

Module: Cloud Engineering Principles and Practice

MSc in Cybersecurity and Digital Forensics

Date of Submission: 01/05/2026 Students

Name: MD. Mehedi Hasan Mixon

Student ID: 2436061

Table of contents

1. Overview of Virtualisation, Containerisation and Cloud Computing -----	04
2. Virtualisation-----	05
2.1 Virtualisation of Shared Hardware -----	05
3. Containerisation -----	07
3.1 Docker Images -----	07
3.2 Docker Volumes-----	11
3.3 Docker Networks -----	13
4. Cloud Migration and Architectural Concepts in the Context of Khan Academy -----	16
5. The Complementary Relationship Between Load Balancing and Scaling-----	18
6. Customer Value in Integrated Cloud Architecture -----	19
7. Key Drivers Behind Cloud Adoption: Cost, Flexibility and Strategic Benefits -----	21
8. Serverless Computing and Its Contrast with Containerisation-----	23
9. Conclusion -----	24
10. References-----	26

List of Figures:

1. Fig.1 File Skeleton for Docker-----	08
2. Fig.2 Constructing a Docker Image-----	08
3. Fig.3 Managing a Docker Container -----	09
4. Fig.4 Docker Container Operating on Localhost-----	09
5. Fig.5 Docker Image Pushing to DockerHub -----	10
6. Fig.6 Docker Image Successfully Published on Docker Hub-----	10
7. Fig.7 Setting up and examining the Docker volume-----	11
8. Fig.8 Attaching the Volume to the Container-----	12
9. Fig.9 After mounting to the container, checking the volume-----	12
10. Fig.10 Creating and Verifying Network Creation -----	13
11. Fig.11 Assigning Network to the Container -----	14
12. Fig.12 Inspecting the Network Configuration -----	14
13. Fig.13 Confirming the IPv4 address assignment for the container ---	15

1. Overview of Virtualization, Containerization, and Cloud Computing

Virtualization, containerization, and cloud computing are the three main technologies that modern applications primarily rely on in today's digital world. Together, these technologies provide computational resources with flexibility, scalability, and on-demand access.

The concept of treating computer resources like a utility service is introduced by cloud computing. Users can obtain processing power, storage, and apps whenever needed and just pay for what they use, as opposed to owning infrastructure [1]. This strategy lowers initial costs and enables businesses to react swiftly to shifting needs.

Virtualisation plays a critical role in enabling this flexibility. It allows a single physical machine to host multiple virtual machines, each operating independently with its own operating system and applications. This is achieved through a hypervisor, which acts as an intermediary layer between hardware and software. By abstracting hardware resources, virtualisation improves utilisation and ensures isolation between different workloads, which enhances both reliability and security [2].

Behind this flexibility, virtualisation plays a foundational role. A hypervisor divides physical hardware into multiple virtual machines (VMs), each running its own operating system independently [2]. This ensures strong isolation between workloads, improving both security and reliability. On the other hand, containerisation provides a more lightweight alternative. Instead of running separate operating systems, containers share the host OS kernel, which significantly reduces overhead and improves performance [3]. Technologies like Docker have made container-based deployment highly efficient and portable.

These technologies work together to provide scalable, affordable, and highly available services in contemporary cloud settings.

2. Virtualisation

The practice of generating virtual replicas of actual computer resources is known as virtualization. It enables the simultaneous operation of several operating systems, each isolated from the others, on a single physical computer. A hypervisor, which controls resource allocation and guarantees that every virtual machine has the required processing power, memory, and storage, makes this feasible [4].

A software layer called a hypervisor controls this process by allocating CPU, memory, and storage resources to each VM [4]. There are two main types of hypervisors:

- Type 1 (bare-metal): Runs directly on hardware
- Type 2 (hosted): Runs on top of an existing operating system

Virtualisation improves resource utilisation because multiple workloads can share the same hardware instead of requiring separate machines.

Benefits:

- Better hardware utilisation
- Strong isolation between systems
- Easy backup and migration using snapshots

Limitations:

- Higher resource overhead (each VM runs a full OS)
- Performance degradation due to abstraction layers
- Complexity in management

2.1 Virtualisation of Shared Hardware

Effective resource management becomes crucial in settings where several virtual machines share the same real hardware. The hypervisor oversees allocating memory, allocating CPU time, and managing input/output operations across several virtual machines. Although this sharing increases overall productivity, there may be drawbacks. The hypervisor must carefully manage resources when several virtual machines (VMs) use the same physical hardware.

- Virtual CPUs (vCPUs) are a subset of CPUs
- Virtual memory approaches are used to manage memory
- I/O devices are shared across VMs

Performance interference, in which the workload of one virtual machine impacts the performance of another, is one such issue. Competition over shared resources like CPU cache, memory bandwidth, or disk I/O is usually the cause of this [5]. This may result in erratic performance in high-demand settings. Another issue is security. Attackers may use side-channel techniques to deduce sensitive information since virtual computers share underlying hardware components. Even while contemporary systems use a variety of mitigating techniques, total isolation is still challenging to achieve [6].

Despite these difficulties, virtualization is a key component of cloud computing, offering the abstraction and control needed for large-scale system management.

3. Containerisation

By emphasizing application-level isolation over hardware-level abstraction, containerization provides a more effective substitute for conventional virtualization. Applications and their dependencies are bundled into containers using this method, so they may function reliably in various contexts. Containers lack a complete operating system, in contrast to virtual machines. Rather, they use technologies like namespaces and control groups to ensure isolation despite sharing the host system's kernel. Containers are ideal for contemporary application designs like microservices because of their design, which drastically lowers resource consumption and enables nearly instantaneous startup [7]. Compared to virtual machines, containers' lightweight design allows for a higher workload density, which allows more apps to run on the same hardware. Because of this, containerization is especially appealing to businesses looking to save expenses and increase efficiency. However, containers present some security issues because they share the host operating system. All active containers may be impacted by a vulnerability in the host kernel. Therefore, while implementing container-based systems, appropriate configuration and security procedures are crucial.

3.1 Docker Images

Docker images serve as the foundation for container deployment. They are essentially read-only templates that contain everything required to run an application, including code, libraries, and configuration settings. These images are built using Dockerfiles, which define the steps needed to create the environment.

Once created, Docker images can be reused, shared, and deployed across different systems. This ensures consistency between development, testing, and production environments. Developers can also upload images to repositories such as Docker Hub, making them easily accessible for collaboration.

While this approach simplifies deployment, it also introduces security concerns. Publicly available images may contain vulnerabilities or unnecessary components, making it important to verify and scan images before use [8].

