

Cloud Engineering Principles and Practice



Assignment Report

Student Name: Abdullah Al Maruf

Student ID: 2526089

Module Code: ACCA7027

Programme: MSc Cybersecurity and Digital Forensics

Lecturer: Renuga Jayakumar

Submission Date: 28th/04/2026

Table of Content

1. Abstract	03
2. Introduction	03
3. Principles of Virtualisation	04
2.1 Virtualisation Architecture	04
2.2 Advantages	05
2.3 Limitations	05
4. Principles and Benefits of Containerisation	05
3.1 Container Architecture	05
3.2 Benefits of Containerisation	06
3.3 Limitation	06
5. Docker, Docker Image and Dockerfile Construction	06
4.1 Docker Image	06
4.2 Dockerfile Example	07
4.3 Build Image	07
6. Images and Publishing Images	07
5.1 Image Workflow	07
5.2 Running a Container	08
5.3 Publishing to DockerHub	08
7. Container Volumes	09
6.1 Creating a Volume	09
6.2 Using a Volume	10
6.3 Benefits	10
8. Container Networking	10
7.1 Create Network	10
7.2 Run Container in Network	11
9. Practical Example: Full Docker Workflow	11
10. Cloud Provider Approach to Load Balancing, Scalability and Application Gateways....	14
9.1 Load Balancing	14
9.2 Scalability	14
9.3 Application Gateways	14
11. Why Load Balancing and Scaling Are Complementary ...	14
12. Benefits of Combining Load Balancing, Scaling, and Application Gateways	15
13. Cloud Growth and Serverless Analysis	15
12.1 Driving Forces behind Cloud Computing Expansion ...	15
12.2 Cost Efficiency	15
12.3 Flexibility and Agility	15
12.4 Scalability and Elasticity	15
12.5 Global Accessibility and Remote Access	16
12.6 Reliability and High Availability	16
12.7 Automation and Integration	16
12.8 Innovation and Competitive Advantage	16
14. Serverless Computing (Functions as a Service)	16
15. Comparison Serverless and Containerisation	17
14.1 Serverless	17
14.2 Containerisation	17
14.3 Critical Comparison in Context of Cloud Growth ...	17
16. Conclusion	17
17. References	18
18. Assessment Video link	18

Abstract

Cloud computing has radically changed how applications are made today by making it possible to access computer resources over the internet in a way that is scalable, flexible, and cost-effective. This research looks at the main technologies that make cloud computing possible, starting with virtualisation and containerisation and going all the way to utilising Docker to put them into practice. Virtualisation is looked at as a way to use hypervisors to make physical hardware less real and support numerous virtual machines. Containerisation is looked at as a lighter option that lets applications run on their own utilising a common operating system kernel.

The paper gives a thorough look at Docker as a top containerisation platform. It talks about important parts including building Dockerfiles, making images, deploying containers, publishing images, container volumes, and networking. These parts are necessary for making apps that are portable, work the same way every time, and are fast. To show the whole container lifecycle, from developing an image to operating and sharing containers, there is a realistic example of how to use Docker to set up a web server.

The paper also looks at advanced cloud computing features like load balancing, scalability, and application gateways that companies like Amazon Web Services and Microsoft Azure use. It also looks into the main reasons why people choose cloud computing, such as lower costs, more flexibility, global access, and high availability. Also, serverless computing models like AWS Lambda and Azure Functions are looked at and contrasted to containerisation in terms of performance, cost, and how hard they are to use.

The research shows how virtualisation, containerisation, Docker technologies, and cloud computing all work together to make contemporary, cloud-native apps that can handle the growing need for reliable and scalable computing solutions.

1. Introduction

In the past, applications were installed directly on physical computers in traditional computing environments. This sometimes led to conflicts between dependencies, inefficient use of resources, limited scalability, and complicated maintenance. These problems make it hard for businesses to manage applications well and keep up with evolving user needs [1].

To get over these problems, virtualisation was added as a basic technology. A hypervisor makes it possible for numerous virtual machines to run on one real system. Each virtual machine runs its own operating system and has its own resources, which makes better use of hardware and provides strong isolation. But virtual machines need a lot of resources because each one has its own operating system, which means more overhead and longer startup times [2].

Containerisation became a better solution. It lets apps function in lightweight, separate environments while sharing the host operating system kernel. This method uses fewer resources and makes things run faster. Docker and other platforms have made containerisation much easier by letting developers put apps and their dependencies into portable containers that can operate the same way in multiple settings [3].

Docker adds a number of important parts that make it easier to deploy modern applications. A Dockerfile tells the computer what to do to make an application image. Docker images are reusable templates that have application code and dependencies on them. Containers are instances of these images that run. Docker also offers volumes, which let you store data outside of the container lifecycle so that it doesn't be lost when the containers are removed. Docker networking also lets containers talk to each other and to outside systems, which is important for running complicated, multi-container apps [4].

Cloud computing has become the most popular way to provide IT services via the internet, building on virtualisation and containerisation. Cloud platforms use these technologies to offer on-demand resources that can be scaled up or down, are flexible, and can be accessed from anywhere in the world.

Top companies like Amazon Web Services and Microsoft Azure offer services like load balancing, auto-scaling, and application gateways that improve system performance, availability, and security. Cloud computing also works with new ways of building things, including microservices and serverless computing. Serverless platforms like AWS Lambda and Azure Functions let developers run code without having to worry about infrastructure. This makes things easier to run and lets developers design applications that respond to events [5].

This report's goal is to look at the ideas behind virtualisation and containerisation, look at Docker's role and its most important parts, like images, volumes, and networking, and talk about how important cloud computing is in today's IT environments. The study shows how these technologies work together to help build efficient, scalable, and cloud-native apps by mixing theoretical ideas with real-world examples.

2. Principles of Virtualisation

Virtualisation is the process of creating virtual versions of physical resources such as servers, storage, and networks. It is achieved using a hypervisor, which manages virtual machines[5].

Hardware virtualization refers to the capability of computer processing chips to run multiple virtual machines simultaneously, enhancing a virtualization platform's ability to switch between different virtual machines. Examples of hardware virtualization technologies include Intel's Virtualization Technology (VT) and AMD Virtualization (AMD-V) [6].

Computer processing chips that offer capabilities to run multiple virtual machines simultaneously are considered a form of hardware virtualization. These capabilities improve a virtualization platform's ability to switch from running one virtual machine to another. Examples of hardware virtualization technologies include Intel's Virtualization Technology (VT) and AMD Virtualization (AMD-V). Some virtualization platforms such as Microsoft Hyper-V require virtualization extensions in order to run. [7].

2.1 Virtualisation Architecture

Physical hardware, a hypervisor layer, and several virtual machines make up the structure of virtualisation architecture. The hypervisor is in charge of managing and assigning system resources including CPU, memory, and storage. It lies between the hardware and the virtual machines. The hypervisor runs each virtual machine separately. Each one has its own operating system, application software, and virtualised hardware resources. This layered method lets more than one system run at the same time on one physical computer [8].

