Evaluación del Uso de Docker en el Despliegue de Aplicaciones: Un Enfoque Práctico con Monitoreo de Recursos

Evaluation of Docker Usage in Application Deployment: A Practical Approach with Resource Monitoring

Brayan Ricardo Chan Tec[1]

[1]*Instituto Tecnológico de Conkal, Yucatán, México*

chantecbrayanricardo@gmail.com

Resumen

Docker se ha consolidado como una tecnología clave en el desarrollo y despliegue de aplicaciones modernas, debido a su enfoque basado en contenedores que permite encapsular aplicaciones junto con sus dependencias, garantizando portabilidad y consistencia entre entornos.

El presente artículo tiene como objetivo evaluar el uso de Docker mediante una implementación práctica, incorporando herramientas de monitoreo como Prometheus y Grafana para analizar el consumo de recursos del sistema.

La metodología empleada combina revisión bibliográfica y experimentación en un entorno Linux, donde se desplegaron contenedores y se monitorearon métricas como CPU, memoria RAM y almacenamiento en tiempo real.

Los resultados evidencian que Docker reduce significativamente los tiempos de despliegue, optimiza el uso de recursos y mejora la observabilidad del sistema. Se concluye que la integración de herramientas de monitoreo fortalece la gestión y control de aplicaciones en entornos modernos.

Palabras clave: Docker, contenedores, DevOps, monitoreo, Prometheus, Grafana.

Abstract

Docker has become a key technology in modern application development and deployment due to its container-based approach, which allows applications and their dependencies to be encapsulated, ensuring portability and consistency across environments.

This article aims to evaluate Docker through a practical implementation, incorporating monitoring tools such as Prometheus and Grafana to analyze system resource usage. The methodology combines literature review and experimentation in a Linux environment, where containers were deployed and metrics such as CPU, RAM, and storage were monitored in real time.

The results show that Docker significantly reduces deployment time, optimizes resource usage, and improves system observability. It is concluded that integrating monitoring tools enhances application management and control in modern environments.

Keywords: Docker, containers, monitoring, DevOps, Prometheus, Grafana.

I. INTRODUCCIÓN

El desarrollo de software moderno requiere soluciones eficientes, escalables y portables. En este contexto, Docker ha emergido como una herramienta fundamental basada en la contenedorización, permitiendo ejecutar aplicaciones en entornos aislados y consistentes.

A diferencia de las máquinas virtuales, los contenedores comparten el núcleo del sistema operativo, lo que reduce el consumo de recursos y mejora el rendimiento. Esto ha impulsado su adopción en prácticas DevOps, especialmente en integración y despliegue continuo (CI/CD).

Además, el monitoreo del rendimiento se ha convertido en un elemento esencial en la gestión de aplicaciones modernas. Herramientas como Prometheus y Grafana permiten recolectar y visualizar métricas en tiempo real, facilitando la toma de decisiones.

El objetivo de este trabajo es evaluar el uso de Docker mediante una implementación práctica, integrando herramientas de monitoreo para analizar el comportamiento del sistema.

II. METODOLOGÍA

La investigación se desarrolló mediante un enfoque descriptivo-experimental. En una primera etapa, se realizó

una revisión bibliográfica sobre contenedorización y tecnologías DevOps.
Posteriormente, se implementó un entorno de prueba en sistema operativo Linux, donde se instaló Docker y se desplegaron contenedores utilizando imágenes oficiales.
Se implementó una aplicación web mediante Nginx y se utilizó Docker Compose para la gestión de servicios.
Adicionalmente, se integraron herramientas de monitoreo como Prometheus y Grafana. Prometheus fue utilizado para la recolección de métricas, mientras que Grafana permitió la visualización mediante paneles interactivos.
Durante la experimentación, se evaluaron variables como tiempo de despliegue, uso de CPU, memoria RAM y almacenamiento.

III. IMPLEMENTACIÓN PRÁTICA

Para validar el funcionamiento de Docker, se desplegó un servidor web utilizando la imagen oficial de Nginx:

```
docker run -d -p 80:80 nginx
```

Asimismo, se utilizó Docker Compose para definir servicios de forma estructurada:

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "80:80"
```

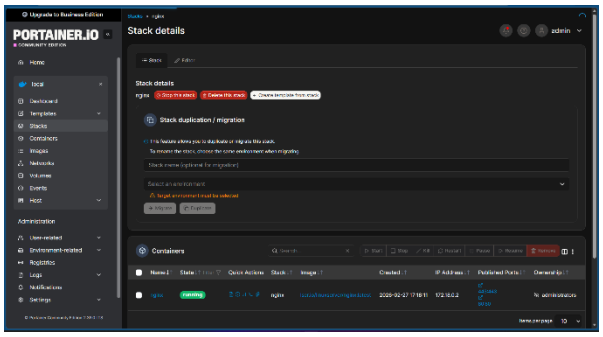

Figura 1. Creación de contenedor Nginx mediante Docker.

El uso de contenedores permitió aislar los servicios sin interferir con el sistema anfitrión, garantizando portabilidad y facilidad de despliegue. Además, Docker Compose facilitó la gestión de múltiples servicios mediante un archivo de configuración centralizado.

Adicionalmente, se configuró un sistema de monitoreo utilizando Prometheus y Grafana:

