https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96

Vol. 3, Núm. 2. abril-junio 2025, págs. 45 - 53

Utilización de Dockers para desplegar en producción sistemas de información heredados

Información del reporte:

Licencia Creative Commons

Para citar este reporte técnico:

Chamú Arias, J. O. (2025). Utilización de Dockers para desplegar en producción sistemas de información heredados. *Cuadernos Técnicos Universitarios de la DGTIC*, 3 (2) páginas (45 - 53).

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96

José Othoniel Chamú Arias

Dirección General de Cómputo y de Tecnologías de Información y Comunicación

Universidad Nacional Autónoma de México

chamu_as@unam.mx

ORCID: 0009-0001-4949-3024

Resumen

Se aborda el análisis de un caso particular en el que se requería extraer un sistema de información heredado que estaba alojado dentro de una computadora personal y transferirlo a un ambiente de trabajo en producción, en máquinas virtuales. Se presentaron incompatibilidades a nivel de sistema operativo, versiones de librerías y lenguajes utilizados ya que, al estar en un equipo personal, esos elementos no se actualizaban, dando como resultado versiones de software antiguas y sin soporte. Se intentó la virtualización sobre Rocky Linux versión 9, pero, al ser reciente el sistema operativo, no incluía el lenguaje de programación PHP 7.4 en el que estaba desarrollado el sistema de información, lo que ocasionó una reacción en cadena en los demás componentes que dependían de dicho módulo, ocasionando que la virtualización no fuera suficiente para resolver la situación; además, utilizar versiones anteriores no era la mejor opción por riesgos de seguridad y falta de soporte. Por ello, se exploró la utilización de la plataforma de contenedores "Dockers", que es un entorno ejecutable que se monta sobre un sistema operativo principal (Rocky Linux) y que reúne los elementos necesarios para poner en operación una aplicación o un sistema de información. Sobre Docker, se montó el sistema operativo original (Debian Linux versión 11) del sistema de información, así como el paquete de las librerías y herramientas necesarias para el funcionamiento de la aplicación. Al estar encapsulado, permitió que el

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 46 - 53

peso de la seguridad se transfiriera al sistema operativo principal y que se pudieran tener los diferentes ambientes de trabajo contenerizados[1]: desarrollo, pruebas y producción dentro de la misma máquina virtual; se ganó tiempo para actualizar la información y facilitar la ejecución de pruebas funcionales y no funcionales al permitir la copia del contenedor principal.

Palabras clave:

Docker, compatibilidad, desarrollo, pruebas, producción, ambiente de trabajo, contenedor, Linux, máquinas virtuales.

Abstract

This report addresses a specific case in which a legacy information system hosted on a personal computer needed to be extracted and transferred to a production environment using virtual machines. Since the operating system, library versions, and languages used were not updated on a personal computer, some incompatibilities were found related to these components, resulting in outdated and unsupported software versions. Virtualization on Rocky Linux version 9 was attempted, but, since the operating system was new, it did not include the PHP 7.4 programming language used to develop the information system. This caused a chain reaction in the other components that depended on that module, getting to an insufficient level of virtualization to solve the problem. Using older versions was not the best option due to security risks and lack of support. Therefore, the use of the "Dockers" container platform was explored. This is an executable environment that runs on a main operating system (Rocky Linux) and brings together the necessary elements to run an application or information system. The original operating system (Debian Linux version 11) of the information system was installed on Docker, along with the package of libraries and tools necessary for the application's operation. Encapsulation allowed the security burden to be transferred to the main operating system and allowed the different work environments to be containerized: development, testing, and production within the same virtual machine. This saved time for updating information and facilitated the execution of functional and non-functional tests by allowing the main container to be copied.

Keywords:

Docker, compatibility, development, testing, production, work environment, container, Linux, virtual machines.

1. INTRODUCCIÓN

Durante los procesos de desarrollo, pruebas, liberación a producción y mantenimiento de las aplicaciones, los grupos de trabajo se enfrentan a problemas de compatibilidad entre versiones del sistema operativo y de las herramientas utilizadas, así como a las diferencias en las configuraciones de cada uno de los ambientes de trabajo como diferentes configuraciones de seguridad requeridas o el sincronizado de configuraciones de las herramientas en cada uno.

