aislados. Por otro lado, los servicios en Docker Compose pueden establecer límites individuales de acceso a los recursos de hardware como CPU, y de llegar a superarlos eliminar los contenedores sin poner en riesgo la estabilidad del sistema operativo. Otra ventaja importante es que todos los contenedores son procesos aislados, lo que evita que algún virus infecte el computador por completo. A continuación, presentamos las métricas del uso de núcleos (hilos) de CPU en porcentajes para la ejecución de los servicios del ambiente de desarrollo.

Tabla 6: Comparación de consumo de CPU entre procesos de despliegue de OdontoApp.

Servicio	Proceso común	Proceso con Docker
Aplicación Frontend	19	25
Aplicación Backend	10	18
Base de datos	5	1
Total	34	44

Fuente: Elaboración propia.

Una vez analizado el uso de los recursos más importantes del computador de un desarrollador, se define la siguiente tabla comparativa resumen que establece el porcentaje general de consumo de recursos de hardware por cada proceso de aprovisionamiento.

Tabla 7: Comparación de uso de recursos total entre procesos de despliegue de OdontoAp.

Indicador	Recursos	Proceso Común		Proceso Docker	
		Valor	Porcentaje	Valor	Porcentaje
Almacenamiento	2048	325	15,87 %	117	5,71 %
RAM	6144	1061	17,27 %	889	14,47 %
CPU	100	34	34,00 %	44	44,00 %
Total			67,14 %		64,18 %

Fuente: Elaboración propia.

Resultados y discusión

La investigación ha permitido determinar que es posible virtualizar en contenedores los servicios del ambiente de desarrollo de OdontoApp para estandarizar su aprovisionamiento, permitiendo optimizar los recursos de hardware de los computadores y el tiempo que los desarrolladores invierten en desplegar sus ambientes de trabajo.

En la tabla 8 podemos resumir que el proceso de aprovisionamiento y despliegue del ambiente de desarrollo en contenedores con Docker, resulta más eficiente con el uso de recursos de hardware del computador, requiere menos intervención del desarrollador para ejecutarlo, además de optimizar uno de los recursos más importantes, el tiempo de aprovisionamiento.

Tabla 8: Resumen comparativo del proceso de despliegue común de OdontoApp y el proceso con Docker.

Indicador	Unidad de medida	Proceso Común	Proceso Docker
Cantidad de operaciones	Tareas	13	3
Tiempos de operación total	Minutos	130	41
Cantidad de recursos	Uso de recursos	67,14 %	64,18 %

Fuente: Elaboración propia.

Esta investigación además quiere exponer a la comunidad científica que el aislamiento de procesos en contenedores y la optimización de recursos de hardware y tiempo del desarrollador, puede habilitar la posibilidad de implementar en un mismo computador varios proyectos de software, aumentado la productividad del desarrollador. También permitiría, a través del registro de imágenes de las aplicaciones, la integración y despliegue continuo, la estandarización del ambiente de producción y su integración con orquestadores de contenedores para su escalamiento horizontal.

Conclusiones

Se ha evidenciado que la virtualización en contenedores de los servicios para el proceso de despliegue del ambiente de desarrollo, sí reducen el tiempo necesario de aprovisionamiento y sí automatiza las tareas que comúnmente el desarrollador tiene que ejecutar para instalar su ambiente de trabajo.

También se ha expuesto que los beneficios de la contenedorización no solo están en los ambientes de desarrollo. Los contenedores pueden estar presentes en todo el ciclo de vida y distribución del software; permitiendo a las áreas de desarrollo, DevOps, QA, entre otras, ser más eficientes en su trabajo diario.

Por último, podemos concluir que la popularidad, el aporte comunitario, el soporte y la integración con herramientas de terceros que disponen los contenedores de Docker, tienen un nivel muy alto y que hoy en día en la industria del software los contenedores son un estándar de facto para el despliegue de aplicaciones nativas en la nube.

