



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

**CONTAINERISATION AND CLOUD COMPUTING: A
PRACTICAL EVALUATION USING DOCKER**

Student Name : Likhitha Pyneni

Student Number : 2509981

Table of Contents

1. Introduction.....	3
2. Virtualisation and Containerisation in Cloud Computing.....	3
2.1 Principles of Virtualisation	3
2.2 Principles and Benefits of Containerisation.....	4
2.3 Docker as a Containerisation Platform	6
3. Practical Docker Implementation and Demonstration.....	7
3.1 Dockerfile Construction and Image Creation	7
3.2 Running Containers and the Image–Container Relationship.....	8
3.3 Publishing Docker Images to Docker Hub	9
3.4 Container Volumes and Persistent Storage	10
3.5 Container Networking.....	11
3.6 End-to-End Example: Containerised Web Server.....	12
4. Load Balancing, Scalability, and Application Gateways	13
4.1 Overview.....	13
4.2 Cloud Provider Approaches	14
4.3 Relationship Between Load Balancing and Scalability	14
4.4 Combined Benefits to Customers	15
5. Driving Forces Behind Cloud Computing and Serverless Architectures	15
5.1 Cost, Flexibility, and Operational Efficiency.....	15
5.2 Serverless Functions vs Containerisation	15
6. Conclusion	16
7. References.....	17
Appendix.....	20

1. Introduction

Cloud computing is one of the key models that have been developed to provide computing services in an elastic, scalable and economical way. Cloud environments allow resources and applications to be dynamically deployed on demand as opposed to the physical infrastructure that uses a fixed set of physical infrastructure. One of such technologies that facilitate this paradigm is containerisation and which enables the packaging of applications along with their dependencies and then running them in a similar location with other environments. A combination of lightweight architecture and a rich image creation, container execution, networking, storage and image distribution made Docker a powerful platform of containerisation. This report evaluates the function of virtualisation and containerisation in cloud computing which is backed by a practical implementation of Docker and a more detailed discussion of the core cloud concepts and drivers.

2. Virtualisation and Containerisation in Cloud Computing

2.1 Principles of Virtualisation

Virtualisation is a fundamental principle in the current computing community that allows the physical hardware facilities to be abstracted and shared among numerous time-isolated loads. Isolating software environments of physical infrastructure, virtualisation enhances the use of hardware, lowers the cost of operations, and enhances resource management flexibility. This abstraction enables the dynamically configured assignment of computing assets like processing capabilities, memory, storage and networking capabilities in cloud environments depending on demand. Consequently, organisations are able to run and operate a variety of systems on a physical server, which has logical isolation of workloads, a feature critical to scalability and optimal utilisation of infrastructure.

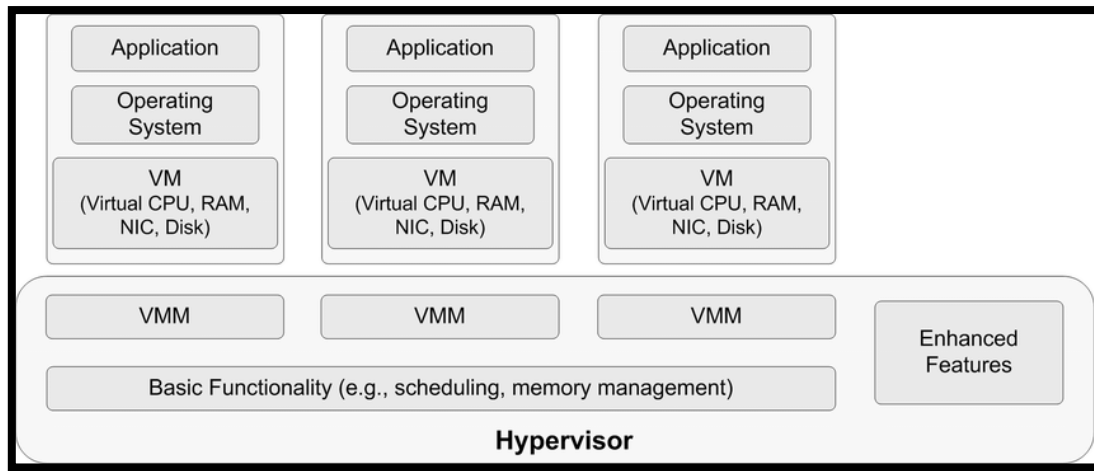


Figure 1: Hypervisor-based Virtualization Architecture

(Source: researchgate.net, 2015)

The most frequently used implementation to traditional virtualisation is by utilising hypervisors which provide and currently maintain the virtual machines by placing them between the physical hardware and the operating systems which are executed at the top. Virtual machines are strongly isolated and secure with each having its own operating system, libraries and applications. This model best fits well in multi-tenant cloud systems whereby the same physical equipment is shared by various users (Zhou, Zhou and Hoppe, 2022). The underlying cost on operating a full-fledged operating system per virtual machine however presents extra overhead which raises resource usage and reduces startup time relative to lightweight solutions.

