

Kalan Final Assignment

by Kalan Hughes

Submission date: 27-Apr-2023 03:10PM (UTC+0100)

Submission ID: 203696634

File name: 17861_Kalan_Hughes_Kalan_Final_Assignment_488474_1957932149.docx (518.65K)

Word count: 6033

Character count: 33333

Assignment 1

Kalan Hughes

1902299

Master of Science: Computing (Software Engineering)

Cloud Engineering Principles and Practice

28 April 2023

Table of Contents

Introduction	3
Task 1: Docker Report	4
1.1 Benefits of Containerisation	4
1.2 Portability.....	5
1.2.1 Construction of a Docker File to Build an Image.....	6
1.3 Images and Publishing Images	7
1.3.1 Pushing an Image to Docker Hub.....	7
1.4 Container Volumes.....	8
1.4.1 Creating a Docker Volume	8
1.5 Container Networking.....	8
1.5.1 Creating a Docker Network	9
1.6 Overall Process.....	10
1.6.1 Building an Image.....	10
1.6.2 Running a Container.....	11
1.6.3 Publishing an Image to Docker Hub.....	12
1.6.4 Providing a Web Server that Displays an HTML Page	13
1.7 Drawbacks of Docker	14
1.8 Conclusion.....	14
Task 2: Video Demonstration.....	15
Task 3: Cloud Computing Concepts	16
3.1 Load Balancing	16
3.2 Scalability	16
3.3 Application Gateways	17
3.4 Cloud Providers Approach (Kubernetes).....	17
3.5 Complementary Concepts.....	18
3.6 Benefits of Combing All Three Technologies	18
3.7 Conclusion.....	18
Task 4: Driving Forces of Cloud Computing	19
4.1 Driving Forces.....	19
4.2 Serverless Functions	19
4.3 Serverless Functions & Containerisation Comparison.....	20
4.4 Conclusion.....	20
Conclusion.....	21
References	22

Introduction

This report is going to contain a variety of information. It will contain a Docker report which will discuss the importance of Docker for running applications effectively in the cloud. It will talk about some important characteristics of Docker and some examples will be given to illustrate the answers. There will also be a link to a YouTube video that demonstrates the examples that were carried out in the Docker report. There will be a section that talks about the three main concepts of cloud computing and the approach that cloud providers can take to implement the concepts. Reasons will be given as to why load balancing and scaling are complementary concepts. An example will be provided which demonstrates the benefits of combining the three technologies together in providing a service to a customer. There will also be a section that talks about the driving forces that are leading to the expansion of cloud computing. There will also be an explanation of serverless functions, and they will be compared with containerisation in identifying the driving forces.

Task 1: Docker Report

This section of the report is going to talk about Docker. It is going to talk about why Docker is an important approach within the cloud. In particular, it is going to talk about the benefits of containerisation, portability, images and publishing images, container volumes, and container networking. There are also going to be examples that are provided to help illustrate the characteristics. The drawbacks of Docker will also be discussed.

1.1 Benefits of Containerisation

The first reason why Docker is an important approach within the cloud is because of containerisation. Containerisation has a variety of different benefits. These benefits are going to be discussed in this section.

The first benefit of containerisation is speed [1]. Speed is a benefit of containerisation because Docker containers allow you to create a master version of an image and it allows you to deploy it quickly on demand. This can help to improve flexibility when you want to create multiple new containerised images. Speed is also a benefit of containerisation because it speeds up the start time as there is no OS that needs to be booted up [1]. This is how containerisation can help to improve speed which is why it is the first benefit of containerisation.

The second benefit of containerisation is improved security [2]. Improved security is a benefit of containerisation because containerisation introduces isolation which provides an additional layer of security [2]. Each container is isolated from the others which means that the applications are running in their own environment that is self-contained [2]. This is a benefit because it means that even if one of the containers is compromised, the other containers on the same host will not be compromised. The containers are also isolated from the host operating system. This means that they can only minimally interact with the computing resources that are available on the operating system [2]. This is how containerisation can help to improve security which is why it is the second benefit of containerisation.

The third benefit of containerisation is scalability [3]. Scalability is a benefit of containerisation because containerisation technology allows for higher application scalability [3]. Containerized applications are able to handle increasing workloads by using a service-oriented application design to reconfigure the existing architecture to enable resources [1]. A container environment enables the addition of new updates, functions, and features instantly without having any effect on the original application [1]. Container technology allows for application scalability with minimum resource usage [3]. This is how containerisation can help to improve scalability which is why it is the third benefit of containerisation.