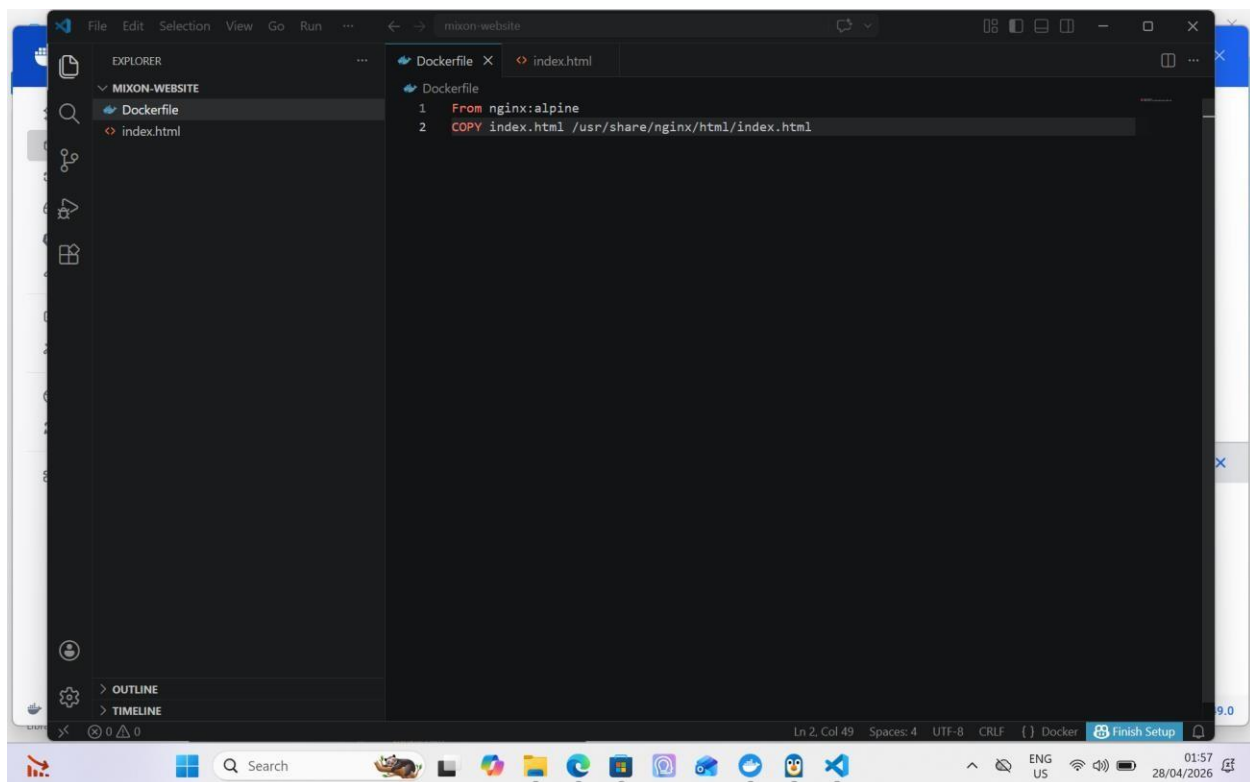


Fig.1 - File Skeleton for Docker

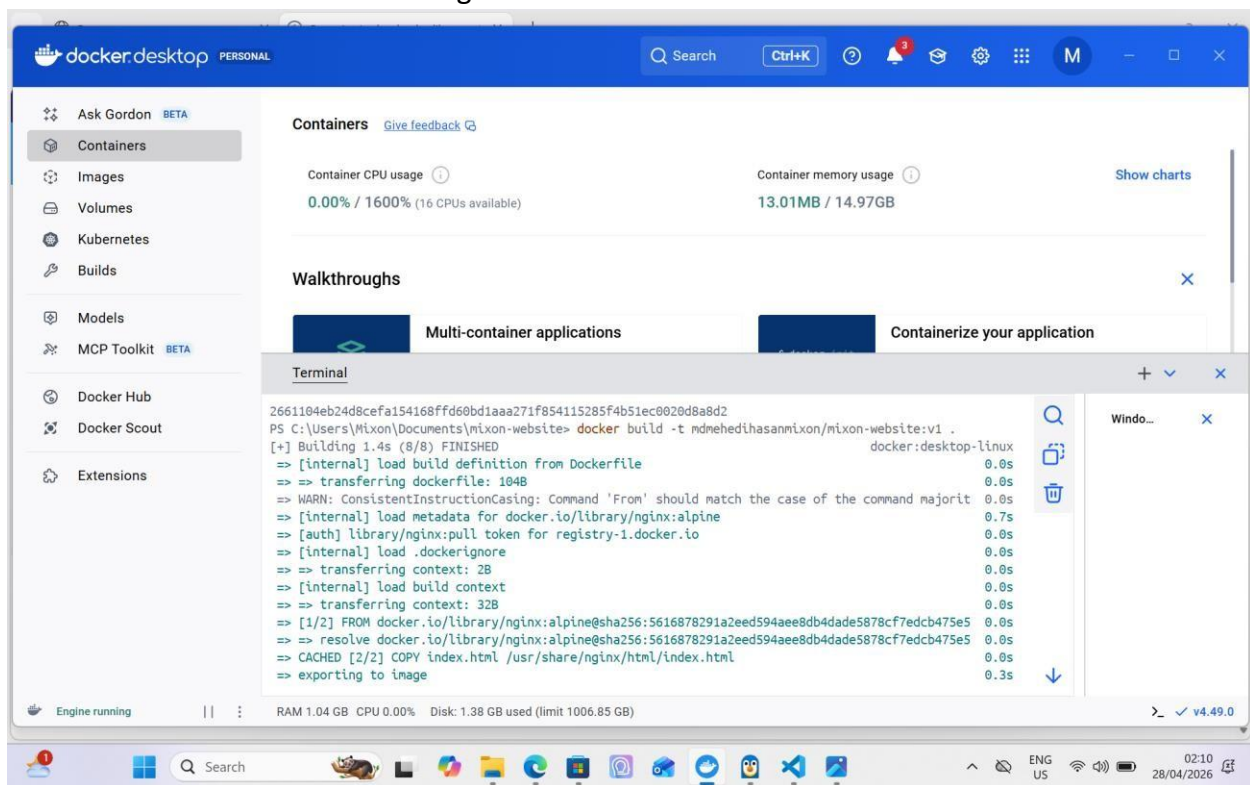


Fig.2 - Constructing a Docker Image

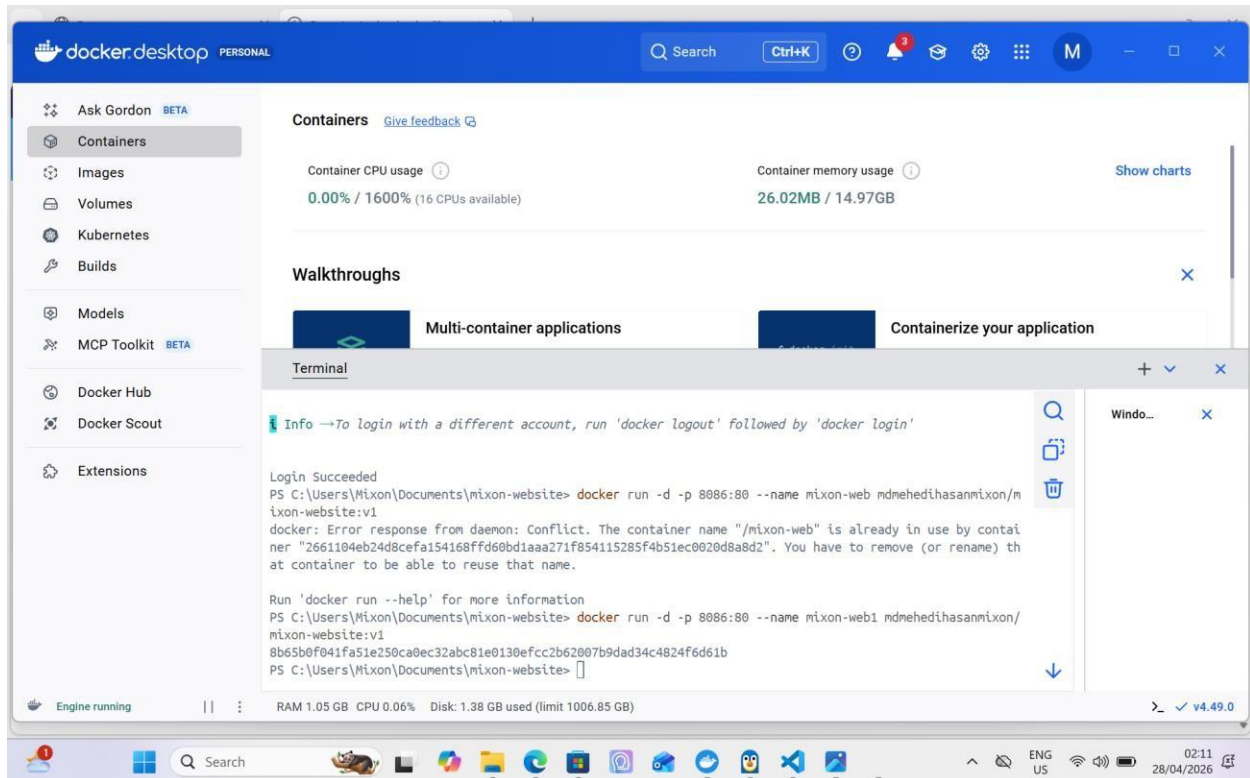


Fig.3 - Managing a Docker Container

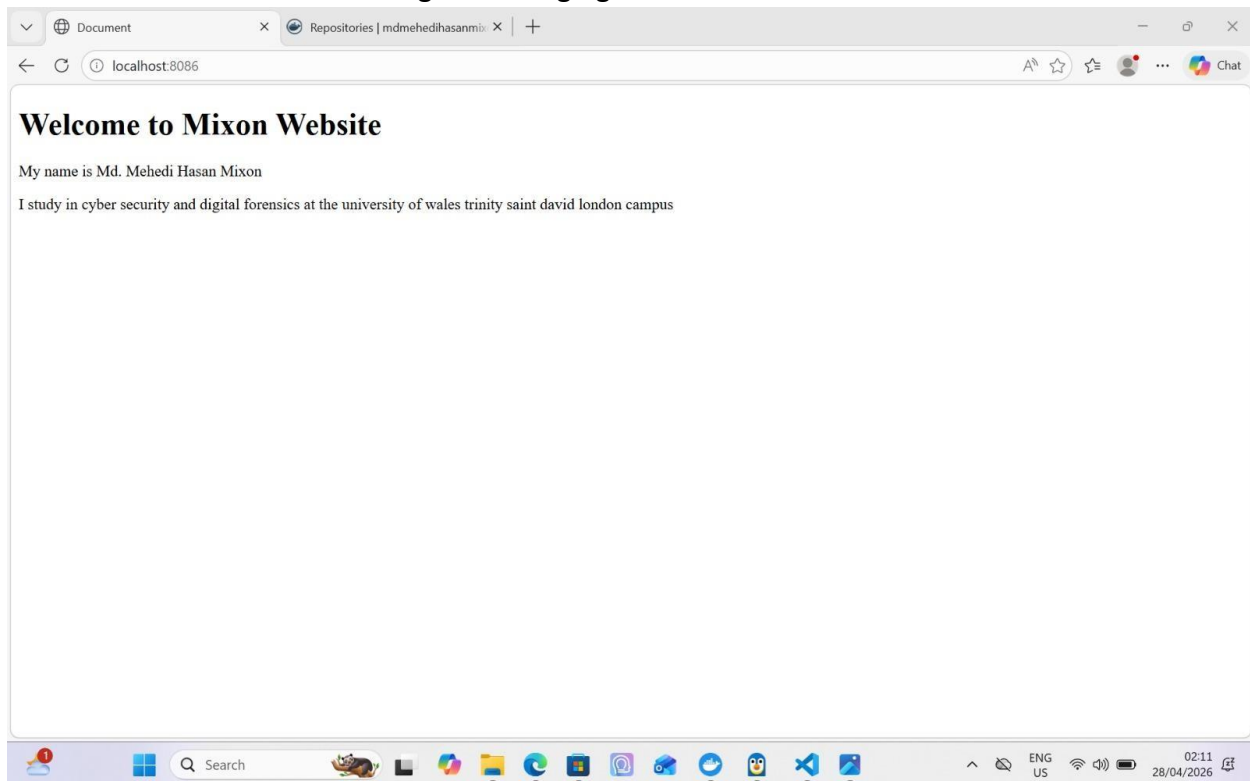


Fig.4 - Docker Container Operating on Localhost

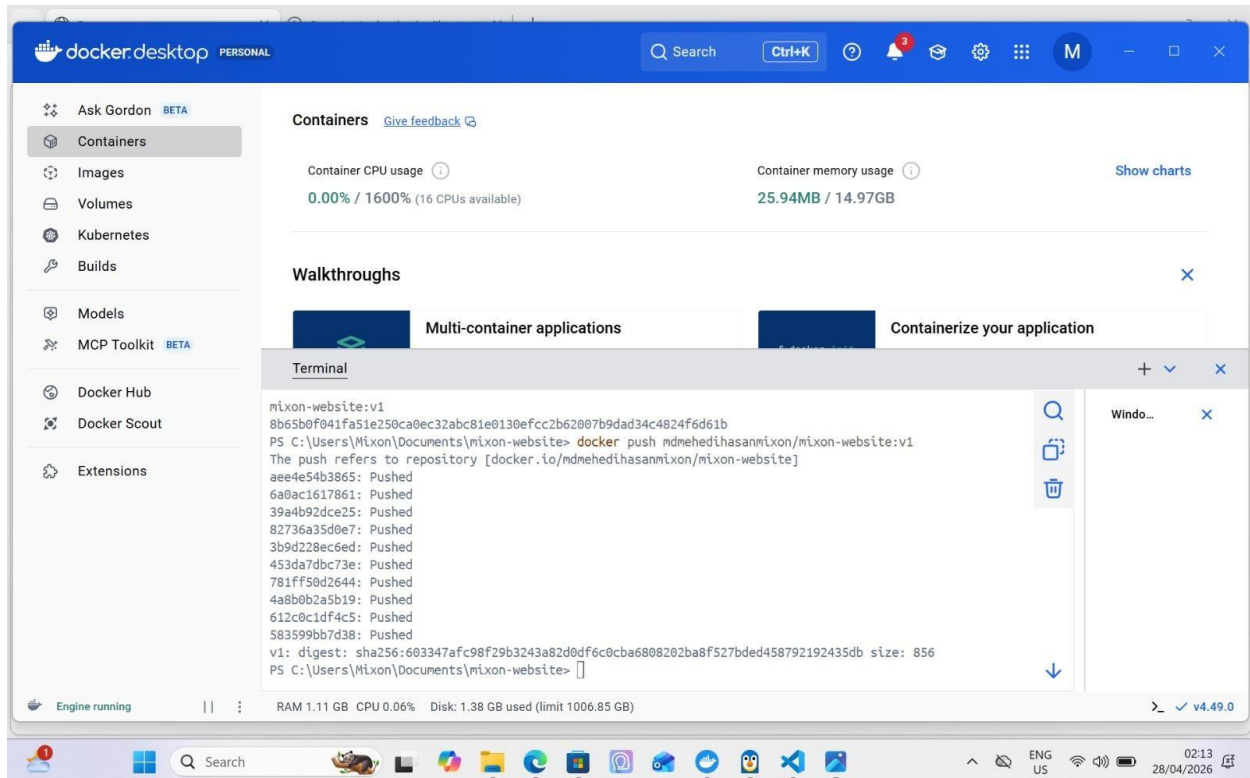


Fig.5 - Docker Image Pushing to DockerHub

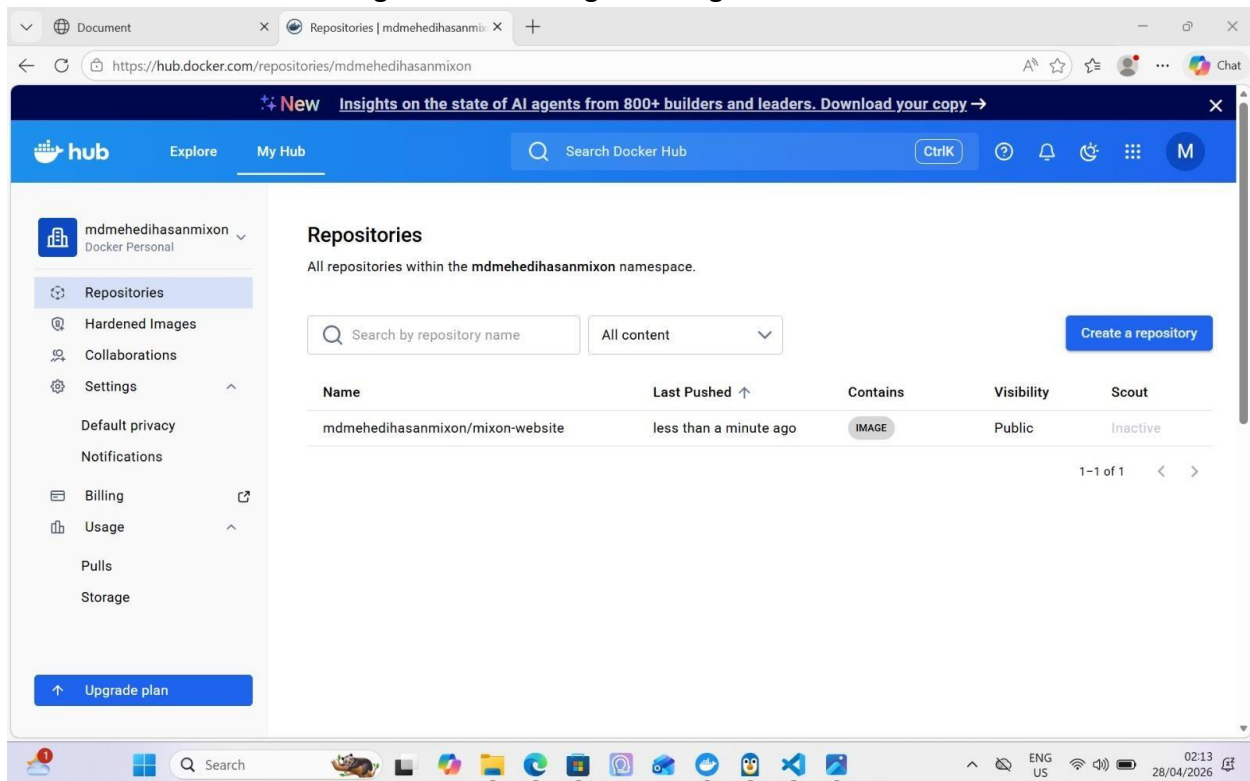
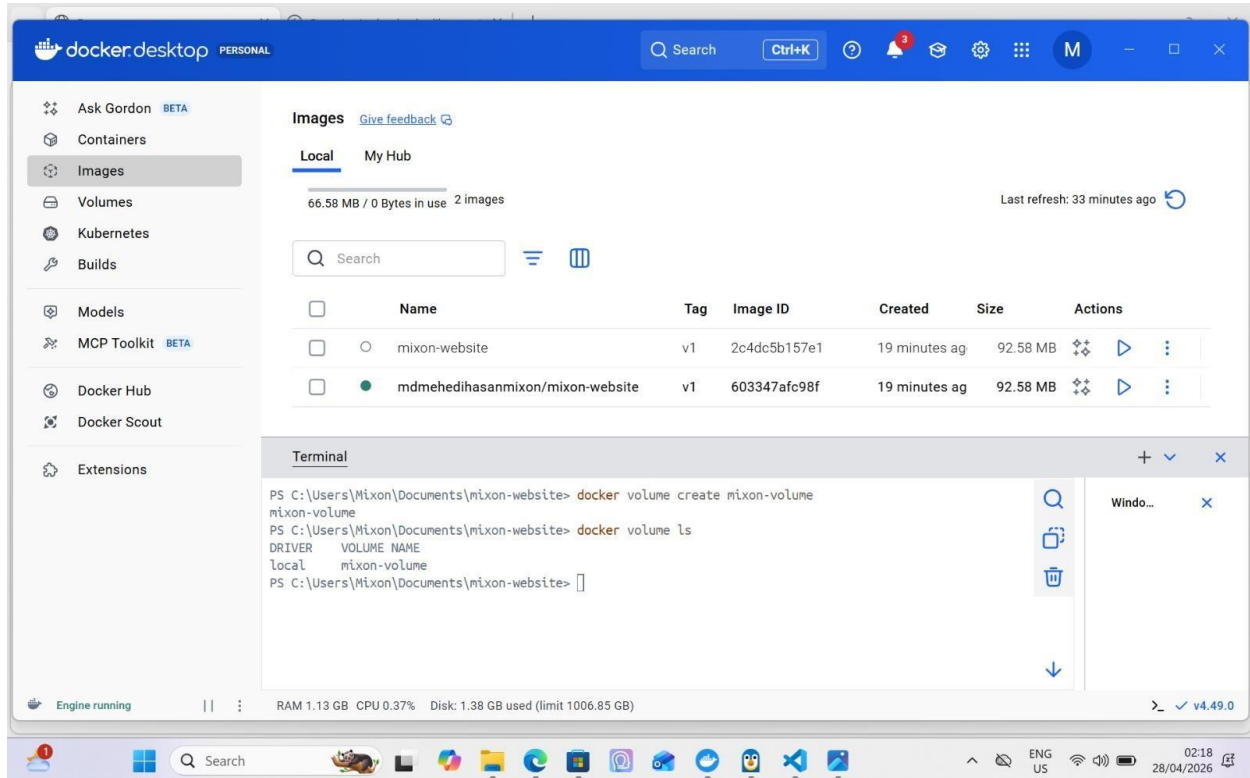


Fig.6 - Docker Image Successfully Published on Docker Hub

3.2 Docker Volumes

Because containers are meant to be transient, any data they contain could be lost if they are stopped or removed. Docker offers volumes, which enable data to remain outside of the container's lifespan, to overcome this