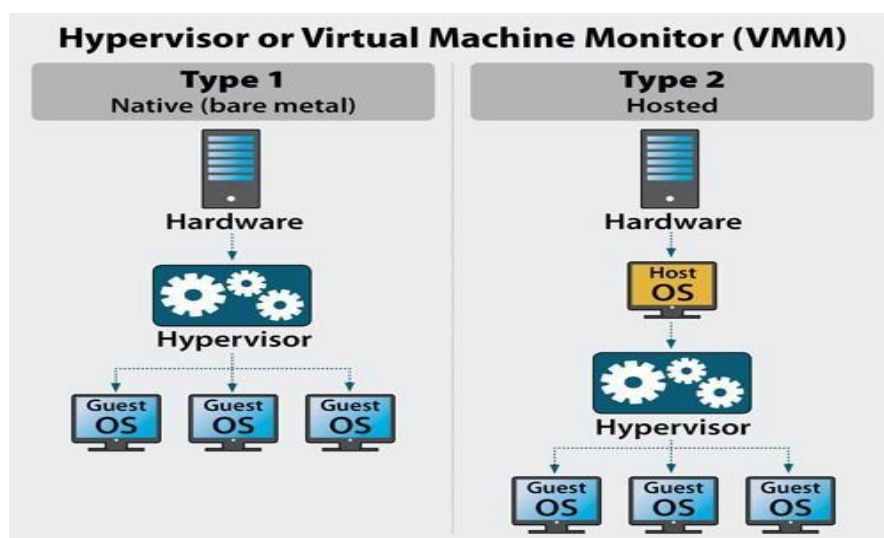


Fig. 1: Virtualisation Architecture

2.2 Advantages

Virtualisation has a number of important benefits in today's computing settings. It makes better use of hardware by letting several virtual machines share a single physical system in a way that works well. It also makes it easier for apps to stay separate from each other, which lowers the chance of system conflicts. It also gives you more options by letting different operating systems and applications run on the same hardware, which meets the needs of different organisations.[9]

2.3 Limitations

Virtualisation has certain drawbacks, even though it offers some benefits. Because each virtual machine needs a whole operating system, they utilise more memory and storage space, which means they demand more system resources. They also take longer to start up than lighter options like containers. Also, managing more than one virtual machine can make the system more complicated, which makes it harder to keep up with and fix [10].

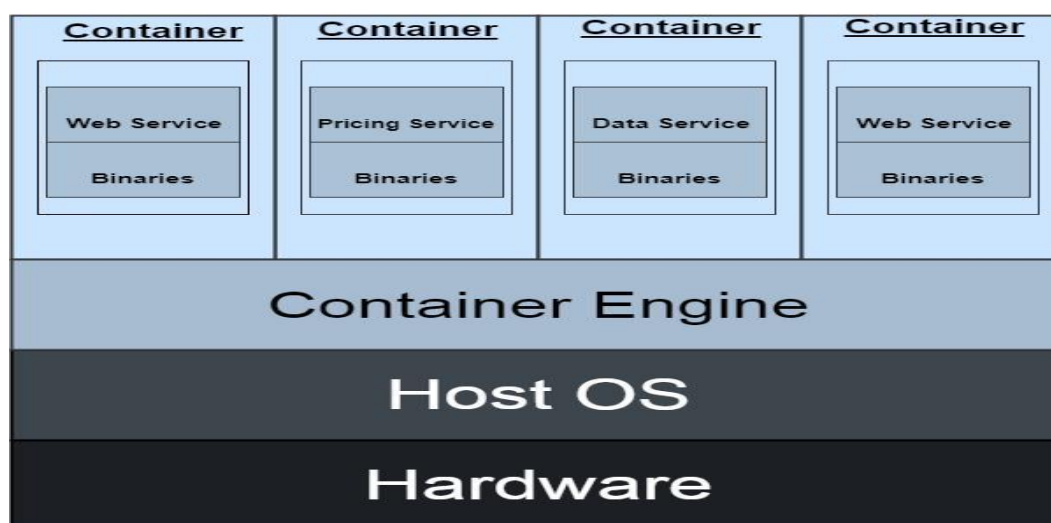
3. Principles and Benefits of Containerisation

Containerisation is a type of virtualisation that works at the operating system level and doesn't take up much space. It lets programs run in separate spaces called containers while still using the same host operating system kernel. Each container wraps up an application with all of its dependencies, libraries, binaries, and configuration files into one unit. This makes sure that the application runs the same way in all environments. This method makes it easier to move things around and makes it less likely that there will be problems with interoperability between development and production systems [11], [12].

There are a few important parts that make up the containerisation ecosystem. Containerd and CRI-O are two examples of container runtimes that run containers. Container images are unchangeable templates that are used to make containers, which makes sure that deployments are always the same. These photos are stored and shared through registries, which makes it easy to share and reuse them. Orchestration technologies like Kubernetes and Docker Swarm also take care of deploying, scaling, and running containers in big environments [12], [13].

3.1 Container Architecture

The actual hardware lies at the bottom of the container architecture, followed by the host operating system, and finally the container engine, which is usually Docker. The container engine is in charge of operating and managing containers on the system. On top of this layer, several containers can run at the same time. Each container has its own application code, dependencies, and runtime environments. Containers don't have a full operating system like virtual machines do. Instead, they share the host OS kernel, which uses fewer resources and makes things run faster. This design lets many containers run smoothly on a single host system while keeping applications separate from each other.



(Fig. 2: Container Architecture)

3.2 Benefits of Containerisation

In modern application development and deployment, containerisation has a number of benefits. One of Containerization has a number of advantages for developing and deploying applications today. Containers are measured in megabytes, while virtual machines are measured in gigabytes. This makes containers lighter and faster to deploy, and they use fewer resources. Containers also start up quickly, usually in seconds or milliseconds, which makes them good for environments that change quickly. Another big benefit is portability, which means that apps can run the same way on other platforms without any changes. Containerization also makes things more efficient by letting numerous containers share the host operating system kernel, which makes better use of hardware [14].

3.3 Limitation

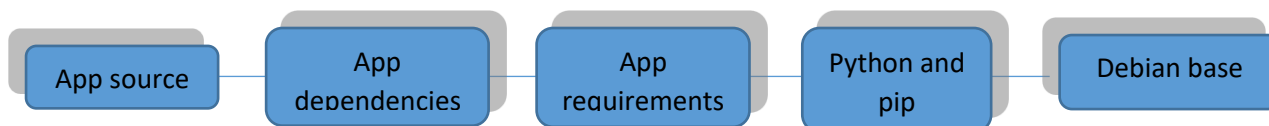
Containerisation has several drawbacks, even though it has some benefits. Because containers share the host operating system kernel, they don't give as much isolation as virtual machines do. If they aren't set up correctly, this could pose security problems. Container environments also rely on the host operating system, which makes it less flexible when different operating systems are needed. To make sure that container technologies are used safely and effectively, it is important to have the right security measures and management processes in place.[14]

4. Docker, Docker Image and Docker file Construction

Docker is a contemporary technology that lets developers create, run, and control programs in containers. It makes it easier to deploy applications by putting code, dependencies, and settings into portable units that can execute the same way in different environments. Docker has a client-server model. The Docker client transmits commands to the Docker daemon, which makes and manages containers. It also uses a Docker registry to keep and share container images, which makes it easy to share and reuse them. This method makes current software development more efficient, portable, and scalable.

4.2 Docker Image

A Docker image has many levels, and each layer shows a different set of changes to the filesystem, including adding, changing, or removing files. A manifest file lists all the layer IDs (digests) needed to generate the image. This layered structure lets images use the same parts over and over, which makes them more efficient and cuts down on duplication. Each layer usually has a specific job. For example, the base layer has the basic system tools and package managers, the runtime layer installs environments like Python and pip, the configuration layer copies files like requirements.txt, the dependency layer installs application libraries, and the application layer has the actual source code. This modular method speeds up builds and makes storage more efficient because shared layers can be utilized in many images. Docker uses content-addressable storage to keep these layers in order and make sure they are always correct. A union filesystem stacks all the levels into one view when a container is made. A writable layer is put on top, which lets you make modifications without changing the original image. With this approach, several containers can run from the same image without affecting performance or isolation [8][15].