Container-based technologies are based on operating system-level virtualisation which provides an alternative of sharing an operating system kernel on the host machine and isolating applications at the process level. This dramatically reduces overhead and allows starting the operations and workloading more rapidly. In cloud computing, the hardware virtualisation layer provides a foundation that allows providing an elastic infrastructure by being able to provision infrastructure, and there is the containerisation which provides support to an elastic infrastructure that can be used to deploy an application very quickly. Collectively, these methods enable cloud solutions to provide the balance between isolation, performance, and efficiency to enable cloud-native and continuous delivery models (Cherukuri, 2024).

2.2 Principles and Benefits of Containerisation

Containerisation is a contemporary application deployment method whereby software along with its dependencies, libraries, and the configuration are bundled into one self-interested unit

called a container. In contrast to the type of deployment that was previously experienced, containerisation makes sure that applications will work in different activities, irrespective of the infrastructures. Images are self-contained with the definition of the application run time and environment, hence they are reproducible and standardised. This type of platforms facilitates quick deployment of applications and is also in line with the modern development styles where automation and consistency are important in all development, testing and production environments (Saboore et al., 2022).

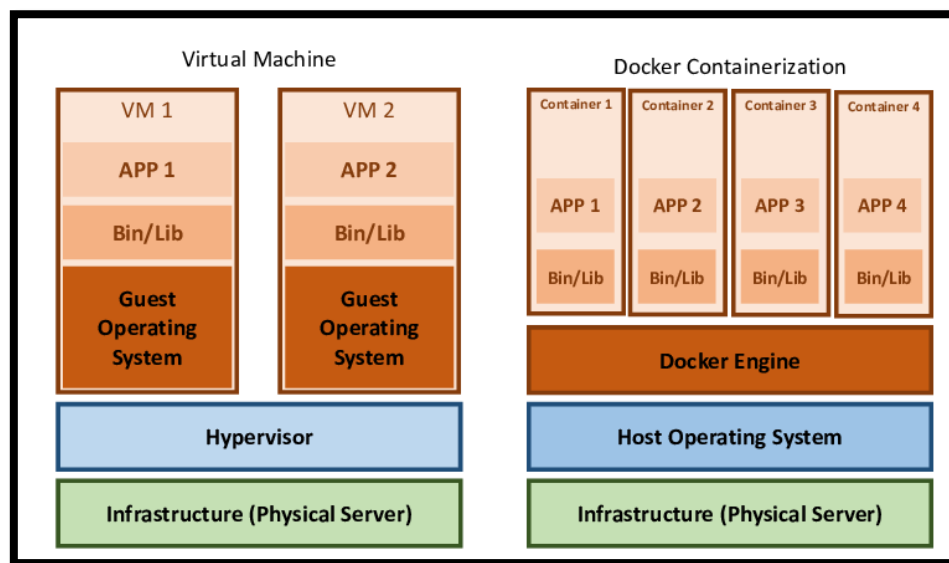


Figure 2: Docker Container VS Virtual Machine

(Source: researchgate.net, 2023)

One of the major differences between containers and virtual machine is the degree of virtualisation. Virtual machines are based on hardware-level virtualisation and they must use an entire guest operating system per and are therefore also more resource consuming and slow to boot. By comparison, containers communicate with the level of the operating system and use the same kernel as the extant hosting system but have process-level isolation. The resulting architectural distinction renders containers much lighter and efficient making it possible to support more workloads over the same physical infrastructure. Consequently, the use of containers allows scaling faster and a better utilisation of computing resources compared to virtual machine.