restriction. Volumes are especially helpful for applications like databases and logging systems that need to store data for an extended period of time.

Volumes guarantee that data is accessible even if the container is regenerated by separating it from the container. Because data may be exchanged between several containers as necessary, this method not only increases flexibility but also reliability.



The screenshot displays the Docker Desktop application window. The left sidebar contains navigation options: Ask Gordon (BETA), Containers, Images (selected), Volumes, Kubernetes, Builds, Models, MCP Toolkit (BETA), Docker Hub, Docker Scout, and Extensions. The main panel shows the 'Images' section with tabs for 'Local' and 'My Hub'. Under 'Local', it indicates '66.58 MB / 0 Bytes in use' and '2 Images'. A table lists the following images:

Name	Tag	Image ID	Created	Size	Actions
mixon-website	v1	2c4dc5b157e1	19 minutes ag	92.58 MB	[Icons]
mdmehedihasanmixon/mixon-website	v1	603347afc98f	19 minutes ag	92.58 MB	[Icons]

Below the images list is a 'Terminal' window with the following commands and output:

```
PS C:\Users\Mixon\Documents\mixon-website> docker volume create mixon-volume
mixon-volume
PS C:\Users\Mixon\Documents\mixon-website> docker volume ls
DRIVER      VOLUME NAME
local       mixon-volume
PS C:\Users\Mixon\Documents\mixon-website>
```

The bottom status bar shows 'Engine running', 'RAM 1.13 GB', 'CPU 0.37%', 'Disk: 1.38 GB used (limit 1006.85 GB)', and 'v4.49.0'. The Windows taskbar at the bottom includes the Start button, search bar, and various application icons.

