Knowledge Engineering and Data Science (KEDS)
Vol 7, No 1, April 2024, pp. 71–85

pISSN 2597-4602
eISSN 2597-4637

Docker Optimization of an Automotive Sector Virtual Server Infrastructure

Leonel Hernandez [a,*], Carlos Eduardo Uc Rios [b,2]

[a]*Institución Universitaria de Barranquilla IUB*
Carrera 45 # 48 – 31, Barranquilla, Colombia
[b]*Universidad Internacional Iberoamericana UNINI*
Calle 15 num. 36, entre 10 y 12, Imi, III, 24560, Campeche, Mexico
[1]*lhernandezc@unibarranquilla.edu.co*;* [2] *carlos.uc@unini.edu.mx*
**corresponding author*

ARTICLE INFO

Article history:
Received 20 June 2024
Revised 08 July 2024
Accepted 20 July 2024
Published online 26 August 2024

Keywords:
IT Infrastructure Optimization
Docker
Virtual Servers
Efficient IT Management

ABSTRACT

Server virtualization is a powerful strategy for optimizing network infrastructure. It allows multiple virtual servers to run on a single physical server, maximizing resource utilization and improving efficiency. Deploying server virtualization using Docker technology offers a lightweight and flexible approach to optimizing network infrastructure. Docker contains package applications and their dependencies, enabling consistent and efficient deployment across various environments. Specifically, optimizing virtual server infrastructure using Docker Technology in the automotive sector focuses on improving the efficiency and management of the company's virtual server resources. By implementing Docker technology, a container platform that allows the packaging and running of applications in a lightweight and secure manner, the project aims to reduce operational costs and increase the agility and scalability of IT services. Adopting Docker will facilitate the rapid deployment of applications, ensuring a consistent and isolated execution environment for each one. This will allow the company to manage its workloads more efficiently and respond quickly to market needs, reassuring the audience about the potential improvements in their work processes. The study is developed under the top-down methodology guidelines for the design of telematics systems. It also includes a detailed analysis of the current server performance, a proposal for restructuring the existing infrastructure, and a plan to implement DevOps practices to optimize development and operational processes. With these changes, a significant improvement in system availability and performance is expected, thus contributing to the company's growth and technological innovation. The benefits of Docker implementation are numerous, including lightweight (containers share the host OS kernel, reducing overhead), portability (consistent environment across development, testing, and production), scalability (effortlessly scale containers horizontally), isolation (each container runs in its isolated environment), and efficiency (optimal resource utilization compared to traditional VMs). These benefits promise a brighter future for the company's IT infrastructure.

I. Introduction

In an information-driven world, a company's technological infrastructure is the core of its daily operations and a fundamental pillar for its growth and adaptation to a constantly evolving business environment. The Company, aware of this reality, is embarking on a strategic initiative to modernize its virtual server infrastructure by implementing Docker technology, a leading solution in virtualization at the operating system level. The company's current infrastructure is heavily reliant on VMware virtualization technology. Despite VMware's provision of a robust server virtualization solution, the scalability, operating costs, and overall business agility have been impacted by several challenges that have arisen over time. These obstacles are summarized below: complexities that impede business agility, increasing operating costs, and scalability limitations. The company acknowledges the necessity of transitioning to a more cost-effective, scalable, and adaptable solution in light of these obstacles. Docker-based server virtualization provides a promising alternative by enabling the operation of lightweight containers in various environments with minimal overhead. This transition aims to address the limitations of the current VMware infrastructure by offering improved

https://doi.org/10.17977/um018v7i12024p71-85

scalability, reduced operating costs, and enhanced business agility through streamlined management and deployment processes.

This project arises in response to the company's current VMWare infrastructure's operational and technical challenges. These include scalability limitations, increasing operating costs, and complexity that tests business agility. Optimizing virtual servers using Docker promises a revolution in application management and how the enterprise approaches operational efficiency and technological innovation.

The emergence and popularization of containerization, spearheaded by Docker, has instigated a revolution in the management and deployment of applications and services. While traditional virtual machines operate under the need for a hypervisor and complete operating systems for each instance, Docker is based on containers that, although operating in sandboxes, share the same operating system. This feature markedly diminishes overhead costs, thereby facilitating the operation of more applications on the same physical infrastructure. Moreover, Docker is distinguished by its portability, which enables the transfer and adaptation of containers across diverse environments.

The challenges identified with the current VMware infrastructure can be directly addressed by transitioning to Docker technology, which offers several advantages. Traditional virtual machines are substantially heavier than Docker containers because they share the host OS kernel. This efficacy enables the execution of a more significant number of containers on a single host than VMs, thereby enhancing scalability and density without necessitating additional hardware. In addition, containers can be rapidly replicated, destroyed, and created, enabling applications' rapid scalability in accordance with demand. This agility allows the organization to adapt to evolving business requirements promptly. Additionally, Docker supports orchestration tools such as Kubernetes and Docker Swarm, which can efficiently manage container clusters. These tools automate deployment, scaling, and management to facilitate the administration of large-scale environments.

Docker is open-source, which eliminates the licensing costs associated with VMware. Although orchestration tools such as Kubernetes may incur associated costs (if managed services are employed), they are generally less expensive than traditional VM licensing fees. As a result of the lightweight nature of containers, the number of physical servers required to support the same number of applications is reduced, resulting in reduced hardware procurement and maintenance costs. An additional suggestion is that reducing the number of servers required and the more efficient utilization of existing resources reduces energy consumption and cooling requirements, resulting in reduced operational costs.

Docker Compose and its standardized container format simplify the deployment process for multi-container applications. This standardization simplifies the process of configuring and administering applications. Containers ensure consistent behavior across development, testing, and production environments by encapsulating all dependencies and configurations. This consistency mitigates the complications resulting from discrepancies in the surrounding environment. Docker is particularly well-suited for microservices, as it enables the deconstruction of applications into more minor, more autonomous services. This architectural approach facilitates more expedient updates and deployments by enhancing modularity and scalability. The portability of microservices is a crucial requirement. Docker is an effective solution for this, as it allows the decomposition of applications into discrete, independent services. This architectural approach facilitates rapid updates and deployments by enhancing modularity and scalability.

Docker's networking capabilities, including overlay networks and service discovery, facilitate the configuration of networks and the implementation of security measures. This simplicity serves to mitigate the risk of misconfiguration and to simplify management. Containers provide process isolation and security features, including user namespaces and second profiles. The segregation of applications and reduction of potential vulnerabilities result in enhanced security. Moreover, Docker is designed to integrate seamlessly with continuous integration and continuous deployment (CI/CD) pipelines, automating the build, test, and deployment processes. Such automation enhances reliability and facilitates the expeditious completion of delivery cycles.

In light of the considerations above, it would be prudent for the company to investigate the potential benefits of adopting Docker to optimize its virtual server infrastructure. This paper examines this issue, identifying the company's current challenges and the potential benefits that could be gained

through implementing Docker. The company can enhance its efficiency, reduce costs, and improve business agility by optimizing its virtual server infrastructure with Docker.