(Fig. 3: Docker Image Layers)

4.3 Dockerfile Example

A Dockerfile is a plain text file that has a list of steps that are used to make a Docker image automatically. It sets the base image, inserts application files, installs dependencies, and tells the container how to run. The Dockerfile's instructions each make a layer in the image, which makes it

reusable and efficient. Using a Dockerfile makes sure that applications are constructed in a consistent and repeatable way, so they work the same way in all settings.[16]

FROM nginx:latest

COPY index.html /usr/share/nginx/html/

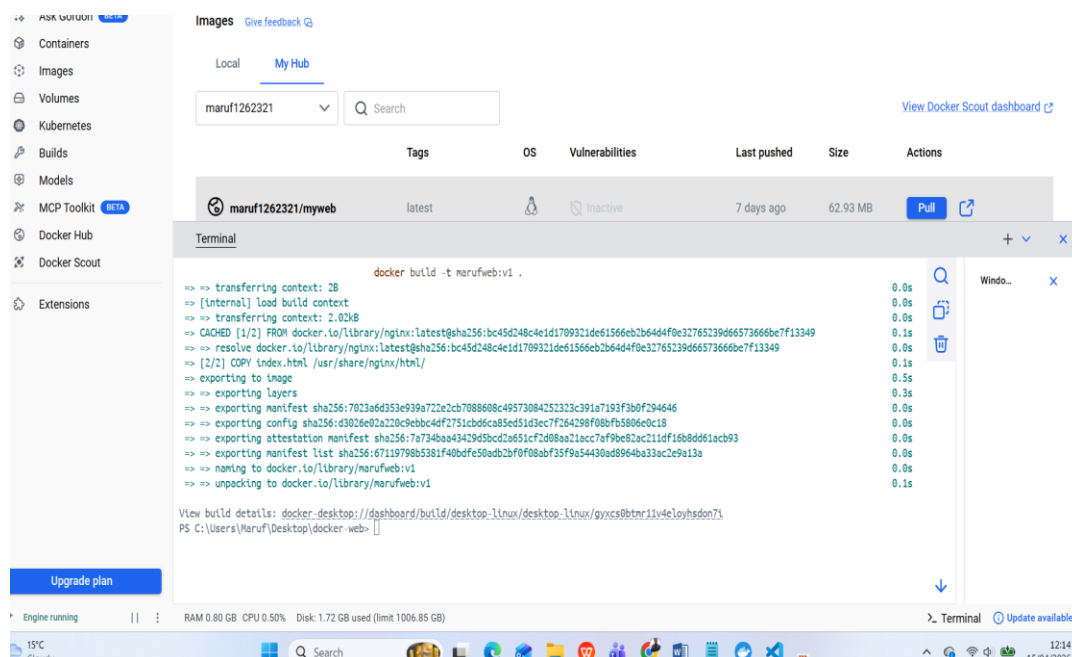
4.4 Explanation

- FROM → Base image
- COPY → Adds files
- CMD → Runs commands

4.4 Build Image

docker build -t myweb:v1 .

This command creates a Docker image from the Dockerfile.



(Fig. 4: Image Build Process)

5. Images and Publishing Images

Docker images are templates used to create containers.

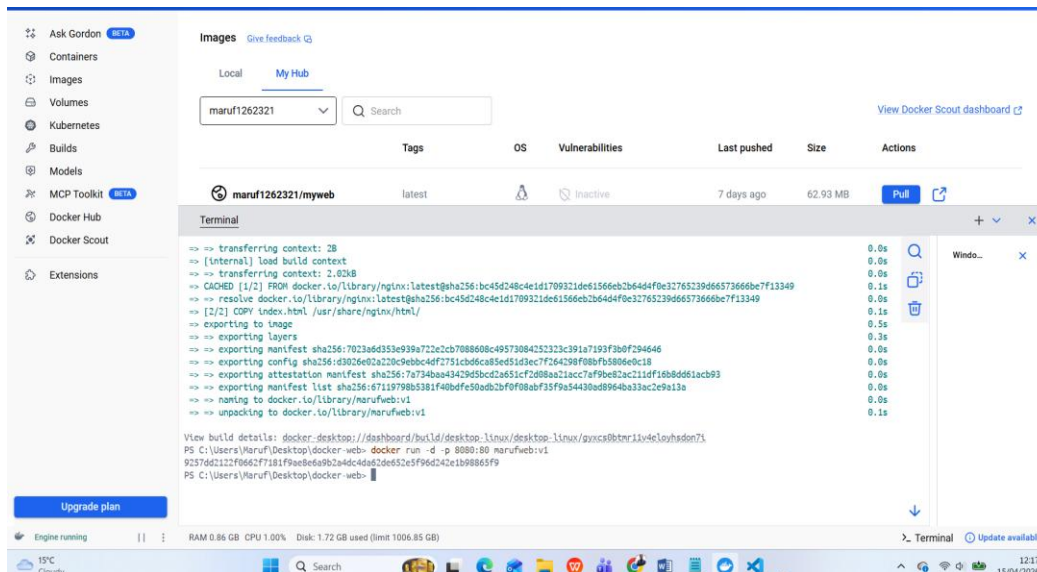
5.1 Image Workflow

This procedure shows how to make and execute a containerised app. A Dockerfile first tells you what you need to do to set up the application environment. The build phase turns these instructions into a Docker image, which is like a template that can be used again and over again. Then, this image is used to execute a container, which makes a live, separate version of the program. The container runs the software the same way in all environments [17].

Dockerfile → Build → Image → Run → Container

5.2 Running a Container

docker run -d -p 8080:80 marufweb:v1



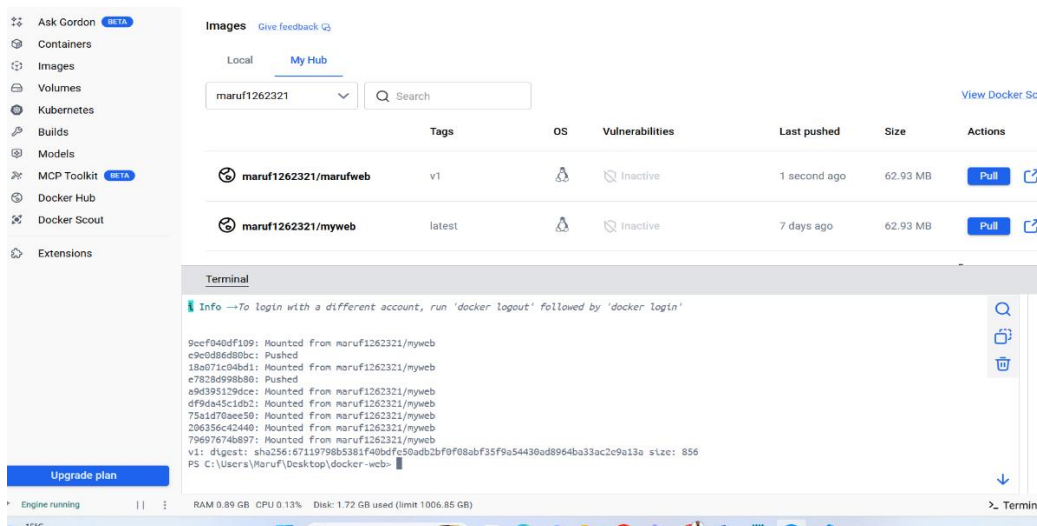
(Fig. 5: Running Container)

5.3 Publishing to DockerHub

A Docker image can be prepared and uploaded to a registry like Docker Hub using these commands. The command `docker tag myweb maruf1232321/myweb` gives the image a new name and links it to a repository under the designated username. The image can then be shared and deployed from other systems by uploading it to the registry using the `docker push maruf1232321/myweb` command[17].

