JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205

Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

INTEGRATING PROMETHEUS MONITORING INTO THE COMPARATIVE FRAMEWORK OF VIRTUAL MACHINES AND DOCKER CONTAINERS

Ibragimov Ulugbek Muradilloyevich

Associate professor in Asia international University

Abstract: This article examines Docker containerization as a modern approach to application deployment and virtualization. Containerization packages applications with their dependencies, system libraries, and configurations into portable containers that ensure consistency across environments. Unlike traditional virtual machines that rely on hypervisors and full operating systems, Docker containers share the host OS kernel, making them smaller, faster to start, and more resource-efficient. The study outlines the four main Docker components—Containers, Client-Server architecture, Images, and Engine—and explains how they interact to build, run, and manage applications. A comparison between virtual machines and containers is presented, highlighting advantages such as scalability, portability, and rapid deployment, alongside challenges including security, storage persistence, and compatibility. The paper also describes methods of creating Docker images through base templates or automated Dockerfiles. Finally, it emphasizes the integration of Docker with Prometheus and Grafana for real-time monitoring, visualization, and performance optimization, showing how containerization supports agile, scalable, and reliable IT infrastructures.

Keywords: Containerization, Docker, Virtualization, Virtual Machines (VM), Docker Engine, Docker Images, Docker Containers, Dockerfile, Client-Server Architecture, Prometheus, Grafana.

Аннотация: В данной статье рассматривается контейнеризация Docker как современный подход к развертыванию и виртуализации приложений. Контейнеризация упаковывает приложения вместе с их зависимостями, системными библиотеками и конфигурациями в переносимые контейнеры, обеспечивающие согласованность в различных средах. В отличие от традиционных виртуальных машин, использующих гипервизоры и полноценные операционные системы, контейнеры Docker используют ядро операционной системы хоста, что делает их компактнее, быстрее запускается и более ресурсоэффективными. В исследовании рассматриваются четыре основных компонента Docker: контейнеры, клиент-серверная архитектура, образы и движок, а также объясняется их взаимодействие при сборке, запуске и управлении приложениями. Представлено сравнение виртуальных машин и контейнеров, в котором подчеркиваются такие преимущества, как масштабируемость, переносимость и быстрое развертывание, а также проблемы, связанные с безопасностью, сохранением данных и совместимостью. В статье также описываются методы создания образов Docker с помощью базовых шаблонов или автоматизированных Dockerfiles. Наконец, особое внимание уделяется интеграции Docker с Prometheus и Grafana для мониторинга, визуализации и оптимизации

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205

Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

производительности в режиме реального времени, демонстрируя, как контейнеризация поддерживает гибкие, масштабируемые и надежные ИТ-инфраструктуры.

Ключевые слова: контейнеризация, Docker, виртуализация, виртуальные машины (ВМ), Docker Engine, образы Docker, контейнеры Docker, Dockerfile, клиент-серверная архитектура, Prometheus, Grafana.

Annotatsiya: Ushbu maqola Docker konteynerizatsiyasini ilovalarni joylashtirish va virtuallashtirishga zamonaviy yondashuv sifatida ko'rib chiqadi. Konteynerlashtirish dasturlarni ularning bog'liqliklari, tizim kutubxonalari va konfiguratsiyalari bilan birga turli muhitlarda muvofiqlikni ta'minlaydigan portativ konteynerlarga joylashtiradi. Gipervisorlar va to'liq huquqli operatsion tizimlardan foydalanadigan an'anaviy virtual mashinalardan (VM) farqli o'laroq, Docker konteynerlari xost operatsion tizimining yadrosidan foydalanadi, bu ularni kichikroq, tezroq ishga tushirish va resurslardan tejamkor qiladi. Tadqiqot Dockerning to'rtta asosiy komponentini - konteynerlar, mijoz-server arxitekturasi, tasvirlar va dvigatelni o'rganadi va ularning ilovalarni yaratish, ishga tushirish va boshqarish uchun o'zaro ta'sirini tushuntiradi. VM va konteynerlarning taqqoslanishi taqdim etilgan bo'lib, unda masshtablilik, portativlik va tezkor joylashtirish kabi afzalliklar, shuningdek, xavfsizlik, ma'lumotlarning barqarorligi va moslik bilan bog'liq muammolar ko'rsatilgan. Maqolada, shuningdek, asosiy shablonlar yoki avtomatlashtirilgan Docker fayllari yordamida Docker tasvirlarini yaratish usullari tasvirlangan. Va nihoyat, konteynerlashtirish moslashuvchan, kengaytiriladigan va ishonchli IT infratuzilmalarini qanday qo'llab-quvvatlashini ko'rsatib, real vaqt rejimida ishlash monitoringi, vizualizatsiya va optimallashtirish uchun Dockerning Prometey va Grafana bilan integratsiyasiga alohida e'tibor qaratiladi.