II. Methods

This research is of paramount importance for the optimization of the company's virtual server infrastructure through the use of Docker technology. As indicated by reference [1], using Docker to optimize virtual server infrastructure offers several advantages. The following benefits include enhanced scalability, simplified application deployment and management, and improved resource utilization. Furthermore, Docker facilitates the portability of applications, allowing for expedient transfer to disparate environments without the hindrance of compatibility concerns.

Docker has transformed the software development, shipping, and deployment process by enhancing the efficacy of development pipelines and ensuring consistency across multiple environments [2]. The optimization of Docker environments is contingent upon many factors, including the reduction of image size, the optimization of resource utilization, the implementation of container orchestration, and the optimization of network performance.

In order to ensure optimal performance and cost-effectiveness, it is essential to guarantee efficient resource utilization in Dockerized environments. Various strategies have been explored to improve the management of CPU, memory, and storage resources within Docker containers. Furthermore, Docker containers facilitate performance isolation and recommend resource limits to prevent contention. In situations of high load, the efficacy of containers can be significantly improved by adjusting the CPU and memory limits. In storage optimization, the advantages of overlay file systems and the recommendation of custom storage drivers to improve I/O performance are exciting [3]. It is paramount to reduce the size of Docker images to facilitate more rapid deployment and reduce storage expenses.

Various methodologies have been proposed to reduce the size of the image. The multistage builds technique significantly reduces the final image size [4], which separates build and runtime dependencies. This method exploits the capability of the Dockerfile to delineate multiple stages and retain solely the most essential components in the concluding stage. Reducing the number of layers can result in more efficient and streamlined images. This involves the incorporation of commands within the Dockerfile to reduce the number of layers generated during the build process [5].

Network performance is critical in containerized applications in microservices architectures, where inter-service communication is frequent [6]. Prior research has addressed Docker networking models, including overlay, bridge, and host networks. The selected networking model has been found to influence throughput and latency significantly. Despite their utility for distributed applications, overlay networks may induce additional latency compared to bridge networks. The integration of SDN: The integration of software-defined networking (SDN) with Docker provides the capability to configure the network dynamically. Integrating SDN with Docker facilitates more efficient traffic management and enhanced network performance in containerized environments [7].

In the context of managing containerized applications at scale, orchestration technologies such as Docker Swarm and Kubernetes are of paramount importance. The optimization of orchestration entails the enhancement of the efficacy of resource allocation, scaling, and scheduling.

Kubernetes Scheduling: The Docker Engine represents the fundamental component that enables containerization and serves as the focal point of Docker's architectural framework. The Docker Engine comprises the Docker Daemon (`dockerd`), a background service that oversees volumes, networks, images, and containers. The Docker Command-Line Interface (CLI), `docker`, provides a means of interacting with the Docker Daemon and managing containers and images. This encompasses the creation, execution, and management of many tasks.

Containers, images, Dockerfiles, volumes, and networks represent some of the most critical components of the Docker ecosystem. Containers are portable, lightweight environments that facilitate consistency across diverse operational contexts by enabling the execution of applications and their associated dependencies. Docker images, which may be considered immutable templates for creating containers, are generated from Dockerfiles that specify the desired environment and dependencies for the application. Docker's networking capabilities facilitate communication between containers and

external entities, while volumes provide persistent storage for data that must be maintained throughout the lifecycle of a container.

Additionally, Docker incorporates a registry system for storing and distributing images. Docker Hub serves as the default public registry, offering users access to a comprehensive repository of images. Private registries allow organizations to administer and store their images securely. Due to the combination of its various elements, Docker is a powerful tool for the consistent, scalable, and efficient development, deployment, and management of applications [8]. In order to make effective scheduling decisions, these algorithms consider the availability of node resources and the requirements of pods. The Kubernetes Horizontal Pod Autoscaler (HPA) is a subject of ongoing investigation, with researchers exploring strategies for dynamically scaling applications by CPU and memory usage [9]. The team proposes that HPA be enhanced by incorporating bespoke metrics and predictive scaling algorithms to oversee workload fluctuations efficiently.

In the context of Docker environments, where security is paramount, concerns about container isolation and image vulnerability represent significant challenges. The process of scanning images for vulnerabilities is known as image vulnerability scanning. A study by [10] underscores the importance of scrutinizing Docker images for potential vulnerabilities. It is recommended that automated tools be employed that are integrated with CI/CD protocols to identify and mitigate security risks at the earliest possible stage of development.

Techniques for Isolation: The discussion will encompass techniques for enhancing container isolation, including using user namespaces and second profiles [11]. These methods provide additional layers of security by restricting containers' capabilities and reducing the attack surface.

As the article by [12] points out, collaborative projects between the company and the educational sector can also be a breeding ground for new ideas and innovations. These projects benefit both parties by developing new products, services, and processes. They can give companies access to recent research while academics gain practical experience.

The optimization project with Docker can help companies reduce operating costs, improve the security of their infrastructure, and increase their responsiveness to market demands [13]. Likewise, it can contribute to the resolution of real-world problems by helping companies improve their efficiency and productivity.

The development of environmental projects in the context of container technologies and cloud computing makes it possible to optimize the virtual server infrastructure. Container technologies enable efficient resource utilization, which can lead to cost savings. It maintains that developing projects in the environmental area is vital due to the need for reproducibility and portability in scientific computing [14]. Therefore, the adoption of container technologies in the scientific computing community has contributed to this need by enabling consistent installations, dependencies, and environments across each image.

In the broad and emerging domain of containerization and virtualization, the development of their research determined that the inherent vulnerabilities in a Docker ecosystem were meticulously explored in the last generation. Therefore, four critical high-demand aspects of Docker vulnerability in an enterprise environment were explored: file system isolation, process and communication isolation, device management and host resource constraints, and network isolation and image transmission.

In the expanding field of software containerization, a notable article explored the relationship between outdated Docker containers, severe vulnerabilities, and bugs. Packaging software in containers is becoming standard practice when deploying services in the cloud and other environments, with Docker images being one of the most popular container technologies [15].

One notable paper explored the rise of serverless containers as a viable approach for scientific workflows in the dynamic realm of cloud computing. The growing popularity of the serverless computing approach has given rise to new cloud infrastructures operating under the container [16] as Service (CaaS) model, such as AWS Fargate, Google Cloud Run, and Azure Container Instances.

Explored the technology of virtual machines and containers [17]. The main objective of the research was to optimize the hardware resources used by virtual machines during the virtualization process, migrating a large volume of services to the virtual container infrastructure.

Focused on using Docker containers to improve software and web engineering research reproducibility [18]. From the above, they noted that reproducibility is an increasingly important issue in many academic disciplines, including computer science and software and web engineering. In these areas, scientific results often depend on developed algorithms, tools and prototypes, quantitative evaluations, and other computational analyses.