Fig.7 - Setting up and examining the Docker volume

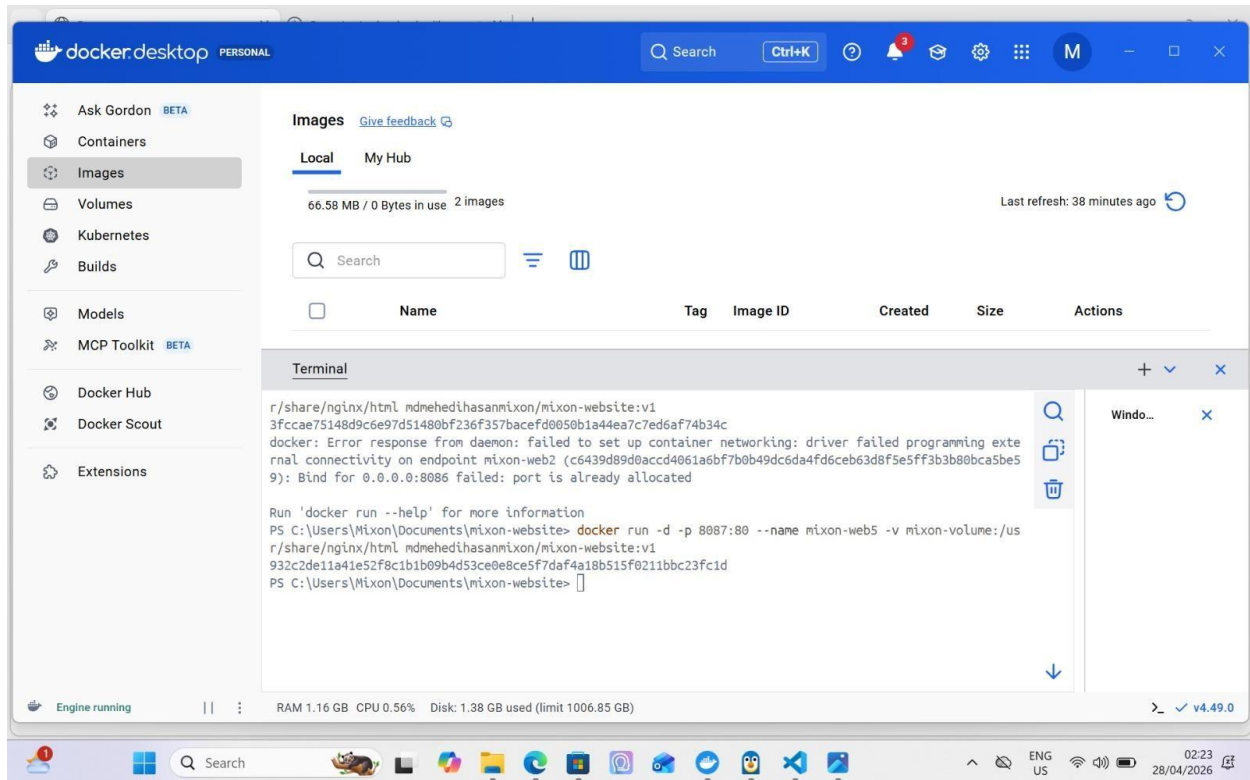


Fig.8 - Attaching the Volume to the Container

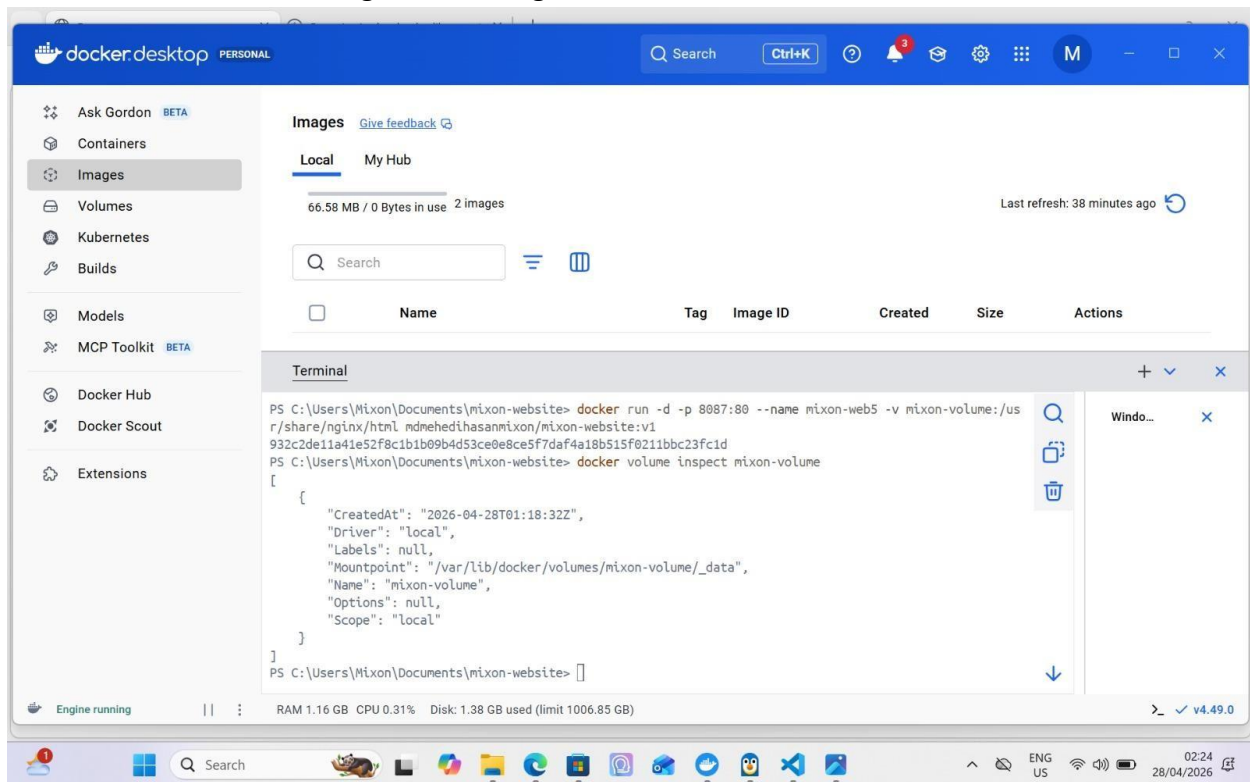


Fig.9 - After mounting to the container, checking the volume

3.3 Docker Networks

Because networking allows containers to communicate with external services, it is a crucial part of container-based systems. From straightforward local connectivity to intricate distributed systems, Docker offers a variety of networking solutions to serve a range of use cases. For instance, overlay networks facilitate communication among several servers in a cluster, whereas bridge networks allow containers to connect within a single host. Containers can be given distinct identities inside a network via more complex setups, increasing control and flexibility. In microservices architectures, where several services must work together smoothly to deliver an entire application, effective networking is especially crucial.