Kalit so'zlar: konteynerlashtirish, Docker, virtualizatsiya, virtual mashinalar (VM), Docker Engine, Docker tasvirlari, Docker konteynerlari, Dockerfile, mijoz-server arxitekturasi, Prometey, Grafana.

Containerization is a technology that combines the application, related dependencies, and system libraries organized to build in the form of a container. The applications which are built and organized can be executed and deployed as a container. This platform is known as Docker, which makes sure that application works in every environment. It also automates the applications that will deploy into Containers. The Docker appends an additional layer of deployment engine over a container environment where the applications are executed and virtualized. To run a code efficiently, Docker helps to provide a quick and lightweight environment. The four main parts of Docker: Docker Containers, Docker Client-Server, Docker Images, and Docker Engine. The following sections will have a detailed explanation of these components

The essential part of the Docker system is Docker Engine, a Client-server application which is installed on the host machine with the following components[1,3].

- Docker Daemon: a type of long-running program (the dockerd command) which helps

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

to create, build, run application.

- A Rest API is used to communicate to the docker daemon.
- A client sends a request to docker daemon through the terminal to access the operations.

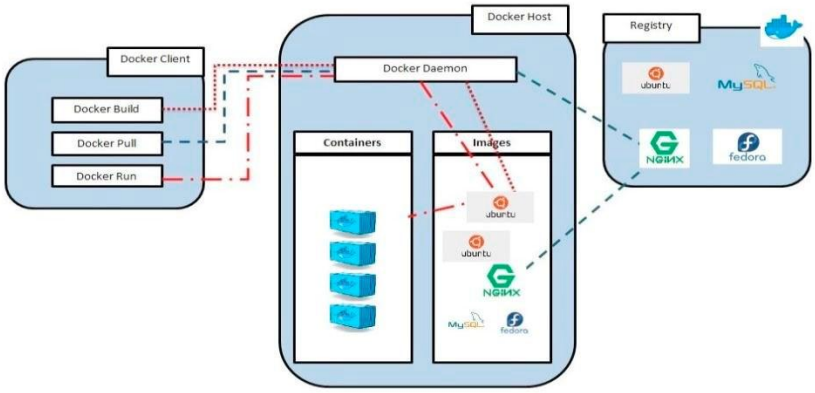

Fig. 1. Architecture of Docker Container

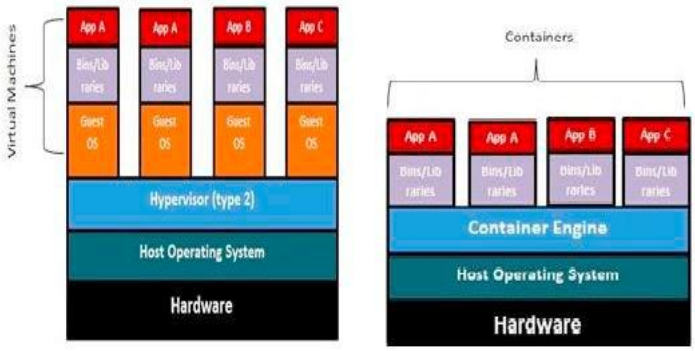

Fig. 2. (a) Hypervisor based architecture (b) Container based architecture

Docker technology mainly refers to client-server architecture. The client communicates to the Docker daemon, which acts as a server that is present within the host machine. The daemon works as three major processes running, building, and distributing containers. Both the Docker

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

container and daemon can be placed in a single machine. Fig . 1. shows the Docker architecture.