Docker is an open-source platform designed to automate applications' deployment, scaling, and management in lightweight, portable containers. Understanding Docker architecture and terminology is essential for effectively leveraging its capabilities. This guide overviews Docker's key components and commonly used terms.

Docker's architecture revolves around the Docker Engine, which makes containerization possible. The Docker Engine consists of the Docker Daemon (`dockerd`), a background service that manages containers, images, networks, and volumes. The Docker CLI (`docker`) is the command-line interface through which users interact with the Docker Daemon to perform various tasks like creating, running, and managing containers and images.

Key components within Docker include containers, images, Dockerfiles, volumes, and networks. Containers are lightweight, portable environments that run applications along with all their dependencies, ensuring consistency across different environments. Docker images, immutable templates for creating containers, are built from Dockerfiles that define the application's environment and dependencies. Volumes provide persistent storage for data that needs to be retained across container lifecycles, while Docker's networking capabilities allow containers to communicate internally and with the outside world.

Additionally, Docker incorporates a registry system for storing and distributing images. Docker Hub is the default public registry where users can access a vast library of images, while private registries allow organizations to manage and store their images securely. Together, these elements make Docker a powerful tool for developing, deploying, and managing applications in a consistent, scalable, and efficient way.

Figure 1 shows the Docker architecture, and Table 1 shows more references that discuss the optimization of virtual server infrastructure using Docker technology.

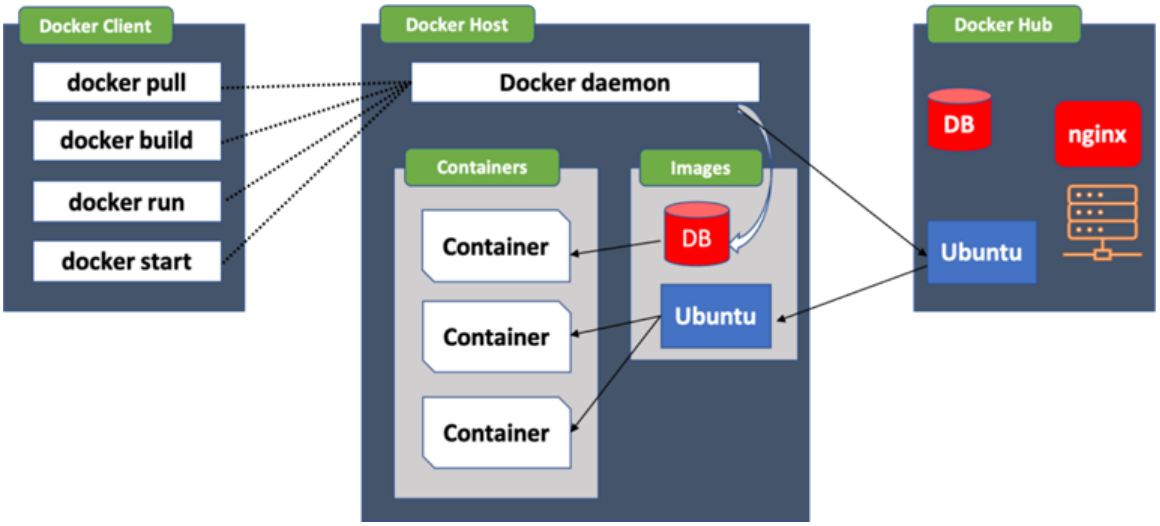


Fig. 1. Docker Architecture

Table 1. More references that discuss the optimization of virtual server infrastructure using Docker technology

Item	Title	Description
1	Apache Spark™: A Unified Engine for Big Data Processing	Discusses how Docker is used to streamline bigdata processing with Apache Spark [19]
2	An Efficient Resource Management Scheme inDocker Container Environments	This paper presents a novel resource managementscheme for Docker environments to enhance efficiency [20]
3	Performance Evaluation ofDocker Container and Virtual Machine in Cloud Computing Environment	A comparative study on the performance of Docker containers and virtual machines in cloud environments [21]
4	Container-based Cloud Resource Management with Deep Reinforcement Learning	Discusses the application of deep reinforcement learning to optimize resource management in Docker-based cloud environments [22]
5	Optimizing Microservices	Examines optimization strategies for deploying microservices with Docker and Kubernetes [23]
6	Deployment Using Docker and Kubernetes	Performance analysis of Docker containers running various applications [1]
7	A Performance Evaluation of Docker Containers for Containerized	Discusses techniques to improve security and performance in Dockerized environments [24]
8	Applications	A comprehensive review of security issues in Docker and potential mitigation techniques [25]
9	Enhancing Security and Performance of Containerized Applications Using	Focuses on cost-saving strategies in Docker container deployments on cloud platforms [26],
10	Docker	Discusses using AI techniques for dynamic resource allocation to optimize Docker-based infrastructures [27]

The Top-Down methodology in this Docker implementation project at The Company refers to a strategic and hierarchical approach to project management and execution. This methodology has been used in similar projects, such as the one developed by [28] and [29]. Transitioning from a VMware-based infrastructure to a Docker-based environment involves a systematic approach to ensure minimal disruption and maximum efficiency. Figure 2 shows the methodology and its phases.

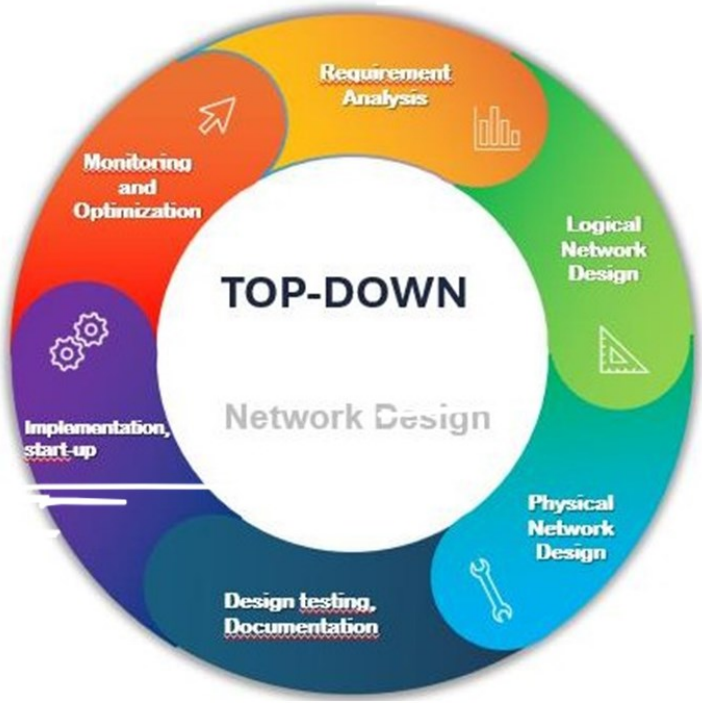


Fig. 2. Top-Down Methodology

Among the methodological aspects conducted to carry out the research, the following stand out:

- Mixed Approach: Combination of qualitative and quantitative methodologies to obtain a complete understanding of the impact of Docker on infrastructure.
- Research Design: Longitudinal study to evaluate performance before and after Docker implementation.
- Data Collection: Through server monitoring tools, surveys, and interviews with the technical staff of The Company
- Data Analysis: Descriptive and analytical statistics are used to evaluate performance improvement, cost efficiency, and thematic content analysis for interviews and survey responses.

