UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

IMPLEMENTACIÓN DE DOCKER: MÉTODOS Y HERRAMIENTAS PARA UN DESPLIEGUE EFICIENTE.

TRABAJO MONOGRÁFICO

PARA OBTENER EL GRADO DE

INGENIERO EN REDES

PRESENTA

Rene Molina Briceño

SUPERVISORES

Dr. Javier Vázquez Castillo

M.T.I. Vladimir Veniamin Cabañas Victoria

M.S.I. Laura Yésica Dávalos Castilla

M.S.I. Rubén enrique González Elixavide

Dr. Jaime Silverio Ortegón Aguilar

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

ÁREA DE TITULACIÓN

CHETUMAL QUINTANA ROO, MÉXICO, SEPTIEMBRE DE 2024

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

División de Ciencias, Ingeniería y Tecnología

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO

DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

TRABAJO MONOGRÁFICO TITULADO

"IMPLEMENTACIÓN DE DOCKER: MÉTODOS Y HERRAMIENTAS PARA UN DESPLIEGUE EFICIENTE"

ELABORADO POR
BR. RENE MOLINA BRICEÑO

BAJO SUPERVISIÓN DEL COMITÉ DEL PROGRAMA DE LICENCIATURA Y APROBADO COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

INGENIERO EN REDES

COMITÉ SUPERVISOR

SUPERVISOR _____
DR. JAVIER VAZQUEZ CASTILLO

SUPERVISOR _____
M.T.I. VLADIMIR VENTAMIN CABAÑAS VICTORIA

SUPERVISORA _____
M.S.I. LAURA YÉSICA DÁVALOS CASTILLA

SUPERVISOR SUPLENTE: _____
M.S.I. RUBÉN ENRIQUE GONZÁLEZ ELIXAVIDE

SUPERVISOR SUPLENTE: _____
DR. JAIME SILVERIO ORTEGÓN AGUILAR

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
ÁREA DE TITULACIÓN

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE QUINTANA ROO
DIVISIÓN DE CIENCIAS, INGENIERÍA Y TECNOLOGÍA

CHETUMAL QUINTANA ROO, MÉXICO, SEPTIEMBRE DE 2024

Resumen

En el primer capítulo veremos una introducción acerca de los grandes beneficios que conlleva la implementación de Docker en los sistemas operativos, así como las desventajas de éste mismo. De igual manera, tendremos como parte del primer capítulo los Objetivos Generales y los Objetivos Particulares.

En el segundo capítulo se describe en gran parte los antecedentes de Docker, conceptos tales como, ¿Qué son los contenedores?, ¿Qué es Docker?, un poco de historia sobre Docker, sus complementos como son el Dockerfile, Docker Compose, Docker Hub y tecnologías similares, tales como, Kubernetes, Máquinas Virtuales, entre otros.

En el tercer capítulo tendremos el desarrollo de la implementación de Docker Engine dentro de un Servidor Ubuntu 22.04 LTS (línea de comandos), en este capítulo contendrá los pasos que deben seguir para una instalación eficiente, las líneas de código estarán descritas para una mayor comprensión. De igual manera, tendremos comandos básicos de Docker los cuales van desde como iniciar Docker hasta como ver sus logs del sistema.

Por último, las conclusiones, tendremos un breve resumen a destacar lo más sobresaliente de los contenedores Docker, el como se gestionan los contenedores a gran escala y la principal idea de cómo surgió este trabajo monográfico.

Agradecimientos

A mis padres:

Agradezco a mis padres, Rene Molina y Ericka Briceño, por todo el apoyo incondicional que me han brindado a lo largo de mi vida. Gracias por sus sabios consejos, por enseñarme el valor del esfuerzo y la dedicación, y por estar siempre ahí para mí en cada paso de mi camino. Su amor y orientación han sido fundamentales para alcanzar mis objetivos y metas.

A mi pareja:

Agradezco a Dulce Núñez, mi compañera de vida, le agradezco por su constante aliento y por creer en mí incluso cuando yo dudaba. Tu confianza en mis capacidades y tu amor incondicional han sido fundamentales para alcanzar mis metas. Gracias por cada momento compartido y por ser mi apoyo inquebrantable.

A mis hermanas:

A Claudia y Odalys, mis hermanas, les agradezco profundamente por su presencia constante en este proceso tan difícil. Sus risas, consejos y amor han sido un refugio en los momentos de dificultad. No podría haberlo logrado sin ustedes a mi lado.

A mi familia:

Agradezco profundamente a mi familia completa, desde primos hasta abuelos. Aunque algunos ya no se encuentren con nosotros, les agradezco su enorme apoyo y experiencias brindadas.

A mis maestros:

A mis maestros, Javier Vázquez y Vladimir Cabañas les expreso mi más sincero agradecimiento por compartir su conocimiento y sabiduría conmigo. Gracias por todos sus apoyos y consejos a lo largo de mi etapa en la universidad.

A mí mismo:

Agradezco profundamente a mí mismo por haber logrado este objetivo tan querido y deseado. Agradezco mi capacidad para enfrentar los desafíos y seguir adelante día con día, siempre con la meta de ser mejor en cada escalón de mi vida. Me siento orgulloso de mi esfuerzo, dedicación y de no haber renunciado a mis sueños, incluso cuando las cosas se pusieron difíciles. Este logro es un testimonio de mi fuerza interior y mi compromiso con mi propio crecimiento y éxito.

A todos aquellos que de alguna manera contribuyeron a mi formación:

Finalmente, quiero agradecer a todos aquellos que de alguna manera han contribuido a mi formación y éxito. Sus palabras de aliento, sus críticas constructivas y su apoyo han sido invaluables para mi crecimiento personal y profesional.

Siempre parece imposible hasta que esté hecho.

Nelson Mandela.

Dedicatoria

A mis padres:

Con infinita gratitud y eterno reconocimiento por el apoyo incondicional que siempre me han brindado. Gracias a ustedes he logrado alcanzar este importante logro en mi carrera profesional. Su amor, sacrificio y constante aliento han sido mi fuerza y mi guía. Ustedes me enseñaron el valor del esfuerzo y la perseverancia, valores que me han acompañado en cada paso de este camino. Su confianza en mí ha sido la mejor de las herencias, y este logro es tanto suyo como mío.

Con todo mi amor y gratitud,

René Molina Briceño.

Índice

Tabla de Ilustraciones

CAPÍTULO 1

CONTENEDORES DOCKER Y SU BENEFICIO EN LA IMPLEMENTACIÓN DE SISTEMAS

CAPÍTULO 1
CONTENEDORES DOCKER Y SU BENEFICIO EN LA IMPLEMENTACIÓN DE SISTEMAS

1.1 Introducción

En la actualidad, los sistemas tradicionales como son servidores, sistemas operativos, aplicaciones, etc., se encuentran en continuo proceso de actualización hacia nuevas tecnologías, por lo que, para optimizar su desarrollo, desempeño y seguridad de los sistemas, se utilizan varias estrategias para el desarrollo e implementación, por lo que, para este contexto, se estaría analizando la adopción de una tecnología de contenedorización, específicamente con la tecnología de Docker.

