



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

Cloud Engineering Principles and Practice Report

Name: Khushi Sangani Navnitbhai

Student ID: 2520615

Course: MSc Cybersecurity and Digital Forensics

Module: Cloud Engineering Principles and Practice

University: University Of Wales, Trinity Saint David

Table of contents		
NO.	Titles	Page No.
	Abstract	3
1	Introduction	4
2	Principles of virtualisation 2.1 What is virtualisation 2.2 Virtualisation of shared hardware 2.3 Limitations of virtualisation	5 to 7
3	Principles and benefits of containerisation 3.1 What is containerisation 3.2 Benefits of containerisation	8 to 9
4	Images and publishing images 4.1 What is a docker image 4.2 Creating a dockerfile 4.3 Building the image 4.4 Running the container 4.5 Publishing the image to dockerhub	10 to 12
5	Container volumes	13
6	Container networking	14
7	Application overview	15
8	Cloud computing concepts 8.1 Load balancing 8.2 Scalability 8.3 Application gateway	16 to 18
9	Load Balancing and Scalability	19
10	Benefits of combining all three technologies	20
11	Driving forces behind cloud computing expansion	21
12	Serverless computing	22
13	Serverless vs Containerisation	23
14	Link to driving forces	24
15	Conclusion	25
16	Reference	26 to 27

Assignment practical link :

<https://youtu.be/yJiPWEEI16o>

Abstract

Cloud computing has transformed the way modern applications are developed, deployed, and managed by providing a scalable, flexible, and cost-efficient computing platform. Among others, this paper examines two key technologies that form the basis of cloud computing—virtualisation and containerisation—and highlights the role of Docker as a major component for successful application deployment .

To illustrate the practical aspects of using containers (Docker), examples such as creating Docker images, publishing them to repositories, managing container volumes, and establishing container networking are presented. In addition, load balancing, scalability, and application gateways, along with how cloud providers use these capabilities to ensure high availability and performance, are also discussed.

1. Introduction

Cloud computing is an important component of modern information systems, providing businesses with the ability to deliver their services and/or applications over the Internet with high levels of flexibility and efficiency. It allows customers to use computing resources without needing to make an initial capital investment in physical hardware, with on-demand access to those resources [1].

In addition to the practical uses of virtualisation and containerisation, such as the creation of Docker images, managing storage for containers, and networking, this study also examines the fundamental principles and benefits of each technology [2].

This paper also discusses three of the main drivers behind the rapid growth of cloud computing: cost effectiveness, flexibility, and scalability. Emerging technologies such as serverless computing are also examined and compared with containerisation through examples such as AWS Lambda [3].

2. Principles of Virtualisation

2.1 What is Virtualisation

Virtualization is the technique according to which it is possible to make virtual copies of real computer resources, such as servers, storage devices, networks, and operating systems. It enhances the use of hardware and efficiency of the system since multiple operating systems and applications can run on the same physical machine [4].

The vital component of the virtualization is the hypervisor or the Virtual Machine Monitor (VMM). The hypervisor controls and allocates resources such as CPU, memory and storage to each of the virtual machines by serving as a software layer between the operating systems and the hardware [4]. As a result, a large number of virtual machines (VMs) can be running simultaneously on the same physical system without affecting each other [5].

Abstraction is one of the basic tenets of virtualization and it is the concept that the physical hardware and the software environment are decoupled. The physical separation allows users and applications to communicate with the virtual resources instead of directly communicating with the physical hardware, which increases the flexibility of the system and makes management easier [6]. A second important principle is isolation, whereby every virtual machine (VM) is independent. This means that failure, crash or security attack in a single VM does not affect other VMs that share the same physical host [4].

In conclusion, virtualization is the process of creating virtual versions of computer resources that enable multiple operating systems and applications to operate efficiently on one physical computer. It is a fundamental technology in the modern computing environment such as cloud computing and corporate data centers [6].

2.2 Virtualisation of Shared Hardware

Virtualization of shared hardware facilitates the use of many operating systems and applications by the efficient sharing of hardware resources of a single physical machine. This is done by a software layer called a hypervisor that monitors and controls the access to the physical hardware, such as CPU, memory, storage, and network interfaces [7].

A given operating system uses all hardware of a normal computing system and this often results in underutilization of resources. By abstracting the hardware and generating numerous virtual machines (VMs), each of which functions as a separate computer system, virtualization gets around this restriction [8]. These virtual computers share actual hardware and have the capability of running numerous operating systems simultaneously.

The hardware is shared although each virtual machine operates in a distinct, secure environment. This enhances the security and reliability of the system since failure or lack of security on one virtual machine (VM) will not affect the use by other VMs on the same system [8].

Even better is shared hardware systems which are supported by hardware-assisted virtualization. Modern processors have virtualization features such as Intel VT-x and AMD-V which reduce the overhead of managing a virtual machine and allow virtual machines to run at near-native speeds [9]. Consequently, the virtualization is more efficient and suitable to high-performance computing environments.