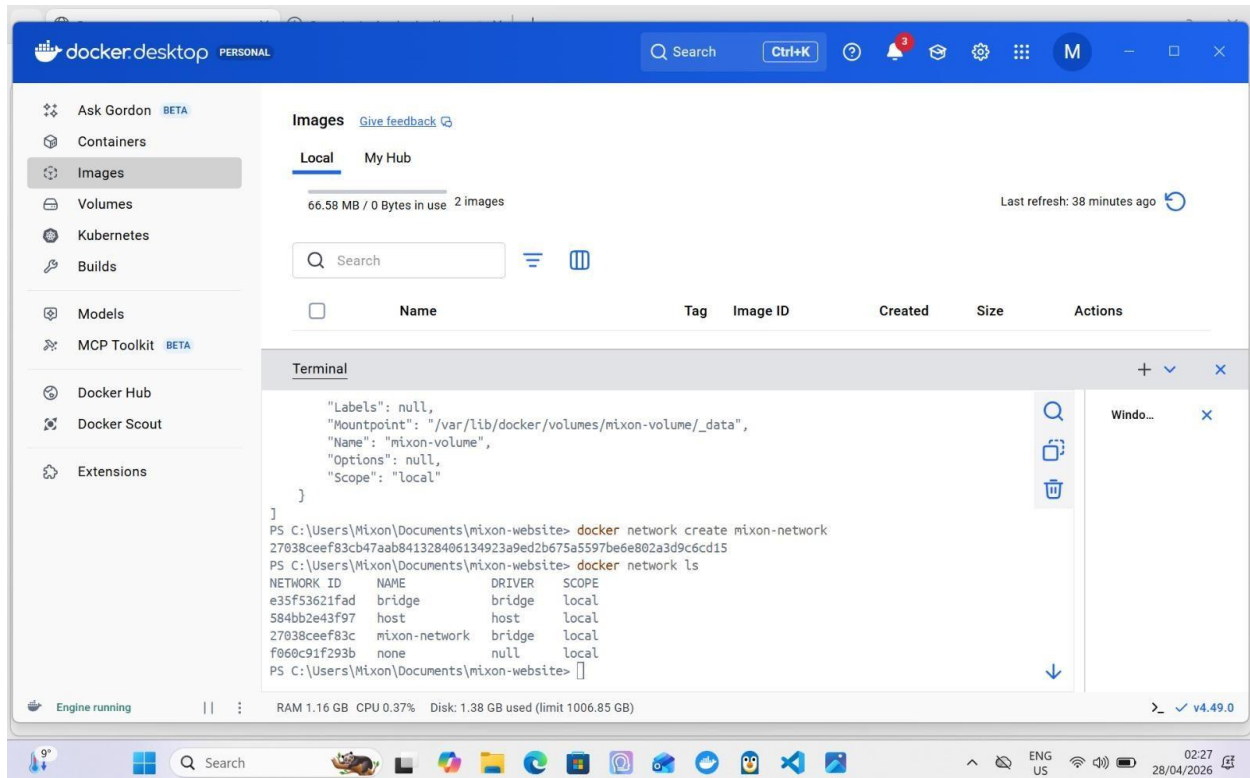


Fig.10 - Creating and Verifying Network Creation

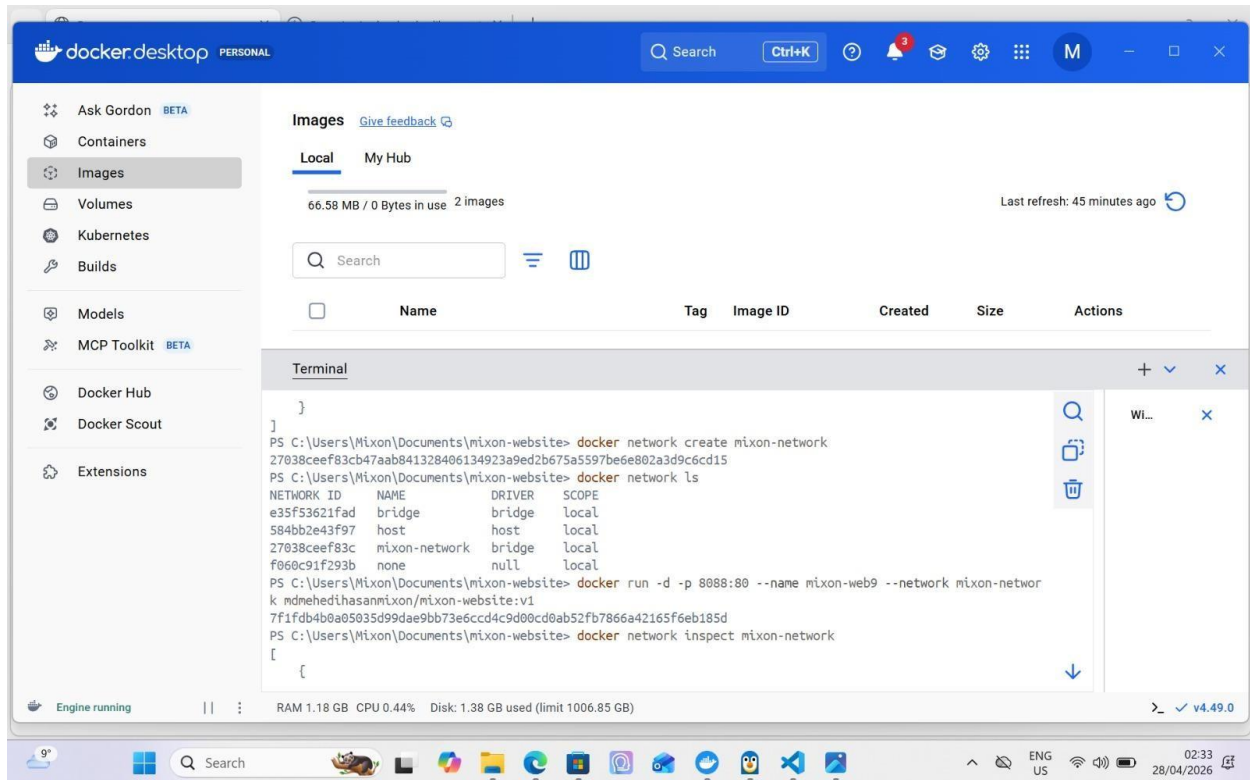


Fig.11- Assigning Network to the Container

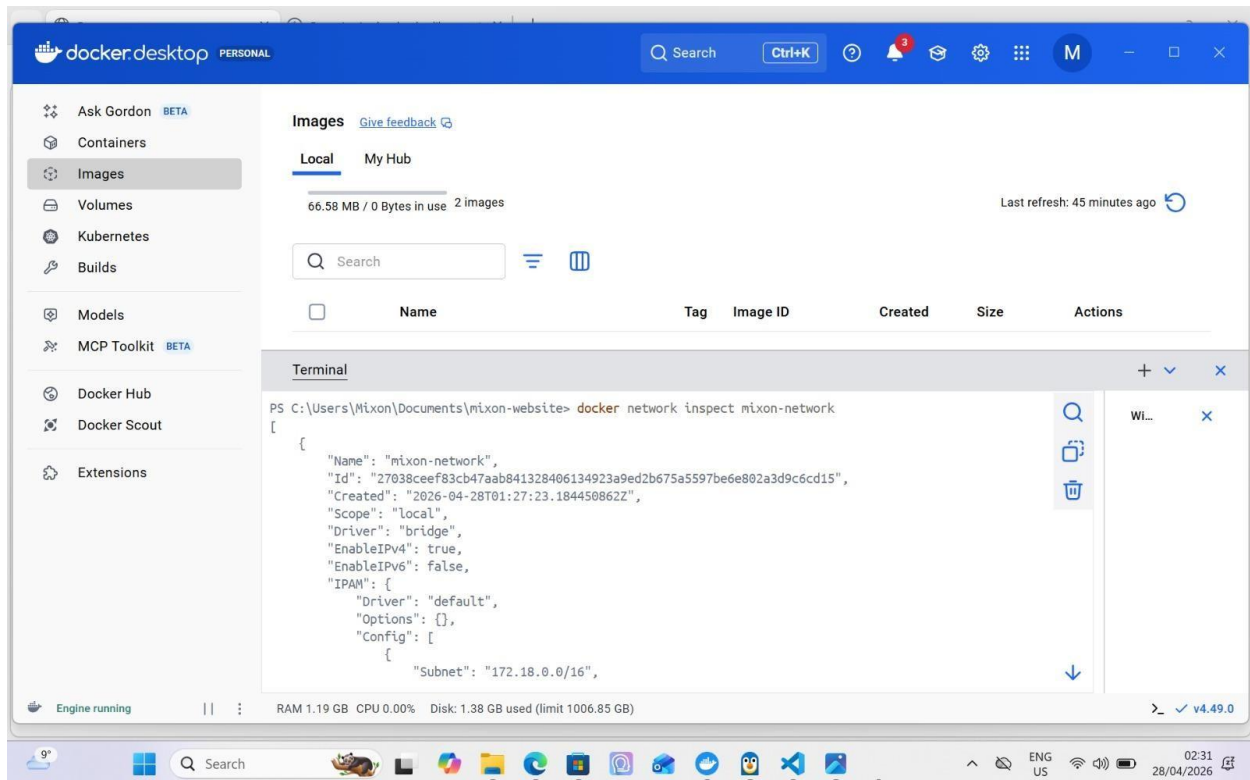


Fig. 12 - Inspecting The Network Configuration

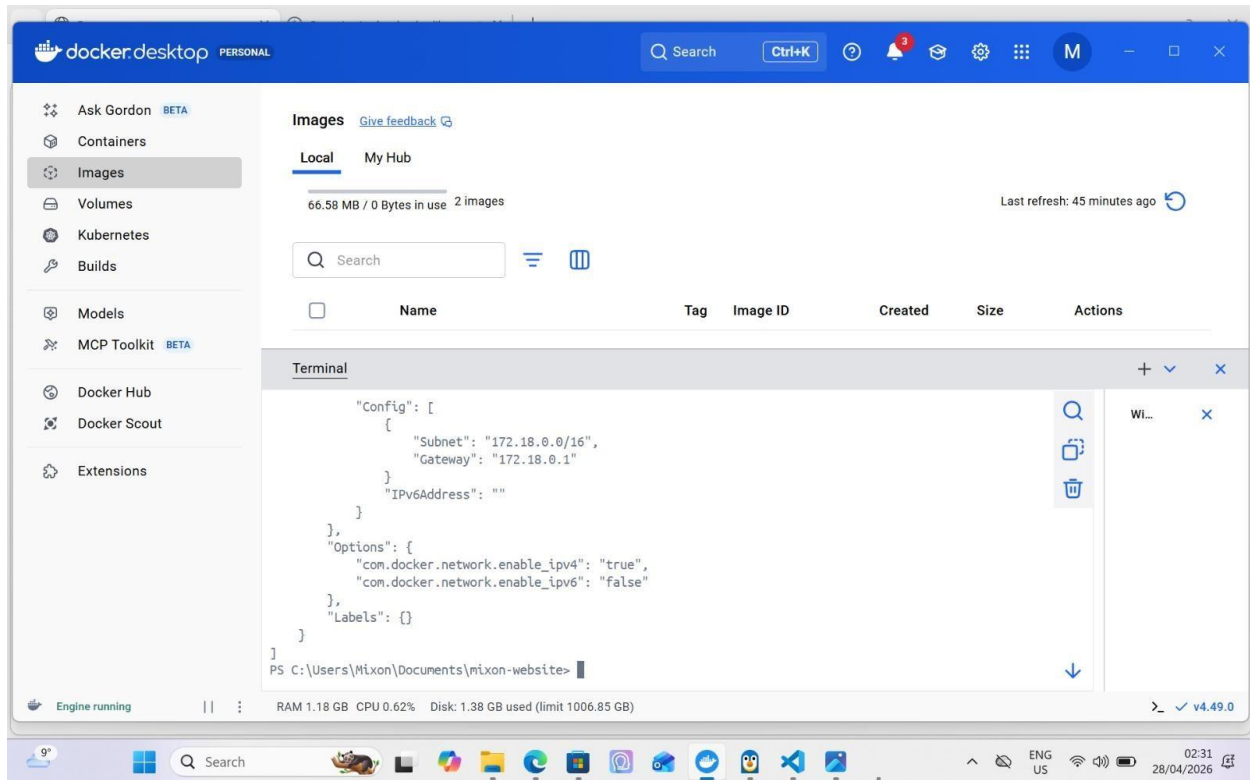


Fig.13 - confirming the IPv4 address assignment for the container

YT-Link: [Docker Demonstration | Md Mehedi Hasan Mixon](#)

4. Cloud Migration and Architectural Concepts in the Context of Khan Academy

Cloud computing has become a fundamental enabler for organisations that need to deliver services at scale, especially those operating in a global and highly dynamic environment. In the case of educational platforms such as Khan Academy, the ability to serve millions of users with varying access patterns requires a robust, flexible, and scalable infrastructure. Traditional on-premise systems would struggle to handle such variability efficiently, which is why cloud migration becomes a strategic necessity rather than just a technological upgrade. At its core, cloud migration involves moving applications, data, and services from local infrastructure to cloud-based platforms such as Microsoft Azure or Amazon Web Services. This transition is not simply a matter of relocation; it requires redesigning system architecture to take full advantage of cloud-native capabilities. These include on-demand resource provisioning, distributed computing, and automated management features that significantly reduce operational overhead.