Docker representa una innovadora propuesta que se centra en la creación de contenedores livianos y portátiles para aplicaciones de software. Estos contenedores poseen la capacidad de ejecutarse en cualquier máquina que cuente con Docker instalado, independientemente del sistema operativo subyacente, ya sea cualquier distribución de Linux, MacOS, Windows, Windows Server, u otros [1].

En comparación con las máquinas virtuales tradicionales, Docker ofrece ventajas sustanciales que impactan positivamente en la eficiencia y manejo de recursos del sistema en cual se encuentre. La ligereza de los contenedores, al compartir el mismo Kernel del sistema operativo subyacente, resulta en un uso más eficiente de recursos como el espacio en disco y la memoria RAM. Esta eficiencia no solo permite ejecutar más aplicaciones en una única máquina, sino que también conlleva a ahorros significativos en costos de hardware, al mismo tiempo que simplifica la administración de recursos [2].

La portabilidad de las aplicaciones es una característica destacada de Docker. Los contenedores encapsulan tanto las aplicaciones como sus dependencias, facilitando así la migración sin inconvenientes entre diversos entornos. Este aspecto

es crucial para adaptarse a las dinámicas demandas de un entorno de desarrollo, donde la movilidad de aplicaciones es esencial.

El aislamiento de alto nivel que proporciona Docker entre aplicaciones y sus dependencias se traduce en una ejecución independiente de cada contenedor. Esto asegura que un contenedor no interfiera con otros en la misma máquina o servidor, promoviendo un ambiente de ejecución estable y seguro [3].

Docker no solo ha simplificado el desarrollo y despliegue de aplicaciones, sino que ha agilizado el ciclo de desarrollo, al permitir a los desarrolladores trabajar en entornos locales idénticos a los de producción, se reduce drásticamente los problemas de compatibilidad, lo que lleva a una aceleración notable en el proceso de desarrollo.

Finalmente, la escalabilidad de las aplicaciones se ve facilitada con Docker, permitiendo la creación de múltiples instancias de un contenedor y distribuyendo eficientemente la carga de trabajo mediante herramientas de orquestación como Kubernetes o Docker Swarm. Esta capacidad de adaptación rápida es esencial en un entorno de desarrollo, donde las demandas pueden variar de manera significativa.

La desventaja que tendría Docker en algunos casos, la complejidad en la configuración inicial de contenedores, ya que puede ser más compleja que en máquinas virtuales o servidores tradicionales.

Otra desventaja es la seguridad, aunque Docker ofrece un buen nivel de aislamiento, los contenedores comparten el mismo Kernel, lo que podría plantear ciertos riesgos de seguridad en comparación con máquinas virtuales.

Docker está destinado principalmente a contenedores aislados con aplicaciones basadas en consola [4].

El presente trabajo presenta una guía de instalación de un sistema basado en Docker.

1.2 Objetivos Generales

- Documentar la guía de instalación de un sistema basado en contenedores con Docker.

1.3 Objetivos Particulares

1) Investigar el estado de arte de contenedores.
2) Documentar la implementación de contenedores mediante Docker en un sistema operativo basado en Linux.
3) Redacción del trabajo de monografía.
4) Presentación del trabajo ante el sínodo.

CAPÍTULO 2
LOS CONTENEDORES Y SUS ANTECEDENTES

CAPÍTULO 2
LOS CONTENEDORES Y SUS ANTECEDENTES

2.1 Contenedor

Los contenedores son unidades de software que encapsulan el código de una aplicación junto con todas sus bibliotecas y dependencias necesarias para su ejecución. Esto permite que el código se ejecute en cualquier entorno, ya sea en una computadora de escritorio, en una infraestructura de TI tradicional o en la nube.

Utilizan una forma de virtualización del sistema operativo, aprovechando características del Kernel del SO como los espacios de nombres y cgroups en Linux, y los silos y objetos de trabajo en Windows, para aislar procesos y controlar los recursos de CPU, memoria y disco que pueden utilizar.

A diferencia de las máquinas virtuales, los contenedores no necesitan incluir un sistema operativo completo en cada instancia, lo que los hace más pequeños, rápidos y portátiles. En su lugar, utilizan las características y recursos del sistema operativo anfitrión.

Aunque los contenedores tienen sus raíces en tecnologías anteriores como FreeBSD Jails y AIX Workload Partitions, la mayoría de los desarrolladores consideran que la era moderna de los contenedores comenzó en 2013 con la introducción de Docker [5].

2.2 Docker

Docker es una plataforma de software de código abierto que permite crear, probar, implementar y administrar aplicaciones rápidamente en contenedores de aplicaciones virtualizados en un sistema operativo común.

Docker empaqueta todo el código y las dependencias de una aplicación en un formato estándar que permite su rápida ejecución y fiabilidad en entornos informáticos (ver Figura 1).

FIGURA 1. LOGOTIPO DE DOCKER

Docker es conocido como un contenedor ejecutable, independiente y ligero, ya que integra todo lo necesario para ejecutar una aplicación, esto incluye sus bibliotecas, herramientas del sistema, código y tiempo de ejecución.

Los servicios de contenedores o *Containers as a Service (CaaS)* son servicios gestionados en la nube que administran el ciclo de vida de los contenedores. Los servicios de contenedores ayudan a orquestar (iniciar, detener, ampliar) el tiempo de ejecución de los contenedores. Con los servicios de contenedor, puede simplificar, automatizar y acelerar el desarrollo de sus aplicaciones y el ciclo de vida de su implementación.

Los motores de Docker funcionan como la aplicación del servidor del cliente que admite contenedores en varios servidores Windows y sistemas operativos Linux, como Oracle Linux, CentOS, Debian, Fedora, RHEL, SUSE y Ubuntu [1].

2.2.1 Historia De Docker

Docker fue concebido por Solomon Hykes como un proyecto interno en dotCloud, una empresa de plataforma como servicio (PaaS). Hykes contó con la colaboración inicial de otros ingenieros de dotCloud, como Andrea Luzzardi y Francois-Xavier Bourlet, así como con el aporte independiente de Jeff Lindsay. El desarrollo de Docker se basó en tecnologías anteriores de código abierto, incluyendo Cloudlets, una tecnología patentada de dotCloud.

El proyecto se liberó como código abierto en marzo de 2013. Con el lanzamiento de la versión 0.9 en marzo de 2014, Docker adoptó su propia biblioteca, libcontainer, en lugar de LXC como su entorno de ejecución por defecto. Esto marcó un hito importante en su evolución.

En 2015, Docker había ganado una considerable popularidad en GitHub, con más de 20,700 estrellas y casi 900 colaboradores. Para 2018, diversas organizaciones, incluyendo Red Hat, Microsoft, IBM y Google, se habían convertido en principales contribuyentes al proyecto, destacando el interés y apoyo de la industria en su desarrollo continuo [6].

2.2.2 Docker CE vs Docker EE

Docker CE (Community Edition) es una plataforma de contenedorización que es gratuita y de código abierto. Esta versión, que se lanzó inicialmente en 2013 bajo el nombre Docker, ha sido renombrada y se mantiene accesible sin costo. Docker CE puede ser utilizado en sistemas operativos como Windows 10 y Mac, así como en servicios en la nube como Azure y AWS, además de distribuciones de Linux como CentOS, Debian, Fedora y Ubuntu. La descarga de Docker CE está disponible a través de Docker Store.