Containerisation has the following advantages especially in the aspect of portability, efficiency, and scalability. Containers have the capability to run the same way both locally in development

setups, on-premise-based data centres and in clouds. Their low weight also makes them overhead friendly and they will start up very fast, an important factor in running dynamic workloads. Horizontal scalability is also promoted by containerisation because then, one can replicate or kill containers fast in reaction to demand fluctuations (Rosa et al., 2023). These features render the containers a good fit to distributed systems and architectures based on microservices.

Containerisation is a very essential aspect in cloud computing to facilitate the flexible application delivery that is resilient. Cloud platforms are also done in such a way that they can offer resources in an elastic manner and containerised applications can fully utilise this. The local image-building system, consistent container execution and distribution across container registries facilitate the efficient deployment processes in clouds. As has been shown in the practical implementation of Docker, the process of containerisation enables applications to be created, run and published in a form that meets the needs of the operation of a cloud environment quite well.

2.3 Docker as a Containerisation Platform

Docker is considered to be one of the most used containerisation platforms as it is quite simple, well tooled, and the ecosystem is well supported. It offers a common platform to construct, operate and distribute containers, and thus containerisation is available to both the developers and operations team. The adoption of Docker has been fuelled by its potential to simplify the complex container operations to simple and repeatable workflows in order to ensure that an application can be packaged and launched with minimal configuration overhead. Such convenience has placed Docker on the platform of cloud-native development techniques.

The fundamental part of Docker is the Docker Engine which is in charge of image building, running containers and controlling the container life cycles. Docker images are immutable templates that specify the way containers get created whereas containers are the running images of containers. Docker also has provided in-built mechanisms in management of both network and persistent storage that allow programmes in distributed environments to perform reliably (Ahamed, Zavarsky and Swar, 2021). The features proven by the creation of images, implementation of containers and utilisation of volumes and networks were used in the practical implementation.

Docker is also relevant to the modern cloud platforms as it is compatible with the cloud-native architecture and models. Docker images are usually supported by the cloud providers as deployable entity and container registries make images visible and reusable across the environment. Publishing images into a container registry The capability to publish images to container registry supports scalable application distribution, which further underlines Docker as a major facilitating technology in the application development and deployment to the cloud (Xia et al., 2021).

3. Practical Docker Implementation and Demonstration

3.1 Docker file Construction and Image Creation

Docker file is the template that determines the way an application setting is developed and that it is run in a container. The arguments used in this implementation include defining a web server-based runtime environment and defining the steps to be followed in creating a functional application image in a Docker file. It makes this possible by specifically defining these steps so that a consistent and repeatable build process becomes possible, and makes sure that the same environment used to build an application will be replicated with reliability across a variety of systems (Kim et al., 2022). This declarative style minimizes configuration drift and facilitates automation of image generation which is pushed in the direction of new cloud deployment deployment mechanisms.

The base image that was selected is that of the Apache HTTP Server image which offers a stable and widely utilised image serving the purpose of serving the static web materials. Using a pre-built, and trusted base image also makes the build simplified as it is possible to use an existing, optimised web server configuration instead of starting to build one. The Docker file will thereupon duplicate the application HTML files to the document root of the web server and the content of the application is embedded into the image (Antonova et al., 2024). This is to ascertain that the application, and that of the runtime environment, are packaged to form one deployable unit.

```

PS C:\Users\WIN 11 PRO> cd D:\cloud-docker-assignment
PS D:\cloud-docker-assignment> dir

Directory: D:\cloud-docker-assignment

Mode                LastWriteTime         Length Name
----                -
d-----          17-12-2025   11:50             web
-a----          17-12-2025   11:46             52 Dockerfile

PS D:\cloud-docker-assignment> docker build -t cloud-web-image .
[+] Building 20.7s (8/8) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile              0.1s
=> => transferring dockerfile: 89B                               0.0s
=> [internal] load metadata for docker.io/library/httpd:2.4     5.7s
=> [auth] library/httpd:pull token for registry-1.docker.io     0.0s
=> [internal] load .dockerignore                                 0.0s
=> => transferring context: 2B                                     0.0s
=> [internal] load build context                                 0.0s
=> => transferring context: 459B                                   0.0s
=> [1/2] FROM docker.io/library/httpd:2.4@sha256:b913eada2685f101f93267e0984109966bbcc3afea6c9b48ed389afb89863aa 13.9s
=> => resolve docker.io/library/httpd:2.4@sha256:b913eada2685f101f93267e0984109966bbcc3afea6c9b48ed389afb89863aa 0.0s
=> => sha256:b282f66bae78cd1b1c9fc369b04c47c02f46c0cb42d9446b257cbdf6504aa73c 2.00MB / 2.00MB 1.2s
=> => sha256:4f4fb70ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 32B / 32B 0.5s

RAM 0.69 GB CPU 0.00% Disk: 2.41 GB used (limit 1006.85 GB)