Docker Images can be built by two methods. The primary method is to build an image with the help of a read - only template. The template consists of base images, either it can be Operating System such as centos, ubuntu or fedora, or any other base OS images that are lightweight. Generally, base images are the foundation of every image. It is necessary to build a new image whenever base images are created from scratch. This type of creating a new image is called "committing a change." The next method is to create a docker file which has all the instructions for creating a docker image. When the docker build command is executed from the terminal, the image will be created with all the dependencies which are mentioned in the docker file. This process is known as a n automated method of building an image[2,4].

Docker Containers are created by Docker image. To run the application in a confined way, every kit required for the application is to be held by the container. The container images can be created based on the service requirement for the application or software. Suppose an application that includes Ubuntu OS and Nginx server should be appended into the docker file. Using the command "docker run," container with the image of Ubuntu OS consisti ng of the Nginx server is created and starts running.

Docker is sometimes referred to as lightweight VMs, but they are not VMs. The underlying technology, as discussed in the below Table 1, differences in virtualization technologies. Fig. 2. shows the architecture of the virtual machine and docker container[5,7].

Table 1. Virtual Machine versus Docker Container

	Virtual Machines	Docker Containers
Isolation Process Level	Hardware	Operating System
Operating System	Separated	Shared
Boot up time	Long	Short
Resources usage	More	Less
Pre-built Images	Hard to find and manage	Already available for home server
Customised preconfigured .	Hard to build	Easy to build
Size	Bigger because they contain whole OS underneath	Smaller with only docker engines over the host OS

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

Mobility	Easy to move to a new host OS	Destroyed and recreated instead of moving.
Creation time	Longer	Within seconds

Advantages of virtual machines

Over the past 20 years, virtual machines have become the de facto standard for enterprise virtualization, offering businesses numerous benefits, including:

- Independence. The isolation of each VM means that failures or crashes in the VM's OS or applications will not impact other VMs on the same machine or on other machines in the datacenter environment.

- Resource Utilization. Because multiple VMs can be provisioned and deployed on a single physical machine, the system can efficiently host multiple workloads without the need to purchase multiple machines. This enables server consolidation within the datacenter.

- Availability. Virtual machine portability allows virtual machines to be balanced across multiple systems to improve performance and support system maintenance. VMs can also be copied to or restored from files, enabling rapid VM protection and recovery.

- Flexibility. Each VM requires its own OS, but each OS can be different. This allows enterprises to use multiple OSs on a single physical machine.

- Security. Hypervisors and the logical isolation they provide have proven to be safe for virtual machines. Even if one VM is infected, it won't infect other VMs.

Disadvantages of virtual machines

Despite significant advantages, virtual machines also have a number of disadvantages:

- Size. VM instances can be large, using multiple processors and significant memory. This is well suited for enterprise-class workloads, but in practice, there is a limit to the number of VMs that can be deployed on a single machine.

- Time. Creating and deploying VMs can take anywhere from a few seconds to several minutes. While this isn't a lot of time in human terms, VMs may not scale quickly enough to meet dynamic or short-term computing needs.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

- Software licensing. Each VM requires an OS and a workload, so OS and application licensing costs can be significant. Enterprises must carefully manage VM deployments to account for licenses and ensure that VMs with expensive OS and application licenses are running and productive for the enterprise, avoiding VM sprawl.

Advantages of containers

Containers have their own unique properties and characteristics:

- Size. Containers use a single, shared OS kernel and don't have their own unique OS, making them much smaller logical units than virtual machines. This allows a single computer to host many more containers simultaneously than a VM.

- Speed. The small size of container instances allows them to be created and destroyed much faster than VMs. This makes containers well-suited for rapid scaling and short-term use, which may be impractical for VMs.

- Unique hypervisor. Specialized hypervisor platforms such as Docker, rkt, and Apache Mesos are used for hosting and managing containers.

- Immutability. Unlike VMs, containers don't change. Instead, a container orchestrator in the container's software layer starts and stops containers as needed. Likewise, the software running in containers isn't updated like traditional software. Instead, updates are included in a new container image that can be deployed where needed.

Disadvantages of containers

Containers have provided tremendous scalability and flexibility for enterprise organizations, but there are also several drawbacks:

- Performance. Containers can be numerous, and containers share a common OS in addition to the container layer. This means containers utilize resources efficiently, but contention can arise between containers when attempting to access hardware resources, such as networks. This contention can impact overall container performance.