En contraste, Docker EE (Enterprise Edition) es la versión premium de Docker CE. Docker EE es una plataforma de contenedores que ofrece una integración completa, certificación y soporte empresarial. Esta versión es compatible con Red Hat

Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), Oracle Linux, Ubuntu, Windows Server 2016, y también se puede ejecutar en Azure y AWS.

Docker CE vs EE: En primer lugar, es importante tener en cuenta que Docker CE no es una versión "diluida" de Docker EE. Tanto CE como EE tienen las mismas características y funciones principales.
Si bien ambas ediciones ofrecen las mismas funciones principales, Docker EE incluye funciones adicionales que pueden ayudar a las empresas a lanzar, administrar y proteger sus contenedores de manera más eficiente [7].

2.2.3 Costos de Docker

Si bien Docker CE es de código abierto, Docker EE tiene una versión comercial ofrecida por Docker.inc (Ver Figura 2.).

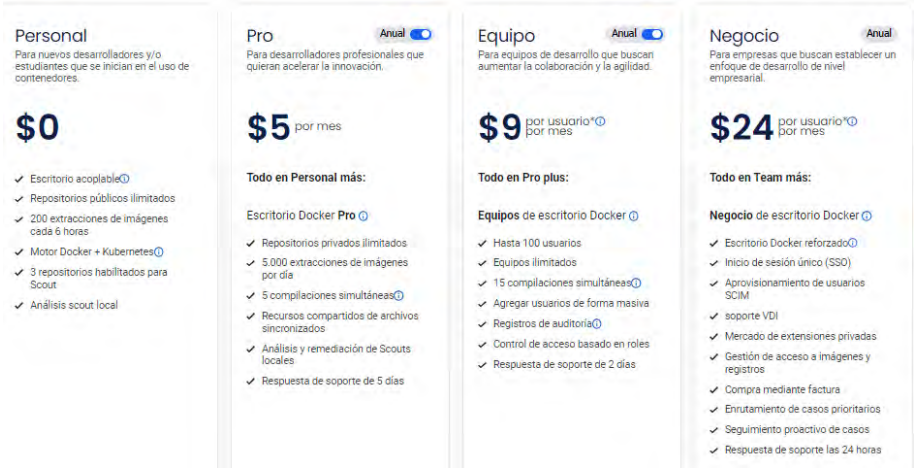


FIGURA 2 SUSCRIPCIONES OFRECIDAS EN DOCKER.INC

Docker Desktop es de uso gratuito como parte de la suscripción a Docker Personal para individuos, desarrolladores de código abierto no comerciales, estudiantes y educadores, y pequeñas empresas de menos de 250 empleados y menos de $10 millones en ingresos. El uso comercial de Docker Desktop en una empresa de más de 250 empleados o más de $10 millones en ingresos anuales requiere una suscripción paga (Pro, Team o Business) [8].

2.3 Complementos de Docker

2.3.1 Dockerfile

Un *Dockerfile* es un archivo de texto plano que contiene una serie de instrucciones necesarias para crear una imagen que, posteriormente, se convertirá en una sola aplicación utilizada para un determinado propósito. Docker puede crear imágenes automáticamente leyendo las instrucciones de un *Dockerfile* [9].

2.3.2 Docker Compose

Docker *Compose* es una herramienta para definir y ejecutar aplicaciones de múltiples contenedores.

Compose simplifica el control de toda su pila de aplicaciones, lo que facilita la administración de servicios, redes y volúmenes en un archivo de configuración YAML único y comprensible *(compose.yaml).* Luego, con un solo comando, crea e inicia todos los servicios desde su archivo de configuración.

Compose trabaja en todos los ambientes; producción, puesta en escena, desarrollo, pruebas y flujos de trabajo de CI. También tiene comandos para gestionar todo el ciclo de vida de su aplicación [10]:

- Iniciar, detener y reconstruir servicios
- Ver el estado de los servicios en ejecución
- Transmita la salida del registro de los servicios en ejecución
- Ejecutar un comando único en un servicio.

2.3.3 Docker Hub

Docker Hub es un registro de contenedores creado para que los desarrolladores y contribuyentes de código abierto encuentren, utilicen y compartan sus imágenes de contenedores. Con Hub, los desarrolladores pueden alojar repositorios públicos que se pueden usar de forma gratuita o repositorios privados para equipos y empresas.

El servicio promociona más de 100,000 aplicaciones disponibles públicamente, así como registros de contenedores públicos y privados [11].

2.4 Kubernetes

Kubernetes, también conocida como “k8s” o “kube” (Ver Figura 3), es una plataforma de orquestación de contenedores diseñada para programar y automatizar el despliegue, la gestión y el escalado de aplicaciones en contenedores [12].

FIGURA 3 LOGOTIPO DE KUBERNETES

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios.

Kubernetes ofrece un entorno de administración centrado en contenedores. Puede orquestar la infraestructura de cómputo, redes y almacenamiento para que las cargar de trabajo de los usuarios no tengan necesidad de hacerlo [13].

2.4.1 Historia de Kubernetes

Kubernetes fue desarrollado por primera vez por ingenieros de Google antes de ser de código abierto en 2014. Es descendiente de Borg, una plataforma de orquestación de contenedores utilizada internamente en Google. Kubernetes significa timonel o piloto en griego, de ahí el timó en el logotipo de Kubernetes [12].

2.5 Máquina Virtual

Una máquina virtual es un entorno informático que funciona como como un sistema aislado con sus propios recursos, este se crea a partir de un conjunto de recursos

de hardware, mediante sistemas denominado hipervisor, este aísla los recursos necesarios para la creación y gestión de las máquinas virtuales.

se denomina maquina host a la maquina física que ejecuta la máquina virtual, y maquina guests a la máquina virtual que se hospeda en la maquina host.

Las máquinas virtuales permiten la ejecución de varios sistemas operativos diferentes a la vez en una misma computadora [14].

2.6 Virtualización

La virtualización es una tecnología que permite crear versiones virtuales de recursos de hardware, sistemas operativos, dispositivos de almacenamiento y redes. En lugar de depender de hardware físico dedicado, la virtualización permite que múltiples sistemas y aplicaciones compartan un único recurso físico de manera eficiente [15].

2.7 Contenedor Docker Vs Máquina Virtual.

Los contenedores Docker y las máquinas virtuales son tecnologías que permiten la independencia de las aplicaciones respecto a la infraestructura de TI [16].