`docker tag myweb maruf1232321/myweb`

`docker push maruf1232321/myweb`



(Fig. 6: Image Push to Docker Hub)

Docker Hub is a repository that stores images and allows sharing with others [17].

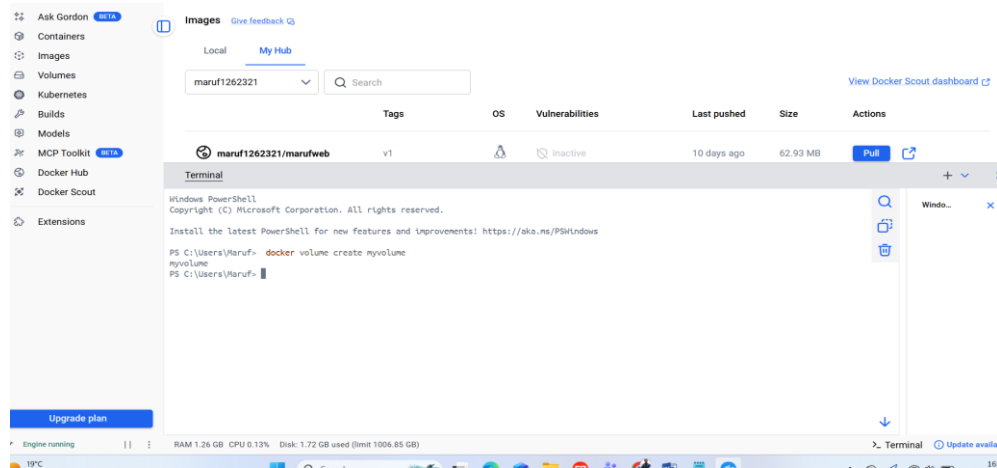
6. Container Volumes

Docker volumes are a technique to store data outside of containers so that important information isn't lost when containers cease working or are destroyed. They are a reliable way to protect data no matter what happens to the container. By default, when a container writes data, it is saved in the container's writable layer. This layer is just temporary, thus any data saved there will be lost if the container is stopped, deleted, or made again. Because of this, it is not appropriate for apps that need to keep data for

a long time. To get around this problem, Docker enables you mount volumes to particular folders in a container's file system. When you attach a volume like way, all the data you write to the directory you choose is stored in the volume instead of the container. So, even if the container is taken away, the data stays safe. Volumes are also versatile because you may easily add them to more containers. This lets one container use or read data from another container, which helps keep things going and makes it easier to manage data across deployments [18].

6.1 Creating a Volume

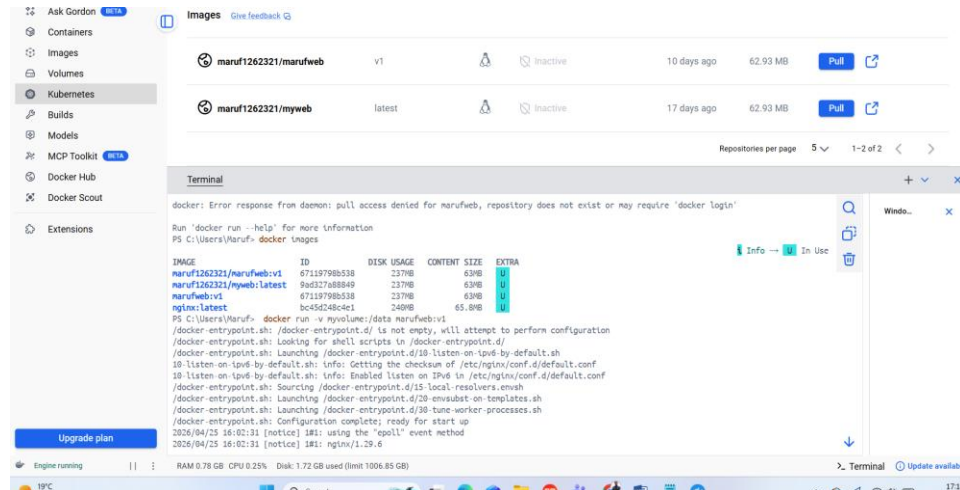
- `docker volume create myvolume`



(Fig. 7: Volume Creation)

6.2 Using a Volume

- `docker run -v myvolume:/data myweb`



(Fig. 8: Volume Mounting)

The picture demonstrates that the Docker image `marufweb:v1` was successfully made and run using a volume (`myvolume:/data`), and the container starts up appropriately with the Nginx server running. It also shows that the problem that happened previously was fixed by utilising the right local image name instead of trying to get an image that doesn't exist from Docker Hub.

6.3 Benefits

Docker volumes keep data safe by keeping it outside of the container. This way, vital information won't be lost when containers are stopped or removed. They also help with backup and recovery because it's easy to copy, move, or restore data stored in volumes when you need to. Volumes also let containers share data, which means that numerous containers can access the same data, which makes working

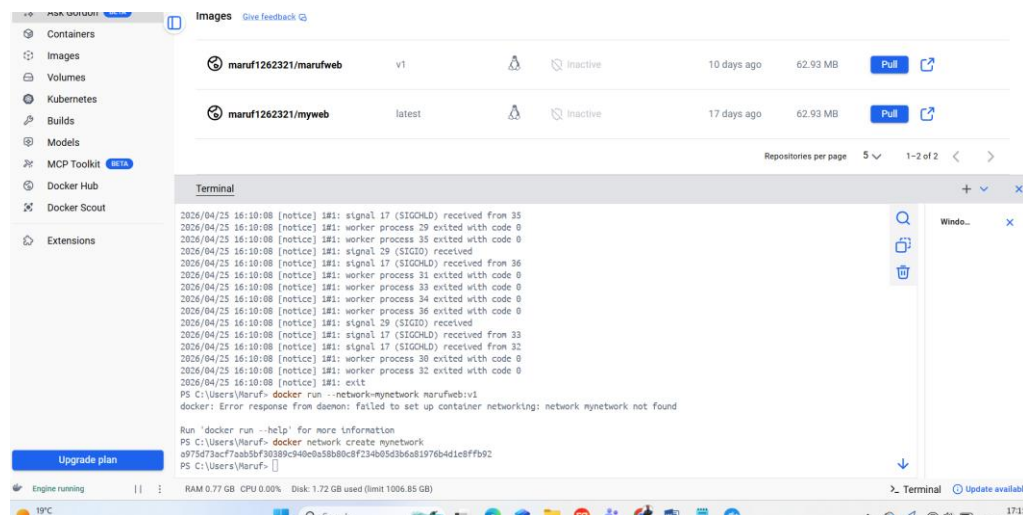
together and getting things done faster. This is why volumes are so important for things like databases, file storage systems, and environments with more than one container.[19]

7. Container Networking

Container networking makes it possible for containers to interact to each other and to systems outside of them by connecting them to both virtual and physical network interfaces. Container networking that works well must have low latency and high throughput, as well as advanced administration and virtualisation options. There are a number of things that affect how well container networking works. These include network interface options like veth and MACVLAN, as well as bridge technologies like Linux bridge and Open vSwitch (OVS). Communication can happen in numerous ways, such as between virtual machines, between separate hosts, or even within the same host. It can use different protocols, such TCP or UDP, and include one or more data flows. Also, networking works differently on different operating systems. For example, Windows and Linux environments have different setups. Bridge networks (the default for communication between a single host), host networks (direct access to host networking), and overlay networks (for communication between multiple hosts and dispersed containers) are all common types of container networks.[20]