- Compatibility. Containers packaged for one platform, such as Docker, may not work with other platforms. Likewise, some container tools may not work with different container platforms. For example, Red Hat OpenShift only works with the Kubernetes orchestrator. Consider the container ecosystem when evaluating container technologies for the enterprise.

- Storage. Containers are designed to be data-independent—the data in a container

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

disappears when the container is destroyed. There are ways to provide persistent storage for containers, such as Docker Data Volumes, but the issue of persistent container storage is often considered a separate issue. • Suitability. Containers are typically small and flexible structures, best suited for application components or microservices. Full-featured enterprise applications typically do not perform well in containers. Consider suitability when planning container deployments[4,6].

Comparison of containers and virtual machines

Containers and virtual machines each have unique characteristics, but there are many aspects to consider when choosing a virtualization technology. The following list highlights some of the most common comparisons:

- Deployment. Both containers and virtual machines can be deployed by loading the appropriate image file into the corresponding virtual instance. The image file contains all the components needed to run the VM or container. VMs can also be deployed manually, as they act as independent computers.

- Fault tolerance. Both containers and virtual machines support fault tolerance techniques such as load-balancing clustering, and both can be easily recovered or restarted from their respective image files. Containers are inherently stateless and typically carry little risk of data loss in the event of a failure. VMs, on the other hand, are persistent and potentially carry some risk of data loss in the event of a failure.

- Isolation. VMs offer complete logical isolation. VMs are unaware of the existence of other VMs in the environment, even if other VMs may be running on the same physical computer. Containers have less isolation because they share a common OS kernel.

- Load Balancing. VMs support live migration and can be moved from one virtualized computer to another to balance compute and network load. Containers are not moved, but rather recreated on target computers before the original instance is destroyed. Thus, containers easily support the concept of immutable infrastructure.

- Networking. Both containers and VMs can communicate between instances, both on the same computer and between computers connected by an IP network. A large number of containers, possible in a datacenter environment, can consume LAN bandwidth faster than a smaller number of VMs.

- Performance. Both containers and VMs provide excellent performance at speeds close to bare metal. VMs have a slight performance advantage because they don't face potential competition from a shared OS.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

- Persistent Storage. VMs include persistent storage with a virtual hard disk. Containers initially run from memory and require careful use of storage tools like Docker Data Volumes to persist container data after container destruction.

- Portability and Compatibility. VMs use common virtualization technologies, often based on or built on open source components. VMs can be portable and compatible with systems using similar virtualization platforms. Containers are highly portable, and packaging components into container image files allows containers to run on virtually any system that supports a container platform. However, container platforms are not fully compatible.

- Resources. Containers use fewer computing resources and typically host smaller applications than virtual machines, so an enterprise can potentially host many more containers on a single machine.

- Scalability. Containers are smaller and require fewer resources than virtual machines, so they can scale—create or destroy—much faster than virtual machines. Containers are often used where short-term compute instances are required.

- Security. VMs use separate OSs and applications, so security flaws in one VM do not propagate to other VMs in the environment. Containers share a common OS and can be susceptible to OS flaws, so they are not as secure as VMs. However, containers can run inside VMs as part of a hybrid deployment approach, which can help isolate potential security vulnerabilities.

- Updates and patching. VMs act as independent, fully functional computers. VM contents, such as the OS, drivers, and applications, can be patched and updated using traditional methods. Containers use a fixed package of libraries and binaries built to run on container platforms like Docker. Containers cannot be easily updated in real time. Instead, container files are updated and brought to specific versions, and then a new image file can be deployed to implement the updates. This also supports the concept of immutable infrastructure[9,11].

Integration of Docker containers and Prometheus

The Docker runtime is monitored to determine the performance and behavior of containers and host systems.

The following will need to be installed on the host system (all downloaded from the official websites):

- Docker to run containers.

- Prometheus to collect Docker environment metrics.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

- Grafana to visualize metrics collected from Prometheus.