The fourth benefit of containerisation is easier management [2]. Easier management is a benefit of containerisation because you are able to automate the installation, management,

and scaling of containerised workloads and services by using a container orchestration platform [1]. Kubernetes is one of the most popular containerisation tools that allows you to perform version updates, debugging, logging, and much more [3]. Using a tool like Kubernetes can help to make managing containers much easier than traditional application management methods [2]. This is how containerisation can help to make management easier which is why it is the fourth benefit of containerisation.

The fifth benefit of containerisation is agility [2]. Agility is a benefit of containerisation because it allows for DevOps workflows to be streamlined [2]. This is because it allows for containers to be created and deployed rapidly to any environment [2]. If there is a task that needs to be completed promptly, you can develop a container quickly and with ease to complete the task [2]. If the container is no longer needed, you can shut it down until it is needed again [2]. Developers can continue using DevOps tools and processes to help with rapid application development and enhancements [1]. This is how containerisation can help to improve agility which is why it is the fifth benefit of containerisation.

The sixth benefit of containerisation is flexibility [2]. Flexibility is a benefit of containerisation because it allows the developers to be more versatile when operating their code either in a virtualised or bare-metal environment [2]. Containerisation can meet the demands of deployment regardless of what they are [2]. This means that the containerised applications are flexible enough to change the environment from metal to virtual or vice versa with ease [2]. Microservices allow for containerised applications to be even more flexible as it allows you to host certain elements on bare metal and it allows you to deploy others to virtual cloud environments [2]. This is how containerisation can help to improve flexibility which is why it is the sixth benefit of containerisation.

The final benefit of containerisation that is going to be talked about is efficiency [1]. Efficiency is a benefit of containerisation because it allows developers to share the application layers across containers because the software that is running in containerised environments shares the host machine's operating system kernel [1]. A single host is able to perform many functions because the isolated containers are able to perform their operations without interfering with the other containers [2]. Containers also help to reduce overhead and minimise resources because they use the host operating system's kernel whereas virtual machines rely on their virtual kernel [2]. This is how containerisation can help to improve efficiency which is why it is the seventh benefit of containerisation.

These benefits of containerisation that have been talked about demonstrate the importance of using Docker to run applications efficiently in the cloud.

1.2 Portability

The second reason why Docker is an important approach within the cloud is because of portability. Docker's container-based platform allows for workloads that are highly portable [4]. Portability is an important concept in containerisation. This is because an application container creates an executable software package that is abstracted away from the host operating system [1]. This means that it is not dependent or tied to the host operating

system which makes it portable, and it allows it to run consistently regardless of the platform or cloud [1]. As long as the host operating system supports the tools that are needed for your containerisation, it will be able to deploy with no issues [2]. A container also bundles all of its dependencies which means that you are able to take the application anywhere without having to rebuild it even if it is in a new environment [2]. Docker's portability and lightweight nature can help to make it easier to manage workloads dynamically as applications and services are able to be scaled up or torn down as the needs dictate, in real-time [4]. The portability characteristic demonstrates the importance of using Docker to run applications effectively in the cloud. Below is an example that demonstrates the construction of a Docker file to build an image.

1.2.1 Construction of a Docker File to Build an Image

This section is going to demonstrate the construction of a Docker file to build an image.

The first step is to create a 'dockerfile'. It is important that the dockerfile does not end in '.txt'. This is the content that has been added to the dockerfile.

 dockerfile

```
1 FROM ubuntu
2 RUN apt-get update && apt-get install -y nginx
3 EXPOSE 80
4 ENTRYPOINT ["nginx", "-g", "daemon off;"]
5
```

The second step is to build the image by opening the command prompt from the file containing the dockerfile. Once the command prompt is open, the image can be built by executing the following command.

```
\my-nginx>docker image build -t my-nginx .
```

```
[+] Building 0.3s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [1/2] FROM docker.io/library/ubuntu
=> CACHED [2/2] RUN apt-get update && apt-get install -y nginx
=> exporting to image
=> => exporting layers
=> => writing image sha256:8bb2f8870c019ab4881937dacba78f45050dd2e4af7dd3084661cbbf0e39365c
=> => naming to docker.io/library/my-nginx
```