The capacity to manage erratic and geographically dispersed user demand is one of the most important features of cloud architecture in this situation. User engagement is not consistent for a platform such as Khan Academy. It varies according to time zones, academic timetables, and outside events like exam periods or worldwide disturbances. An infrastructure that can dynamically adapt itself without human involvement becomes necessary as a result. Load balancing plays a central role in achieving this. It ensures that incoming user requests are distributed evenly across multiple servers or instances, preventing any single component from becoming a bottleneck. By intelligently routing traffic, load balancers help maintain consistent response times and reduce the risk of system failure. In cloud environments, this process is often automated and continuously optimised based on real-time performance metrics. This is closely related to the idea of scalability, especially elasticity. Systems with scalability can adjust their processing resources in response to demand. For instance, more virtual machines or container instances can be automatically launched to manage increasing traffic during peak usage hours. These excess resources can be eliminated as demand declines, guaranteeing cost effectiveness. One of the key benefits of cloud computing is its capacity to scale dynamically, which is crucial for systems with erratic consumption patterns.

Another key component is the application gateway, which operates at a higher level of the network stack. Unlike basic load balancers, application gateways can inspect incoming requests at the application layer (Layer 7), allowing for more advanced routing decisions.

They can direct traffic based on URL paths, manage SSL termination, and enforce security policies such as web application firewalls. This not only improves performance but also strengthens the security posture of the system by filtering malicious traffic before it reaches backend services. When these components are integrated effectively, they create a resilient and adaptive architecture. In the context of Khan Academy, this means users from different parts of the world

can access learning resources with minimal delay, even during periods of high demand. The system can automatically respond to fluctuations in traffic, maintain high availability, and ensure a secure environment for users.

Ultimately, cloud migration is not just about moving systems to a new environment; it is about transforming how those systems operate. By leveraging load balancing, scalability, and intelligent traffic management, organisations can build architectures that are not only efficient but also future-ready.

5. The Complementary Relationship Between Load Balancing and Scaling

Although load balancing and scaling are frequently seen as distinct methods in cloud-based systems, they are intricately linked and must work together to guarantee optimal system performance. Designing systems that can successfully manage workloads in the real world requires an understanding of their connection.

The main goal of load balancing is to divide incoming traffic among several computing resources. Its primary goal is to prevent any one server from becoming overloaded, which contributes to the stability and responsiveness of the system. Load balancing lowers latency and enhances user experience by distributing requests equally. Nevertheless, load balancing itself does not boost the system's overall capacity. It merely maximizes the usage of available resources. Conversely, system capacity is directly addressed by scaling. To meet the present demand, it entails modifying the quantity of resources that are accessible. This can be accomplished either horizontally by expanding the system's machine count or vertically by boosting the power of already-existing machines. Horizontal scaling is increasingly widely used in contemporary cloud systems due to its increased flexibility and fault tolerance.

The limitation of using load balancing without scaling becomes evident when demand exceeds the system's total capacity. Even if traffic is distributed perfectly, the system will still struggle if there are not enough resources to handle the load. This can lead to increased response times, service degradation, or even complete system failure. Similarly, scaling without proper load balancing can result in inefficient resource utilisation. For example, new instances may be added to the system, but if incoming traffic is not distributed correctly, some servers may remain underutilized while others are overloaded. This not only wastes resources but also fails to improve performance effectively.

The true strength of cloud architecture lies in combining these two mechanisms. When scaling is integrated with load balancing, the system can both expand its capacity and distribute workloads efficiently. As demand increases, new resources are automatically added, and the load balancer ensures that traffic is evenly distributed across all available instances. When demand decreases, resources can be scaled down, maintaining cost efficiency without compromising performance. This relationship becomes particularly important in scenarios involving unpredictable workloads. For example, during a sudden surge in users, such as during an online learning campaign or exam preparation period, the system must respond quickly to maintain service quality. Load balancing ensures that incoming requests are handled smoothly, while scaling provides the additional capacity required to sustain the increased demand.

In large-scale platforms like Khan Academy, this integration is essential for delivering a consistent user experience. The system must be capable of handling millions of simultaneous requests while

maintaining low latency and high availability. By combining load balancing with dynamic scaling, cloud platforms can achieve this balance effectively. In conclusion, load balancing and scaling should not be viewed as independent solutions but as complementary components of a unified system. Their combined use enables cloud infrastructures to be both efficient and resilient, ensuring that they can adapt to changing demands while maintaining performance and reliability.

6. Customer Value in Integrated Cloud Architecture

The value that cloud computing provides to end users is a better indicator of its genuine impact than its technological efficiency alone. A system that directly improves user experience in ways that are frequently imperceptible but extremely important is created when several cloud components, such as load balancing, scalability mechanisms, and application gateways, are integrated into a single design. Enhanced performance is one of the most direct advantages. When servers are overcrowded or have inadequate resources to fulfill incoming requests, customers frequently encounter delays in traditional systems. Cloud-based systems, on the other hand, intelligently divide workloads among several resources. While autoscaling technologies automatically expand capacity when necessary, load balancers make sure that no single server is

overloaded. Users benefit from quicker reaction times and more seamless interactions as a result, even during moments of high demand.

Another important factor in consumer value is reliability. Failures in large-scale systems are unavoidable, whether they result from network disruptions, software defects, or hardware problems. The capacity of contemporary cloud systems to function in spite of such setbacks is what sets them apart. The architecture incorporates redundancy, which allows other components to take over in the event of a component failure without disrupting the service. This dependability is crucial for a platform like Khan Academy since users always rely on continuous access to learning materials and places. Also, security is a major factor in determining user trust. Before harmful requests reach backend systems, application gateways and integrated security layers examine incoming traffic and remove them. Web application firewalls, SSL termination, and identity-based access control are examples of features that assist safeguard private information and preserve system integrity. Although these methods work quietly in the background from the user's point of view, they are crucial for maintaining a secure and reliable environment. The uniformity of experience across many places is another crucial factor. Applications can serve customers from geographically closer areas thanks to cloud providers' worldwide data center operations. This lowers latency and guarantees consistent performance no matter where the user accesses the service. This is a significant benefit over conventional centralized systems for globally distributed platforms.

Beyond performance and reliability, cloud architecture also simplifies complexity for end users. One of the defining characteristics of cloud computing is abstraction. Users do not need to understand how resources are managed or how traffic is routed; they simply interact with a system that appears seamless and responsive. This hidden complexity is managed entirely by the underlying architecture, which continuously adapts to maintain optimal performance.

In essence, integrated cloud architecture transforms technical capabilities into tangible user benefits. It ensures that services are fast, reliable, secure, and consistently available, which ultimately enhances user satisfaction and engagement.

7. Key Drivers Behind Cloud Adoption: Cost, Flexibility and Strategic Benefits

Cloud computing's quick uptake across industries is not coincidental; rather, it is a highly appealing solution for contemporary businesses due to a confluence of technical, strategic, and economic considerations. Gaining an understanding of these factors helps explain why cloud technologies are now the norm for deploying systems on a wide scale. Cost effectiveness is one of the most important criteria. Conventional IT infrastructure necessitates a large initial outlay for gear, upkeep, and physical space. Businesses must forecast future demand and allocate resources appropriately, which frequently results in overprovisioning or underutilization. This model is changed by cloud computing to an operational expense approach, in which resources are used as needed and billed appropriately. In addition to lowering financial risk, this pay-as-you-go strategy enables businesses to directly match spending to actual consumption [10].