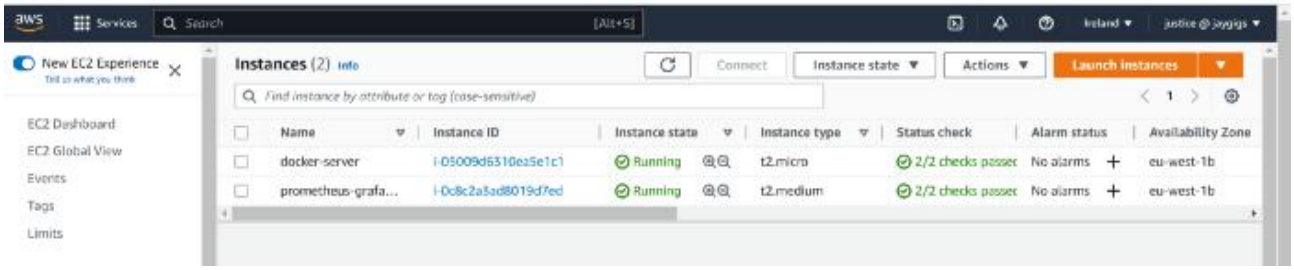

Fig. 3. Installing the Docker server.

Step 1. Installing the Prometheus Server

Prometheus is a monitoring system for collecting and storing time series data, with tools for querying and visualizing the data.

By default, Prometheus listens for incoming HTTP requests on port 9090. Navigating to http://localhost:9090 in your browser allows you to access the Prometheus web interface.

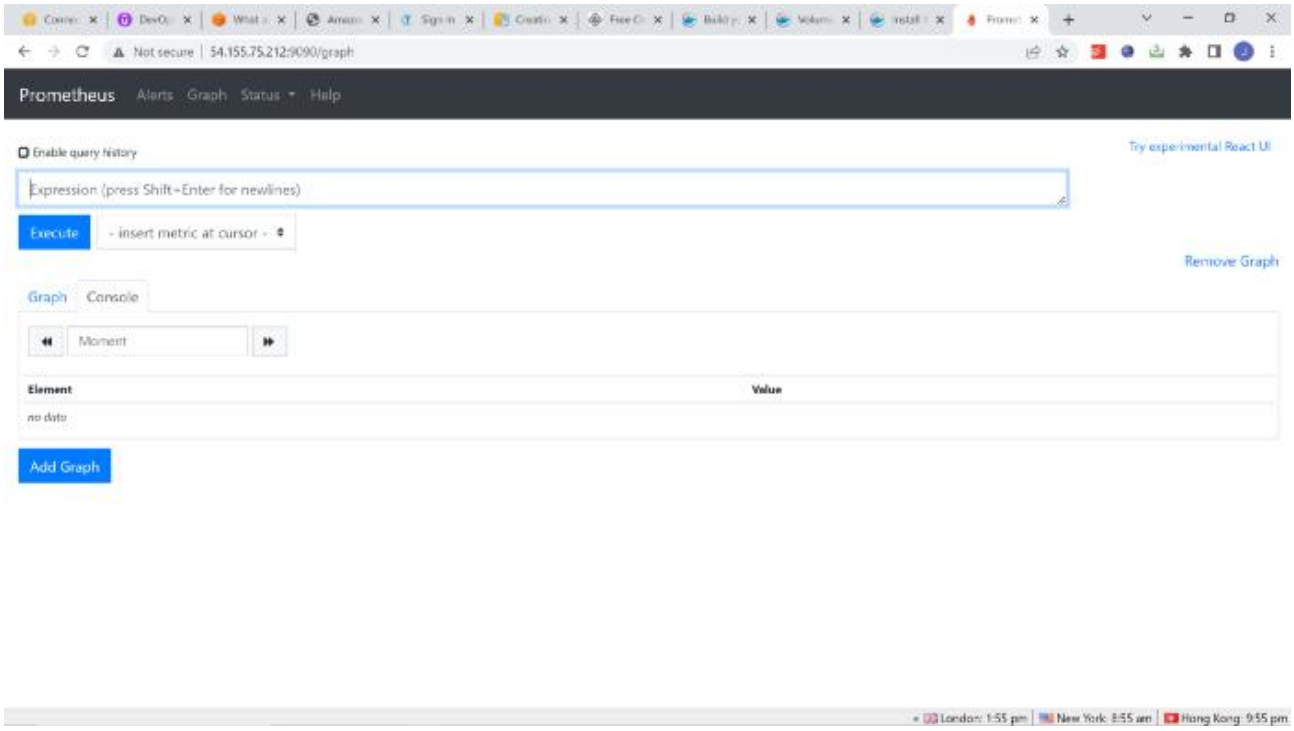

Fig.4. Prometheus system.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

Step 2. Installing the Grafana Server

Grafana is a popular open-source data visualization and tracking tool for creating dashboards and graphs.