The implementation of the Docker strategy for infrastructure optimization at Willard can be visualized in Figure 3.

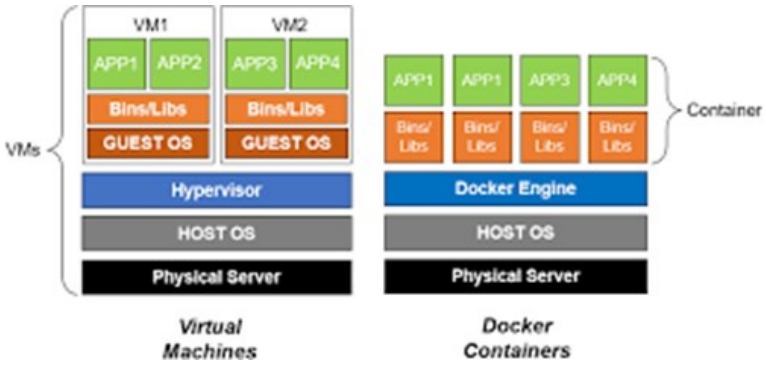


Fig. 3. Docker Implementation (Own Development)

Deploying server virtualization using Docker technology offers a lightweight and flexible approach to optimizing network infrastructure. Docker contains package applications and their dependencies, enabling consistent and efficient deployment across various environments. Table 2 shows a comprehensive guide to deploying server virtualization through Docker.

Docker is defined as an open-source containerization platform that allows developers to package applications in containers, standardizing their execution on any compatible operating system. The study's primary purpose is to present how Docker containers can overcome reproducibility challenges in web and software engineering research and discuss their applications in the field, like the study conducted by [30]. Docker containers were introduced as lightweight virtual machines that allow you to configure a complete computing environment, including dependencies, configurations, code, and necessary data, all within a single unit called an image.

Data were gathered from various sources to evaluate the Docker implementation comprehensively. These included server monitoring tools like Prometheus, Grafana, and Docker's native monitoring features, which provided real-time metrics like CPU usage, memory consumption, network I/O, and disk I/O. Surveys and interviews were conducted with IT staff, developers, and end-users to gather qualitative insights on performance, ease of use, and any challenges encountered. Financial records from the finance department were analyzed to track cost efficiency. At the same time, performance logs from VMware and Docker environments offered a detailed comparison of system metrics before and after Docker implementation.

Various tools and instruments were utilized to collect data. Prometheus was employed to gather essential metrics on resource usage, which were then visualized and analyzed using Grafana. Docker Stats, a command-line tool, was used for real-time container performance monitoring. Surveys and questionnaires were distributed among IT staff and end-users to assess management ease, deployment speed, and perceived performance. In-depth interviews with key stakeholders, including IT managers and developers, provided further qualitative insights. Cost data was collected and analyzed using financial software, encompassing hardware, software licensing, energy consumption, and maintenance.

In order to guarantee an exhaustive assessment, numerous sampling methodologies were implemented during the data collection phase. The perspectives of each group were represented

through stratified sampling, which enabled the selection of IT staff, developers, and end-users from a range of demographic groups. Purposive sampling was employed to interview key stakeholders directly involved in the Docker implementation. A random sampling approach was employed To obtain feedback from a diverse range of end users regarding the effectiveness of the application. This method ensured the fair representation of all pertinent viewpoints, enhancing the evaluation's robustness.

Table 2. Comprehensive guide to deploying server virtualization through Docker

Item	Procedure	Description
1	Assess Infrastructure Needs	Evaluate Current Setup: Understand your existing infrastructure, applications, and workloads.
		Identify Goals: Define the objectives for using Docker (e.g., improved deployment speed, resource efficiency, scalability)
2	Install Docker	System Requirements: Ensure your servers meet Docker's system requirements
		Install Docker Engine: Install Docker on servers. For Linux, it can follow the official installation guide.
3	Plan the Container Architecture	Microservices Design: Break down applications into smaller, manageable microservices if applicable
		Resource Allocation: Plan how to allocate CPU, memory, and storage resources to the containers. Dockerfile: Create Dockerfiles to define the application environments.
4	Create Docker Images	Dockerfile: Create Dockerfiles to define the application environments. # Example Dockerfile FROM ubuntu:20.04 RUN apt-get update && apt-get install -y \ nginx \&& rm -rf /var/lib/apt/lists/* COPY. /usr/share/nginx/HTML CMD ["Nginx," "-g," "daemon off;"]
		Build Images: Build Docker images using Dockerfiles. Docker build -t myapp: latest.
5	Deploy Containers	Run Containers: Deploy containers using the Docker run command or Docker Compose for multi-container applications. docker run -d -p 80:80 myapp: latest
		Docker Networking: Configure Docker networks (bridge, host, overlay) based on the needs.
6	Network Configuration	docker network create my_network docker run -d --network my_network --name my_container myapp:latest
7	Store Management	Volumes: Use Docker volumes to persist data. docker volume create my_volume docker run -d -v my_volume:/data myapp: latest
8	Orchestration and Scaling	Docker Swarm: Use Docker Swarm for orchestration and scaling. docker swarm init docker service create --name my_service -- replicas three myapp: latest
		Kubernetes: For more advanced orchestration, consider using Kubernetes. # Example Kubernetes Deployment apiVersion: apps/v1 kind: Deployment metadata: Name: my-deployment spec: Replicas: 3 selector: matchLabels: App: myapp template: metadata: Labels: App: myapp spec: containers: - name: myapp image: myapp: latest ports: - containerPort: 80
9	Monitoring and Logging	Monitoring Tools: Implement monitoring with tools like Prometheus, Grafana, or Docker's built-in stats.
		Logging: Use centralized solutions such as ELK Stack (Elasticsearch, Logstash, Kibana).
10	Security Best Practices	Least Privilege: Run containers with the least privilege necessary
		Regular Updates: Keep Docker, images, and dependencies up to date
		Image Scanning: Use tools like Docker Security Scanning or Clair to scan images for vulnerabilities
11	Backup and Recovery	Data Backup: Regularly back up data from Docker volumes.
		Disaster Recovery: Plan for container recovery and implement failover strategies
12	Documentation and Training	Documentation: Document the Docker setup, processes, and best practices.
		Training: Train the team on Docker management troubleshooting

The data collection process was designed to encompass pre- and post-Docker implementation periods. Prior to the deployment of Docker, baseline data was collected using VMware vCenter, Prometheus, and Grafana. The data set included metrics such as CPU utilization, memory consumption, network I/O, and system uptime. The same instruments and metrics were utilized for the post-implementation data collection to ensure a direct comparison. The precise capture and analysis of any performance changes resulting from Docker were guaranteed by the continuous monitoring of both segments over one month.