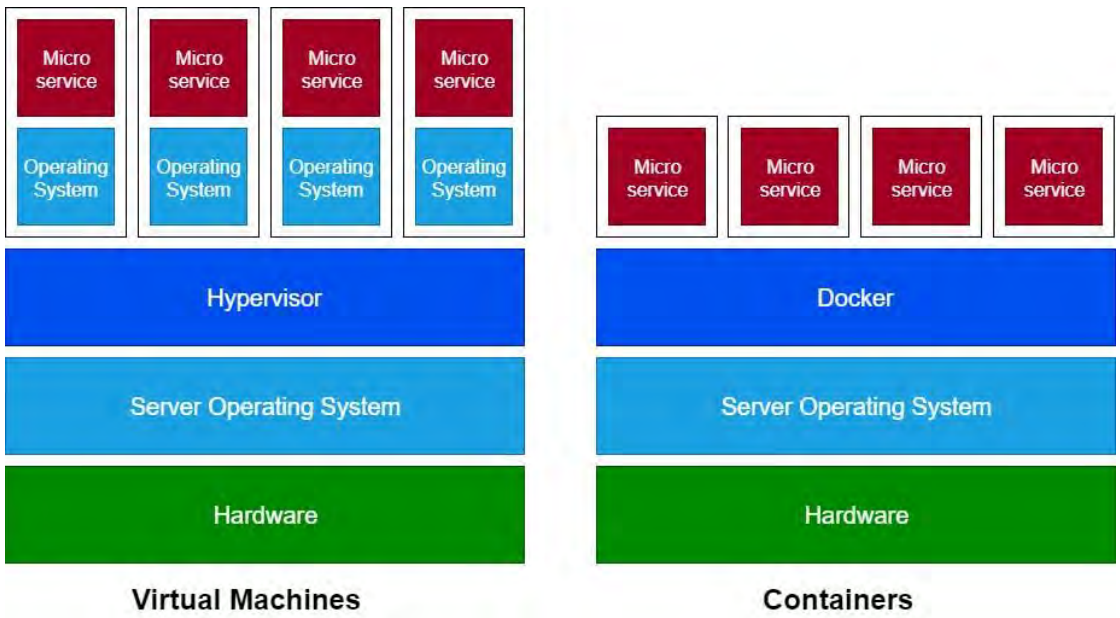


FIGURA 4 MÁQUINA VIRTUAL VS DOCKER

Características	Contenedores	Máquinas Virtuales
Definición	Paquetes de software que contienen el código de una aplicación, sus bibliotecas y dependencias.	Copias digitales de máquinas físicas con sistemas operativos propios.
Portabilidad	Permiten que el mismo código se ejecute en cualquier dispositivo.	Permiten usar eficazmente los recursos de hardware físico.
Tecnología Principal	Utilizan un motor de contenedores (como Docker) que actúa como intermediario entre los contenedores y el sistema operativo.	Utilizan hipervisores que coordinan el uso de recursos entre el sistema operativo invitado y el host.
Tamaño	Son ligeros, generalmente medidos en MB.	Son más grandes, generalmente varios GB.
Funcionamiento	Crean paquetes de software autosuficientes que funcionan de manera uniforme en cualquier máquina.	Implican la instalación de software de virtualización en una máquina física.
Configuración	Ofrecen definiciones estáticas de las configuraciones.	Ofrecen más control sobre el entorno de la aplicación, permitiendo la instalación manual de software y la restauración de estados.
Velocidad de Desarrollo	Se pueden modificar e iterar rápidamente.	Son más laboriosas y lentas de crear o regenerar.
Escalabilidad	Ocupan menos espacio y permiten un control detallado sobre la escalabilidad mediante microservicios.	Ocupan más espacio de almacenamiento y requieren más hardware, aunque migrar a la nube puede reducir costos.

CAPÍTULO 3
INSTALACIÓN DE DOCKER EN DISTRIBUCIÓN LINUX

CAPÍTULO 3

INSTALACIÓN DE DOCKER EN DISTRIBUCIÓN LINUX

3.1 Instalación de Docker Engine en Ubuntu Server 22.04 live-server-amd64

3.1.1 Instalar usando el repositorio APT.

Antes de la instalación de Docker Engine por primera vez en una nueva máquina host, se debe preparar la configurar del repositorio de Docker. Luego, se puede instalar y actualizar Docker desde el repositorio.

1. **Configuramos el repositorio apt de Docker.**
    1. *sudo apt-get update*

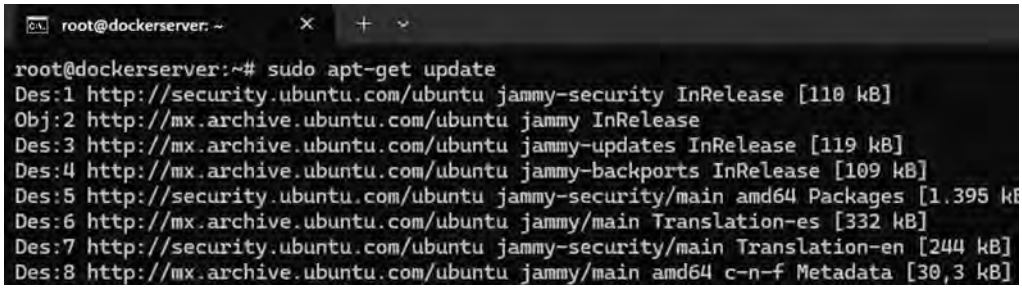


FIGURA 5 REPOSITORIO APT

- Este comando actualiza la lista de paquetes disponibles para instalar en tu sistema Ubuntu (Ver Figura 5).
- El comando **apt-get** es una herramienta de gestión de paquetes en sistemas basados en Debian, como Ubuntu.
- **sudo** se usa para ejecutar el comando con privilegios de superusuario, lo que permite realizar cambios en el sistema.

2. *sudo apt-get install ca-certificates curl*

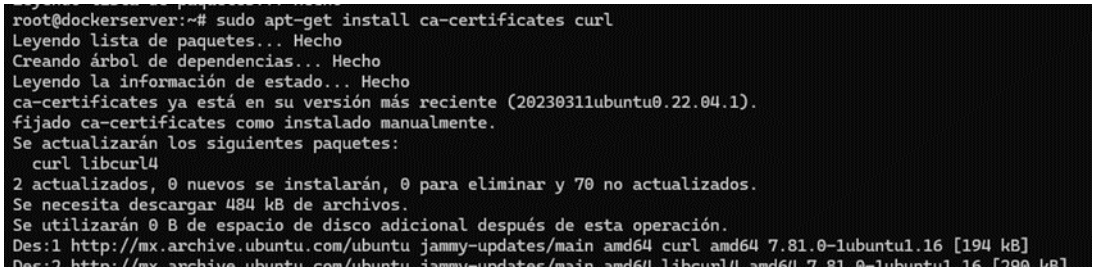


FIGURA 6 INSTALACIÓN DE CERTIFICADOS CA Y CURL

Esta línea instala dos paquetes en tu sistema:

- **ca-certificates:** Este paquete contiene certificados de autoridad de certificación (CA) comunes utilizados para autenticar la identidad de los sitios web seguros. Estos certificados son utilizados por aplicaciones como curl y apt para verificar la autenticidad de los servidores remotos.
- **curl:** Es una herramienta de línea de comandos para transferir datos desde o hacia un servidor. Se utiliza comúnmente para descargar archivos desde la web, y en este caso, será utilizado para descargar la clave de GPG (Gnu Privacy Guard) de Docker.

3. *sudo install -m 0755 -d /etc/apt/keyrings*

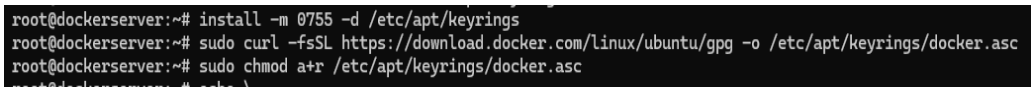


FIGURA 7 MÚLTIPLES COMANDOS

Este comando crea un directorio llamado */etc/apt/keyrings* en tu sistema (Ver Figura 7). Los archivos de claves de GPG (*Gnu Privacy Guard*) se almacenan comúnmente en este directorio en sistemas Debian/Ubuntu.

