https://doi.org/10.22201/dgtic.ctud.2023.1.1.15

Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 72 - 81

Uso de contenedores para la construcción de productos de software

Información del reporte:

Para citar este reporte técnico:

Barajas González, L. D. (2023). Uso de contenedores para la construcción de productos de software. *Cuadernos Técnicos Universitarios de la DGTIC, 1* (1), páginas (72 - 81). https://doi.org/10.22201/dgtic.ctud.2023.1.1.15

Luis Daniel Barajas González

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

ldanielbg@comunidad.unam.mx

ORCID: 0009-0001-7515-4667

Resumen:

La tecnología de contenedores ha surgido como una alternativa efectiva para agilizar el desarrollo de *software*. Encapsular las herramientas y librerías de programación en entornos de ejecución virtuales, permite que sean compartidos en el equipo de trabajo sin requerir tiempo o esfuerzo excesivo para su instalación y configuración. Lo anterior crea ambientes de trabajo homogéneos que previenen problemas de incompatibilidad.

Palabras clave:

Contenedores, Docker, ambientes, software.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 73 - 81

1. INTRODUCCIÓN

Las herramientas de software mínimas que conforman un ambiente de desarrollo para sistemas web transaccionales son:

a) un servidor web que habilite el acceso al sistema vía Internet,

b) un manejador de bases de datos para almacenar y recuperar datos,

c) un lenguaje de programación para construir componentes ejecutables de *software*,

d) un servidor de correo para el envío y recepción de mensajes del sistema,

e) un editor de código, y

f) un programa que permita interactuar con la base de datos.

Figura 1

Conjunto de herramientas mínimas para desarrollo de un sistema web

Todos estos elementos deben ser instalados en el equipo de cómputo de cada programador que participe en el desarrollo de un sistema web transaccional.

Los principales retos a los que se enfrenta un programador al trabajar en un esquema como el antes descrito son:

1. Instalar y configurar cada uno de estos programas en el equipo de cómputo asignado.
2. Lograr que todos los programadores cuenten con un ambiente de trabajo configurado con los mismos elementos de *software*.

Surgen problemas de incompatibilidad en los componentes de *software* construidos cuando los ambientes de programación divergen, por ello, la adopción de prácticas y herramientas de trabajo permite superar los retos antes planteados a través del encapsulamiento de programas de *software* dentro de contenedores.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 74 - 81

2. OBJETIVO

Compartir el conocimiento adquirido sobre el uso de contenedores que permite reducir el tiempo requerido para instalar y configurar ambientes de programación para sistemas web en equipos de trabajo.

3. ANTECEDENTES

La herramienta *XAMPP* era utilizada para programar sistemas web en la Subdirección de Sistemas Integrados de la DGTIC; esta herramienta es un paquete de *software* que simplifica la instalación de un servidor web *Apache* configurado para interpretar al lenguaje *PHP*. También se instalaba un manejador relacional de bases de datos como *PostgreSQL* o *MySQL* para persistir los datos del sistema. Estos programas eran instalados directamente sobre el sistema operativo del equipo y sus versiones podían variar conforme a los requerimientos de cada proyecto.

Constantemente se desarrollaban nuevos sistemas y se modificaba la funcionalidad de sistemas legados. Por lo tanto, era necesario tener disponibles distintas versiones de las herramientas. Por ejemplo, para modificar un sistema construido con *PHP* versión 7 y al mismo tiempo trabajar en un nuevo sistema con *PHP* versión 8.2, se necesita contar la instalación de ambas versiones del lenguaje.

3.1 PROBLEMÁTICA

A. Instalar y configurar ambientes locales para sistemas distintos.

Para utilizar diferentes versiones de *PHP* en un mismo equipo era necesario instalar más de una instancia de *XAMPP*, configurar los puertos de los servidores web *Apache* y *PostgreSQL* para evitar colisiones con instalaciones anteriores de ambos programas, así como establecer la ruta de ejecución de *PHP* en el sistema operativo para que coincida con el proyecto en turno.