After installation, open a browser and navigate to http://localhost:3000. The Grafana login page will appear. Enter the default username "admin" and password "admin."

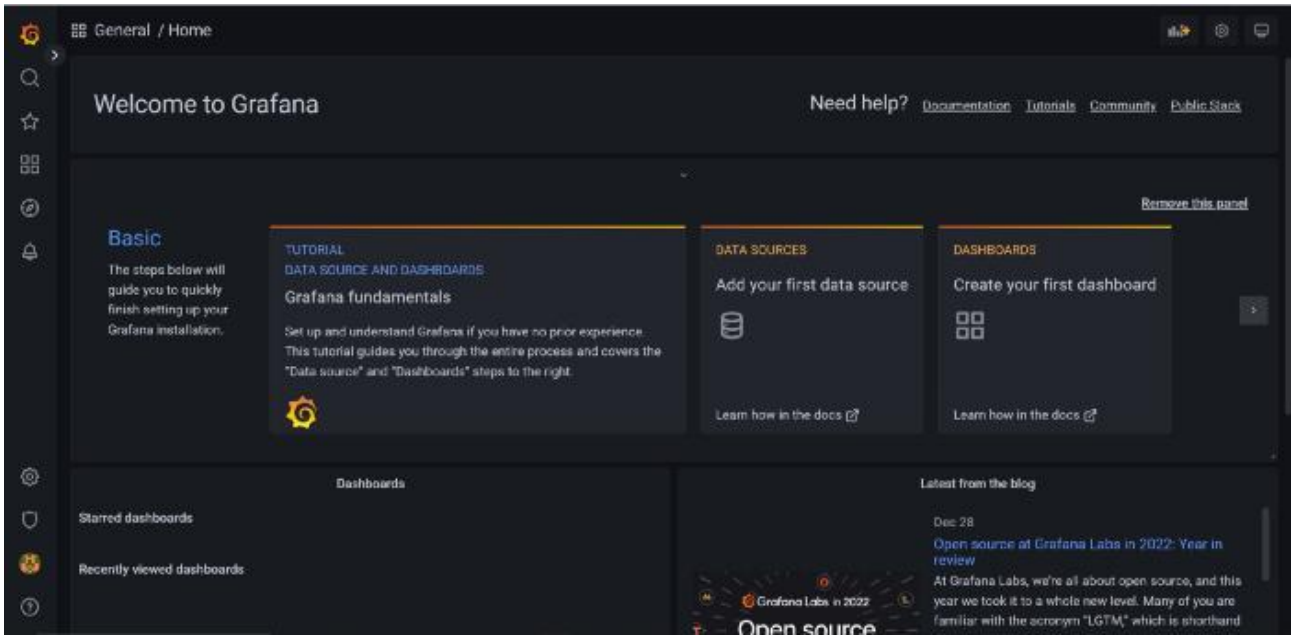

Fig. 4. Grafana system.

Step 3. Configuring Prometheus to retrieve metrics from the Docker runtime

The Prometheus and Grafana servers are running. Now let's create a Prometheus configuration file that specifies the targets for retrieving metrics from the Docker runtime and other parameters.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