7.2 Create Network

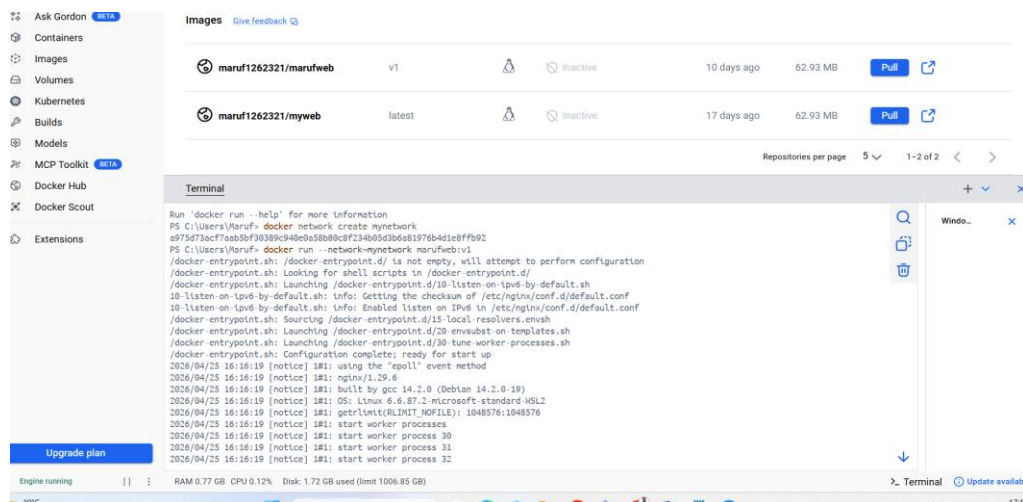
- **docker network create mynetwork**



(Fig. 9: Network Creation)

7.3 Run Container in Network

- **docker run --network=mynetwork marufweb:v1**



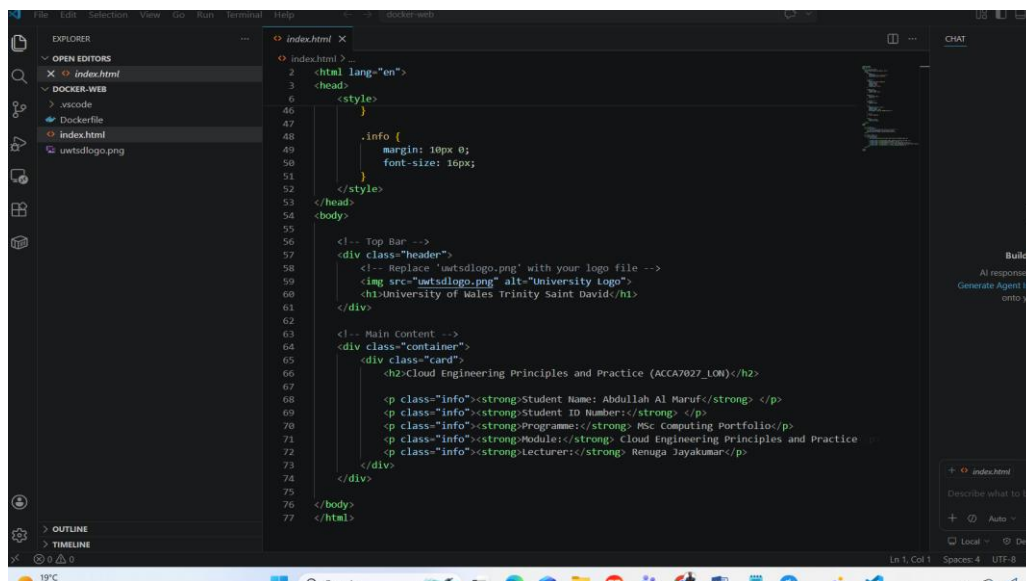
(Fig. 10: Container Networking)

Networking enables applications like web servers and databases to interact. The picture demonstrates that the Docker image marufweb:v1 was successfully made and run using a volume (myvolume:/data), and the container starts up appropriately with the Nginx server running. It also shows that the problem that happened previously was fixed by utilising the right local image name instead of trying to get an image that doesn't exist from Docker Hub.

8 Practical Example: Full Docker Workflow

This example demonstrates the complete process.

Step 1: Create HTML File



```

1 <html lang="en">
2 <head>
3   <style>
4     .info {
5       margin: 10px 0;
6       font-size: 16px;
7     }
8   </style>
9 </head>
10 <body>
11   <!-- Top Bar -->
12   <div class="header">
13     <!-- Replace 'uetsdlogo.png' with your logo file -->
14     
15     <h1>University of Wales Trinity Saint David</h1>
16   </div>
17
18   <!-- Main Content -->
19   <div class="container">
20     <div class="card">
21       <h2>Cloud Engineering Principles and Practice (ACCA7027_10M)</h2>
22
23       <p class="info"><strong>Student Name:</strong> Abdullah Al Maruf</strong></p>
24       <p class="info"><strong>Student ID Number:</strong> </p>
25       <p class="info"><strong>Programme:</strong> MSc Computing Portfolio</p>
26       <p class="info"><strong>Module:</strong> Cloud Engineering Principles and Practice</p>
27       <p class="info"><strong>Lecturer:</strong> Renuga Jayakumar</p>
28     </div>
29   </div>
30 </body>
31 </html>

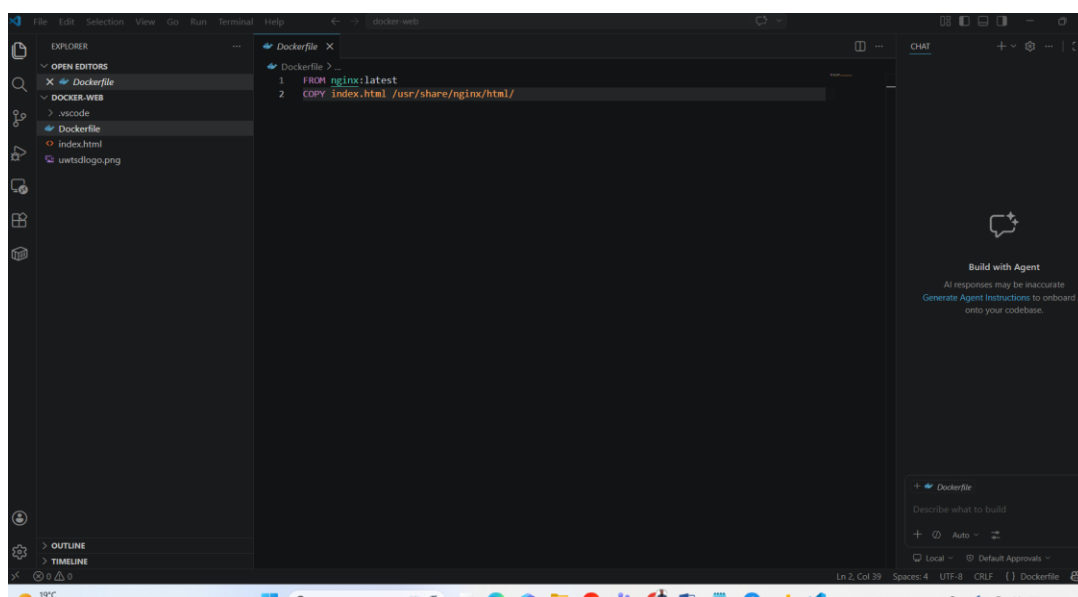
```

(Fig. 11: HTML File Setup)