Lo anterior presentaba algunos problemas, como el uso de espacio en disco para programas duplicados y la tarea adicional de modificar la configuración de variables del sistema operativo cuando era necesario pasar de un proyecto a otro para que funcionara sobre el ambiente adecuado. En ocasiones era importante desinstalar herramientas de versiones nuevas y reemplazarlas con otras antiguas para mantener la compatibilidad.

B. Ambientes de programación en equipos de trabajo

Al trabajar con otros programadores, un reto es lograr que todos y cada uno de ellos tenga los elementos de *software* instalados y configurados con las mismas versiones y parámetros. Aunque esto pudiera parecer solamente una cuestión de buena comunicación entre personas, en la realidad el ambiente de cada integrante del equipo es diferente: pueden tener distintos sistemas operativos (*Windows*, *Linux* o *MacOS*), o bien, diferentes versiones de un mismo sistema operativo (*Windows 10* o *Windows 11*); también pueden tener instaladas y configuradas versiones antiguas del lenguaje de programación, debido a su participación en proyectos previos.

Cuando divergen demasiado los ambientes de programación de un mismo equipo de trabajo, aparece un fenómeno en el cual la funcionalidad construida por un programador solo funciona en su equipo, lo cual es inaceptable porque los componentes de *software* construidos entre todos los miembros del equipo deben funcionar en armonía sobre un ambiente tecnológico homogéneo.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 75 - 81

3.2 ALTERNATIVAS DE SOLUCIÓN

A continuación, se desglosan algunas soluciones que se exploraron para subsanar la problemática antes descrita.

A. Uso de máquinas virtuales

El aislamiento de ambientes de ejecución de sistemas de *software* tiene sus orígenes en diversos esfuerzos orientados a la virtualización de equipos de cómputo donde destacan nombres como *VMWare*, *Oracle VirtualBox* y *Virtuozzo*. La principal ventaja que ofrece la virtualización es compartir recursos de almacenamiento, memoria y procesador de un solo equipo físico entre muchos equipos virtuales cuya demanda de recursos sea significativamente moderada (Younge, et al, 2011).

El análisis de esta solución concluyó que el uso de máquinas virtuales para el desarrollo de *software* es una solución parcial al problema de instalar los componentes del ambiente de desarrollo sin afectar al equipo anfitrión. Adicionalmente, se identificó que esta opción consume demasiados recursos de almacenamiento y memoria debido a que implica la instalación de un sistema operativo completo, que se ejecutará sobre el equipo anfitrión. Para el desarrollo de *software*, no son necesarias muchas de las librerías, programas y funcionalidades instaladas como parte del sistema operativo de la máquina virtual, y constituyen un desperdicio de recursos. Es por lo anterior, que la virtualización completa de equipos no es la mejor solución para aislar ambientes de desarrollo de *software*.

B. Tecnología de Contenedores

La tecnología de contenedores ha sido un parteaguas para el desarrollo de *software*, porque permite aislar el ambiente de ejecución de una aplicación, del resto del sistema sobre el que funciona. Esto permite conformar arquitecturas de sistemas conectando servidores web, manejadores de bases de datos, lenguajes de programación y cuanto *software* sea necesario sin afectar directamente el ambiente operativo del equipo anfitrión.

Respecto al uso de recursos, los contenedores ocupan menos espacio que las máquinas virtuales debido a que no instalan programas adicionales innecesarios para el desarrollo de *software*. Además, todos los contenedores comparten el mismo *kernel* del sistema operativo para acceder a los recursos computacionales del equipo donde se ejecutan (Preeth, Mulerickal, et al., 2015).

C. Paquete de software *XAMPP*

La herramienta *XAMPP* es un apilamiento (*stack*) de tecnologías que se usan para desarrollar sistemas web, que pueden instalarse fácilmente a través de un programa que guía paso a paso el proceso. La ventaja principal de este tipo de herramientas es que simplifican la puesta en marcha del ambiente de desarrollo, y conjunta un servidor web que además interpreta *scripts* de *PHP* y con múltiples librerías adicionales al lenguaje. La desventaja está en que los programas se instalan directamente sobre el equipo de cómputo y quedan acoplados al equipo, a su ruta de ejecución (*PATH*) y a los puertos de red. En caso de necesitar versiones más nuevas o antiguas de *XAMPP* es mandatorio instalar una instancia diferente y modificar la configuración del equipo.