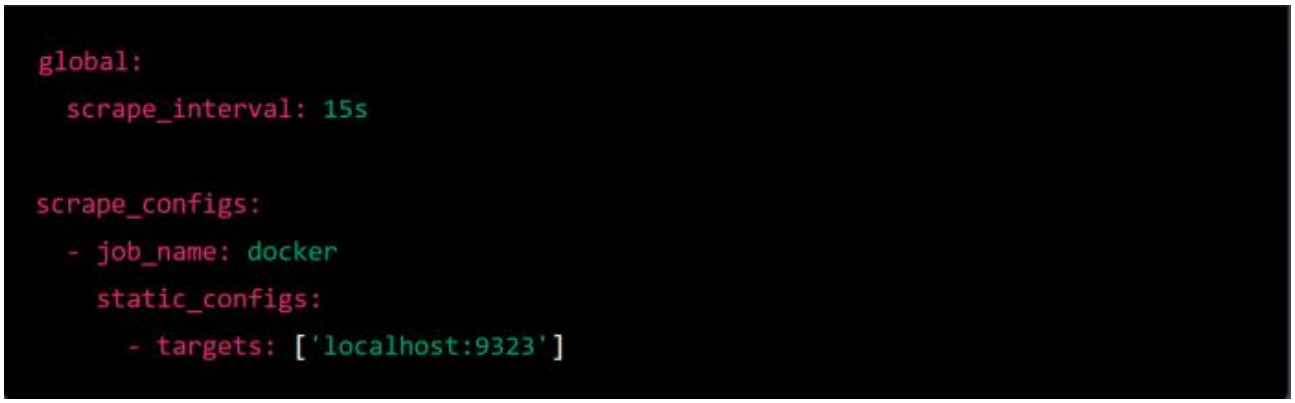

Fig. 5. Prometheus configuration file.

In this configuration file, we've configured Prometheus to fetch metrics from the Docker runtime every 15 seconds. The "targets" field specifies the host and port to fetch metrics from. Here, we're fetching metrics from the Docker daemon running on localhost at port 9323.

Step 4. Starting the Docker Firewall Configuration

To fetch metrics from the Docker runtime using Prometheus, we'll start the Docker Firewall. This dedicated tool provides metrics from the Docker runtime in a format suitable for fetching by Prometheus.

Start the Docker Firewall with the following command:

Fig. 6. Launching the firewall.

Step 5. Creating a Grafana Dashboard

After configuring Prometheus and running the Docker exporter, create a dashboard in Grafana to visualize metrics.

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205
Volume 15 Issue 09, September 2025
Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