Step 2: Create Dockerfile

FROM nginx:latest

COPY index.html /usr/share/nginx/html/



```

1 FROM nginx:latest
2 COPY index.html /usr/share/nginx/html/

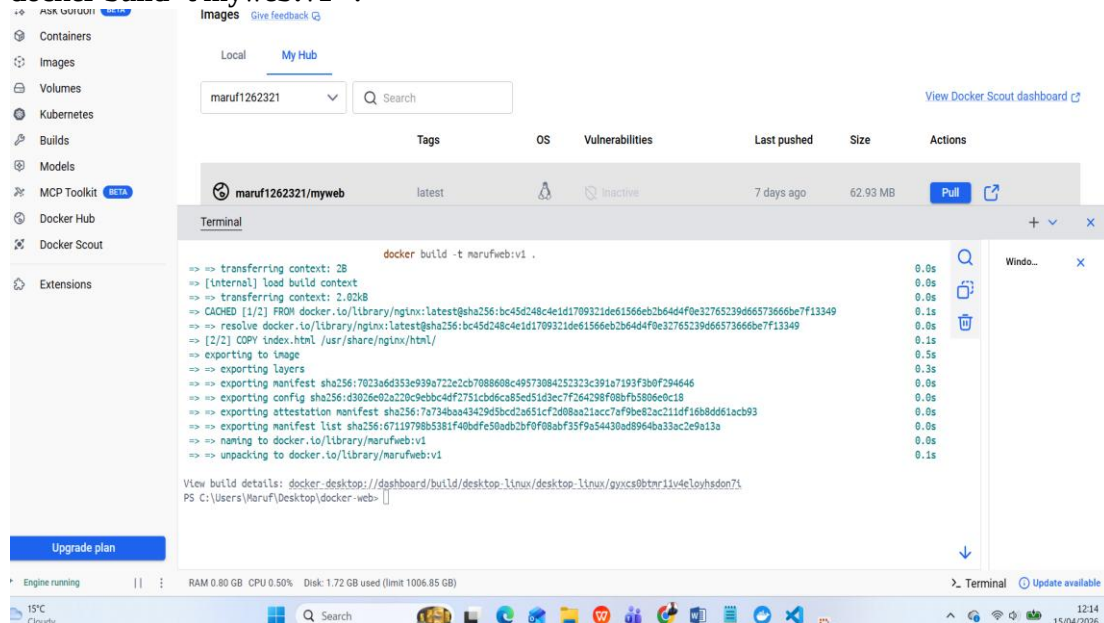
```

(Fig. 12: Dockerfile Setup)

This screenshot illustrates a simple Dockerfile made in VS Code that uses the base image `nginx:latest` to make a container for a web server. The `COPY` command moves the `index.html` file to the default Nginx directory. With this approach, you can quickly deploy a static web app with Docker.

Step 3: Build Image

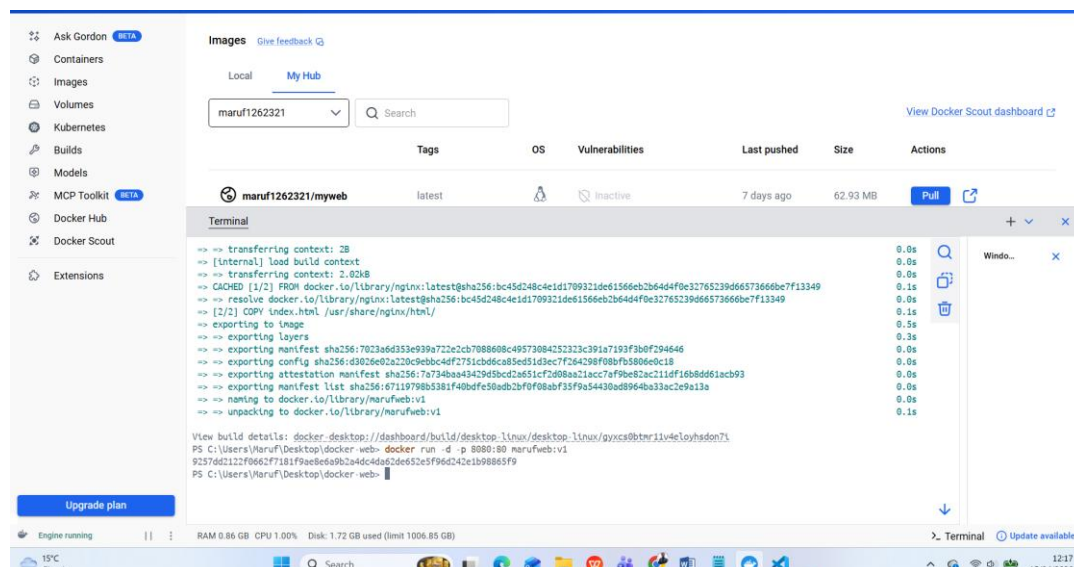
docker build -t myweb:v1 .



(Fig. 13: Image Build Command)

Step 4: Run Container

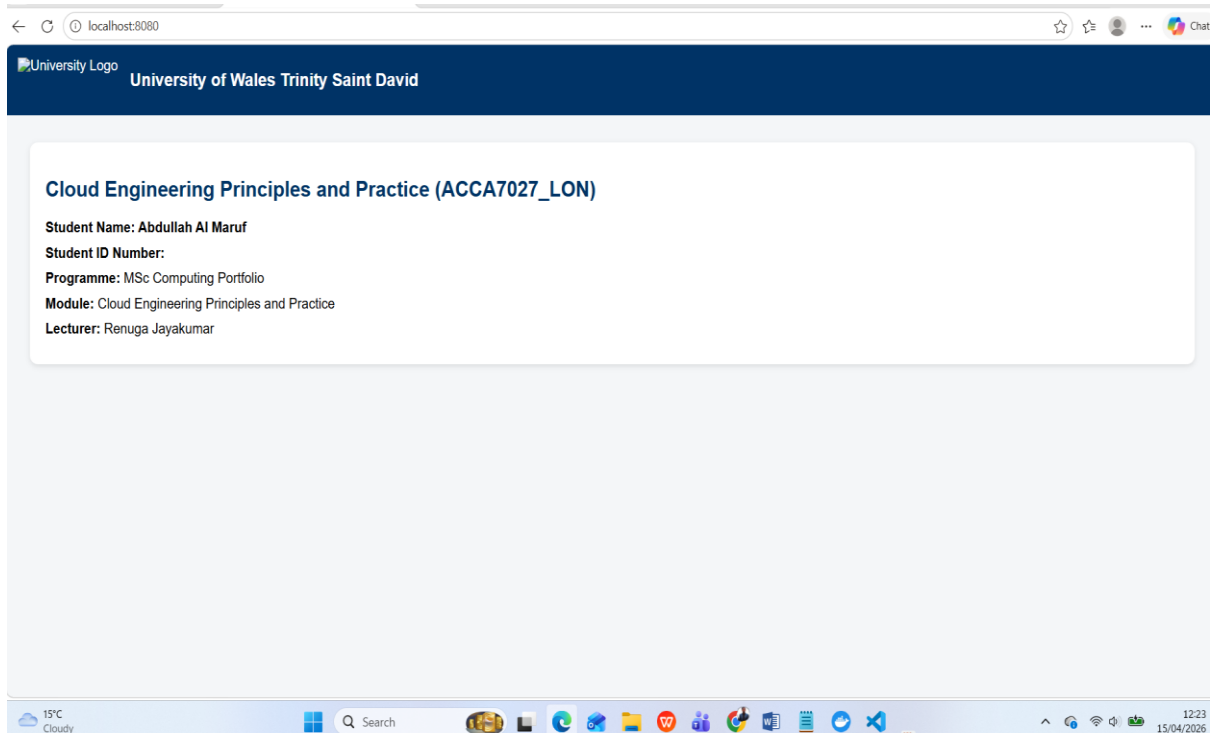
docker run -d -p 8080:80 myweb:v1



(Fig. 14: Container Execution)