Conclusively, virtualization of shared hardware enables a number of virtualization machines to share a common physical system effectively through abstraction, allocation of resources and isolation. In modern computing systems, it is crucial in the improvement of hardware use, flexibility and scalability [9].

2.3 Limitations of Virtualisation

Although virtualization is a potent technology that increases flexibility and resource efficiency, there are a number of drawbacks that should be taken into account when creating and overseeing virtualized systems.

One major disadvantage of virtualization is that it causes performance overhead. A hypervisor may slightly reduce the performance of a system when compared to executing programs directly on actual hardware due to the extra processing layer that exists between the hardware and the virtual machines [10]. This overhead is more evident when more virtual machines are running at the same time.

Security risk is still another important constraint. Although virtual computers are isolated, a hypervisor becomes a necessity. The systems may be highly vulnerable to the exploitation of the hypervisor which would compromise all virtual machines running on that system posing serious security vulnerabilities [6].

Resource contention is another problem of shared hardware systems. Multiple virtual computers compete over the same physical resources, e.g. CPU, memory, and storage. Inappropriate management of resources can lead to performance reduction and system instability [5].

Moreover, it is challenging to manage with virtualization. The work of multiple virtual machines, the arrangement of hypervisors, and the monitoring of the distribution of resources require skilled administrators and special tools. This increases the complexity of operations compared to the traditional systems [6].

The single point of failure is another drawback. Various virtual machines rely on a single physical host and hypervisor, thus in case of failure of the host system, all the virtual machines running on the host system might be affected [5]. Due to this reason, the importance of backup and disaster recovery plans is essential.

To sum up, virtualization has numerous benefits, and to ensure the correct functioning and safety, the disadvantages, like the overhead of performance, security risks, and resource contention, complexity in administration and dependency on the hypervisor should be properly addressed [4].

3. Principles and Benefits of Containerisation

3.1 What is Containerisation

Applications can operate in separate contexts known as containers thanks to containerization, a lightweight type of virtualization. In order to have the program running everywhere, development, testing and production, the application along with all its dependencies, libraries and configuration files are in each container [11].

Containers do not require different operating systems in each case as with traditional virtual machines. Instead, they maintain isolation of the applications but share the host operating system kernel. Due to this, containerization proves to be better with regard to performance and resource consumption [8].

One of the prime advantages of containerization is portability. There are no compatibility issues since the container will be able to run on any operating system that supports the container engine since the program and its dependencies are packed as a single unit [12]. Another important thing is isolation that enables each of the containers to work separately and does not influence the failures in one container on the other [5].

Containerization has other advantages, such as scalability and fast deployment. Containers are perfect for contemporary cloud computing environments and microservices architectures since they can be started or halted in a matter of seconds [4].

It is critical to the development of scalable applications, DevOps and cloud computing [12].

3.2 Benefits of Containerisation

Containerization is the solution that is suggested to be used in the modern software development, deployment and cloud computing environment because it has numerous advantages. It increases the effectiveness, scalability and consistency of applications across multiple platforms.

One of the key benefits of containerization is that it is lightweight and effective. Containers do not execute their own operating system kernel (as is done with virtual machines), sharing the host operating system kernel, which reduces resource consumption and improves system performance [13]. This enables the use of the same hardware for several applications.

This eliminates any compatibility issues that are commonly faced with the traditional deployment methods [5]. Scalability and quick deployment are more benefits of containerization. By being able to build, start and destroy containers within a few seconds, it is simple to scale an application up or down as needed. It is particularly useful with dynamic workloads and cloud environments [7].

Isolation is another great advantage. Since all the containers work independently, the issues or failures in a single container will not affect the other containers within the same system. This improves stability and reliability of the system [8].

While there are many benefits to containerization, it still requires proper security and management procedures, especially for larger deployments [8]. However due to advantages, it has become a vital part of modern day information technology infrastructure.

In summary, containerization is an important element of cloud computing and DevOps practices as it allows for light-weighted running, mobility, fast deployment, separation, efficient resource utilization, and support for microservices [7]