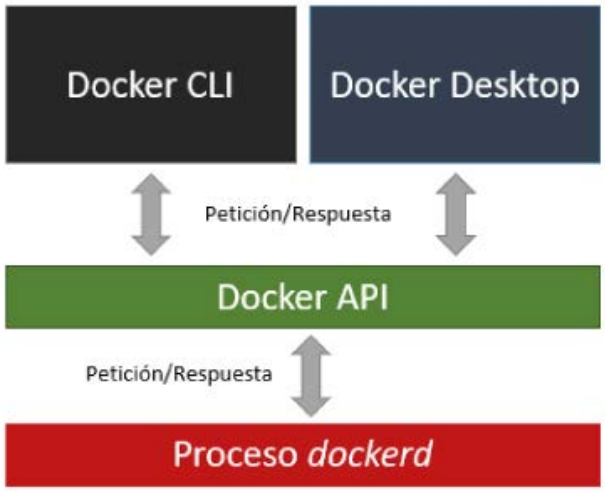
https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 76 - 81

4. LA PLATAFORMA DOCKER

La plataforma que popularizó el uso de contenedores fue *Docker* en el 2013 al lanzar su "motor" de contenedores conocido como "*docker engine*", el cual permite crear y ejecutar contenedores de aplicaciones. *Docker* funciona como una arquitectura cliente-servidor, donde existe un proceso servidor conocido como "*dockerd*" que se ejecuta en segundo plano y es el responsable de construir y ejecutar los contenedores. Los clientes utilizan la interfaz de programación de aplicaciones (*API*) de *Docker* para crear y utilizar los contenedores. Existen clientes gráficos tales como *Docker Desktop*, pero también es posible utilizar la interfaz de línea de comandos (*CLI*) de *Docker* para interactuar con la herramienta.

Figura 2

Arquitectura cliente-servidor de la plataforma Docker



4.1 COMPONENTES DE *DOCKER* COMO PLATAFORMA

Docker proporciona tres herramientas principales. La primera de ellas es el motor (*Docker engine*) que funciona como el gestor principal de otros elementos relacionados con la tecnología de componentes, tales como las imágenes, los contenedores, los volúmenes y las redes virtuales. La segunda herramienta son los clientes que se mencionaron antes: la *API* de línea de comandos y el cliente gráfico *Docker Desktop*, los cuales permiten el acceso a la funcionalidad del motor. La tercera herramienta es el servicio de alojamiento de imágenes conocido como *Docker Hub*, el cual es de uso público y permite subir y compartir imágenes con la comunidad de usuarios (Overview of Docker Hub, 2023).

4.1.1 OBJETOS DE *DOCKER*

Al trabajar con contenedores para el desarrollo de sistemas se identifican cuatro tipos de objetos comunes que son: 1) las imágenes, 2) los contenedores, 3) los volúmenes, y 4) las redes.

Las imágenes pueden comprenderse como una especie de plantillas a partir de las cuales se crean los contenedores. Las imágenes contienen los archivos binarios y librerías necesarias para ejecutar un tipo

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 77 - 81

específico de aplicación. A partir de una imagen es posible crear múltiples contenedores para aplicaciones distintas. Haciendo un símil, las imágenes son parecidas a los archivos *ISO* de las distribuciones *Linux* que pueden descargarse de Internet para generar un disco de arranque ya sea sobre un disco compacto o sobre un dispositivo *USB*. Estos archivos *ISO* contienen el *software* necesario para instalar y hacer funcionar *Linux* en una computadora personal. Es posible crear imágenes personalizadas a partir de archivos conocidos como *Dockerfiles*, los cuales son archivos de texto plano que contienen las instrucciones de instalación de los archivos binarios y las librerías con la imagen en cuestión. Posteriormente, estos archivos son procesados por un comando de *Docker* para generar la imagen.

Los contenedores son ambientes de ejecución para aplicaciones, estos nos permiten "encapsular" un sistema a la medida junto con sus dependencias para que se ejecute correctamente sin ser afectado por los cambios en la configuración del equipo anfitrión. Los contenedores se crean a partir de una imagen base y es posible configurar variables propias de cada instancia, por ejemplo, los puertos de red que mantendrán abiertos, o las contraseñas de los servicios que contienen.