The surveys and questionnaires were designed with great care to collect both qualitative and quantitative data. The questionnaire was distributed via online platforms, including SurveyMonkey and Google Forms, and focused on pivotal areas, such as operational challenges, deployment speed, and management ease. End-user surveys were randomly distributed, while IT staff surveys were directed at those responsible for administering the VMware and Docker environments. The data collection period was two weeks, deemed sufficient to obtain an adequate sample size for analysis.

A semi-structured interview guide was developed to address critical areas, including operational challenges, cost, and performance. Individual interviews were conducted with IT administrators, developers, and finance personnel. The interviews were recorded and transcribed for in-depth analysis with consent, lasting between 30 and 45 minutes. This method allowed for examining individual perspectives on the Docker implementation, providing detailed insights.

The acquisition of cost data encompassed both the baseline and post-implementation phases. The costs associated with the hardware, software licensing, energy consumption, and maintenance were recorded in the baseline data prior to the deployment of Docker. The same financial software and methodologies were utilized to monitor post-implementation costs. This method guaranteed data collection consistency, enabling a precise comparison of costs before and after the implementation of Docker.

The collected data underwent various analyses. Performance data were subjected to comparative analysis, using statistical methods to identify improvements. Grafana was employed to create visual dashboards that illustrated these changes. Survey responses were analyzed quantitatively through descriptive statistics, while interview transcripts were coded and examined for recurring themes. Cost data were compared pre- and post-implementation to quantify savings, and ROI was calculated to evaluate the financial benefits of Docker implementation.

III. Results and Discussion

Optimizing Docker environments can significantly improve application performance, resource utilization, and operational efficiency. Optimizing resource limits for Docker containers can significantly enhance CPU and memory usage. Containers are designed to consume only the necessary resources, which prevents resource contention and boosts overall system performance. Research indicates that setting appropriate resource limits can improve performance by 15-30%, depending on the workload and resource constraints. Moreover, deploying optimized storage drivers and techniques, such as overlay file systems, can mitigate I/O overhead and enhance the responsiveness of applications. This method results in accelerated read/write operations, which can enhance the efficacy of I/O-intensive applications by up to 20%.

By minimizing the number of layers in Docker images and implementing multistage builds, it is possible to reduce the size of the images substantially. This is particularly advantageous in continuous integration and deployment (CI/CD) pipelines, as smaller images result in faster download and startup periods. The reduction in image size typically results in a 30-50% reduction in deployment duration. Furthermore, leveraging Docker's layer caching mechanism can accelerate the build process. This optimization results in more expedient deployments and rebuilds, enhancing developer productivity and reducing the overall build time by 40-60%.

The selection of an appropriate networking model, such as a bridge, host, or overlay, based on the requirements of the application in question can result in a reduction of latency and an optimization of throughput. Research has demonstrated that optimized network configurations can increase throughput by between 25 and 35 percent and reduce latency by between 20 and 40 percent. Integrating software-defined networking (SDN) with Docker enables more efficient traffic

management and the mitigation of network congestion. In the context of distributed applications, this integration has the potential to enhance network performance by 15–25%. SDN represents a novel paradigm in network infrastructure development, offering substantial advantages as evidenced by research.

Orchestration tools such as Kubernetes can facilitate optimal resource allocation and enhance application performance by implementing sophisticated scheduling algorithms. Implementing an efficient scheduling system can result in a 10-25% increase in resource utilization while simultaneously ensuring the high availability and reliability of services. Moreover, applications can efficiently manage diverse workloads through dynamic autoscaling based on predictive algorithms and custom metrics. This reduces operational costs by up to 30-40%, which in turn leads to improved performance during periods of peak demand and cost savings during periods of low utilization.

Implementing routine vulnerability scans of Docker images represents an effective strategy for mitigating security risks. Integrating automated tools with CI/CD pipelines facilitates the early identification and mitigation of vulnerabilities, enhancing the overall security posture and reducing the attack surface. Namespaces, groups, and security profiles such as Seccomp and AppArmor can be employed to improve container isolation, thereby reducing the likelihood of security vulnerabilities. These methods guarantee that the system's overall security and reliability are enhanced, even if a single container is compromised, as it does not impact others.

Resource Cost Savings: Organizations can decrease the quantity of physical or virtual machines required to execute their duties by optimizing resource utilization. This results in immediate infrastructure cost savings. According to reports, infrastructure costs can be reduced by 20-50% in optimized Docker environments.

Management and Upkeep: The overhead of manual maintenance is reduced by simplifying and automating Docker container management through orchestration tools, which enables DevOps teams to concentrate on higher-value tasks and improves operational efficiency.

The technological infrastructure modernization initiative at Willard Batteries substantially improves operational efficiency and resource optimization. The transition from a VMware-based infrastructure with 16 virtual machines (VMs), which presently necessitates 256GB of RAM, 16TB of disk storage, and 64 CPU cores, to a Docker-based solution demonstrated a substantial enhancement in resource allocation computing and more efficient application management.

The company aims to minimize its hardware footprint by utilizing only 8 CPU processors, reducing the required storage to 10TB, and reducing the RAM to 64GB. This will be achieved without compromising the performance and availability of the 16 business-critical applications. This modification has the additional benefit of reducing capital investment (CAPEX) by eliminating the need for tangible hardware. Furthermore, it is anticipated that operating costs (OPEX) associated with power consumption, cooling, and maintenance will also decline.

The project's impact extends beyond the immediate reduction in costs. Using Docker containers enables Willard Batteries to create a more scalable and agile application environment. In a business environment requiring flexibility and responsiveness to evolving market demands, containers are indispensable. They facilitate faster deployment and simplified administration, which is crucial for maintaining competitiveness in today's rapidly changing market.

Moreover, this technological transformation allows for continuous integration and continuous delivery (CI/CD) practices, which enhances collaboration between teams and accelerates the software development lifecycle. Moreover, the enterprise's capacity to monitor, scale, and recover from incidents is further enhanced using Docker, which provides access to more extensive orchestration and monitoring tools.

Security, a fundamental component of any technological infrastructure, is also reinforced. Docker offers robust mechanisms for application isolation, thereby reducing the attack surface and improving vulnerability management. At the same time, Docker's ability to integrate with modern security tools allows Willard Batteries to maintain a secure operating environment and comply with regulatory standards. Figure 4 shows the proposed design for infrastructure optimization with Docker.