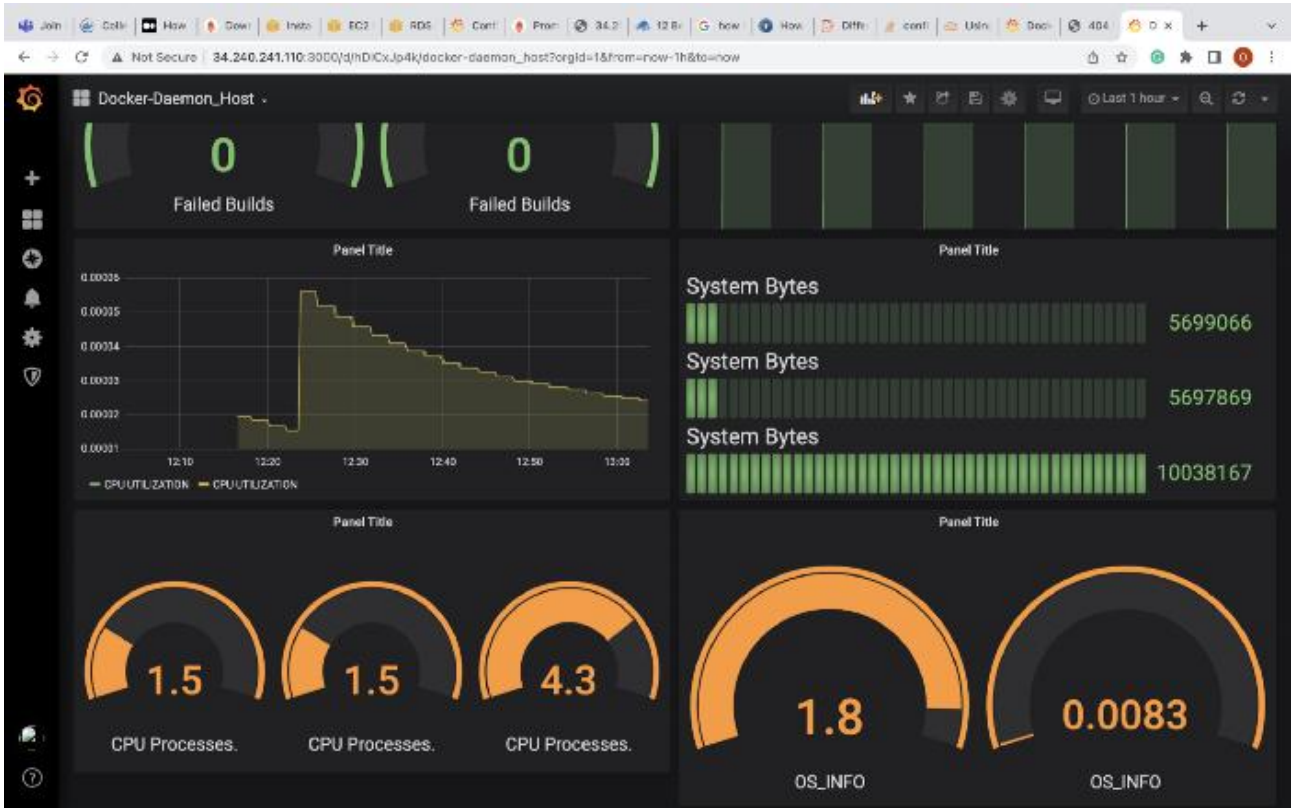

Fig.7. Grafana dashboard.

The Grafana dashboard shows metrics for failed builds, CPU usage, system bytes, CPU processes, and operating system information[10,12].

In conclusion, Docker containerization offers a lightweight, efficient, and scalable alternative to traditional virtualization, enabling faster deployment, portability, and simplified application management. While virtual machines remain valuable for their strong isolation and security, containers excel in speed, resource utilization, and adaptability for modern microservices-based architectures. By integrating Docker with monitoring tools like Prometheus and Grafana, organizations can gain real-time visibility into container performance, ensuring optimized operations, proactive issue detection, and improved reliability in dynamic IT environments.

References:

1. Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Journal, 2014(239).

2. Pahl, C. (2015). Containerization and the PaaS Cloud. IEEE Cloud Computing, 2(3), 24–31. https://doi.org/10.1109/MCC.2015.51

JOURNAL OF APPLIED SCIENCE AND SOCIAL SCIENCE
eISSN 2229-3113 pISSN 2229-3205

Volume 15 Issue 09, September 2025

Impact factor: 2019: 4.679 2020: 5.015 2021: 5.436, 2022: 5.242, 2023: 6.995, 2024 7.75

3. Dua, R., Raja, A. R., & Kakadia, D. (2014). Virtualization vs Containerization to Support PaaS. In 2014 IEEE International Conference on Cloud Engineering (IC2E), 610–614. https://doi.org/10.1109/IC2E.2014.41

4. Turnbull, J. (2014). The Docker Book: Containerization is the New Virtualization. James Turnbull.

5. Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes. ACM Queue, 14(1), 70–93. https://doi.org/10.1145/2898442.2898444

6. Chandramouli, R., Millar, T., & Ghosh, A. (2016). Containers: Challenges and Opportunities. NIST Special Publication, 800-190.

7. Polat, H., & Doğan, İ. (2020). A Comparative Study on Docker Swarm and Kubernetes. International Journal of Computer Applications, 975, 8887.

8. Mavridis, I., & Karatza, H. D. (2019). Performance and Security Issues of Containers in Cloud Computing. Future Generation Computer Systems, 94, 122–134. https://doi.org/10.1016/j.future.2018.11.005

9. Гуляев, Р. А., Ибрагимов, У. М., & Исмойилов, Ҳ. Б. (2023). Элементы автоматизации как помощники цифровизации агропромышленности. Science and Education, 4(3), 282-287.

10. Ibragimov, U. M. (2022). ARCHITECTURE FOR BUILDING THE SYSTEMS OF STORAGE AND ANALYSIS OF BIG DATA. Экономика и социум, (5-1 (96)), 205-208.

11. Gulyaev, R. A., Ibragimov, U. M., & Ismoyilov, H. B. (2022). The use of BIG DATA processing in a digitalized agro-industry system. Journal: INTERNATIONAL BULLETIN OF APPLIED SCIENCE AND TECHNOLOGY. ISSN, 2750-3402.

12. Ibragimov, U. M. (2022). ARCHITECTURE FOR BUILDING THE SYSTEMS OF STORAGE AND ANALYSIS OF BIG DATA. Экономика и социум, (5-1 (96)), 205-208.