```

Figure 3: Docker Image Creation Using Dockerfile

(Source: Self-Made using Docker Desktop)

Docker image when created is an unchangeable artefact, which is a representation of the application, dependencies, and configuring the application. Containerisation is based on the principle of immutability because once an image is created then it is never changed. Any modification of the application will involve a rebuild of the image, which is favourable as it enforces a version control and traceability approach. The methodology facilitates consistent deployments and adheres to the cloud-native patterns because images may be run as containers in alternative settings in predictable behaviour.

3.2 Running Containers and the Image–Container Relationship

Docker images and containers are two similar but different notions of the model of containerisation. A container is a running result of an image, which is read-only, whereas an image is a template that outlines an application and an instance on which it is executed. This segregation enables pictures to be reposed to generate numerous containers, which can run on their own. The association of images with containers also gives scalability and stability since the same instance of an application can be initiated by using the same image at any time there is need.

Container execution consists of the creation of an image into a container and provides the container with the required system resource to execute it. Here by implementation the container is given in detached mode, therefore making it be executed at the back and then serve

application requests. Port mapping is employed to open port of the internal web server inside the container to the host system so that the application can be accessed with the help of a web browser (Ajasa, Chizari and Alam, 2025). This mapping shows how the services of containerisation can interoperate with external networks without exposing internal container ports in any manner at all.

Containers execution is based on the advantage of isolation. All containers have their own process space and operate in an isolated environment such that the application is not dependent on other tasks running on the host system. Such isolation increases and improves security and stability because what is wrong in one container does not necessarily encounter other containers. Containers are also lightweight at the same time, which enables efficient utilisation of resources, as well as application separation.

3.3 Publishing Docker Images to Docker Hub

An important phase in application distribution in the clouds is publishing Docker images into a container registry to facilitate this. Any image to be published should be so tagged beforehand with a naming convention which includes identifying the target registry and image version. Image tagging can be used to manage several applications of a single application and ease the process of tracing the application in the event of development and deployment phases. After tagging the image, the image can be pushed into Docker Hub which acts as a centralised container registry where container images can be stored and distributed (Đorđević, Kraljević and Kuk, 2024).

Docker Hub offers the means of storing the images in a secure place and so they could be accessed by the authorised users or could be made publicly available. It allows sharing of images in various environments without necessarily having to connect to the original build system since it is a registry. The capability helps in maintaining the consistency in deployment practises and makes the process of distributing containerised applications easier. Public and private repositories are supported and sometimes organisations can decide on the open distribution and restricted access through security and intellectual property needs.

```
PS D:\cloud-docker-assignment> docker login
Authenticating with existing credentials... [Username: sunangrm]

Info → To login with a different account, run 'docker logout' followed by 'docker login'

Login Succeeded
PS D:\cloud-docker-assignment> docker tag cloud-web-image sunangrm/cloud-web-image:latest
PS D:\cloud-docker-assignment> docker images
```

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
cloud-web-image:latest	f33f11c776c5	175MB	45.2MB	U
sunangrm/cloud-web-image:latest	f33f11c776c5	175MB	45.2MB	U
vulnerables/web-dwa:latest	dae203fe1164	935MB	178MB	

Figure 4: Docker Image Tagging and Authentication

(Source: Self-Made using Docker Desktop)

Cloud-based application delivery is based on the use of container registries like Docker Hub. Deploying applications quickly to cloud or other environments with support of containers can be implemented by just hosting images in a registry. This solution separates the allocation of applications and infrastructure enabling scale-able and repeatable deployments (Lohumi et al., 2023). The practical release of the Docker image illustrates the ways in which containerised applications are made ready to be deployed to the cloud employing an industry-standard tooling.

```
The push refers to repository [docker.io/sunangrm/cloud-web-image]
d57e09c0863c: Waiting
5c8e7be3e66c: Waiting
4c913e0e89a2: Waiting
d57e09c0863c: Pushed
5c8e7be3e66c: Pushed
4c913e0e89a2: Pushed
1733a4cd5954: Pushed
202c0e1a6b4d: Pushed
4f4fb700ef54: Pushed
b282f66bae78: Pushed
a1c7286d48a4: Pushed
latest: digest: sha256:f33f11c776c5b46fe68de7a0e59acab20b76d1c94f6605289a78c585665db2c8 size: 856
PS D:\cloud-docker-assignment> 
```