4. *sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc*

   Este comando descarga la clave de GPG de Docker desde el sitio web oficial de Docker *(https://download.docker.com/linux/ubuntu/gpg)* y la guarda en el archivo */etc/apt/keyrings/docker.asc.* (Ver Figura 7).

   Aquí está el desglose de los componentes:

   - **curl**: Es una herramienta para transferir datos desde o hacia un servidor mediante URL.
   - **-fsSL**: Son opciones utilizadas con curl:
     - -f: Hace que curl falle silenciosamente en caso de errores en la transferencia.
     - -s: Hace que curl funcione en modo silencioso, sin mostrar progreso ni mensajes de error.
     - -S: Muestra un mensaje de error si curl falla en la transferencia.
     - -L: Hace que curl siga redirecciones HTTP.
   - **https://download.docker.com/linux/ubuntu/gpg:** Es la URL de la clave de GPG de Docker.
   - **-o /etc/apt/keyrings/docker.asc:** Indica que la salida de curl (la clave de GPG descargada) debe guardarse en el archivo */etc/apt/keyrings/docker.asc.*

5. *sudo chmod a+r /etc/apt/keyrings/docker.asc*

   Este comando establece permisos de lectura (a+r) para el archivo */etc/apt/keyrings/docker.asc*, lo que permite que cualquier usuario pueda leer este archivo (Ver Figura 7).

   Esto es necesario para que apt pueda usar esta clave para verificar la autenticidad de los paquetes de Docker que descargues e instales en tu sistema.

2. **Agregue el repositorio a las fuentes de Apt:**

1. *echo *
   *"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu *
   *$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | *
   sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

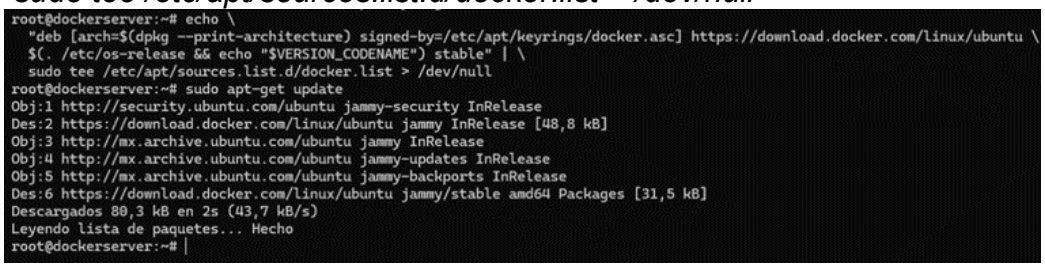


FIGURA 8 FUENTES DE APT

Generar una línea de texto para APT:

- *echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable"*
  - Crea una línea que define un repositorio de Docker para APT.
  - *$(dpkg --print-architecture):* obtiene la arquitectura del sistema (como amd64).
  - *$(. /etc/os-release && echo "$VERSION_CODENAME"): obtiene el nombre de la versión de Ubuntu.*

Agregar esta línea a un archivo de configuración de APT:

- | *sudo tee /etc/apt/sources.list.d/docker.list*
- Usa *tee* con sudo para escribir la línea generada en un nuevo archivo llamado *docker.listen* el directorio *`/etc/apt/s*

2. *Sudo apt-get update*
   Actualizamos los repositorios

3. Instale los paquetes de Docker.
    1. Para instalar la última versión, ejecute

    sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

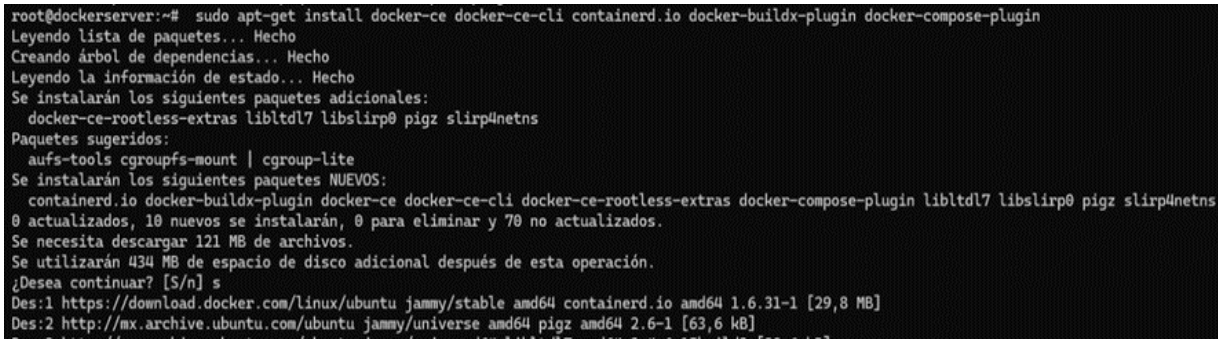


FIGURA 9 INSTALACIÓN DE VERSIÓN DE DOCKER

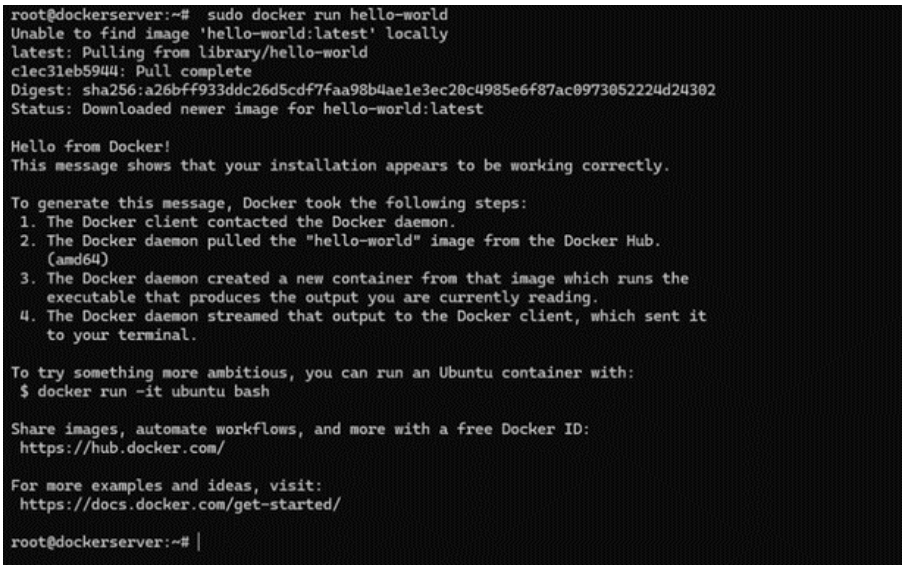
Este comando instalará varios componentes relacionados con Docker en el sistema Ubuntu. Aquí está el desglose de cada uno de los paquetes que estás instalando:

- **docker-ce:** Este es el paquete principal de Docker Community Edition (CE). Contiene el motor de Docker, que es el componente responsable de ejecutar y administrar contenedores Docker en tu sistema.
- **docker-ce-cli**: Este paquete proporciona la interfaz de línea de comandos (CLI) para interactuar con el motor de Docker. Te permite ejecutar comandos como docker run, docker build, docker pull, etc.
- **containerd.io:** Containerd es un contenedor de alto rendimiento y estándar de la industria que forma parte de la infraestructura de Docker. Es responsable de administrar los contenedores en ejecución en un sistema.
- **docker-buildx-plugin:** Este es un plugin para Docker que extiende las capacidades de construcción de imágenes.