Step 5: Access Application

Open browser → localhost:8080



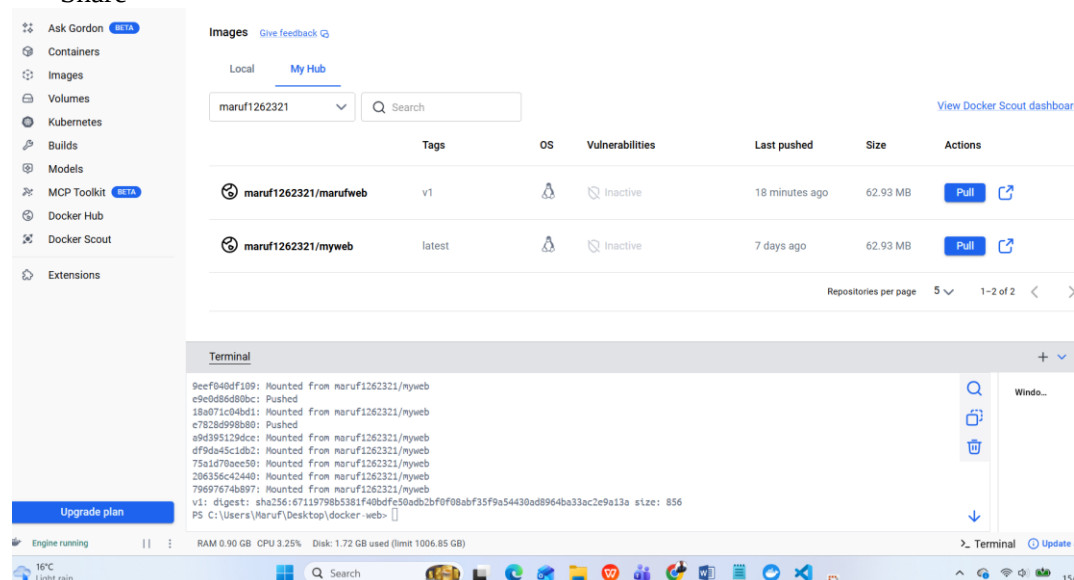
(Fig. 15: Web App Output)

Step 6: Push to DockerHub

`docker push maruf1262321/myweb:v1`

This demonstrates the full lifecycle:

- Build
- Run
- Deploy
- Share



(Fig. 16: Docker Image Push)

9 Cloud Provider Approach to Load Balancing, Scalability and Application Gateways

Cloud providers like Amazon Web Services, Microsoft Azure, and Google Cloud Platform use fully managed services to make infrastructure management easier. These services include load balancing, scalability, and application gateways.

9.1 Load Balancing

Managed services like AWS Elastic Load Balancer and Azure Load Balancer help cloud providers balance load. These services send incoming user requests to several servers so that no one server gets too many requests. For instance, load balancers send traffic to the least busy servers in a system like Netflix, which has millions of customers at the same time. This makes sure that videos stream smoothly [21].

9.2 Scalability

Auto-scaling methods in cloud computing make it possible to scale up or down automatically based on demand. AWS Auto Scaling and Azure Virtual Machine Scale Sets are examples of services that automatically add or remove servers to keep performance up. For instance, when Netflix releases new material, more servers are instantly added to manage the extra traffic and make sure that users have a good experience[22].

9.3 Application Gateways

Application gateways work at the application layer (Layer 7) and add additional routing and security features to cloud apps. They provide features like URL-based routing, SSL termination, and a Web Application Firewall (WAF) that improve both speed and security. For instance, application gateways on platforms like Netflix send user requests to specialised services like authentication, video streaming, and recommendation algorithms. This makes sure that services are delivered quickly and safely [23].

10 Why Load Balancing and Scaling Are Complementary

Load balancing and scalability are quite similar and must work together to make sure the system works well. Load balancing makes sure that incoming traffic is spread fairly across servers, while scalability makes sure that there are enough resources to meet growing demand. If you utilise load balancing without scaling, servers can get too full. If you use scaling without load balancing, traffic may not be sent out in the best way. When Netflix gets a lot of traffic, for example, the system scales by adding more servers and load balancers spread the traffic between them. This integrated method makes sure that performance, reliability, and resource use are all at their best [24].

11 Benefits of Combining Load Balancing, Scaling, and Application Gateways

A cloud system that is very stable and efficient can be made by combining load balancing, scalability, and application gateways. These technologies work together to make sure that services are always available, even if some servers go down. Load balancing spreads traffic evenly over several servers, which speeds up the whole system. Scalability lets the system dynamically change resources dependent on how much demand there is. Also, application gateways make things safer by adding capabilities like traffic filtering and protection against threats that come from the web.

From the user's point of view, this integration makes for a smooth and dependable experience. For services like Netflix, consumers benefit from little buffering and access that doesn't stop, even when the service is busy. Services are always available, and sensitive data is safe from possible cyber threats. In general, these technologies work together to make sure that current cloud environments have the best system performance, dependability, and security [25].

12 Cloud Growth and Serverless Analysis

12.1 Driving Forces behind Cloud Computing Expansion

Several important things have led to the rapid growth of cloud computing, including cost-effectiveness, flexibility, scalability, and advances in technology. These drivers have changed the way businesses

plan, build, and run their IT infrastructure. There are a few main reasons why cloud computing is growing so quickly [25].

12.2 Cost Efficiency

Cloud computing gets rid of the requirement for expensive physical infrastructure, which lowers capital expenses by a lot. It works on a pay-as-you-go basis, which means that businesses only pay for the resources they utilize. This lowers both capital expenditures (CapEx) and ongoing operating costs. Amazon Web Services and Microsoft Azure are examples of services that don't require an upfront investment and take care of maintenance, upgrades, and energy use. Because of this, cloud computing is very cost-effective and especially helpful for small enterprises and startups [26].

12.3 Flexibility and Agility

Cloud computing is quite popular because it is very flexible. Businesses can easily deploy, change, and grow programs without having to buy and set up physical hardware. Developers can quickly set up resources, test apps, and make changes in real time. This helps with modern methods like DevOps and CI/CD, which speed up innovation and cut down on time to market. Cloud platforms also offer a wide range of services, such as virtual machines, containers, and serverless computing. This lets businesses pick the best options for their needs [26].

12.4 Scalability and Elasticity:

Cloud computing can scale up almost without limit, so businesses can handle different amounts of work easily. When demand is high, resources can be increased, and when demand is low, they can be decreased. This flexibility makes sure that resources are used in the best way possible and that infrastructure is not over- or under-used. For instance, an e-commerce site can automatically increase its resources during busy shopping times and then lower them again later to save money and improve performance [26].

12.5 Global Accessibility and Remote Access

With cloud computing, you may use apps and data from anywhere in the globe as long as you have an internet connection. This has become more significant as more people work from home and work with people from all over the world. No matter where they are, employees can safely access systems, work together in real time, and stay productive. This level of accessibility has played a big role in the worldwide use of cloud technologies [27].

12.6 Reliability and High Availability

Cloud providers make sure that their services are very reliable by using distributed data centers and redundancy systems. Even if hardware fails, services are designed to keep working. Automated backups, failover systems, and disaster recovery are some of the features that keep businesses running. This dependability is highly important for businesses that need services to be delivered without interruption [28].

12.7 Automation and Integration with Modern Technologies

Automation and Integration with New Technologies: Cloud platforms work well with new technologies like AI, machine learning, and big data analytics. They also support automation tools. This lets businesses automate processes that are done again and over again, make workflows more efficient, and learn from data. Automation also cuts down on mistakes made by people and makes operations run more smoothly, which is why cloud computing is such an important part of current IT plans [29].