Figure 5: Successful Publication of a Docker Image to Docker Hub

(Source: Self-Made using Docker Desktop & Docker Hub)

3.4 Container Volumes and Persistent Storage

Containers are in nature volatile and hence all data in the writable layer of a container is lost when the container is ceased to run or when it is deleted. Though this behaviour is beneficial

to stateless application design and can ease container lifecycle management, it provides challenge to an application that needs durable data storage. As a solution to this weakness, Docker offers volumes as a data storage method through which data can be stored outside the container filesystem and still accessed by other containers that are running.

Docker volumes are also controlled by Docker Engine and they are not tied to particular containers. They may also be constructed on the fly and be attached into containers, enabling data to be stored even when a given instance of a container is no longer in use. During this implementation, a volume is generated and bound to the document root of the container, and will guarantee the availability of application data even when the container is re-created (Rabiu, Yong and Mohamad, 2022). This shows how the issue of storage can be decoupled with the execution of the application to enhance reliability and maintainability.

```
PS D:\cloud-docker-assignment> docker volume create cloud-web-volume
cloud-web-volume
PS D:\cloud-docker-assignment> docker volume ls
DRIVER      VOLUME NAME
local       cloud-web-volume
PS D:\cloud-docker-assignment> docker run -d -p 8080:80 -v cloud-web-volume:/usr/local/apache2/htdocs --name cloud-web-container cloud
-web-image
ad5588ec6a829c4e09947a082af77d362dd10e10b3679b373e9bd4e40b323368
PS D:\cloud-docker-assignment> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                                                                 NAMES
ad5588ec6a82   cloud-web-image "httpd-foreground"      21 seconds ago Up 20 seconds  0.0.0.0:8080->80/tcp, [::]:8080->80/tcp   cloud-web-container
PS D:\cloud-docker-assignment>
```

Figure 6: Creation and Use of a Docker Volume

(Source: Self-Made using Docker Desktop)

Persistent storage is especially essential when use is made of stateful applications that depend on the consistency of data, as in the case of a web service that stores user-created content or configuration files. With the help of volumes, it is possible to replace or scale containers without the risk of information being lost. This feature is consistent with the principles of cloud computing as it facilitates resiliency, flexibility, and continuity of operations together with the advantages of containerisation (Mohammadian et al., 2021).

3.5 Container Networking

Container networking allows communication between the containers and isolation between the containers and the host system and other networks. Docker bridge networks offer a default framework of linking containers inside a controlled virtual network, and containers are capable of communicating safely in the usage of internal IP address. With this, network traffic can be

put aside other containers and applications executing in a host through attachment of containers to a custom bridge network as a security measure in case of unintended interference as well as increasing security. Such isolation is namely significant in those cases when the multi-container environment should be logically separated.

```
PS D:\cloud-docker-assignment> docker run -d -p 8080:80 --network cloud-web-network -v cloud-web-volume:/usr/local/apache2/htdocs --name cloud-web-container cloud-web-image
4afc4805590ab897b2b8cabb9425b1b2242bec75dceb8698ff9dd71fba4e56e8
PS D:\cloud-docker-assignment> docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
4afc4805590a	cloud-web-image	"httpd-foreground"	8 seconds ago	Up 6 seconds	0.0.0.0:8080->80/tcp, [::]:8080->80/tcp	cloud-w