```
global:
  scrape_interval: 120s
  evaluation_interval: 120s

  external_labels:
    monitor: 'my-project'

scrape_configs:
  - job_name: 'prometheus'
    scrape_interval: 120s

    static_configs:
      - targets:
          - 'localhost:9090'
          - 'cadvisor:8080'
          - 'node-exporter:9100'
          - 'nginx-exporter:9113'
```

Con el comando Docker ps, podemos identificar si esta corriendo los servicios de Grafana y Prometheus.
Estas herramientas permitieron recolectar y visualizar métricas del sistema en tiempo real, proporcionando información clave sobre el comportamiento de los contenedores.

III. RESULTADOS

Mediante el uso de Grafana y Prometheus, se visualizaron métricas del sistema en tiempo real, incluyendo el consumo de CPU, memoria RAM y almacenamiento. Los paneles generados permitieron analizar el comportamiento de los contenedores durante su ejecución.

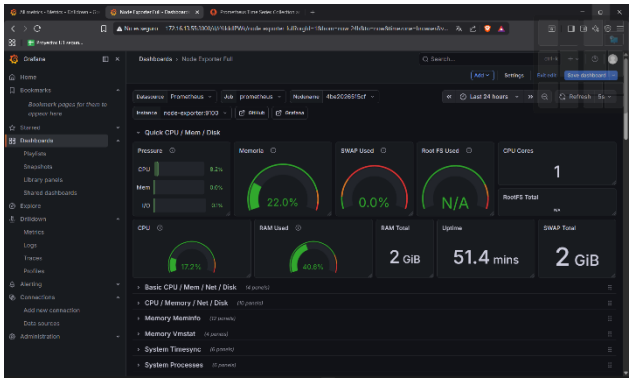

Figura 2. Dashboard de Grafana mostrando el uso de recursos en tiempo real.

Se observó que el uso de recursos se mantuvo estable, con variaciones controladas en función de la carga del sistema. Esto confirma la eficiencia de los contenedores en comparación con entornos tradicionales.
Los dashboards de Grafana facilitaron la identificación de patrones de uso, permitiendo validar el rendimiento del sistema y detectar posibles cuellos de botella.
Asimismo, el monitoreo continuo permite identificar comportamientos anómalos en el consumo de recursos, los

cuales podrían estar asociados a fallos del sistema, sobrecarga de procesos o posibles incidentes de seguridad, como uso no autorizado de la infraestructura. En entornos empresariales, este tipo de análisis resulta fundamental para prevenir riesgos y garantizar la estabilidad del sistema**.**

IV. DISCUSIÓN

Los resultados obtenidos muestran que el uso de Docker mejora el despliegue de aplicaciones mediante contenedores, ya que estos comparten el sistema operativo y consumen menos recursos que las máquinas virtuales. Esto hace que los despliegues sean más rápidos y eficientes, especialmente en entornos donde el tiempo es importante.

Por otra parte, Docker Compose facilita la gestión de múltiples servicios al permitir definir configuraciones de forma ordenada y reproducible. Esto ayuda a reducir errores y mantiene la consistencia entre los entornos de desarrollo y producción.

En cuanto al monitoreo, la integración de Prometheus y Grafana permitió visualizar métricas del sistema en tiempo real. Esto ayudó a analizar el comportamiento de los contenedores y detectar posibles problemas de rendimiento o consumo de recursos.

Las métricas de CPU, memoria y almacenamiento mostraron un comportamiento estable, con variaciones normales según la carga del sistema. El monitoreo continuo también permite identificar comportamientos anómalos que podrían indicar fallos o mal uso de recursos.

Sin embargo, el uso de contenedores también presenta retos, sobre todo en seguridad y en la administración de varios servicios al mismo tiempo. En este sentido, herramientas como Kubernetes pueden complementar la solución al mejorar la orquestación y escalabilidad.

V. CONCLUSIONES

Se concluye que Docker mejora la eficiencia en el despliegue de aplicaciones al permitir la ejecución en contenedores ligeros, reduciendo el consumo de recursos y los tiempos de implementación frente a métodos tradicionales.

La integración de Prometheus y Grafana facilita el monitoreo en tiempo real del sistema, permitiendo analizar el uso de CPU, memoria y almacenamiento, lo que contribuye a una mejor detección del comportamiento del entorno.

En conjunto, Docker se consolida como una solución efectiva para entornos modernos de desarrollo y despliegue, especialmente cuando se complementa con herramientas de monitoreo para mejorar el control del sistema.

REFERENCIAS

Merkel, D. (2014). Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal, 2014(239), 2–7.

Pahl, C. (2015). Containerization and the PaaS cloud. IEEE Cloud Computing, 2(3), 24–31.

Zhang, Q., Chen, M., & Li, L. (2021). Container technologies in cloud computing: A survey. Journal of Cloud Computing, 10(1), 1–17.

Turnbull, J. (2019). The Docker Book: Containerization is the new virtualization.

Docker Inc. (2024). Descripción general de Docker. https://www.docker.com/

Boettiger, C. (2015). Introducción a Docker para la investigación reproducible. ACM SIGOPS Operating Systems Review, 49(1), 71–79.

Burns, B., Beda, J., & Hightower, K. (2022). Kubernetes: En producción (3.ª ed.). O'Reilly Media.

Wiggins, A. (2017). Aplicación de doce factores (The Twelve-Factor App). https://12factor.net/

Morris, K. (2020). Infraestructura como código: gestión de servidores en la nube. O'Reilly Media.

Brazil, B. (2018). Prometheus: En funcionamiento (Up & Running). O'Reilly Media.

Grafana Labs. (2024). Documentación de Grafana. https://grafana.com/docs/

Sarkar, A., & Kumar, R. (2020). DevOps y orquestación de contenedores: una revisión sistemática. Revista Internacional de Computación y Servicios en la Nube, 9(2), 45–58.