4. Images and Publishing Images

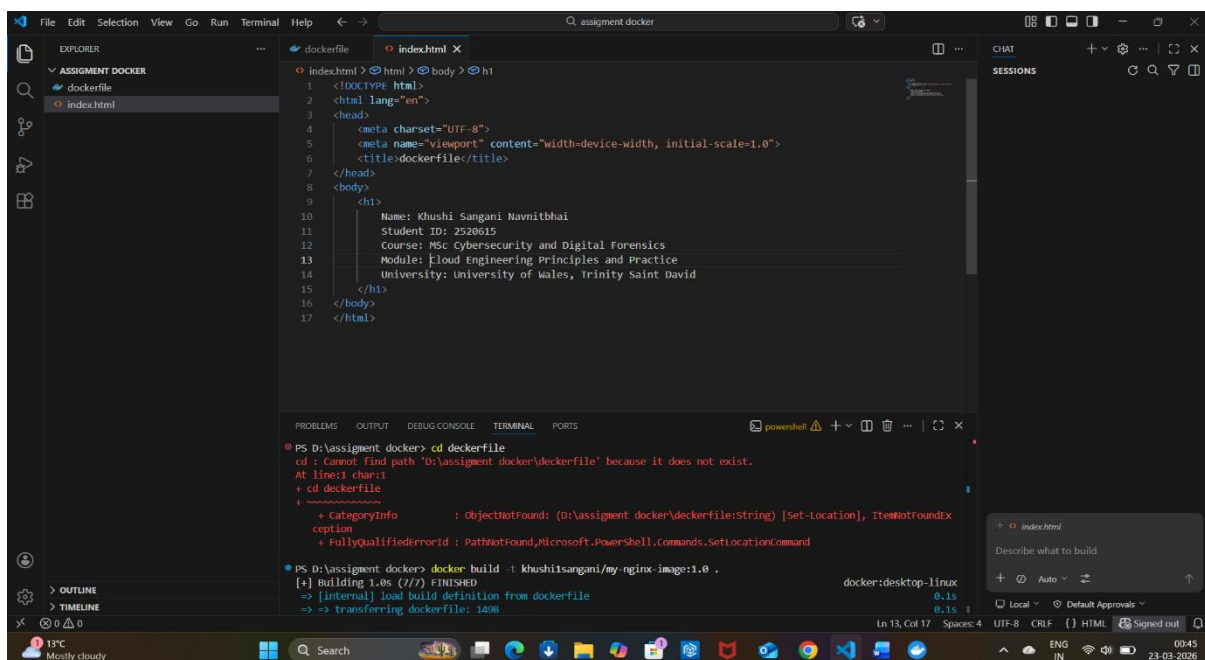
4.1 What is a Docker Image

A Docker image is an essential part of containerisation technology that helps in the efficient deployment of an application. An image is a lightweight, standalone, and executable package that contains all the necessary components to run an application.

The components include the code for the application, system libraries, runtime environment, configuration files, and dependencies. Docker images are used in conjunction with a containerisation platform called Docker to create containers that can function in any environment.

4.2 Creating a Dockerfile

Create a HTML file



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'ASSIGNMENT DOCKER' with files 'dockerfile' and 'index.html'. The main editor displays the 'index.html' file with the following content:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>dockerfile</title>
7 </head>
8 <body>
9   <h1>
10     Name: Khushi Sangani Navnitbhai
11     Student ID: 2520615
12     Course: MSc Cybersecurity and Digital Forensics
13     Module: Cloud Engineering Principles and Practice
14     University: University of Wales, Trinity Saint David
15   </h1>
16 </body>
17 </html>
```

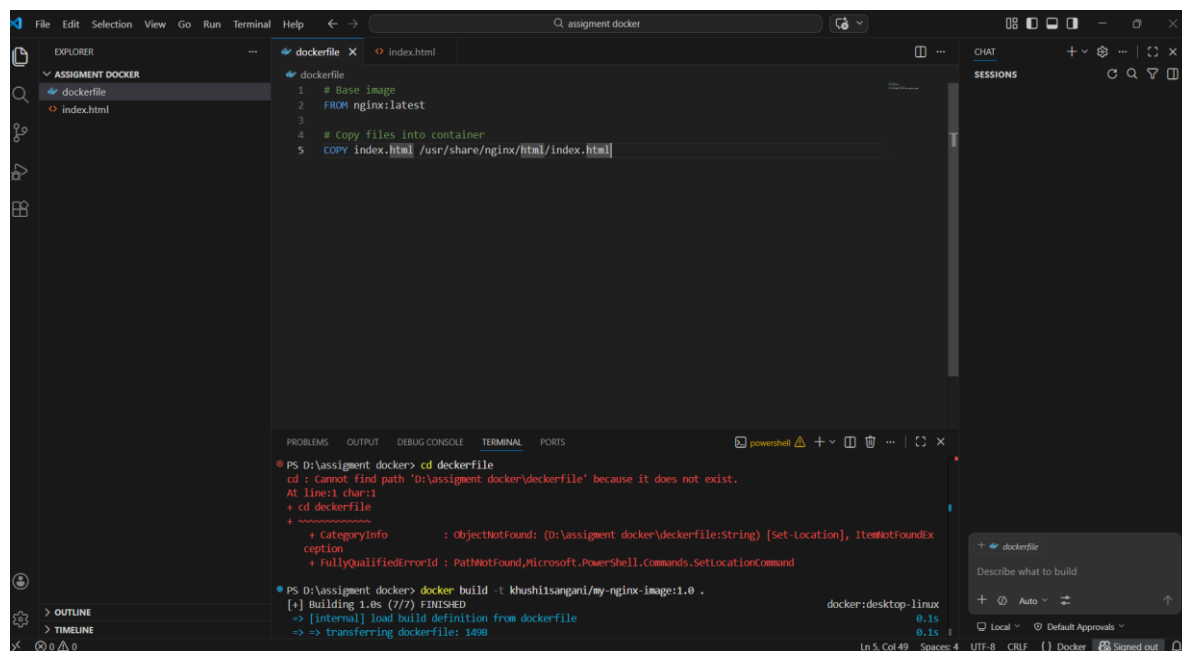
The terminal at the bottom shows the following commands and output:

```
PS D:\assignment docker> cd dockerfile
cd : Cannot find path 'D:\assignment docker\dockerfile' because it does not exist.
At line:1 char:1
+ cd dockerfile
~
+ CategoryInfo          : ObjectNotFound: (D:\assignment docker\dockerfile:String) [Set-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS D:\assignment docker> docker build -t khushisangani/my-nginx-image:1.0 .
[+] Building 1.0s (??? ) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 140B
```

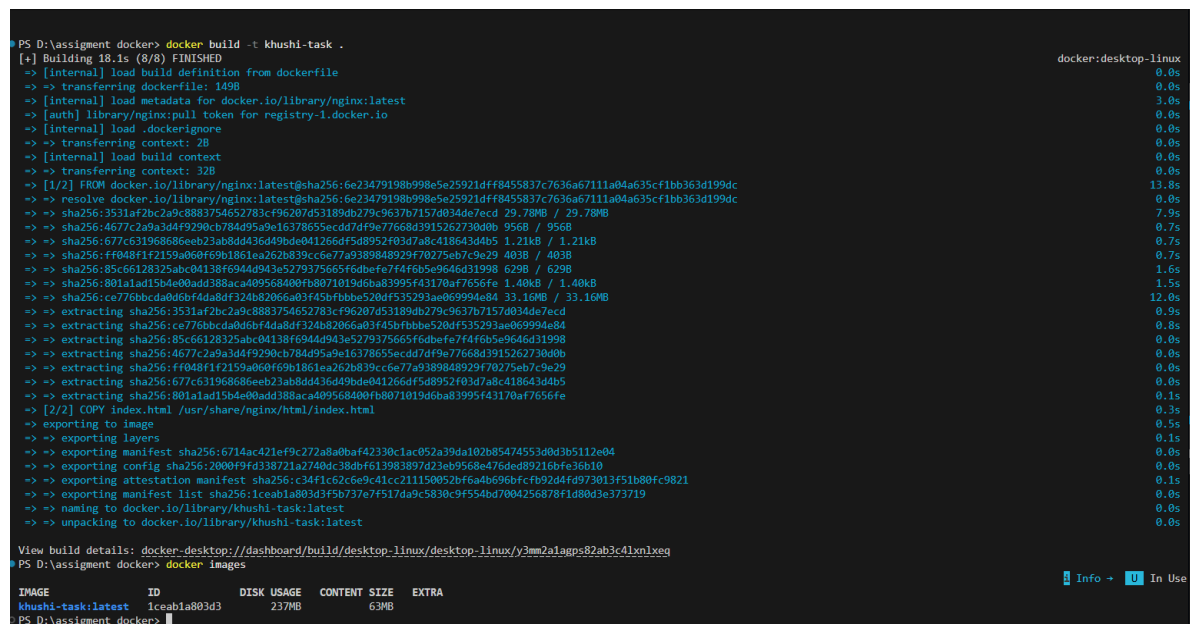
The status bar at the bottom indicates the file is UTF-8, CRLF, and HTML. The system tray shows a temperature of 13°C and the date 23-03-2026.

Create a Dockerfile



4.3 Building the Image

docker images



4.4 Running the Container

docker run -d -p 8080:80 --name my-container my-web-app



Open browser:

http://localhost:8080



Name: Khushi Sangani Navnitbhai
Student ID: 2520615
Course: MSc Cybersecurity and Digital Forensics
Module: Cloud Engineering Principles and Practice
University: University of Wales, Trinity Saint David

4.5 Publishing the Image to DockerHub

```
PS D:\assignment docker> docker tag khushi-task khushi1sangani/khushi-task
PS D:\assignment docker> docker push khushi1sangani/khushi-task
Using default tag: latest
The push refers to repository [docker.io/khushi1sangani/khushi-task]
801a1ad15b4e: Pushed
a284f508acc4: Pushed
ff048f1f2159: Pushed
3531af2bc2a9: Pushed
ce776bbcdad0: Pushed
4677c2a9a3d4: Pushed
677c63196868: Pushed
aa8df8f19d62: Pushed
85c66128325a: Pushed
latest: digest: sha256:1ceab1a803d3f5b737e7f517da9c5830c9f554bd7004256878f1d80d3e373719 size: 856
PS D:\assignment docker>
```

5. Container Volumes

docker volume create my-volume

docker run -d -p 8081:80 --name volume-container -v my-volume:/usr/share/nginx/html nginx

docker volume ls