```
eb-container
PS D:\cloud-docker-assignment>
```

Figure 7: Running a Docker Container

(Source: Self-Made using Docker Desktop)

Container networking is extremely applicable in the current cloud architectures particularly those anchored on microservices. Under these kinds of architectures applications can be split into smaller independently deployable services that are required to communicate reliably over the network. Docker networks form the basis of this communication, which is the discovery of services and controlled connexion (Sriram, 2022). In cloud systems, these networking principles can be extended to higher-order solutions, which can support distributed systems, which must have inter-service communication that is secure, flexible and resilient.

3.6 End-to-End Example: Containerised Web Server

The enterprise deployment practises are evidenced by the practical application of end-to-end containerisation workflow. The initial HTML application was developed and declared in a Docker file which was also used to create a Docker image. This picture was subsequently implemented as a container where port mapping has been used to make external access of web server possible. Long term storage was implemented with the help of a Docker volume that was used whereas the network isolation was implemented with the help of a custom Docker network. Lastly, the photograph was labelled and posted to Docker Hub, which allowed sharing of applications (Ghandour, El Kafhali and Hanini, 2023). This workflow is accompanied by a video demonstration which can be watched, and the sequence of stages of the process is explained step by step and the ideas are clearly explained.

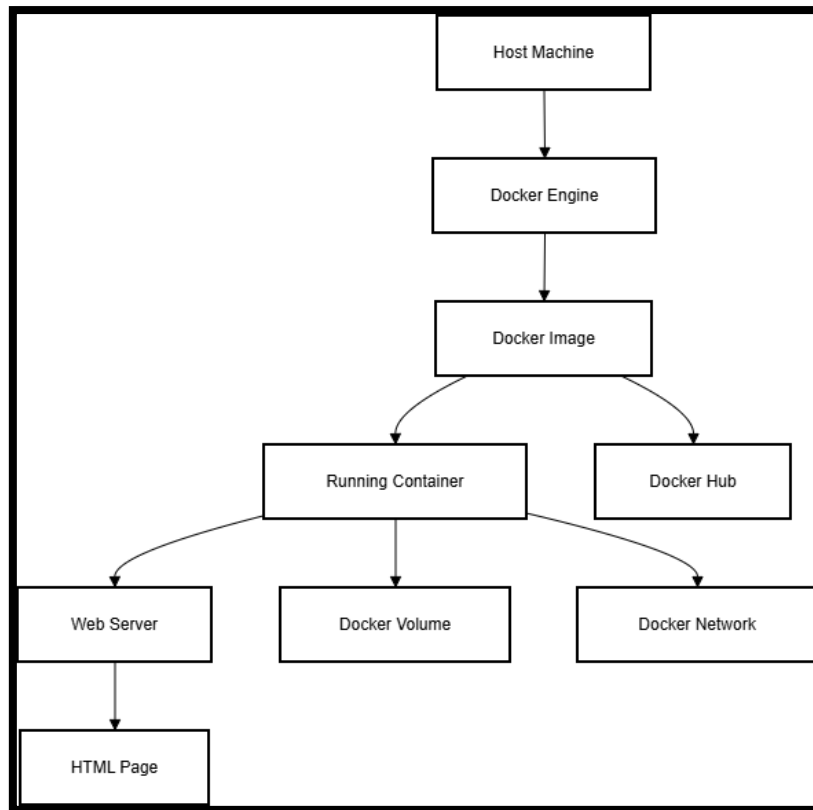


Figure 8: Docker Architecture

(Source: Self-Made using Draw.io)

4. Load Balancing, Scalability, and Application Gateways

4.1 Overview

The basic concepts that help cloud platforms to provide reliable and responsive services are load balancing, scalability and application gateway. Load balancing is the procedure that allocates incoming traffic to a number of application instances to ensure that any of the specific components of the application is not congested. Scalability is the capability of a system to dynamically respond to varying demand by responding dynamically in terms of resource capacity. Application gateways work at a higher layer in the network and give intelligent routing of traffic depending on application-based information (Castellanos-Rodríguez et al., 2024). Collectively, such concepts are vital in clouds where load is elastic and services need to be accessible and scalable in response to user demand on the infrastructure, as well as in response to variation in infrastructure conditions.

4.2 Cloud Provider Approaches

These managed services include load balancing, scalability, and application gateways implemented by major cloud providers to make applications deployment and operation easier. The load balancing service is meant to spread out the traffic received by the network to a number of application instances with the aim of enhancing availability and resiliency. These services work at diverse network levels, and, after all, they both distribute traffic rudimentary and provide more sophisticated request processing according to application needs. Cloud providers have the benefit that a scaling application does not need any human intervention or fancy administration to achieve the scaling of the traffic.

Scalability is most often attained by means of automated scaling processes that scale the number of running application instances depending upon pre-configured metrics like resource usage or request rate. Auto-scaling helps cloud systems to adjust dynamically to new workload requirements to be in a good position when demand soars to high levels and can also help minimise resources when activities are low (Kaur et al., 2024). This is an elastic behaviour that is a unique feature of cloud computing and assists in efficient management of costs in the utilisation of resources.

Application gateways add load balancing features since they operate inside the application layer. They offer specifications like request routing depending on URLs, host name, or protocol allowing finer control of traffic. Application gateways can also be optimised to provide security, performance, including closing client connexions and providing efficient handling of backend traffic. Collectively, these managed services enable the cloud provider to help in providing complex distributed application architectures to the users without them having to directly manage the underlying networking infrastructure (Kithulwatta et al., 2021).