1 La contenerización se refiere al empaquetado de código de software con las bibliotecas del sistema operativo y las dependencias necesarias para ejecutar dicho código, a fin de crear un único ejecutable ligero, llamado contenedor, que se ejecuta en cualquier infraestructura.

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 47 - 53

Por ejemplo, al tener que preparar los ambientes de desarrollo para aplicaciones que utilizan la versión de PHP 7.4, se evidenció que éstas ya no cuentan con soporte en Rocky Linux 9.x y, al ser un requerimiento para el funcionamiento de la aplicación, no podrían actualizarse a la versión de PHP más reciente, por lo que se tuvo que explorar la utilización de plataformas tecnológicas que permitieran el uso de la versión de PHP con versiones anteriores del sistema operativo, sin comprometer la seguridad.

Al utilizar los Dockers como una plataforma tecnológica que apoyara el desarrollo de aplicaciones, hubo resistencia al cambio por parte de algunos desarrolladores al tener que modificar la forma de trabajo respecto al despliegue de las aplicaciones, además de introducir la curva de aprendizaje que se tiene que enfrentar al comprender el uso de contenedores y de los comandos que se tienen que utilizar para su gestión.

Por lo anterior, se comparte la experiencia a través de este reporte técnico, cuyo objetivo es mostrar el uso de Docker como una plataforma tecnológica viable en el despliegue de aplicaciones nuevas o heredadas, que ofrece portabilidad, escalabilidad y una mejor eficiencia en el uso de los recursos utilizados, además de permitir agilizar el desarrollo y liberación de las mismas al quedar empaquetadas con sus dependencias en contenedores portátiles.

2. DESARROLLO TÉCNICO

2.1 ANTECEDENTES

Actualmente, algunos de los sistemas operativos Linux han tenido un cambio en el modelo de licenciamiento por parte de los dueños de los derechos, o bien, son fusionados a otros con licenciamiento comercial, lo que obliga a cambiar los desarrollos o los sistemas de información a otras plataformas. Tal es el caso de CentOS, que era distribuido por la empresa Red Hat como software libre y de código abierto, pero que fue adquirida por IBM, la cual cambió el modo de distribución.

El tema que se presenta en este reporte técnico deriva de los cambios de licencia que han afectado a los sistemas operativos y, por consiguiente, al desarrollo de las aplicaciones. Para evitar el riesgo de quedarse sin soporte, se buscaron opciones diferentes a CentOS, entre ellas: Ubuntu, FreeBSD, Debian, Rocky y AlmaLinux. Tener múltiples opciones trae consigo complicaciones de incompatibilidad en los ambientes de trabajo, los cuales, en ocasiones, se colocan en equipos de cómputo personales, trayendo como consecuencia que la aplicación funcione solamente en su ambiente de trabajo local debido a las diferentes versiones de librerías, plataformas y sistemas operativos que se usan, lo que afecta la puesta en producción.

Para subsanar las situaciones antes mencionadas, se evaluaron algunas características que se muestran a continuación:

Máquinas virtuales

Las máquinas virtuales presentan varias ventajas, entre las que destaca que alojan un sistema operativo completo con manejo de errores y reporte de logs, lo que permite la creación de una plataforma completa. Además, el hardware completo es emulado y así se facilita la creación de tantas instancias como se requieran; además, pueden ser transferidas a diferentes equipos, siempre y cuando trabajen Linux con QEMU/KVM.

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 48 - 53

Sin embargo, también tienen desventajas: reservan recursos exclusivamente para el funcionamiento del sistema operativo huésped, como CPU, memoria RAM y disco duro; además, se requiere un proceso de conversión de formato de disco duro en caso de que un elemento del equipo tenga otro virtualizador; consumen una cantidad considerable de recursos de red para poder compartir la máquina entre diferentes equipos, lo cual requiere reservar espacio para emular el disco duro; y, asimismo, es más complejo controlar las versiones de las diferentes máquinas y equipos en la nube o en la empresa.

Dockers

La utilización de Dockers presenta varias ventajas, entre las que destaca que ésta ayuda a homologar los ambientes de trabajo de desarrollo y mejora la precisión del control de versiones; además, ofrece mayor comodidad a los equipos de desarrollo para emplear el sistema operativo que mejor les funcione; La creación, actualización y borrado de las imágenes es más ágil y centralizada, lo que permite abordar el problema de la incompatibilidad: eliminar los efectos de las diferencias entre los equipos y mitigar los riesgos al colocar el sistema en producción; también ofrece características que permiten crear múltiples instancias de los contenedores, esto es, incrementar o decrementar su número según se requiera.