```
PS D:\assignment docker> docker volume create my-volume
my-volume
PS D:\assignment docker> docker volume create khushi-data
khushi-data
PS D:\assignment docker> docker run -d -p 8081:80 --name volume-container -v my-volume:/usr/share/nginx/html nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
Digest: sha256:6e23479198b998e5e25921dff8455837c7636a67111a04a635cf1bb363d199dc
Status: Downloaded newer image for nginx:latest
648cb8bd3d57e04468bf9cf9bd44cb65707977b06976746c06a5a253f4f65ac7
PS D:\assignment docker> docker volume ls
DRIVER      VOLUME NAME
local       khushi123
local       khushi-data
local       my-volume
local       mydata
PS D:\assignment docker>
```

6. Container Networking

Docker network create network-test

```
PS D:\assignment docker> docker network create network-test
4a8cedd88432e40f1f0a02b1766e10ffc391fda69ca1ed6230fe8dc23f921bf7
```

Docker run -d --name web02 --network network-test nginx

```
PS D:\assignment docker> docker run -d --name web02 --network network-test nginx
af5ec2635ac0c6b55b664e4abc53652453550f781a73e84d93abd4e5c7cbb59a
PS D:\assignment docker> docker run -it --name tester --network network-test busybox
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
481282afbc43: Pull complete
aecba0016ef2: Download complete
Digest: sha256:1487d0af5f52b4ba31c7e465126ee2123fe3f2305d638e7827681e7cf6c83d5e
Status: Downloaded newer image for busybox:latest
/ # ping web02
PING web02 (172.20.0.2): 56 data bytes
64 bytes from 172.20.0.2: seq=0 ttl=64 time=1.206 ms
64 bytes from 172.20.0.2: seq=1 ttl=64 time=0.148 ms
64 bytes from 172.20.0.2: seq=2 ttl=64 time=0.145 ms
64 bytes from 172.20.0.2: seq=3 ttl=64 time=0.140 ms
64 bytes from 172.20.0.2: seq=4 ttl=64 time=0.159 ms
64 bytes from 172.20.0.2: seq=5 ttl=64 time=0.221 ms
^C
--- web02 ping statistics ---
6 packets transmitted, 6 packets received, 0% packet loss
round-trip min/avg/max = 0.140/0.336/1.206 ms
/ # exit
```

7. Application overview

In this assignment, the application of interest is an E-commerce Web Application (Online Shopping System). This application is very popular and evidently illustrates the concept of cloud computing like load balancing, scalability and application gateways.

E-commerce application allows users to navigate through products, place items into a shopping cart and conclude online purchases. Typically, Rephrase It consists of a few elements that include a web interface, application logic and a database system. These elements work together to deliver services to the users through the internet [13].

Users who are customers or administrators of the system do so through a web browser or a mobile application. Their requests are sent to the web frontend which is the user interface and deals with user interaction [14].

This application frontend communicates with the application server, which does business logic of order management, authentication, and payment processing. The application server will then be connected to a database that contains vital information such as product details, customer information and transaction records. This is the division of parts, which increases the organization and scalability of the system [13].

A cloud environment is one where a whole application is hosted on cloud infrastructure, which allows resources to be dynamically allocated as needed. The system is able to handle several users efficiently and maintain high availability using this approach [9].

E-commerce systems are dependent on cloud computing because there have been enormous changes in the level of user traffic. The system should be able to handle a larger traffic without deteriorating performance during peak traffic times like sales events or holidays. At this point, cloud technologies like load balancing and scalability are essential [14].

When summarizing, an e-commerce application is a multi-tier system that contains users, a frontend interface, backend logic, and a database and will be deployed on cloud infrastructure. The architecture is an excellent example of how to demonstrate major concepts of cloud computing [14].

8. Cloud computing concepts

8.1 Load Balancing

One of the concepts of cloud computing is load balancing, which splits the network traffic that enters onto a group of servers or computer resources. Its primary goal is to ensure no single server is overloaded to improve the availability, performance and reliability of the system [15].

A load balancer is an intermediary between back end servers and users in a cloud setup. A request made by a user (say to access a web site) is received by the load balancer which based on a predetermined algorithm (say round-robin, least connection or IP hashing) forwards the request to one of a number of available servers [13]. This will ensure that servers are equally distributed among the load.

Load balancing is necessary to ensure high availability. The load balancer automatically sends the traffic to the other servers that are running in case one of the servers dies or becomes unresponsive. This ensures continuous service that is very important in applications such as online services and e-commerce websites [4].

Another benefit of load balancing is enhanced performance. It ensures that consumers gain faster and reliable access to applications, through distribution of traffic. Moreover, it helps in maximizing resource utilization as there is only efficient utilization of all the servers and not having few idle and many overworked servers [1].

Implementation of load balancing can be done in many different ways, including:

Hardware load balancers are special purpose physical equipment.

Server based applications are software load balancers.

Load balancers are offered by the cloud platforms as cloud-based load balancers.

Cloud providers such as Amazon Web Services offer managed load balancing services (such as Elastic Load Balancer), which automatically allocate traffic and can be connected to other cloud services (such as auto scaling).

To sum up, load balancing plays a vital role in controlling high traffic, performance, ensuring fault tolerance, and maintaining the reliability of the cloud-based systems. It is an essential component in modern cloud designs and works in close coordination with scalability to manage the changing workloads in an efficient way [13].

8.2 Scalability