4.3 Relationship Between Load Balancing and Scalability

Load balancing and scalability are two complimentary concepts that interact and ensure both maintainability of application performance and availability. Load balancing is the process by which the incoming traffic is evenly handled in the available resources so that the occurrence of the overloading of individual cases is avoided. The system, through scalability can raise or lower the number of available resources based on demand. In the absence of load balancing, it is impossible to make effective use of additional resources, and in the absence of scalability, load balancing has a fixed capacity. Together these mechanisms enable cloud systems to manage changeable workloads whilst being fault tolerant and service available.

4.4 Combined Benefits to Customers

The combination of load balancing, scalability, and application gateways provides immense customer, who utilises cloud-based services with benefits. Distribution of traffic and allocation of resources dynamically are a means of improving performance to ensure responsiveness is achieved with the high demands. Availability is achieved through decreasing single points of failure and providing automatic recovery when there is a problem in the infrastructure (Lakkadwala and Lakkadwala, 2024). Elastic scaling makes a company cost effective as it ensures that customers pay only the resources they need. The combination of these technologies leads to the creation of a stable and flawless user experience, which supports the importance of cloud computing as a service delivery model.

5. Driving Forces Behind Cloud Computing and Serverless Architectures

5.1 Cost, Flexibility, and Operational Efficiency

The growth of cloud computing has been caused mainly by the cost-effectiveness, adaptability and minimised overhead cost. Cloud computing works under the pay-as-you-go service that enables organisations to only pay as much as they use and not to invest on a fixed infrastructure. Dynamic scaling through the allocation of resources elastically also allows dynamically scaled systems to attain the successful utilisation of computing capacity without reduced performance. As well, cloud services eliminate the aspect of maintenance, updates and availability since the organisational nature of physical hardware and underlying infrastructure is transferred to the provider. This operational efficiency enables teams to be able to concentrate on application development and service delivery as opposed to the need to maintain an infrastructure.

5.2 Serverless Functions vs Containerisation

Another form of cloud computing execution model which is known as serverless computing involves the deployment of applications as functions, which are run only when required. AWS Lambda, Azure Functions, among other services, provide an entirely automated approach to management and scaling of servers and, in doing so, resource allocation. It is a good fit when dealing with event-based and short-lived workloads and where one wants little control over the issue of the execution environment. Containerisation, in its turn is more flexible and controllable since it bundles applications and dependencies in reusable images. Containers feature more operational management than serverless functions do, but have more diverse

workload support. Both methods are concerned with scalability in clouds, as well as cost effectiveness, however, they are applicable based on the needs of the applications.

6. Conclusion

This report has looked at the theme of virtualisation and containerisation in cloud computing which has found relevance in Docker as a solid and extensively used containerisation system. Using an elaborate practical implementation process, some of the concepts of Docker that were illustrated included image creation, container execution, persistent storage, networking and image publishing. The discussion load balancing, scalability and application gateway explained how the performance and availability of the cloud platforms are handled at scale. Besides, a review of cloud adoption drivers and serverless computing strategies highlighted the value of flexibility and efficiency. In general, containerisation is significant to facilitate the development of modern cloud-native applications and scalable service delivery.

7. Video Link :

https://youtu.be/dt1_-4BP8eM?si=qOCrH55mylcqCu0k

8. References