En caso de cambios de licenciamiento o de políticas del proveedor del software, la comunidad de desarrollo cuenta con varias alternativas de plataformas compatibles que ofrecen el uso de contenedores. De acuerdo con Bruno y Bruno (2020), entre otras opciones, destacan RKT, PodMan, Singularity, Linux Containers – LXC y OpenVz. El Docker es utilizado por grandes compañías como ING, PayPal, ADP y Spotify (Novoseltseva, 2023). Por otro lado, presenta un comportamiento similar a Java, ya que es multiplataforma y consume pocos recursos. "Para ejecutar un contenedor no se necesita hypervisor porque no se ejecuta un sistema operativo invitado y no hay que simular hardware" (Gallego & Chico Guzmán, 2017).

Los contenedores son elementos ligeros en el uso de recursos del sistema operativo y su despliegue es más ágil una vez creada la imagen, es decir, la plantilla con las instrucciones que definen un contenedor. Además, permite desplegar ambientes heredados y, de acuerdo con Dock, M. (2025) "no tiene costo para organizaciones pequeñas o medianas con menos de 250 empleados y que generen ingresos menores a 10 millones de dólares anuales. Para organizaciones mayores, implica usar algún plan empresarial". En términos de seguridad, al estar encapsulado en un sistema operativo huésped, se delegan las actualizaciones y medidas de seguridad perimetrales como son el cierre de puertos, las actualizaciones de librerías, la segmentación de redes internas, el uso de proxy para administrar el certificado o el uso de un *firewall* de aplicaciones web; la escalabilidad que se tiene con esta tecnología es más práctica al aumentar el número de contenedores en lugar de crear más máquinas virtuales. Al no tener que emular el hardware con todos sus componentes, el rendimiento es mucho más ligero, pues se consumen menos recursos.

Sin embargo, el uso de Docker también presenta desventajas, como la curva de aprendizaje para su utilización, que es más alta, ya que requiere conocimientos básicos de servidores y *firewalls* por parte de los equipos de desarrollo. Los contenedores, al ser sistemas recortados, pueden apagarse al generarse un error, debido a que no tienen todos los elementos para generar un *core dump*[2]; aunado a esto, modificar un archivo de configuración es una tarea que requiere mayor experiencia técnica, pues implica

2 Core dump: También conocido como volcado de memoria, es una copia del estado de la memoria que se crea cuando se produce un error o fallo de un programa o aplicación.

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 49 - 53

muchos pasos para recrear el contenedor con las nuevas características, lo cual implica borrarlo, crearlo, respaldarlo, restaurarlo, ajustar puertos y, finalmente, verificar que todo funcione adecuadamente.

La protección de la seguridad es más compleja ya que, dependiendo de cómo se crea un contenedor, hay que cerrar o abrir múltiples puertos; también se debe considerar lo que se requiere para que, en caso de intrusión, ésta no absorba todos los recursos del equipo huésped, por mencionar algunos.

A partir de lo anterior, se presenta la Tabla 1 para visualizar las ventajas de seleccionar Dockers en lugar de máquinas virtuales:

Tabla 1

Comparativa entre un Docker y una máquina virtual

Tecnología	Emulación de hardware	Ejecución de múltiples sistemas operativos en un mismo equipo físico	Separar componentes de hardware	Ejecución de una aplicación en ambiente productivo	Ejecutar en diferentes tipos de hardware y sistemas operativos	Capacidad de Escalamiento
Máquina Virtual	Sí	Sí	Sí	Sí	Sí	Sí
Docker	No	No	Sí	Sí	Sí	Sí

Es importante destacar que Docker se ejecuta en diversas plataformas de hardware sin emular todo el componente físico, dando como resultado menor consumo de recursos.

Asimismo, si se cambian las reglas del *firewall*, éstas se borran y hay que reiniciar los servicios y contenedores. Si se requiere un cambio en la imagen del contenedor, es probable que se tenga que reconstruir. La compatibilidad de las versiones antiguas está atada a la versión del cliente con la que se creó y sobre la que se requiere ejecutar. Para evitar problemas como los mencionados, es necesario un orquestador, ya sea Kubernetes, Docker Swarm, entre otros.