Los volúmenes son espacios de almacenamiento administrados por *Docker* que permiten almacenar archivos de trabajo tales como el código fuente para usarlos desde un contenedor. Para usarlos, es necesario crearlos y "montarlos" sobre algún directorio del contenedor durante la creación del mismo. El contenedor tendrá acceso a los archivos del volumen de forma transparente.

Docker permite crear **redes virtuales** con el propósito de comunicar contenedores entre sí. Por ejemplo, un contenedor "A" que ejecuta una aplicación web, conectado a la misma red virtual que un contenedor "B" que ejecuta un manejador de bases de datos, podrá establecer conexión para consultar y almacenar datos en este último.

Figura 3

Objetos de Docker

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 78 - 81

5. CASO DE USO

A continuación, se presenta la creación de un ambiente de desarrollo basado en contenedores de *Docker* para una aplicación web. Está conformado por una red virtual de *Docker*, un contenedor para la base de datos, otro contenedor para un servidor de correo electrónico y uno más para ejecutar el sistema web. Este ambiente sirvió para el desarrollo de un sistema para el archivo histórico de un cuerpo colegiado de la Universidad, el cual fue desarrollado con el marco de trabajo *Laravel* que utiliza los lenguajes de programación *Javascript* y *PHP*.

Primeramente se creó una red virtual para conectar y comunicar todos los contenedores del ambiente de desarrollo.

Figura 4

Instrucción para crear la red virtual con Docker

```
docker network create local-net
```

Se utilizó una imagen del manejador de bases de datos *PostgreSQL* versión 15 para crear un contenedor para la base de datos. Se utilizó un volumen de *Docker* para que los datos quedaran almacenados independientemente del contenedor, lo que permite destruir y volver a generar el mismo sin perder información del sistema.

Figura 5

Instrucciones para la creación del contenedor de la base de datos

```
docker run -itd --name postgres-15 \
    --network local-net \
    -e POSTGRES_USER=postgres \
    -e POSTGRES_PASSWORD=s3cret \
    -p 5432:5432 \
    -v postgres-data:/var/lib/postgresql/data \
    -h postgres.hostname \
    postgres-15:latest
```

También se creó un servidor de correo electrónico falso utilizando la imagen *Docker* del *software Mailhog*. Este *software* recibe las peticiones de envío de correo electrónico del sistema y muestra los mensajes en una página web simulando ser un cliente de correo real.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 79 - 81

Figura 6

Instrucciones para la creación del contenedor de Mailhog

```
docker run -d \
    -e "MH_STORAGE=maildir" \
    -h mailhog.hostname \
    --network local-net \
    -p 1025:1025 \
    -p 8025:8025 \
    -v mailhog-maildir:/maildir \
    mailhog/mailhog
```

El contenedor que ejecuta la aplicación web está basado en una imagen personalizada que conjunta un servidor web *Apache*, el lenguaje de programación *PHP* en su versión 8, el entorno de ejecución *Node JS,* el manejador de dependencias *Composer* y diversas librerías para que funcione el marco de desarrollo *Laravel*. Esta imagen personalizada está disponible para descarga desde *Docker Hub* a través de la dirección electrónica https://hub.docker.com/repository/docker/danielbg1409/webphp-8.2/general.

Figura 7

Instrucciones para crear el contenedor de la aplicación web

```
docker run -itd \
    --name junta-gobierno \
    -v v-junta-gobierno:/var/www/html \
    --network local-net \
    -p 80:80 \
    -p 5173:5173 \
    -h junta-gobierno.hostname \
    danielbg1409/web-php-8.2:latest
```