You can check to see if the image was built correctly by listing all of the images.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker image ls -a
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
my-nginx             latest      8bb2f8870c01  14 hours ago  177MB
```

This concludes the example that demonstrates the construction of a Docker file to build an image.

1.3 Images and Publishing Images

The third reason why Docker is an important approach within the cloud is because of images and publishing images. A Docker image is a file that is used to execute code inside a Docker container [5]. It is a read-only template that contains a set of instructions for creating a container that is able to run on the Docker platform [6]. A docker image contains application code, tools, libraries, dependencies, and other files that are used to run an application [5]. Docker images are a reusable asset which means that the user does not have to recreate the image from scratch which can help to save a lot of time [5]. Docker containers are dependent on images as the containers need a runnable image to exist [5]. Containers also need images to construct runtime environments and to run an application [5].

Docker images are able to be published to a container registry which is also known as a repository [6]. The three main registry types are Docker Hub, third-party registry services, and self-owned registries [6]. Docker Hub is Docker's own registry. Docker Hub allows access to more than 100,000 container images that have been shared by open-source projects, Docker users, and software vendors [6]. Docker Hub can also be used to host and manage your own private images. You can create a repository in Docker Hub which can help to make it easy to share images with your team, customers, or the Docker community [7]. The images and publishing images characteristic demonstrates the importance of using Docker to run applications effectively in the cloud. Below is an example that demonstrates pushing an image to Docker Hub.

1.3.1 Pushing an Image to Docker Hub

This section is going to demonstrate pushing an image to Docker Hub.

To push an image to Docker Hub, you first need to choose which image you want to push. You can do this by listing all of the images.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker image ls
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
my-nginx             latest      8bb2f8870c01  15 hours ago  177MB
```

You then need to tag the image by entering the following command.

```
\my-nginx>docker image tag my-nginx kaland01/my-nginx
```

Once the image has been tagged, it can be pushed to Docker Hub by entering the following command.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker push kaland01/my-nginx
Using default tag: latest
The push refers to repository [docker.io/kaland01/my-nginx]
643372886766: Pushed
b93c1bd012ab: Mounted from library/ubuntu
latest: digest: sha256:838a720dc999e2dc563a00bc64acd5a4927d39d8280af746ffac4cf90293c7b5 size: 741
```

This concludes the example that demonstrates pushing an image to Docker Hub.

1.4 Container Volumes

The fourth reason why Docker is an important approach within the cloud is because of container volumes. Volumes are the preferred mechanism for persisting data that is generated by and used by Docker containers [8]. By default, all of the changes that are made inside the container are lost when the container stops. Docker volumes and bind mounts can help to keep the data between runs [9]. A Docker bind mount is a high-performance connection from the container to one of the directories on the host machine [9]. Bind mounts allow the host to share its own file system with the container which is able to be made read-only or write-only [9]. Docker volumes are native to Docker, unlike bind mounts which use the host file system [9]. The volume has a longer lifecycle than the container which allows it to remain until it is no longer needed [9]. One advantage that volumes have over bind mounts is that they are easier to back up or migrate [8]. Volumes are also a better choice than persisting data in a container's writable layer. This is because a volume does not increase the size of the container that is using it. The volume's content is also able to exist outside of the container's lifecycle [8]. The volumes characteristic demonstrates the importance of using Docker to run applications effectively in the cloud. Below is an example that demonstrates creating a Docker volume.

1.4.1 Creating a Docker Volume

This section is going to demonstrate creating a Docker volume.

To create a docker volume, you need to enter the following command. You also need to give the volume an appropriate name.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker volume create my_volumeA
my_volumeA
```

You can check to see if the volume was created correctly by listing all of the volumes.

```
local      my_volumeA
local      volume1
```

This concludes the example that demonstrates creating a Docker volume.

1.5 Container Networking

The fifth reason why Docker is an important approach within the cloud is because of container networking. Docker networking is used to establish communication between other Docker containers and the outside world via the host machine which is running the Docker daemon [10]. One advantage of Docker networking is that it helps to deliver software quickly [11]. Another advantage of Docker networking is that it helps in application portability. Docker uses network drivers to support networking for its containers [11]. The different types of network drivers are bridge, host, none, overlay, and macvlan [11]. The most common network type is bridge networking [12]. Bridge networks are easy to create,

troubleshoot, and manage as it is limited to containers within a single host that is running the Docker engine [12]. The port mapping needs to be configured in order for the containers on the bridge network to communicate with the outside world [12]. For example, if a Docker container is running on port 80 on a web service, a port on the host system would need to be mapped to port 80 on the container for outside traffic to be able to reach the web service [12]. The networking characteristic demonstrates the importance of using Docker to run applications effectively in the cloud. Below is an example that demonstrates creating a Docker network.

1.5.1 Creating a Docker Network

This section is going to demonstrate creating a Docker network.

To create a docker network, you need to enter the following command. You also need to give the network an appropriate name.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker network create some-network
2004a21208e51f0e11ee0613684fbab9bf06b785600f0dcf2c4a73be097ade4d
```

You can check to see if the network was created correctly by listing all of the networks.