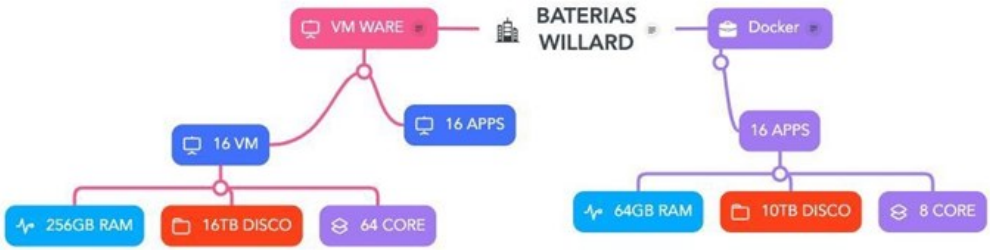


Fig. 4. IT Optimization with Docker (Own Development)

Figure 5, Figure 6, and Figure 7 show the creation of Docker images, the metrics and resource consumption, and the list of containers created for the initial tests of the optimization process, which have been satisfactory.

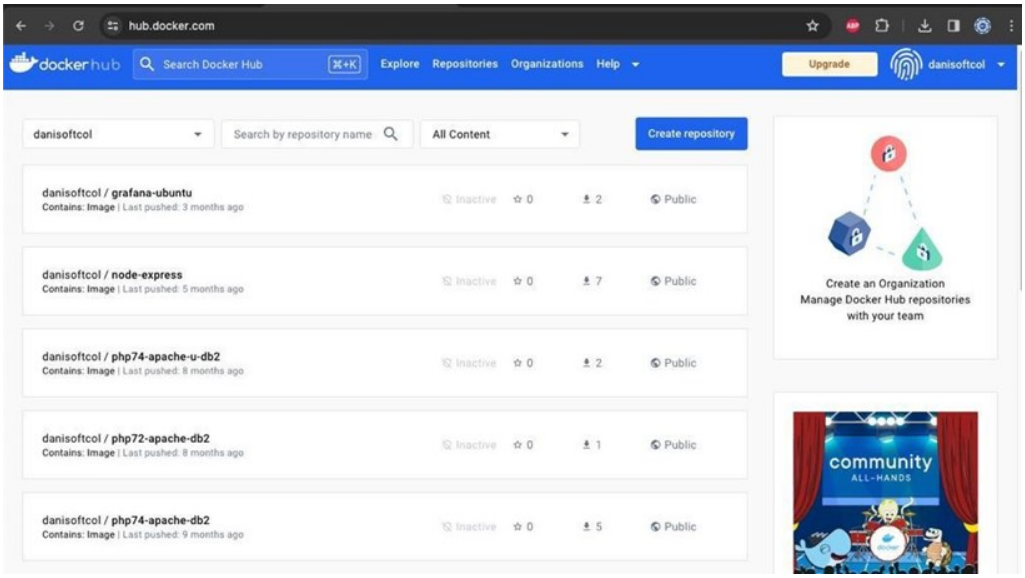
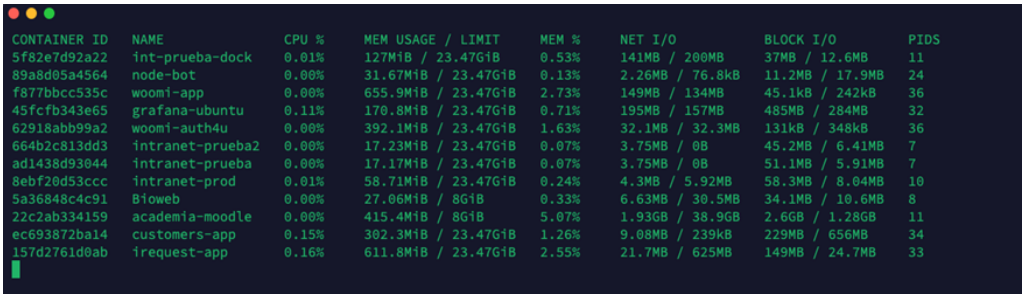


Fig. 5. Creation of Docker hub images (Own creation)



CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
5f82e7d92a22	int-prueba-dock	0.01%	127MiB / 23.47GiB	0.53%	141MB / 200MB	37MB / 12.6MB	11
89a8d05a4564	node-bot	0.00%	31.67MiB / 23.47GiB	0.13%	2.26MB / 76.8kB	11.2MB / 17.9MB	24
f877bbcc535c	woomi-app	0.00%	655.9MiB / 23.47GiB	2.73%	149MB / 134MB	45.1kB / 242kB	36
45fcfb343e65	grafana-ubuntu	0.11%	170.8MiB / 23.47GiB	0.71%	195MB / 157MB	485MB / 284MB	32
62918abb99a2	woomi-auth4u	0.00%	392.1MiB / 23.47GiB	1.63%	32.1MB / 32.3MB	131kB / 348kB	36
664b2c813dd3	intranet-prueba2	0.00%	17.23MiB / 23.47GiB	0.07%	3.75MB / 0B	45.2MB / 6.41MB	7
ad1438d93044	intranet-prueba	0.00%	17.17MiB / 23.47GiB	0.07%	3.75MB / 0B	51.1MB / 5.91MB	7
8ebf20d53ccc	intranet-prod	0.01%	58.71MiB / 23.47GiB	0.24%	4.3MB / 5.92MB	58.3MB / 8.04MB	10
5a36848c4c91	Bioweb	0.00%	27.06MiB / 8GiB	0.33%	6.63MB / 30.5MB	34.1MB / 10.6MB	8
22c2ab334159	academia-moodle	0.00%	415.4MiB / 8GiB	5.07%	1.93GB / 38.9GB	2.6GB / 1.28GB	11
ec693872ba14	customers-app	0.15%	302.3MiB / 23.47GiB	1.26%	9.08MB / 239kB	229MB / 656MB	34
157d2761d0ab	irequest-app	0.16%	611.8MiB / 23.47GiB	2.55%	21.7MB / 625MB	149MB / 24.7MB	33

Fig. 6. Metrics and resource consumption (Own elaboration)

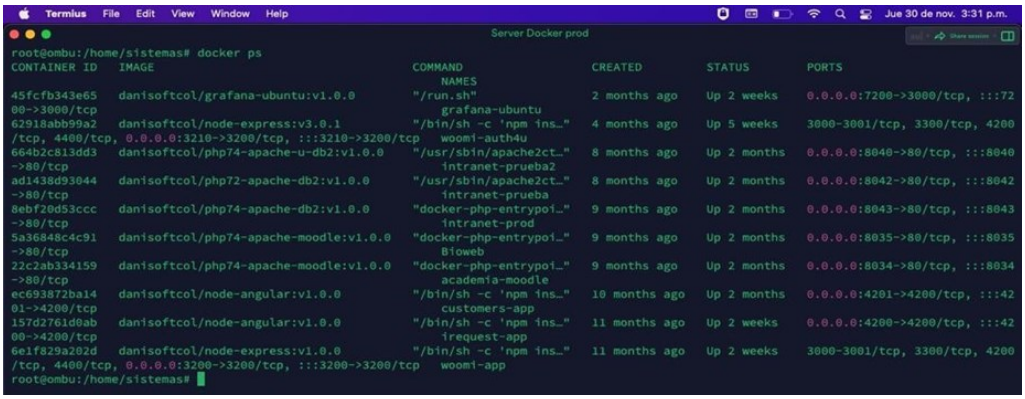


Fig. 7. Docker Resource Consumption

Figure 8 shows the custom images installed on the main Docker machine, which is Dockerhub, with their respective description and size.