Permite la construcción de imágenes de Docker para múltiples plataformas arquitectónicas de forma simultánea.

- **docker-compose-plugin:** Este plugin agrega compatibilidad con Docker Compose, una herramienta para definir y ejecutar aplicaciones Docker de múltiples contenedores. Permite definir la configuración de la aplicación en un archivo YAML y luego ejecutarla con un solo comando.

2. Verifique que la instalación de Docker Engine sea exitosa ejecutando la imagen *"hello-world"*.
    - Sudo docker run hello-world



Este comando descarga una imagen de prueba y la ejecuta en un contenedor. Cuando el contenedor se ejecuta, imprime un mensaje de confirmación y sale [17].

3.2 Comandos básicos de Docker en servidores Linux

En este apartado veremos los comandos básicos que pueden ser utilizados en Ubuntu Server o cualquier distribución de Linux. Estos comandos van desde como iniciar Docker hasta como ver los logs o registros de entrada y salida de un contenedor Docker.

1. **Comandos para Docker**
    1. Iniciar Docker:
        - *$ sudo systemctl start Docker*

```
root@dockerserver:~# sudo systemctl start docker
root@dockerserver:~#
```

FIGURA 10 INICIAR DOCKER

    2. Detener Docker
        - *$ sudo systemctl stop Docker*

```
root@dockerserver:~# sudo systemctl stop docker
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
root@dockerserver:~#
```

FIGURA 11 DETENER DOCKER

    3. Revisar la versión de Docker
        - *$ docker --version*

```
root@dockerserver:~# docker --version
Docker version 26.1.1, build 4cf5afa
root@dockerserver:~#
```

FIGURA 12 VERSIÓN DE DOCKER

2. **Imágenes Docker**
    1. Ver imágenes
        - *$ docker images*

```
root@dockerserver:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
httpd         2.4       67c2fc9e3d84   5 weeks ago     147MB
hello-world   latest    d2c94e258dcb   12 months ago   13.3kB
root@dockerserver:~#
```

FIGURA 13 VER IMÁGENES DOCKER

    2. Borrar imagen

- *$ docker rmi <image>*

```
root@dockerserver:~# docker rmi 67c2fc9e3d84
Untagged: httpd:2.4
Untagged: httpd@sha256:36c8c79f900108f0f09fd4148ad35ade57cba0dc19d13f3d15be24ce94e6a639
Deleted: sha256:67c2fc9e3d849b21a35b4c96f5ad8ec4dc9b73ca44c56537039e8c6f3054db0a
Deleted: sha256:f936a5242a64ac31b67a82b97893ed11b3714567e39362890bad496374029486
Deleted: sha256:87a86ebdb0e8643ca908f058814c42066199d4b348258a8018aad90c68803591
Deleted: sha256:8cbaa3d8954887877518ed80f6807ecb380f67b618820d91c627a15b90d60079
Deleted: sha256:8b2b346ba36bb622138e1fcc1a3e419df2c340f34ee47d1849ed0b5a49b9d8b5
root@dockerserver:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    1d668e06f1e5   10 days ago     188MB
hello-world   latest    d2c94e258dcb   12 months ago   13.3kB
root@dockerserver:~#
```

FIGURA 14 BORRAR IMAGEN DOCKER

Notas adicionales:

Si intentas eliminar una imagen que está siendo utilizada por un contenedor activo, Docker mostrará un mensaje de error. En ese caso, debes detener y eliminar todos los contenedores que están utilizando esa imagen antes de intentar eliminar la imagen.

3. Crear una imagen desde un contenedor
   - *$ docker commit -a “creador” –m “comentario” <contenedor> <imagen_a_crear>*
     docker commit -a "renemolina" -m "pruebas" addf88d6e7ea imagen-prueba

```
root@dockerserver:~# docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
nginx         latest    1d668e06f1e5   10 days ago     188MB
hello-world   latest    d2c94e258dcb   12 months ago   13.3kB
root@dockerserver:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
root@dockerserver:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
addf88d6e7ea   nginx     "/docker-entrypoint.…"   27 minutes ago   Up 23 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp
root@dockerserver:~# docker commit -a "renemolina" -m "pruebas" addf88d6e7ea imagen-prueba
sha256:b6a92aa3f22256890709b6397bf3d7a801a61539a0f25aec6e989303c03beeb5
root@dockerserver:~# docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
addf88d6e7ea   nginx     "/docker-entrypoint.…"   27 minutes ago   Up 23 minutes   0.0.0.0:8080->80/tcp, :::8080->80/tcp
root@dockerserver:~# docker images
REPOSITORY      TAG       IMAGE ID       CREATED              SIZE
imagen-prueba   latest    b6a92aa3f222   About a minute ago   188MB
nginx           latest    1d668e06f1e5   10 days ago          188MB
hello-world     latest    d2c94e258dcb   12 months ago        13.3kB
root@dockerserver:~#
```

FIGURA 15 CREAR UNA IMAGEN DESDE UN CONTENEDOR

4. Crear una imagen desde un Dockerfile
   - *$ docker build -t <nombre_imagen> -f <dockerfile> <path_de_destino_de_la_imagen>*

```
root@dockerserver:~# docker build -t imagen-apartir-dockerfile -f php-server/dockerfile .
[+] Building 0.9s (7/7) FINISHED
 => [internal] load build definition from dockerfile
 => => transferring dockerfile: 425B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [1/3] FROM docker.io/library/ubuntu:latest@sha256:3f85b7caad41a95462cf5b787d8a04604c8262cdcdf9a472b8c52ef83375fe15
 => => resolve docker.io/library/ubuntu:latest@sha256:3f85b7caad41a95462cf5b787d8a04604c8262cdcdf9a472b8c52ef83375fe15
 => [internal] load build context
 => => transferring context: 2B
```

FIGURA 16 CREAR UNA IMAGEN DESDE UN DOCKERFILE

3. Contenedores Docker

1. Ver contenedores inactivos
    - *$ docker ps –a*

```
root@dockerserver:~# docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED      STATUS                      P
ef0283254334   hello-world   "/hello"   6 days ago   Exited (0) 6 days ago
0c0a78e7e1b3   hello-world   "/hello"   6 days ago   Exited (0) 6 days ago
root@dockerserver:~#
```

FIGURA 17 VER CONTENEDORES INACTIVOS

2. Ver contenedores activos
    - *$ docker ps*

```
root@dockerserver:~# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@dockerserver:~#
```

FIGURA 18 VER CONTENEDORES ACTIVOS

3. Borrar contenedor:
    - *$ docker rm <contenedor id>*

```
root@dockerserver:~# docker rm ef0283254334
ef0283254334
root@dockerserver:~#
```

FIGURA 19 BORRAR CONTENEDOR

4. Detener contenedor:
    - *$ docker stop <contenedor id>*

```
root@dockerserver:~# docker stop 0c0a78e7e1b3
0c0a78e7e1b3
root@dockerserver:~#
```