Scalability is a vital concept in cloud computing, which refers to the ability of a system to cope with varying workloads by adjusting its resources accordingly. It ensures that apps are able to run and be used even when user demand changes radically [16].

Scalability allows systems in clouds to portably allocate resources such as CPU, memory, and storage on demand in real time. This is especially essential in applications such as e-commerce platforms, where traffic of users may peak at certain times such as holidays or sales [4].

Scalability is in two main forms:

1. Vertical scaling (upwards) capability.

Vertical scalability is increasing the ability of a single server by adding additional CPU, RAM, or storage. This approach has shortcomings since only a particular degree of improvement can be achieved on a single machine although it is simple to do so [1].

2. Horizontal scaling (scaling out) capability.

Horizontal scalability is the addition of more servers or instances to share the load. This approach allows systems to deploy enormous traffic volumes across multiple workstations, increasing its flexibility and being more popular in cloud computing [13].

Cloud providers offer automatic scaling capabilities that can adjust resources automatically. Services such as auto scaling e.g. monitor system performance and automatically add or remove instances based on its traffic. This will in addition to reducing costs when the demand is low, ensure that programs are in a position to cope with sudden increases in demand [13].

Scalability has several important benefits. It improves performance by ensuring that it has sufficient resources to respond to user demands. It also enhances the cost efficiency since a business is only charged what resources it uses. Moreover, scalability also promotes high availability ensuring that programs can still operate even in highly demanded periods [10].

To sum up, scalability is essential to modern cloud-based systems as it allows applications to grow and adapt to changing requirements. Scalability plays a vital role in the delivery of reliable and responsive services as it allows managing resources effectively and maintaining the performance of a system [13].

8.3 Application gateway

An application gateway is a network component that controls and manages incoming web traffic at the application layer (Layer 7 of the OSI model). Being a reverse proxy, it receives requests made by clients and according to a pre-defined set of rules, forwards it to the corresponding backend servers [13].

The process of intelligent routing of traffic in cloud systems is handled by an application gateway. Unlike simple load balancers which would evenly distribute traffic, application gateways can route requests based on:

URL routes, e.g. /login and /products.

HTTP headers

Session information

This increases efficiency and performance as different backend services can serve different elements of an application [4].

One of the main features of an application gateway is security. It often includes a Web Application Firewall (WAF) that protects apps against common cyberattacks, such as SQL injection, cross-site scripting (XSS), and other attacks. Due to this, it is important in protecting sensitive applications such as banking and e-commerce [1].

Another important feature is SSL/TLS termination where the application gateway controls data encryption and decryption. This improves the performance of the entire system and reduces the workload on the backend servers [13].

Application gateways also provide session persistence in order to ensure that user requests are always sent to the same backend server when required to do so. This is essential in maintaining user sessions in applications such as online shopping carts [4].

Amazon Web Service (AWS) has application gateways that perform the same functions as an Application Load Balancer (ALB), which combines load balancing with sophisticated routing and security capabilities.

An application gateway also provides improved security, performance, and smart traffic routing for Cloud applications.

9. Cloud provider implementation

Feature	Load Balancing	Scalability
Definition	Distributes incoming traffic across multiple servers	Adjusts system resources to handle workload changes
Main Purpose	Prevents any single server from being overloaded	Ensures system can handle increasing or decreasing demand
Function	Traffic management	Resource management
Focus	Efficient distribution of requests	Increasing or decreasing capacity
Dependency	Requires multiple servers to distribute load	Can involve adding or upgrading servers
Example	Sending user requests to different servers	Adding more servers during high traffic

The relationship between load balancing and scalability.

The concepts of scalability and load balancing are complementary although they are different:

Scalability enables the addition and removal of servers.

Load balancing is used to distribute traffic across those servers.

Without scalability, the capability of load balancing to spread out traffic would be limited. Without load balancing, additional resources provided by scaling would not be put into good use [13].

To sum up, load balancing and scalability play separate, yet closely related roles in cloud computing. Whereas scalability ensures that the resources are adequate to service the demand, load balancing focuses on efficient traffic allocation.

10. Benefits of combining all three technologies

Technology	Individual Role	Combined Benefit
Load Balancing	Distributes traffic across servers	Prevents overload and improves performance
Scalability	Adds/removes resources based on demand	Handles traffic spikes efficiently
Application Gateway	Routes and secures application traffic	Enhances security and intelligent routing

Availability improvement would be one important benefit from this integration. Scalability provides more computing power when required whereas load balancing ensures sharing of workloads among various systems at any given time. The system can still work despite failures or high load because the application gateway ensures proper routing and security [13].

Another benefit of this system would be better efficiency. Scalability ensures there is enough capacity for all users' needs whereas load balancing reduces response time as it evenly distributes traffic. By improving traffic routing, the application gateway improves system performance and speed [17].