Flexibility is another critical driver. Cloud environments enable rapid deployment and modification of applications, allowing organisations to respond quickly to changing requirements. In traditional systems, introducing new features or scaling infrastructure often involves lengthy procurement and setup processes. In contrast, cloud platforms provide immediate access to resources, enabling developers to experiment, iterate, and deploy solutions at a much faster pace. This agility is particularly valuable in competitive environments where time-to-market is a key factor. Scalability further enhances this flexibility. As user demand fluctuates, cloud systems can automatically adjust resource allocation without manual intervention. This ensures that applications remain responsive during periods of high demand while avoiding unnecessary costs during low usage periods. For organisations serving global audiences, this capability is essential for maintaining consistent performance. Another important driver is global accessibility. Cloud

providers maintain data centers in multiple geographic regions, allowing applications to be deployed closer to end users. This reduces latency and improves overall performance, especially services that rely on real-time interaction. For platforms like Khan Academy, which serve users across different countries, this global infrastructure ensures equitable access to resources regardless of location.

Cloud adoption is significantly influenced by automation as well. Infrastructure management, scaling, and monitoring technologies are often integrated into cloud services. These tools minimize the possibility of human error and lessen the necessity for manual intervention. Without interfering with the service, automated systems can identify performance problems, modify resources, and even implement security updates. Once thought to be a deterrent to cloud adoption, security is now one of its advantages. Cloud providers make significant investments in cutting-edge security measures like identity management, encryption, and ongoing surveillance. The level of security provided by major cloud platforms frequently surpasses what individual organizations can accomplish on their own, even though no system is totally impervious to threats [11]. Finally, cloud computing supports long-term strategic growth. It enables organisations to innovate more freely, expand into new markets, and adapt to technological changes without being constrained by physical infrastructure. This strategic advantage makes cloud adoption not just a technical decision, but a business-driven one.

8. Serverless Computing and Its Contrast with Containerisation

New architectural models have surfaced to further streamline application deployment and maintenance as cloud technologies continue to advance. Among these, serverless computing signifies a substantial change in the development and implementation of programs. However, containerization is still a popular method for creating portable and scalable systems. Both models are part of the cloud ecosystem, but they serve different purposes and have unique benefits. Despite its name, serverless computing does not completely do away with servers. Rather, it separates developers from server management. Applications are run in this approach in reaction to particular events, like user input or system triggers. Infrastructure management, scaling, and resource allocation are all handled automatically by the cloud provider. This frees developers from worrying about server provisioning and maintenance so they can concentrate just on writing code. The cost effectiveness of serverless computing is one of its main benefits. There is no expense for idle time because resources are only used when functions are carried out. Because of this, serverless computing is especially well suited for applications that have erratic or sporadic workloads, like background operations, notifications, or data processing chores [12]. Furthermore, serverless systems could grow automatically at a very fine-grained level, reacting to variations in demand nearly quickly. However, this abstraction comes with certain limitations. Developers have less control over the execution environment, which can make it difficult to optimise performance for specific use cases. There are also constraints related to execution time, resource limits, and dependency management. These factors can make serverless less suitable for long-running or complex applications. Containerisation offers a different approach. By packaging applications along with their dependencies into containers, developers gain greater control over the runtime environment. Containers can run consistently across different platforms, making them ideal for applications that require portability and predictable behaviour. Unlike serverless functions, containers can support long-running processes and more complex system architectures.

In terms of scalability, both models offer strong capabilities, but they operate differently. Serverless systems scale automatically based on individual events, while container-based systems typically rely on orchestration tools such as Kubernetes to manage scaling at the application level. This provides more flexibility but also introduces additional complexity.

From a cost perspective, the choice between serverless and containerisation depends on the nature of the workload. Serverless is generally more cost-effective for short-lived or infrequent tasks, while containers may provide better efficiency for continuously running applications. In practice, many organisations adopt a hybrid approach, using serverless for specific functions and containers for core services. Ultimately, serverless computing and containerisation should not be seen as competing technologies, but as complementary solutions within the broader cloud

ecosystem. By selecting the appropriate model for each use case, organisations can achieve a balance between efficiency, flexibility, and control.

9. Conclusion

The core technologies supporting contemporary cloud-based systems have been examined in this research, with an emphasis on virtualization, containerization, and cloud computing as interrelated building blocks rather than discrete ideas. This conversation makes it evident that

computing has evolved from inflexible, hardware-dependent settings to adaptable, service-driven architectures that put user experience, scalability, and efficiency first. This change began with virtualization, which allows several systems to run on a single physical machine while preserving isolation and control. Its limits in terms of overhead and scalability underscore the need for more lightweight alternatives in highly dynamic systems, even though it is still a potent tool for resource management and security. Many of these issues are resolved by containerization, which provides a more effective and portable method of application deployment. Containers facilitate quicker deployment, more efficient use of resources, and enhanced uniformity between development and production environments by lowering reliance on complete operating systems and emphasizing application-level packaging. Nevertheless, there are trade-offs associated with this efficiency, especially regarding shared kernel hazards and a greater dependence on appropriate setup and security procedures.

The discussion then moved beyond individual technologies to examine how they operate within cloud environments. Using the example of large-scale platforms such as Khan Academy, the report highlighted how cloud architecture enables systems to handle unpredictable demand while maintaining performance and reliability. Key mechanisms such as load balancing, scalability, and application gateways were shown to play a central role in achieving this, working together to ensure that services remain accessible and responsive even under heavy workloads.

A critical insight from this analysis is that no single technology is sufficient on its own. Instead, the effectiveness of modern systems depends on how these components are integrated. The relationship between load balancing and scaling, for instance, demonstrates that efficient resource distribution must be combined with adequate system capacity to achieve true elasticity. Similarly, the comparison between containerisation and serverless computing shows that different architectural approaches are suited to different types of workloads, and that hybrid solutions often provide the best balance between control, efficiency, and cost. From a broader perspective, cloud computing represents not just a technological advancement, but a shift in how organisations approach system design and service delivery. The ability to access resources on demand, scale dynamically, and deploy applications globally has transformed both operational strategies and user expectations. End users now expect services to be fast, reliable, and continuously available, and cloud-based architectures are specifically designed to meet these expectations.

In conclusion, the integration of virtualisation, containerisation, and cloud-native technologies forms the foundation of modern digital systems. While each technology has its own strengths and limitations, their combined use enables organisations to build architectures that are resilient, scalable, and adaptable to future challenges. As cloud technologies continue to evolve, the ability to understand and strategically apply these concepts will remain essential for designing efficient and sustainable computing solutions.

10. References

- [1] P. Mell and T. Grance, “The NIST definition of cloud computing,” National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, Special Publication 800-145, 2011.
- [2] R. Buyya, J. Broberg, and A. M. Goscinski, Cloud Computing: Principles and Paradigms. Hoboken, NJ, USA: Wiley, 2010.
- [3] D. Merkel, “Docker: Lightweight Linux containers for consistent development and deployment,” Linux Journal, no. 239, 2014.
- [4] J. E. Smith and R. Nair, “The architecture of virtual machines,” IEEE Computer, vol. 38, no. 5, pp. 32–38, May 2005.

- [5] A. Gulati, A. Holler, M. Ji, G. Shanmuganathan, C. Waldspurger, and X. Zhu, "VMware distributed resource management: Design, implementation and lessons learned," *VMware Technical Journal*, vol. 1, no. 1, pp. 45–64, 2012.
- [6] Y. Yarom and K. Falkner, "Flush+Reload: A high resolution, low noise, L3 cache sidechannel attack," in *Proc. 23rd USENIX Security Symp.*, San Diego, CA, USA, 2014, pp. 719– 732.
- [7] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," in *Proc. IEEE Int. Conf. Cloud Eng. (IC2E)*, Tempe, AZ, USA, 2015, pp. 386–393.
- [8] S. Zhang, X. Chen, and J. Wu, "Security analysis of Docker containers," *IEEE Cloud Computing*, vol. 5, no. 3, pp. 48–56, May–Jun. 2018.
- [9] Microsoft Azure, "Application Gateway documentation," Microsoft, 2024.
- [10] G. Reese, *Cloud Application Architectures: Building Applications and Infrastructure in the Cloud*. Sebastopol, CA, USA: O'Reilly Media, 2009.
- [11] C. Pahl, "Containerization and the PaaS cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May–Jun. 2015.
- [12] Amazon Web Services, "AWS Lambda Developer Guide," AWS, 2024.
- [13] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in *Proc. 10th Computing Colombian Conf. (CCC)*, 2015, pp. 1–9.
- [14] N. Krishnan, "Load balancing techniques in cloud computing: A survey," *IEEE Systems Journal*, vol. 10, no. 2, pp. 1–10, 2016.
- [15] L. M. Vaquero, L. Roderio-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, Jan. 2009.