Docker, la herramienta seleccionada, funciona en los sistemas operativos más populares y utilizados en la UNAM. Racero & Som (2022) mencionan que "los contenedores son un paquete de elementos que permiten ejecutar una aplicación determinada en cualquier sistema operativo", ya que permiten crear un ambiente desde un repositorio central, donde se puede controlar y administrar el control de las versiones. Al estar centralizado, se puede tener un control más preciso de los cambios en los ambientes de los diferentes equipos y procesos. Además, esta herramienta permite que los equipos de trabajo usen los sistemas operativos que tienen en sus equipos personales, por ejemplo: personal de desarrollo que trabaje en Windows, personal de aseguramiento de calidad que ocupe MacOS, expertos en seguridad que usen Ubuntu y los responsables del ambiente de producción que trabajen en Debian. Dicho de otra manera, el motor y la imagen funcionan siempre y cuando se encuentren dentro de la lista de los sistemas operativos más usados por las comunidades de desarrollo.

Se utiliza Dockers cuando se requieren despliegues de ambientes en poco tiempo, independientes de plataforma, con facilidad de escalamiento tanto vertical como horizontalmente. Por ello, la aplicación debe estar preparada para funcionar en dicho entorno, por ejemplo, si es una aplicación que guarda el

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 50 - 53

estado del usuario, debe almacenarlo en base de datos y las credenciales de conexión a los recursos deben ser configurados con variables de ambiente para reducir o evitar modificaciones a los contenedores.

Sobre los ambientes de trabajo, Tirado (2017) destaca a "Docker y docker-compose, herramienta que nos ayuda a gestionar aplicaciones multicontenedor, en este caso, nos ayudan a la administración de los ambientes de desarrollo, facilitando la forma de trabajo y minimizando los recursos que necesitamos para la estandarización de los ambientes de trabajo".

Como plataforma, ayuda a poner en marcha sistemas monolíticos, y permite un mejor aprovechamiento de los recursos de hardware de los servidores. Conforme pasa el tiempo, se va adoptando cada vez más: un estudio de 2018 arroja que hay "alrededor de 700 millones de contenedores en uso" (MuyComputerPro, 2018).

3. IMPLEMENTACIÓN

Para el uso de Dockers dentro de los ambientes de trabajo (desarrollo, pruebas, pre-producción o producción), se requiere de la instalación en una máquina virtual o servidor físico y del desarrollo de las siguientes etapas:

- Creación de la imagen del Docker
- Ejecución del contenedor
- Publicación y distribución de la imagen

3.1 CREACIÓN DE LA IMAGEN DEL DOCKER

Para la creación de la imagen del Docker, se eligió como sistema operativo Rocky Linux 9, que es una versión libre compatible con Red Hat del cual derivan diversas distribuciones de Linux y una alternativa a CentOS, el cual fue descontinuado y que se utilizaba en los desarrollos dentro del área donde trabajo. Sin embargo, también puede utilizarse en otras distribuciones de Linux como son Ubuntu y Debian.

Se instaló Rocky Linux 9 en una máquina virtual, siguiendo los pasos predeterminados del instalador, sin modificaciones ni configuraciones adicionales. Para la creación de la imagen del Docker, es necesario instalar algunas librerías y clientes (Docker Engine) que son descargados del Repositorio de Docker[3] de acuerdo con el sistema operativo utilizado.

A continuación, se debe configurar el ambiente del contenedor en el sistema operativo para otorgar privilegios a la cuenta de usuario que va a gestionar el Docker, y se abren los puertos de comunicación que utilizará el contenedor de acuerdo a los requerimientos de la aplicación a desarrollar o que residirá en él.