In addition to this, it is more secure. Web Application Firewalls (WAF) that protect from web attacks is an element of application firewalls. [1].

They are economical as well. Efficiency reduces costs by limiting resource usage to necessity. Although the application gateway reduces overhead by efficiently managing traffic, load balancing ensures that resources are used optimally [14].

Better customer service also counts as a benefit too. They receive nonstop services with fewer mistakes as well faster responses from us. [13].

Thus, an architecture combining applications gateways with techniques of scaling out as well as load balancers appears to be very reliable, safe for use, easily scaleable alongwith being efficient. These technologies work well together and are needed to provide customers with reliable cloud-based services [1].

11. Driving forces behind cloud computing expansion

One of the primary incentives for cloud computing is its cost efficiency; organizations can now utilize cloud computing without having to invest heavily into the infrastructure of the organization.

Flexibility is another vital characteristic of a cloud platform. Flexibility allows companies to rapidly pivot to meet changing business requirements by providing or removing resources on demand. The ability to adapt quickly fosters faster cycles of development and innovation [32].

Scalability is another important characteristic. Automatic adjustments of resource levels in cloud systems (either up or down) provide the necessary resources to run all applications effectively whether they are under heavy or light load conditions. This feature will be particularly useful in applications that experience sporadic workloads.

The ability to access applications and data from any place where there is an internet connection is another way cloud services provide global accessibility. This makes it easier for people to work remotely and collaborate with people in different countries, both of which have become essential to businesses today [32].

Another important aspect of cloud computing is how reliably and consistently it can deliver its services. Cloud service providers have utilized redundancy strategies and distributed data centers to ensure that their services remain available even if a data centre or hardware fails [31].

Finally, the speed and agility with which businesses can bring products to market has been a dominant influence in determining whether or not organisations will adopt cloud technologies. Since applications can be deployed quickly, businesses can get their products to market much more quickly than if they had to configure complex, on-premises infrastructure [32].

Ultimately, the driving factor behind the growth of cloud computing is its ability to provide businesses with a cost-efficient solution; flexibility; scalability; accessibility; and reliability. Accordingly, cloud computing will remain an indispensable tool for 21st century enterprises and applications [32].

12. Serverless computing

Using this model, developers can entirely devote themselves to developing and deploying code as the cloud provider provides the infrastructure, scaling, and maintenance [18].

Examples of serverless platforms include Microsoft Azure Functions and Amazon Web Services (AWS) Lambda, which allow for execution of event-driven code in response to events like HTTP requests, database changes, and file uploads.

Serverless computing consists of four key features which include:

- Developers don't have to manage servers
- Functions automatically scale up or down depending on usage
- Users are only billed for how long the functions ran
- Functions only execute when there is a trigger to initiate the function

These features make serverless computing cost-effective and efficient [4].

Though there are many advantages associated with serverless computing, there are also some significant drawbacks to consider:

- Cold start latency - time taken for function invocation
- Short execution time
- Less control over the underlying hardware

These factors may render serverless computing impractical for highly tailored or time-consuming applications [4].

In summary, serverless computing represents a modern-day cloud model for running applications, enabling developers with their applications, without having to worry about maintaining any of the physical resources that make up those solutions. Serverless is suitable for building event-driven applications and for building and deploying many other types of modern-day cloud services, as it provides automatic scaling, cost efficiency, and rapid delivery. [13]

13. Serverless vs Containerisation

Feature	Serverless (FaaS)	Containerisation
Definition	Runs code without managing servers	Packages applications with dependencies into containers
Infrastructure	Fully managed by cloud provider	Managed by user (or partially automated)
Scaling	Automatic and instant	Manual or auto-scaling configuration
Cost Model	Pay per execution	Pay for running containers/resources
Control	Limited control over environment	Full control over runtime and environment
Startup Time	Fast (but may have cold starts)	Slightly slower but consistent
Use Case	Event-driven, short tasks	Long-running applications, microservices
Maintenance	Minimal	Requires updates and monitoring

In summary, containerisation and serverless computing are both popular cloud technologies. However, the two have very different functionalities. As such, containerisation is better suited to more involved, longer-lasting applications that require greater control over the resource environment; while serverless computing is better suited to event-driven, temporary workloads that require lower maintenance efforts. The choice between the two ultimately depends on scalability, cost and application needs [19].

14. Link to driving forces

One of the primary motivations behind the explosion of cloud computing is cost effectiveness. This is accomplished through the pay-as-you-go pricing model of serverless computing, meaning you only pay for your resources once you've executed your code, effectively eliminating the cost of keeping servers sitting idle. On the other hand, while containerization does improve efficiency, it still incurs ongoing costs due to having to always be operational [20].