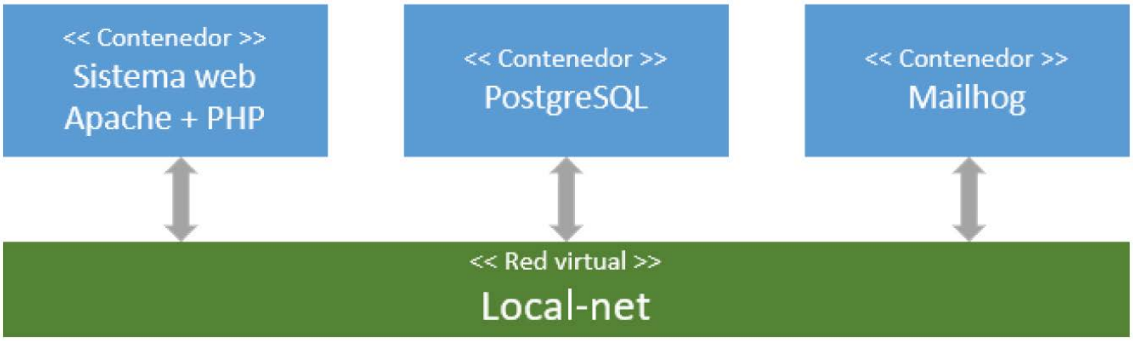
Este último contenedor también utilizaba un volumen de *Docker* para mantener el código fuente del sistema desacoplado del contenedor.

Al momento de crear cada contenedor se conectaba a la red mediante el parámetro *"network"*, esto permitió que el contenedor del sistema web estableciera conexión con la base de datos y con el servidor de envío de correo.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 80 - 81

Figura 8

Los contenedores del ambiente conectados entre sí



Una vez creados los contenedores, se utilizaron editores de código y clientes de bases de datos para trabajar directamente sobre ellos. El código fuente, así como los datos quedaban guardados en volúmenes de *Docker*, de tal forma que, si el contenedor era destruido, no había ninguna pérdida de información porque el contenedor y el volumen son gestionados por *Docker* como objetos independientes.

Otra de las ventajas de *Docker* fue que al separar los componentes en tres contenedores distintos se reutilizaron los contenedores de *PostgreSQL* y de *Mailhog* en otros proyectos. También se reutilizó la imagen personalizada del servidor web en otros proyectos que utilizan el mismo conjunto de tecnologías.

Debido a que los programas quedan encapsulados en contenedores, no afectan a otros programas instalados en el equipo anfitrión, tampoco alteran sus variables de entorno ni la ruta de ejecución del sistema operativo. Esto permite tener contenedores con ambientes de desarrollo para el mantenimiento de sistemas legados junto con ambientes actualizados sin que se afecten entre ellos.

Los programadores del equipo de trabajo instalaron *Docker* en sus computadoras y posteriormente ejecutaron los comandos antes mencionados para habilitar su ambiente de desarrollo con los mismos elementos de *software*, evitando así problemas de incompatibilidad en los artefactos de *software* que construyeron.

6. CONCLUSIONES

El uso de contenedores para desarrollo de *software* facilita al programador la instalación de las herramientas necesarias para aplicaciones cuya arquitectura requiere diversos servicios adicionales. También facilita trabajar sobre sistemas legados porque encapsula los archivos binarios y librerías que estos necesitan para funcionar, en un entorno independiente al sistema operativo que lo contiene, a diferencia de otro tipo de soluciones basadas en *stacks* de tecnologías como *XAMPP*. Los contenedores solucionan la incompatibilidad entre ambientes de programación en proyectos con más de un programador ya que pueden crear contenedores con las mismas características, independientemente del sistema operativo o configuraciones propias de cada uno de sus equipos de cómputo.

https://doi.org/10.22201/dgtic.ctud.2023.1.1.15 Vol. 1, núm. 1. Octubre-diciembre 2023, págs. 81 - 81

REFERENCIAS BIBLIOGRÁFICAS

Overview of Docker Hub. (2023, September 19). *Docker Documentation*. https://docs.docker.com/docker-hub/

Preeth, E. N., Mulerickal, F. J. P., Paul, B. and Y. Sastri, (2015). *Evaluation of Docker containers based on hardware utilization*, International Conference on Control Communication & Computing India (ICCC), Trivandrum, India, 2015, pp. 697-700, doi: 10.1109/ICCC.2015.7432984

Younge, A. J., Henschel, R. Brown, J. T. Laszewski, G. von. Qiu, , J. and G. C. Fox. (2011). *Analysis of Virtualization Technologies for High Performance Computing Environments*, IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 2011, pp. 9-16, doi: 10.1109/CLOUD.2011.29