FIGURA 20 DETENER CONTENEDOR

5. Arrancar contenedor detenido:
    - *$ docker start <contenedor id>*

```
root@dockerserver:~# docker start 0c0a78e7e1b3
0c0a78e7e1b3
root@dockerserver:~#
```

FIGURA 21 ARRANCAR CONTENEDOR DETENIDO

6. Iniciar contenedor:
    - *$ docker run --name <contenedor> -p <puerto_origen>:<puerto_contenedor> <imagen_que_queremos_utilizar>*

```
root@dockerserver:~# docker run --name nginx-container -p 8080:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
b0a0cf830b12: Already exists
4d84de5fb9b2: Pull complete
2818b7b6a9db: Pull complete
1e5314d67f16: Pull complete
8066e07ce4f2: Pull complete
05f7109fea9e: Pull complete
e58cbd904f7f: Pull complete
Digest: sha256:32e76d4f34f80e479964a0fbd4c5b4f6967b5322c8d004e9cf0cb81c93510766
```

FIGURA 22 INICIAR CONTENEDOR

7. Entrar en un contenedor iniciado:
    - *$ docker exec -it <contenedor> sh*

```
root@dockerserver:~# docker exec -it addf88d6e7ea sh
#
```

FIGURA 23 ENTRAR EN UN CONTENEDOR INICIADO

8. Copiar fichero externo del contenedor a dentro del contenedor:
    - *$ docker cp file <container-name>:/directorio/*

```
root@dockerserver:~# ls
php-server  snap
root@dockerserver:~# docker cp php-server/dockerfile addf88d6e7ea:/
Successfully copied 2.05kB to addf88d6e7ea:/
root@dockerserver:~#
```

FIGURA 24 COPIAR FICHERO EXTERNO DEL CONTENEDOR A DENTRO DEL CONTENEDOR

9. Mostrar el registro de logs del contenedor:
    - *$ docker logs <container>*

```
root@dockerserver:~# docker logs addf88d6e7ea
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/05/13 21:08:29 [notice] 1#1: using the "epoll" event method
2024/05/13 21:08:29 [notice] 1#1: nginx/1.25.5
2024/05/13 21:08:29 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/05/13 21:08:29 [notice] 1#1: OS: Linux 5.15.0-105-generic
2024/05/13 21:08:29 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/05/13 21:08:29 [notice] 1#1: start worker processes
2024/05/13 21:08:29 [notice] 1#1: start worker process 30
2024/05/13 21:08:29 [notice] 1#1: start worker process 31
2024/05/13 21:08:29 [notice] 1#1: start worker process 32
2024/05/13 21:08:29 [notice] 1#1: start worker process 33
2024/05/13 21:10:32 [notice] 1#1: signal 2 (SIGINT) received, exiting
```

FIGURA 25 MOSTRAR EL REGISTRO DE LOGS DEL CONTENEDOR

10. Mostrar estadísticas de recursos del contenedor:
    - *$ docker stats <container>*

```
root@dockerserver:~# docker stats addf88d6e7ea
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.02%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.02%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O         BLOCK I/O     PIDS
addf88d6e7ea   nginx-container   0.00%     4.395MiB / 3.82GiB    0.11%     1.16kB / 0B     0B / 8.19kB   5
```

FIGURA 26 MOSTRAR ESTADÍSTICAS DE RECURSOS DEL CONTENEDOR

[18]

Dockerfile

Un Dockerfile o imagen de Docker es un archivo de texto que contiene una serie de instrucciones para crear una imagen de Docker. Esta imagen es un archivo ejecutable e independiente que se utiliza para crear un contenedor. El Dockerfile no tiene una extensión específica y puede ser creado con cualquier editor de texto, dentro del archivo se encontrarán una serie de instrucciones para poder crear una imagen.

Se definirán los pasos a seguir para construir una imagen:

- **Selección de una imagen base:** Elegir una imagen base desde la cual se construirá la nueva imagen.
- **Copiar archivos y directorios:** Incluir archivos y directorios necesarios para la aplicación en la imagen.
- **Instalación de software:** Instalar el software necesario para la aplicación dentro de la imagen.
- **Configuración de variables de entorno:** Definir variables de entorno que serán usadas por la aplicación.
- **Especificación de comandos:** Determinar los comandos que deben ejecutarse cuando se inicie un contenedor basado en la imagen [19].

SIN MUCHA IMPORTANCIA		
INSTRUCCIÓN	EXPLICACIÓN	EJEMPLO
MAINTAINER	Se agregan los datos de quien ha creado el Dockerfile y la imagen	MAINTAINER Nombre <correo@ejemplo.com>
OBLIGATORIO		
FROM	Se especifica la imagen base que se utilizará para construir la nueva imagen. Esta instrucción es una de las más importantes en Dockerfile porque	FROM debian:latest FROM ubuntu:20.04

	establece el punto de partida para la construcción de tu imagen.	
DE EJECUCIÓN		
RUN	Ejecuta comandos en una nueva capa y crea un nuevo contenedor intermedio. Normalmente es usado para instalar nuevos paquetes.	RUN apt-get update && apt-get install -y paquete1 paquete2
CMD	El comando que se especifique a continuación se ejecutara cuando haga Docker run si no se especificar otro comando.	CMD apt-get update
ENTRYPOINT	Igual que CMD pero no es ignorado si indicamos otro comando.	ENTRYPOINT apt-get update
DATOS		
COPY	Copia archivos o directorios desde la máquina local (donde está instalado Docker) al contenedor Docker.	COPY texto.txt /directorio/ejemplo
ADD	Similar a COPY. Añade más funcionalidades. Permite 2 fuentes además de la posibilidad de que sean URLs. Se pueden usar escapes. si se usa con un archivo comprimido te lo descomprime.	

OTROS		
EXPOSE	Indica (no expone) el puerto del contenedor. Útil para que los puertos no se expongan aleatoriamente. Es complementario a Docker *run -p.* Si es -p (minúscula) tienes que indicar puerto entrada:puerto salida. Si es -P (mayúscula) el de salida es fijo y el de entrada aleatorio.	EXPOSE 8080
VOLUME	Indica el destino de un volumen dentro del contenedor. Puede ser uno o varios.	VOLUME /unadireccion
WORKDIR	Especifica el directorio desde el que se van a ejecutar las ordenes de: ▪ RUN ▪ CMD ▪ ENTRYPOINT ▪ COPY ▪ ADD Sustituye al comando cd. Y si no existe el directorio, lo crea.	WORKDIR /midirectorio
USER	Indica el nombre o UID que va a ejecutar las acciones del Dockerfile.	USER renemolina

CONCLUSIONES

CONCLUSIONES

Docker es una plataforma de software de código abierto que permite crear, probar, implementar y administrar aplicaciones rápidamente en contenedores de aplicaciones virtualizados en un sistema operativo común, Docker es conocido por ser un contenedor ejecutable, independiente y ligero, ya que este integra todo lo necesario para la ejecución de una aplicación, el empaquetamiento incluye sus bibliotecas, herramientas del sistema, código y tiempo de ejecución.