In addition to cost savings, the second major driver for cloud computing is flexibility. Serverless solutions allow developers to rapidly implement functionality by eliminating concerns about underlying infrastructure since development can occur independently from the server environment. However, containerization allows a greater degree of flexibility than serverless platforms because it provides the developer with complete control over their application's dependencies and runtime environment [14].

Scalability is yet another key factor driving the demand for serverless computing. Due to the fact that serverless computing is designed to automatically scale up or down based upon incoming requests, serverless computing is a very viable solution for fluctuating workloads. Containerization does provide some level of scalability, but requires additional orchestration tools or manual/preset methods to actually achieve scalability [19].

Serverless computing will facilitate the rapid deployment of small event-based processes compared to containerization, which is more suitable for deploying complete applications and microservice architectures [21].

The other major difference between maintenance versus management is the level of operational burden placed on the organization. Containerization requires monitoring, upgrading, and orchestrating the management of container environments, which adds to the operational burden of the organization, while serverless computing has less overhead, as the cloud provider is responsible for managing and maintaining the underlying infrastructure [17].

To sum up, both containerization and serverless computing help propel cloud-computing developments via different methods. Containerization is a more dynamic and flexible facility—that also offers precision, manual oversight, and extensive support to assist projects that require complex infrastructures—whereas serverless computing is largely about ease of use, efficient cost management, and the ability to expand automatically. The integration of container-based applications with serverless-based services will enable businesses to develop scalable cloud solutions that are effective and affordable [22].

Conclusion

To summarize, this paper has shown that virtualization and container technology are essential components of developing an efficient and scalable cloud computing platform. Container technology enables portable and lightweight ways to deploy applications using products such as Docker, while virtualization allows multiple systems to effectively share hardware resources.

The examples provided illustrate how virtualization and containerization can be used in practice, including creating and distributing Docker images, controlling volumes, and establishing container networks.

This research also examined a number of essential concepts within cloud computing, such as load balancing, scalability, and application gateways. These concepts all work together to produce high levels of performance, reliability, and security when an organization integrates multiple technologies into its operations.

Considering everything, cloud computing technologies continue to provide sustained benefit for developing and implementing modern applications. Ongoing advancements will play a significant role in shaping the evolution of digital services and IT infrastructure.

Reference

- [1] T. Erl, R. Puttini, and Z. Mahmood, *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall, 2013.
- [2] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment,” *Linux Journal*, no. 239, pp. 2–11, 2014.
- [3] I. Baldini et al., “Serverless Computing: Current Trends and Open Problems,” *Research Advances in Cloud Computing*, Springer, pp. 1–20, 2017.
- [4] P. Barham et al., “Xen and the Art of Virtualization,” *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, Oct. 2003.
- [5] R. P. Goldberg, “Survey of Virtual Machine Research,” *IEEE Computer*, vol. 7, no. 6, pp. 34–45, Jun. 1974.
- [6] W. Stallings, *Operating Systems: Internals and Design Principles*, 9th ed. Harlow, UK: Pearson, 2018.
- [7] K. Adams and O. Agesen, “A Comparison of Software and Hardware Techniques for x86 Virtualization,” *ACM SIGOPS Operating Systems Review*, vol. 40, no. 5, pp. 2–13, Oct. 2006.
- [8] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment,” *Linux Journal*, no. 239, 2014.
- [9] P. Pahl, “Containerization and the PaaS Cloud,” *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May 2015.
- [10] J. Baumgartner, M. P. C. T. A. V. P. D. S. and J. H. H. P. R. J. B. "Performance losses with virtualization: Comparing bare metal to VMs and containers," *iCoSys*, pp. 1–10, 2023.
- [11] C. Pahl, "Containerization and the PaaS Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May–June 2015, doi: 10.1109/MCC.2015.51.
- [12] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, Omega, and Kubernetes,” *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
- [13] M. Armbrust et al., “A View of Cloud Computing,” *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.
- [14] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” NIST Special Publication 800-145, 2011.

- [15] S. Singh and I. Chana, "A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, June 2016, doi: 10.1007/s10723-015-9359-2.
- [16] C. Xue, C. Lin and J. Hu, "Scalability analysis of request scheduling in cloud computing," *Tsinghua Science and Technology*, vol. 24, no. 2, pp. 249–261, April 2019, doi: 10.26599/TST.2018.9010069.
- [17] P. Pahl, "Containerization and the PaaS Cloud," *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, May 2015.
- [18] J. M. Hellerstein et al., "Serverless Computing: One Step Forward, Two Steps Back," *arXiv preprint arXiv:1812.03651*, 2018.
- [19] E. Jonas et al., "Cloud Programming Simplified: A Berkeley View on Serverless Computing," *arXiv preprint arXiv:1902.03383*, 2019.
- [20] Y. Li, Y. Lin, Y. Wang, K. Ye and C. Xu, "Serverless Computing: State-of-the-Art, Challenges and Opportunities," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1522–1539, March-April 2023, doi: 10.1109/TSC.2022.3166553.
- [21] C. Fan, A. Jindal, and M. Gerndt, "Microservices vs Serverless: A Performance Comparison on a Cloud-native Web Application," *Proceedings of the 10th International Conference on Cloud Computing and Services Science*, pp. 204-215, 2020.
- [22] A. Vora, "Serverless computing and container synergy in network architectures," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 22, no. 1, pp. 1–12, Jan. 2024.