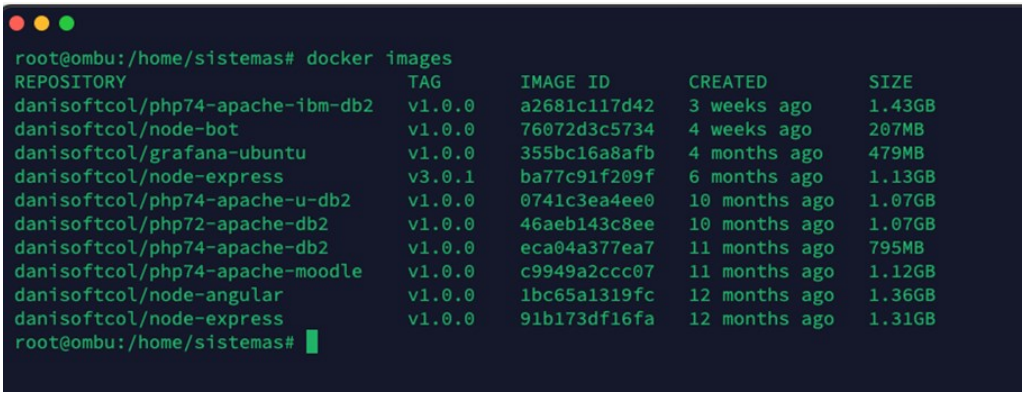


Fig. 8. Images in Docker

Figure 9 shows the information on the central Docker server and the allocated and available resources.

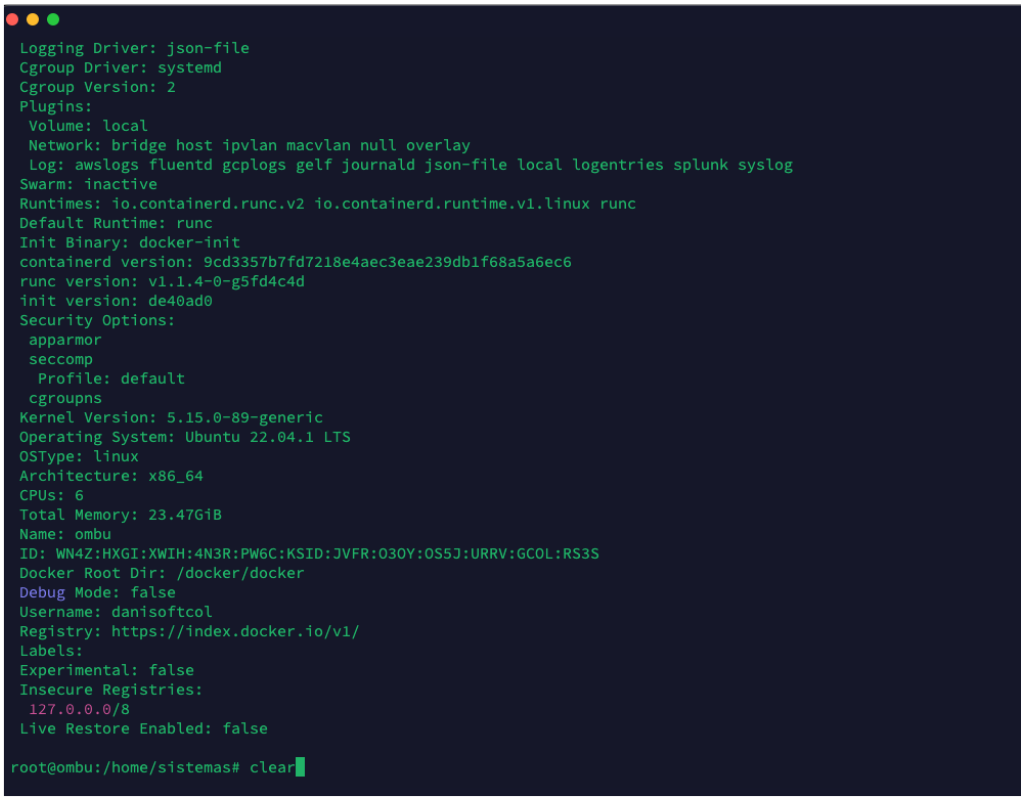


Fig. 9. Docker central server

The study's findings on implementing Docker technology to optimize virtual server infrastructure have significant practical implications for the company in the automotive sector. Resource efficiency, scalability, and security improvements can yield numerous cost savings, operational agility, and innovation benefits. By optimizing resource utilization through containerization, the company can reduce the number of physical servers required, leading to lower capital expenditures on hardware. Efficient resource utilization reduces power and cooling requirements, resulting in significant energy cost savings.

Docker enables rapid deployment of applications and updates. This agility allows the company to respond quickly to market demands and reduces time to market for new features and products. The contemporary DevOps practices, including continuous integration and continuous deployment

(CI/CD), are integrated seamlessly with the adoption of Docker. This facilitates enhanced collaboration between the development and operations teams and optimizes the development pipeline.

The company in the automotive sector will experience substantial practical implications due to the study's findings on the optimization of virtual server infrastructure through the use of Docker technology. It is possible to identify several advantages regarding operational agility, innovation, and cost reductions through enhancements to resource efficiency, scalability, and security. The company may reduce the number of physical servers required, thereby reducing capital expenditures on hardware, by optimizing resource utilization through containerization. The efficient utilization of resources results in significant energy cost savings, as it reduces the need for power and ventilation.

Docker enables the expeditious deployment of applications and updates. This agility allows the company to respond rapidly to market demands and reduce the time it takes to market for new products and features. Incorporating Docker facilitates the seamless integration of contemporary DevOps practices within the organizational structure, including continuous integration and continuous deployment (CI/CD). This enhances collaboration between the development and operations teams and optimizes the development pipeline.

IV. Conclusions

The implementation of Docker represents a significant milestone in the company's pursuit of operational efficiency and innovation. It reflects the ambitious project to modernize Willard Batteries' virtual server infrastructure. This initiative has not only facilitated technological advancements but has also served to reaffirm the company's dedication to remaining at the vanguard of its industry. The achievements are notable for their impact on reducing costs, creating a more agile platform for future growth, improving resource efficiency, scalable operations, and enhancing infrastructure security.

However, the journey was not without its challenges. The team was confronted with several technical challenges, was obliged to adapt to new methodologies and tools, and was responsible for ensuring the continuity of business operations during the transition period. These challenges highlighted the importance of effective team collaboration, adaptability, and meticulous planning. The initiative has furnished the company with invaluable experience, enabling it to navigate the ever-changing business landscape and embrace emerging technologies confidently.