12.8 Innovation and Competitive Advantage

Cloud computing gives businesses access to modern tools and infrastructure that can grow with them, making it possible for them to quickly create new apps and innovate. It gives businesses a big edge over their competitors by letting them try out new ideas, swiftly deploy apps, and respond well to changes in market demand. Its pay-as-you-go strategy makes it cost-effective, which lowers both capital and

operating costs. Platforms like Amazon Web Services and Microsoft Azure cut down on big upfront costs and make better use of resources. Cloud environments are very versatile and make it easy to quickly install, update, and scale programs. They also let systems dynamically change resources based on demand, which makes them scalable and flexible. Also, cloud computing is a crucial driver of digital transformation and long-term competitiveness since it is easy to access from anywhere in the world and is always available [29].

13 Serverless Computing (Functions as a Service)

Serverless computing is a cloud-based architecture that lets developers run code without having to worry about servers. The cloud provider takes care of all the infrastructure. AWS Lambda and Azure Functions are two examples of services that make this possible by letting apps execute as discrete, separate functions. In this architecture, developers write code as functions that run automatically when certain events happen, including HTTP requests, file uploads, or database updates. The cloud provider automatically handles resource allocation, scaling, and execution, making sure that everything runs smoothly without any help from users. This architecture is based on events, which means that programs can automatically grow or shrink based on demand and only use resources when they are needed. A serverless function, for instance, can dynamically resize uploaded photos without needing a separate server. This makes it a very efficient and cost-effective option for modern cloud apps [30].

14 Comparison Serverless and Containerisation

14.1 Serverless:

In serverless computing, the cloud provider takes care of all the underlying infrastructure, so developers just have to worry about creating and delivering code. This method runs applications as functions that are triggered by certain events, like HTTP requests or file uploads. One of the best things about serverless computing is that it can automatically scale, meaning that resources are assigned based on demand without any manual setup. Also, the cost model is based on pay-per-execution, which means that users are only charged when their code runs. This makes it very cost-effective for workloads that are not always the same or that change often. However, serverless computing doesn't give developers a lot of freedom because they have to operate within the limits set by the cloud provider. This paradigm works best for tasks that happen when something happens, such as processing data, handling APIs, and doing background jobs [31].

14.2 Containerisation

Containerisation is a way to deploy apps by putting them and their dependencies into containers that work the same way in all contexts. Containerisation, on the other hand, needs users to manage the infrastructure that runs the containers. Tools like Docker make this easier. Scaling in containerised environments can be set up and is commonly done with orchestration platforms like Kubernetes to manage more work. The cost model usually means paying for resources that are always operating. This may not be the best way to do quick activities, but it is better for long-running applications. Containerisation gives developers a lot of freedom because they can manage the runtime environment, configurations, and dependencies. This method is great for putting out full applications, microservices architectures, and systems that need to work well all the time and be easy to control [32].

14.3 Comparison in Context of Cloud Growth

Serverless computing and containerisation both help cloud computing grow, but they do it in different ways. With serverless computing, developers only have to worry about running code, which cuts down on operational costs and infrastructure administration. Its pay-per-use model also makes it very affordable for startups and lightweight, event-driven apps. Containerisation, especially with Docker, on the other hand, gives you more control and flexibility over the application environment. This makes it better for microservices architectures and complicated, long-running systems. In the actual world, businesses generally use both methods to get the best performance and efficiency. Netflix might employ containers for its main streaming services, which need to be stable and always running, while serverless functions are used for background tasks like managing data or sending notifications. This mixed

strategy makes the best use of resources, scales well, and works the best, which is a big part of modern cloud growth [33].

15. Conclusion

In the final analysis, virtualisation and containerisation are important technologies that make modern cloud computing possible. Virtualisation lets numerous virtual machines run on the same hardware with tight isolation. Containerisation, on the other hand, lets applications run in separate environments utilising a shared operating system kernel, which is a lightweight and efficient way to do things. Technologies like Docker make this process easier by providing capabilities like images, containers, volumes, and networking. This makes sure that applications are deployed in a consistent, portable, and efficient-way.

Cloud platforms like Amazon Web Services and Microsoft Azure make these features even better by providing scalable and reliable services like load balancing, auto-scaling, and application gateways. Cost-effectiveness, flexibility, and worldwide accessibility are what make cloud computing so popular. Models like AWS Lambda and Azure Functions make operations even simpler. Overall, these technologies work together to make it possible to build cloud-native, scalable, and efficient apps.

References:

- [1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST, 2011.
- [2] Amazon Web Services, "Overview of Cloud Computing," 2024.
- [3] Microsoft Azure, "What is Cloud Computing?," 2024.
- [4] IBM, "Benefits of Cloud Computing," 2024.
- [5] ScienceDirect, "Hardware Virtualization," 2026. [Online]. Available: <https://www.sciencedirect.com/topics/computer-science/hardware-virtualization>
- [6] F. Rodríguez-Haro et al., "A Summary of Virtualization Techniques," Procedia Technology, 2012.
- [7] M. Leppälä, "Virtualization and Cloud Computing," Aalto University, 2021.
- [8] Docker Inc., "Docker Overview," 2024. [Online]. Available: <https://docs.docker.com/>
- [9] GeeksforGeeks, "Containerization using Docker," 2026.
- [10] IBM, "Containerization Explained," 2024.
- [11] Networkers Home, "Docker Fundamentals," 2024.
- [12] D. Manor et al., "Containerization in Cloud Computing: A Review," 2025.
- [13] O. Bentaleb et al., "Containerization Technologies: Taxonomies, Applications and Challenges," J. Supercomput., 2021.
- [14] Docker77, "Docker Volumes: With Real-Time Examples," 2024. [Online]. Available: <https://docker77.hashnode.dev/docker-volumes-with-real-time-examples>
- [15] Y. Zhao et al., "Performance of Container Networking Technologies," HotConNet, 2017.
- [16] V. Medel et al., "Performance Modeling of Container-Based Systems," ACM, 2016.
- [17] M. Arango et al., "Performance Evaluation of Container-Based Virtualization," 2017.
- [18] Amazon Web Services, "Elastic Load Balancing Documentation," 2024.
- [19] Microsoft Azure, "Azure Load Balancer Documentation," 2024.
- [20] Microsoft Azure, "Azure Application Gateway Overview," 2024.
- [21] Netflix, "Cloud Architecture Overview," 2023.
- [22] ISO, "Information Technology—Cloud Computing Vocabulary," ISO/IEC 22123-1, 2023.
- [23] ISO, "Virtualization Software Definitions," ISO/IEC TS 23167, 2020.
- [24] ISO, "Cloud Computing Definition and Characteristics," 2023.
- [25] Amazon, "AWS Lambda Documentation," 2024.
- [26] M. Spigolon et al., "Serverless Architectures on AWS," O'Reilly, 2024.
- [27] AWS, "Firecracker MicroVM Architecture," 2018.
- [28] Serverless Research, "Serverless Cloud Computing: State of the Art," 2023.
- [29] T. A. Bodhanya, "Comparing Cloud Orchestrated Container Platforms," 2022.
- [30] Y. Mao et al., "DRAPS: Resource-Aware Placement for Docker Containers," 2018.
- [31] C. Novaes et al., "Docker and Kubernetes Orchestration in Cloud Systems," 2020.
- [32] A. Rahman et al., "Cloud Native Architecture and Software-Defined Infrastructure," 2023.
- [33] NITRD, "Containerization and Virtualization Summary Report," 2020.

Video Presentation Link:

- <https://youtu.be/xT5RByrwbec?si=I85PYAzkjcSh6ggG>