- Ahamed, W.S.S., Zavarsky, P. and Swar, B., 2021, May. Security audit of docker container images in cloud architecture. In *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)* (pp. 202-207). IEEE.
- Ajasa, A.D., Chizari, H. and Alam, A., 2025. Database Security and Performance: A Case of SQL Injection Attacks Using Docker-Based Virtualisation and Its Effect on Performance. *Future Internet*, 17(4), p.156.
- Antonova, V.M., Egorov, M.A., Blinov, V.P., Malikova, E.E. and Malikov, A.Y., 2024, March. Studying the Principles of Infocommunication Network Virtualisation Using the Docker Platform. In *2024 Systems of Signals Generating and Processing in the Field of on Board Communications* (pp. 1-5). IEEE.
- Castellanos-Rodríguez, Ó., Expósito, R.R., Enes, J., Taboada, G.L. and Touriño, J., 2024. Serverless-like platform for container-based YARN clusters. *Future Generation Computer Systems*, 155, pp.256-271.
- Cherukuri, B.R., 2024. Containerization in cloud computing: Comparing Docker and Kubernetes for scalable web applications. *International Journal of Scientific Research and Application*, 13(01), pp.3302-3315.
- Dorđević, B., Kraljević, N. and Kuk, K., 2024, March. Performance of Docker Containerization Tool on Different Linux Distributions. In *2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH)* (pp. 1-6). IEEE.
- Ghandour, O., El Kafhali, S. and Hanini, M., 2023. Computing resources scalability performance analysis in cloud computing data center. *Journal of Grid Computing*, 21(4), p.61.
- Kaur, J., Cordingly, R., Hung, L.H. and Lloyd, W., 2024, June. Towards Low-Cost Global Highly Available Large Container-Based Serverless Functions. In *2024 IEEE Cloud Summit* (pp. 7-12). IEEE.
- Kim, B.S., Lee, S.H., Lee, Y.R., Park, Y.H. and Jeong, J., 2022. Design and implementation of cloud docker application architecture based on machine learning in container management for smart manufacturing. *Applied Sciences*, 12(13), p.6737.

Kithulwatta, W.M.C.J.T., Wickramaarachchi, W.U., Jayasena, K.P.N., Kumara, B.T.G.S. and Rathnayaka, R.M.K.T., 2021. Adoption of docker containers as an infrastructure for deploying software applications: A review. *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*, pp.247-259.

Lakkadwala, A. and Lakkadwala, P., 2024, September. Accelerating Cloud Deployment Through Containerization Technologies. In *2024 International Conference on Advances in Computing Research on Science Engineering and Technology (ACROSET)* (pp. 1-9). IEEE.

Lohumi, Y., Gangodkar, D., Srivastava, P., Khan, M.Z., Alahmadi, A. and Alahmadi, A.H., 2023. Load balancing in cloud environment: A state-of-the-art review. *Ieee Access*, *11*, pp.134517-134530.

Mohammadian, V., Navimipour, N.J., Hosseinzadeh, M. and Darwesh, A., 2021. Fault-tolerant load balancing in cloud computing: A systematic literature review. *IEEE Access*, *10*, pp.12714-12731.

Rabiu, S., Yong, C.H. and Mohamad, S.M.S., 2022. A cloud-based container microservices: A review on load-balancing and auto-scaling issues. *International Journal of Data Science*, *3*(2), pp.80-92.

researchgate.net, 2015. *Hypervisor-based Virtualization Architecture*. Available at: https://www.researchgate.net/figure/Hypervisor-based-virtualization-architecture_fig2_276849142 [Accessed on: 15.12.2025]

researchgate.net, 2023. *Docker Container VS Virtual Machine*. Available at: https://www.researchgate.net/figure/Docker-Container-vs-Virtual-Machine_fig2_369061128 [Accessed on: 15.12.2025]

Rosa, G., Scalabrino, S., Bavota, G. and Oliveto, R., 2023. What Quality Aspects Influence the Adoption of Docker Images?. *ACM Transactions on Software Engineering and Methodology*, *32*(6), pp.1-30.

Saboor, A., Hassan, M.F., Akbar, R., Shah, S.N.M., Hassan, F., Magsi, S.A. and Siddiqui, M.A., 2022. Containerized microservices orchestration and provisioning in cloud computing: A conceptual framework and future perspectives. *Applied Sciences*, *12*(12), p.5793.

Sriram, G.S., 2022. Challenges of cloud compute load balancing algorithms. *International Research Journal of Modernization in Engineering Technology and Science*, 4(1), pp.1186-1190.

Xia, W., Ling, X., Wang, J., Ma, X., Wang, Z., Chen, H., Tao, L., Huang, Y. and Ye, R., 2021, March. Design and Implementation of Docker-based Image Repository Management System of Dispatching and Control Cloud. In *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (Vol. 5, pp. 512-516). IEEE.

Zhou, N., Zhou, H. and Hoppe, D., 2022. Containerization for high performance computing systems: Survey and prospects. *IEEE Transactions on Software Engineering*, 49(4), pp.2722-2740.

Appendix