In the future, the results of this initiative have identified numerous areas that require further research and exploration, including multi-cloud deployments, performance optimization, and advanced orchestration. By continuing to innovate and optimize Docker technology, Willard Batteries can enhance operational efficiency, reinforce security, and achieve sustainable growth. In conclusion, the successful implementation of Docker has established a robust foundation for a more efficient, secure, and scalable infrastructure, which aligns with the company's long-term vision of excellence and leadership in the automotive sector.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

References

[1] P. Kaur, J. K. Josan, and N. Neeru, "Performance analysis of docker containerization and virtualization," in Proceedings of Third International Conference on Communication, Computing and Electronics Systems: ICCCES 2021, Springer, 2022, pp. 863–877.

[2] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: a performance comparison," in 2015 IEEE International Conference on cloud engineering, IEEE, 2015, pp. 386–393.

[3] F. Guo, Y. Li, M. Lv, Y. Xu, and J. C. Lui, "HP-mapper: A high performance storage driver for docker containers," in Proceedings of the ACM Symposium on Cloud Computing, 2019, pp. 325–336.

[4] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," Linux j, vol. 239, no. 2, p. 2, 2014.

[5] C. Boettiger, "An introduction to Docker for reproducible research," ACM SIGOPS Oper. Syst. Rev., vol. 49, no. 1, pp. 71–79, 2015.

[6] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: a state-of-the-art review," IEEE Trans. Cloud Comput., vol. 7, no. 3, pp. 677–692, 2017.

[7] M. S. Bonfim, K. L. Dias, and S. F. Fernandes, "Integrated NFV/SDN architectures: A systematic literature review," ACM Comput. Surv., vol. 51, no. 6, pp. 1–39, 2019.

[8] K. Senjab, S. Abbas, N. Ahmed, and A. U. R. Khan, "A survey of Kubernetes scheduling algorithms," J. Cloud Comput., vol. 12, no. 1, p. 87, 2023.

[9] A. Nakarmi, H. Kesharwani, T. Mallick, S. Jhingran, and G. Raj, "A Comprehensive Study on Optimization Techniques for Microservices Deployment," in 2024 Sixth International Conference on Computational Intelligence and Communication Technologies (CCICT), IEEE Computer Society, 2024, pp. 133-140.

[10] R. Shu, X. Gu, and W. Enck, "A study of security vulnerabilities on docker hub," in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017, pp. 269–280.

[11] A. R. Manu, J. K. Patel, S. Akhtar, V. K. Agrawal, and K. B. S. Murthy, "Docker container security via heuristics-based multilateral security-conceptual and pragmatic study," in 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), IEEE, 2016, pp. 1–14.

[12] F. Dobslaw, K. Angelin, L. M. Öberg, and A. Ahmad, "The Gap between Higher Education and the Software Industry—A Case Study on Technology Differences," in Proceedings of the 5th European Conference on Software Engineering Education, 2023, pp. 11–21.

[13] M. Al-Rakhami, M. Alsahli, M. M. Hassan, A. Alamri, A. Guerrieri, and G. Fortino, "Cost efficient edge intelligence framework using docker containers," in 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech, IEEE, 2018, pp. 800–807.

[14] D. Moreau, K. Wiebels, and C. Boettiger, "Containers for computational reproducibility," Nat. Rev. Methods Prim., vol. 3, no. 1, p. 50, Jul. 2023.

[15] A. Zerouali, T. Mens, G. Robles, and J. M. Gonzalez-Barahona, "On the relation between outdated docker containers, severity vulnerabilities, and bugs," in 2019 ieee 26th international conference on software analysis, evolution and reengineering (saner), IEEE, 2019, pp. 491–501.

[16] P. Z. Vaillancourt, J. E. Coulter, R. Knepper, and B. Barker, "Self-scaling clusters and reproducible containers to enable scientific computing," in 2020 IEEE High-Performance Extreme Computing Conference (HPEC), IEEE, 2020, pp. 1–8.

[17] A. V. Hernández, L. V. Ledo, and J. P. León, "Implementación de la herramienta de gestion de redes OpManager en contenedores Docker," Tono, Rev. Técnica la Empres. Telecomunicaciones Cuba SA, vol. 18, no. 2, pp. 37–49, 2022.

[18] J. Cito and H. C. Gall, "Using docker containers to improve reproducibility in software engineering research," in Proceedings of the 38th international conference on software engineering companion, 2016, pp. 906–907.

[19] M. Zaharia et al., "Apache spark: a unified engine for big data processing," Commun. ACM, vol. 59, no. 11, pp. 56–65, 2016.

[20] D. K. Kang, G. B. Choi, S. H. Kim, I. S. Hwang, and C. H. Youn, "Workload-aware resource management for energy efficient heterogeneous docker containers," in 2016 IEEE Region 10 Conference (TENCON), IEEE, 2016, pp. 2428–2431.

[21] A. M. Potdar, D. G. Narayan, S. Kengond, and M. M. Mulla, "Performance evaluation of docker container and virtual machine," Procedia Comput. Sci., vol. 171, pp. 1419–1428, 2020.

[22] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," IEEE Cloud Comput., vol. 4, no. 6, pp. 60–69, 2017.

[23] G. Sayfan, Hands-On Microservices with Kubernetes: Build, deploy, and manage scalable microservices on Kubernetes. Packt Publishing Ltd, 2019.

[24] C. Puliafito, C. Vallati, E. Mingozzi, G. Merlino, F. Longo, and A. Puliafito, "Container migration in the fog: A performance evaluation," Sensors, vol. 19, no. 7, p. 1488, 2019.

[25] D. P. VS, S. C. Sethuraman, and M. K. Khan, "Container security: precaution levels, mitigation strategies, and research perspectives," Comput. Secur., p. 103490, 2023.

[26] Z. Zhong and R. Buyya, "A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources," ACM Trans. Internet Technol., vol. 20, no. 2, pp. 1–24, 2020.

[27] Z. Zaman, S. Rahman, F. Rafsani, I. R. Rahman, and M. Naznin, "DeepVRM: Deep Learning Based Virtual Resource Management for Energy Efficiency," J. Netw. Syst. Manag., vol. 31, no. 4, p. 66, 2023.

[28] L. Hernandez and G. Jimenez, "Characterization of the current conditions of the ITSA data centers according to standards of the green data centers friendly to the environment," in Cybernetics and Mathematics Applications in Intelligent Systems: Proceedings of the 6th Computer Science On-line Conference 2017 (CSOC2017), Vol 2 6, Springer, 2017, pp. 329–340.

[29] L. Hernandez, G. Jimenez, C. Baloco, A. Jimenez, and H. Hernandez, "Characterization of the Use of the Internet of Things in the Institutions of Higher Education of the City of Barranquilla and Its Metropolitan Area," in HCI International 2018–Posters' Extended Abstracts: 20th International Conference, HCI International 2018, Las Vegas, NV, USA, July 15-20, 2018, Proceedings, Part III 20, Springer, 2018, pp. 17–24.
[30] H. Pan, W. Hu, Q. Yang, and K. Zhang, "Design and Implementation of Server Management System Based on Docker," in 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, 2019, pp. 48–52.