La arquitectura de Docker incluye el Kernel, el sistema operativo host, el Docker Engine y los contenedores de microservicios, en contraste con las máquinas virtuales que constan del Kernel, el sistema operativo host, el hipervisor, el sistema operativo invitado y sus servicios. La principal diferencia entre Docker y las máquinas virtuales es que Docker no utiliza un hipervisor, sino que aprovecha características del Kernel de Linux como los cgroups y namespaces para crear contenedores ligeros y aislados.

Una de las principales ventajas de Docker sobre las máquinas virtuales tradicionales es su ligereza. Los contenedores Docker son generalmente medidos en megabytes (MB), mientras que las máquinas virtuales se miden en gigabytes (GB), debido a que una máquina virtual requiere un sistema operativo completo para funcionar, mientras que Docker utiliza el sistema operativo subyacente de la máquina host. Esta característica también mejora la portabilidad, ya que los contenedores Docker pueden ejecutarse en cualquier máquina con Docker instalado, a diferencia de las máquinas virtuales que requieren trasladar un sistema operativo completo, lo cual puede ser tedioso debido al tamaño excesivo.

El funcionamiento de Docker es bastante sencillo, este crea paquetes de software autosuficientes que funcionan de manera uniforme en cualquier máquina. De igual forma la portabilidad que entre estas dos es abismal, Docker tiene una gran portabilidad ya que este permite ejecutarse en cualquier máquina que contenga Docker instalado, mientras que una máquina virtual es llevar un sistema operativo

completo lo que en ocasiones puede ser bastante tedioso la portabilidad debido al excesivo peso de los sistemas operativos.

La escalabilidad juega un rol importante a la hora de implementar nuevas tecnologías, Docker ocupa menos espacio que una maquina virtual y este permite un control detallado sobre la escalabilidad mediante microservicios.

Sin embargo, Docker tiene una desventaja comparada con su competencia, ya que comparte el mismo kernel del sistema operativo host, lo que implica un riesgo en caso de fallos. A pesar de esto, Docker es eficiente en términos de aislamiento, ya que los contenedores están completamente aislados. Los cgroups limitan y aíslan el uso de recursos (CPU, memoria, I/O, etc.) para un conjunto de procesos, asegurando que cada contenedor utilice solo los recursos asignados, evitando que un contenedor consuma todos los recursos del sistema y afecte a otros contenedores.

Para gestionar aplicaciones compuestas por múltiples contenedores, Docker Compose es una herramienta esencial. Docker Compose permite definir y ejecutar aplicaciones Docker multi-contenedor, facilitando la configuración y la orquestación de múltiples servicios a través de un archivo YAML. Esto simplifica enormemente el desarrollo, las pruebas y el despliegue de aplicaciones complejas, asegurando que todos los contenedores involucrados se inicien en el orden correcto y con las configuraciones adecuadas.

Además, para la orquestación de contenedores a gran escala, Kubernetes se ha convertido en la solución líder. Kubernetes automatiza la implementación, escalado y operación de aplicaciones en contenedores, proporcionando herramientas robustas para gestionar la infraestructura de contenedores en entornos de producción. Con Kubernetes, se pueden gestionar miles de contenedores en clústeres de máquinas, ofreciendo alta disponibilidad, escalabilidad automática y recuperación ante fallos.

La idea de implementar Docker en un sistema operativo surgió durante mis prácticas profesionales en el área de Innovación y Tecnología en la Universidad Autónoma del Estado de Quintana Roo. La motivación principal fue el uso excesivo de los recursos del servidor, lo que llevó a considerar la implementación de Docker en los servidores. A pesar de la difícil configuración de Docker, he optado por realizar esta monografía como una guía de instalación para Docker en distribuciones Linux, con el objetivo de facilitar su adopción y mejorar la eficiencia en el uso de los recursos del servidor.

Fuentes de información

Bibliografía

[1] J. Garzas, «javiergarzas.com,» 24 07 2015. [En línea]. Available: https://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html. [Último acceso: 16 11 2023].

[2] Avi, «GEEKFLARE,» 04 11 2023. [En línea]. Available: https://geekflare.com/es/docker-vs-virtual-machine/. [Último acceso: 16 11 2023].

[3] «Dimensiona,» 29 09 2023. [En línea]. Available: https://www.dimensiona.com/es/que-es-docker-y-cuales-son-sus-ventajas/#:~:text=Docker%20utiliza%20menos%20recursos%20en,en%20disco%20y%20memoria%20RAM.. [Último acceso: 16 11 2023].

[4] O. Romanyuk, «freecodecamp,» 30 01 2023. [En línea]. Available: https://www.freecodecamp.org/espanol/news/7-casos-en-los-que-no-deberias-usar-docker/. [Último acceso: 16 11 2023].

[5] «¿Qué son los contenedores?,» [En línea]. Available: https://www.ibm.com/mx-es/topics/containers. [Último acceso: 27 Mayo 2024].

[6] «wikipedia,» 27 Febrero 2024. [En línea]. Available: https://es.wikipedia.org/w/index.php?title=Docker_(software)&oldid=158117738#Historia.

[7] C. Weerasinghe, «stack overflow,» 2024 Mayo 2024. [En línea]. Available: https://stackoverflow.com/questions/45018786/what-is-the-exact-difference-between-docker-ee-enterprise-edition-docker-ce#:~:text=Docker%20CE%20(Community%20Edition)%20is,EE%20for%20versions%20%3C%3D%201.13..

[8] docker.inc, «docker,» 27 Mayo 2024. [En línea]. Available: https://www.docker.com/pricing/.

[9] X. Rodríguez, «OpenWebinars,» 24 Julio 2019. [En línea]. Available: https://openwebinars.net/blog/que-es-dockerfile/.

[10] «docker.docs,» 28 Febrero 2024. [En línea]. Available: https://docs.docker.com/compose/.

[11] «Docker Hub,» 28 Febrero 2024. [En línea]. Available: https://www.docker.com/products/docker-hub/.

[12] «Kubernetes,» 05 Marzo 2024. [En línea]. Available: https://www.ibm.com/mx-es/topics/kubernetes.

[13] «¿qué es kubernetes?,» 17 Julio 2022. [En línea]. Available: https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/.

[14] «Las Maquinas Virtuales,» 31 Enero 2023. [En línea]. Available: https://www.redhat.com/es/topics/virtualization/what-is-a-virtual-machine.

[15] «¿Qué es la virtualización?,» 27 Mayo 2024. [En línea]. Available: https://aws.amazon.com/es/what-is/virtualization/.

[16] «¿Cuál es la diferencia entre los contenedores y las máquinas virtuales?,» 27 Mayo 2024. [En línea]. Available: https://aws.amazon.com/es/compare/the-difference-between-containers-and-virtual-machines/.

[17] «Install Docker Engine on Ubuntu,» 06 Mayo 2024. [En línea]. Available: https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository.

[18] R. A. Aviles, «onesait platform,» 08 Julio 2022. [En línea]. Available: https://blog.onesaitplatform.com/2022/07/08/guia-comandos-para-docker/. [Último acceso: 29 Mayo 2024].

[19] RedxLus, «Crear Dockerfile,» 09 Julio 2019. [En línea]. Available: https://luisiblogdeinformatica.com/crear-dockerfile/. [Último acceso: 29 Mayo 2024].