Posteriormente se creará el contenedor a partir de una imagen de prueba que se encuentra en el Repositorio de Docker, tomando como base el archivo llamado Dockerfile, el cual contiene la receta para crear la imagen como se muestra a continuación (Creación de Imágenes A Partir de un Dockerfile, s. f.):

```
FROM debian:buster-slim
```

3 Repositorio de Docker: https://docs.docker.com/engine/

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 51 - 53

```
MAINTAINER José Domingo Muñoz "josedom24@gmail.com"
RUN apt-get update  && apt-get install -y  apache2
COPY index.html /var/www/html/
CMD ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

La línea "MAINTAINER" puede ser modificada u omitida por quien genere la imagen del Docker.

Para ejecutar el archivo Dockerfile se utiliza el comando:

```
docker build -t usuario/Prueba:1.0
```

con el cual se leen las instrucciones del archivo. Si no encuentra errores, crea la imagen del contenedor.

Se puede verificar su creación a través de:

```
docker image ls
```

3.2 EJECUCIÓN DEL CONTENEDOR

Una vez creado el contenedor, se tiene que poner en funcionamiento a través del siguiente comando:

```
docker run -d  --name  incripcionesweb1    usuario/Prueba:1.0
```

Para verificar que se encuentre en ejecución el Docker, se puede verificar con el comando:

```
docker ps
```

Éste mostrará una lista de los contenedores que se encuentran en ejecución.

3.3 PUBLICACIÓN Y DISTRIBUCIÓN DE LA IMAGEN

Una vez realizado el paso anterior, se puede publicar el resultado en un repositorio privado como GitHub, o bien, en Docker Hub, que es un sitio público o privado donde se concentran las imágenes para compartirlas desde la nube, el cual es "el repositorio en la nube de Docker donde se encuentran todas las imágenes de los contenedores" (Ponsico Martín, 2017). Por ejemplo, la liga https://hub.docker.com/r/jchamu/php8-usuario3 contiene una imagen para ejecutar ambientes de PHP y es accesible para todas las áreas de trabajo. De igual forma, cuenta con características de versionado, descripción de contenido, instrucciones para crear las imágenes e instrucciones para su puesta en operación.

La implementación realizada se usó en el portal web de un proyecto a cargo de la DGTIC, el cual tenía las limitantes de rescatar un sistema de información con las versiones exactas, que para este caso eran CouchDB 2.3.1, Directus 9.12.1, PHP 7.4 y Mysql 8.0, con restricciones de tiempo. Se trabajó en dos ambientes (desarrollo y producción), la meta del tiempo se logró y todas las versiones del software funcionaron correctamente con el uso de dicha tecnología, lo que permitió simplificar los retos técnicos derivados de que la versión "PHP 7.4" ya no tenía soporte en Rocky Linux 9, que era la versión mínima que se tenía en el ambiente de producción. La forma de subsanar estas restricciones y evitar más incompatibilidades fue el uso de los contenedores. La seguridad se distribuyó teniendo un sistema operativo reciente con todas

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 52 - 53

las actualizaciones, como es el caso de Rocky 9 y en el contenedor incorporando un *firewall* web en el Apache como lo es mod_security[4].

4. RESULTADOS

El uso de Docker, en el caso mencionado anteriormente, permitió el funcionamiento de una aplicación heredada que requería versiones anteriores, manteniendo la seguridad a través del sistema operativo de la máquina virtual que lo aloja y permitiendo su traslado a un ambiente de producción sin modificaciones, es decir, sin tener que agregar o quitar librerías, así como cambiar valores en variables de ambiente.

Asimismo, se lograron reducir los tiempos de uno a dos días a una hora por ambiente, debido a que se cuenta con un ambiente estandarizado en el contenedor que se podría portar sin modificaciones a desarrollo, pruebas y producción, es decir, al no tener que estar instalando y configurando cada tipo de ambiente, ajustando variables y permisos en directorios, validar versiones de librerías, entre otras cosas.

Lo anterior simplifica las labores de administración de las máquinas virtuales, porque agiliza la atención de solicitudes de ambientes de desarrollo, pruebas y producción, evitando tener solicitudes de ambientes de aplicaciones encoladas por más de dos días.

De igual manera, el uso de Dockers ayudó a reducir las incompatibilidades que se tenían entre los diferentes ambientes de trabajo, a centralizar las imágenes del sistema en un repositorio, controlar los cambios al tener versionado de dichas imágenes y abrió las puertas a un nuevo tipo de despliegue de aplicaciones distribuidas. Hoy, se cuenta con una nueva opción que no implica solamente máquinas virtuales y que ha sido una experiencia favorable en casi un año de uso de los contenedores. En contraste, en el modelo de trabajo anterior, se tenían problemas de compatibilidad entre ambientes de trabajo en todos los proyectos de desarrollo de software.

Esto permitió mantener el control del versionado del contenedor al momento de entregarlo a los diferentes equipos de trabajo, donde participan al menos 15 personas con equipos de cómputo mixtos, sin generar conflictos. Esta característica no está limitada a este número de participantes, ya que puede ser mayor.

5. CONCLUSIONES

Las bondades del despliegue ágil de esta tecnología han sido aplicadas de manera exitosa en otras aplicaciones de la Universidad, a cargo de la DGTIC. Si bien el uso de Docker y sus bondades de encapsulado para controlar diferentes ambientes de trabajo no es nuevo, ha aportado mejoras significativas, reduciendo los tiempos de despliegue y la posibilidad de incompatibilidades.

Respecto al uso de Dockers, se encontró que, si no se tienen conocimientos de sistemas operativos, configuración de la red y nociones básicas de la operación de *firewalls*, se enfrenta una curva de aprendizaje más alta.

4 Proyecto de mod_security: https://modsecurity.org/

https://doi.org/10.22201/dgtic.30618096e.2025.3.2.96 **Vol. 3, Núm. 2.** abril-junio 2025, págs. 53 - 53

Los Dockers son una herramienta útil para agilizar la labor en la administración de los ambientes de trabajo de desarrollo, pruebas, pre-producción y producción, que permite reducir tiempos en la implementación, optimizar recursos hardware al admitir la ejecución de múltiples instancias de una aplicación en diferentes contenedores y permitir que las aplicaciones puedan ejecutarse en diferentes sistemas operativos y entornos sin modificaciones.

De igual forma, proporcionan el aislamiento de recursos entre contenedores, lo que permite evitar que un contenedor afecte a otros en su operación al tener diferentes sistemas operativos, versiones de herramientas y librerías o configuraciones de seguridad.

Por último, los Dockers son una plataforma tecnológica que se utilizará más ampliamente a futuro, integrada con el uso de Kubernetes y de nuevas herramientas que permitan implementar aplicaciones que se centran en tareas de alto impacto, que implican la gestión de grandes cantidades de datos, que requieren un nivel de disponibilidad que permita a una aplicación seguir funcionando mientras una de sus partes se encuentra en actualización o reparación, o que requiera un esquema robusto de seguridad. De igual forma, se tendrá una mayor integración en nubes públicas o privadas o híbridas.

REFERENCIAS

Bruno, V., & Bruno, V. (2020, 30 marzo). Top 5 Alternativas a Docker. Infranetworking. https://blog.infranetworking.com/top-5-alternativas-a-docker/

Creación de imágenes a partir de un Dockerfile. (s. f.). Docker. https://iesgn.github.io/curso_docker_2021/sesion6/dockerfile.html.

Dock, M. (2025, 23 febrero). Pricing | Docker. Docker. https://www.docker.com/pricing/

Gallego, M., & Chico Guzmán, P. (2017). Seminarios OfiLibre Docker y Kubernetes. Universidad Rey Juan Carlos. https://tv.urjc.es/uploads/material/5fb81bcbd68b14ac6f8b4be2/Docker.pdf

MuyComputerPro. (2018, Julio 3). Estadísticas de uso de Docker año 2018. https://www.muycomputerpro.com/2018/07/03/adopcion-docker-estadisticas

Novoseltseva, E. (2023, 22 junio). Utilizar Docker: beneficios, estadísticas y historias de éxito | Apiumhub. Apiumhub. https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-utilizar-docker/

Ponsico Martín, P. (2017). Tecnología de Contenedores Docker [Tesis de grado de Ingeniería Telemática, Universitat Politècnica de Catalunya]. https://upcommons.upc.edu/bitstream/handle/2117/113040/Degree_thesis.pdf?sequence=1

Racero, A., & Som, A. (2022, mayo). Contenedores: Iniciación a Docker y casos de uso prácticos. Oficina de Software Libre, Universidad de Granada. https://osl.ugr.es/wp-content/uploads/2022/05/Contenedores-Iniciacio%CC%81n-a-Docker-y-casos-de-uso-pra%CC%81cticos-Mayo2022.pptx.pdf

Tirado, L. M. (2017). Implementación de Docker en la gestión del entorno de desarrollo [Tesina de Ingeniería, Universidad Politécnica de Sinaloa, Programa Académico de Ingeniería en Informática]. http://repositorio.upsin.edu.mx/formatos/TesinaLiliaMariaTirado3453.pdf