Referencias

1. Bhardwaj, A., & Krishna, C. R. (2019). A Container-Based Technique to Improve Virtual Machine Migration in Cloud Computing. IETE Journal of Research, 0(0), 1–16. https://doi.org/10.1080/03772063.2019.1605848

2. Guzmán, P. C., Gorostiaga, F., & Sánchez, C. (2018). i2kit: A Deployment Tool with the Simplicity of Containers and the Security of Virtual Machines. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11233 LNCS, 81–95. https://doi.org/10.1007/978-3-030-02922-7_6
3. Kaur, R., & Sengupta, J. (2010). Phased model for component driven approach in software development lifecycle. ICCET 2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings, 4, 113–116. https://doi.org/10.1109/ICCET.2010.5485696
4. Khalyly, B. El, Belangour, A., Erraissi, A., & Banane, M. (2020). Devops and microservices based internet of things meta-model. International Journal of Emerging Trends in Engineering Research, 8(9), 6254–6266. https://doi.org/10.30534/ijeter/2020/217892020
5. Kotouza, M. T., Psomopoulos, F. E., & Mitkas, P. A. (2020). A dockerized framework for hierarchical frequency-based document clustering on cloud computing infrastructures. Journal of Cloud Computing, 9(1). https://doi.org/10.1186/s13677-019-0150-y
6. Marathe, N., Gandhi, A., & Shah, J. M. (2019). Docker swarm and kubernetes in cloud computing environment. Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019, 2019-April(Icoei), 179–184. https://doi.org/10.1109/icoei.2019.8862654
7. Mens, T., Cataldo, M., & Damian, D. (2019). The social developer: The future of software development [guest editors' introduction]. IEEE Software, 36(1), 11–14. https://doi.org/10.1109/MS.2018.2874316
8. Mohan, V., Ben Othmane, L., & Kres, A. (2018). BP: Security concerns and best practices for automation of software deployment processes: An industrial case study. Proceedings - 2018 IEEE Cybersecurity Development Conference, SecDev 2018, 21–28. https://doi.org/10.1109/SecDev.2018.00011
9. Pfeiffer, R. H. (2020). What constitutes Software?: An Empirical, Descriptive Study of Artifacts. Proceedings - 2020 IEEE/ACM 17th International Conference on Mining Software Repositories, MSR 2020, 481–491. https://doi.org/10.1145/3379597.3387442
10. Piedade, B., Dias, J. P., & Correia, F. F. (2020). An empirical study on visual programming docker compose configurations. Proceedings - 23rd ACM/IEEE International Conference on

Model Driven Engineering Languages and Systems, MODELS-C 2020 - Companion Proceedings, 403–412. https://doi.org/10.1145/3417990.3420194

11. Schmitt, A., Theobald, S., & Diebold, P. (2019). Comparison of Agile Maturity Models. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11915 LNCS. https://doi.org/10.1007/978-3-030-35333-9_52
12. Watada, J., Roy, A., Kadikar, R., Pham, H., & Xu, B. (2019). Emerging Trends, Techniques and Open Issues of Containerization: A Review. IEEE Access, 7, 152443–152472. https://doi.org/10.1109/ACCESS.2019.2945930
13. Yadav, R. R., Sousa, E. T. G., & Callou, G. R. A. (2018). Performance comparison between virtual machines and docker containers. IEEE Latin America Transactions, 16(8), 2282–2288. https://doi.org/10.1109/TLA.2018.8528247
14. Zhao, X., Tian, J., & Xue, L. (2020). Herding and Software Adoption: A Re-Examination Based on Post-Adoption Software Discontinuance. Journal of Management Information Systems, 37(2), 484–509. https://doi.org/10.1080/07421222.2020.1759941
15. Zheng, Y., Jin, D., & Nicol, D. M. (2012). Validation of application behavior on a virtual time integrated network emulation testbed. Proceedings - Winter Simulation Conference. https://doi.org/10.1109/WSC.2012.6465240
16. Zhou, X., & Liu, Y. (2010). Toward proactive knowledge protection in community-based software development. Proceedings - International Conference on Software Engineering, 76–83. https://doi.org/10.1145/1833310.1833323
17. Zhu, H., & Bayley, I. (2018). If Docker is the Answer, What is the Question? Proceedings - 12th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2018 and 9th International Workshop on Joint Cloud Computing, JCC 2018, 152–163. https://doi.org/10.1109/SOSE.2018.00027