```
2004a21208e5  some-network  bridge  local
```

To test that the network works, you can create two different containers and ping one container from the other.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker container run --network some-network -itd --name unix1 debian
```

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker container run --network some-network --name unix2 debian ping unix1
```

This shows that one container is pinging from the other which means that the network is working correctly.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker exec -ti unix1 ping unix2
PING unix2 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=6.029 ms
64 bytes from 172.18.0.3: seq=1 ttl=64 time=0.148 ms
64 bytes from 172.18.0.3: seq=2 ttl=64 time=0.124 ms
64 bytes from 172.18.0.3: seq=3 ttl=64 time=0.131 ms
64 bytes from 172.18.0.3: seq=4 ttl=64 time=0.120 ms
64 bytes from 172.18.0.3: seq=5 ttl=64 time=0.177 ms
64 bytes from 172.18.0.3: seq=6 ttl=64 time=0.129 ms
64 bytes from 172.18.0.3: seq=7 ttl=64 time=0.136 ms
```

This concludes the example that demonstrates creating a Docker network.

1.6 Overall Process

This section is going to demonstrate the overall process. It is going to demonstrate building an image, running a container, how to publish an image to Docker Hub, and providing a web server that displays an HTML page.

1.6.1 Building an Image

The first step of the overall process is to create a folder where you will store the Dockerfile and any other resources.

 my-nginx

The second step is to create a 'dockerfile'. It is important that the dockerfile does not end in '.txt'. This is the content that has been added to the dockerfile.

 dockerfile

```
1 FROM ubuntu
2 RUN apt-get update && apt-get install -y nginx
3 EXPOSE 80
4 ENTRYPOINT ["nginx", "-g", "daemon off;"]
5
```

The third step is to build the image by opening the command prompt from the file containing the dockerfile. Once the command prompt is open, the image can be built by executing the following command.

```
\my-nginx>docker image build -t my-nginx .
```

```
[+] Building 0.3s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> [1/2] FROM docker.io/library/ubuntu
=> CACHED [2/2] RUN apt-get update && apt-get install -y nginx
=> exporting to image
=> => exporting layers
=> => writing image sha256:8bb2f8870c019ab4881937dacba78f45050dd2e4af7dd3084661cbbf0e39365c
=> => naming to docker.io/library/my-nginx
```

You can check to see if the image was built correctly by listing all of the images.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker image ls -a
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
my-nginx             latest      8bb2f8870c01  14 hours ago  177MB
```

This is how to construct a dockerfile to build an image.

1.6.2 Running a Container

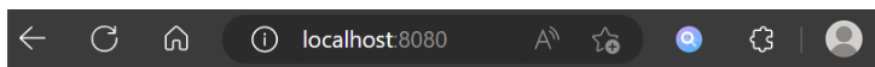
The next step of the overall process is to run the image in a container. This is done by running the following command. This command maps port 80 which is exposed in the dockerfile as port 8080 on the host.

```
\my-nginx>docker container run --name=my-nginx -d -p 8080:80 my-nginx
```

Once the command has been executed, you can check to see if the container is running by entering the following command. This shows that the container is successfully running.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker container ls
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                    NAMES
a0fdbel9613   my-nginx  "nginx -g 'daemon of..." 26 seconds ago Up 23 seconds  0.0.0.0:8080->80/tcp    my-nginx
```

You can test to see if it is working by going to 'localhost:8080' from the browser on the host. This shows that the my-nginx image is successfully running in the container.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

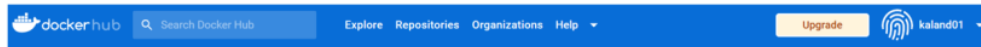
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

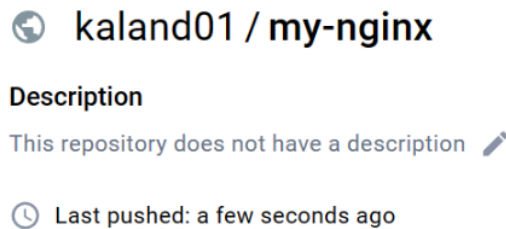
This is how an image is run in a container.

1.6.3 Publishing an Image to Docker Hub

The next step of the overall process is to publish an image to Docker Hub. The first step is to create a Docker Hub account.



The second step is to create a repository. This repository is called 'my-nginx'.



The third step is to push the image to the repository. To do this, you first need to log in.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker login
Authenticating with existing credentials...
Login Succeeded
```

You then need to choose the image that you want to push. This can be done by listing all of the images.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker image ls
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
my-nginx             latest          8bb2f8870c01   15 hours ago   177MB
```

You then need to tag the image by entering the following command.

```
\my-nginx>docker image tag my-nginx kaland01/my-nginx
```

Once the image has been tagged, it can be pushed to Docker Hub by entering the following command.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker push kaland01/my-nginx
Using default tag: latest
The push refers to repository [docker.io/kaland01/my-nginx]
643372886766: Pushed
b93c1bd012ab: Mounted from library/ubuntu
latest: digest: sha256:838a720dc999e2dc563a00bc64acd5a4927d39d8280af746ffac4cf90293c7b5 size: 741
```

You will then be able to see on Docker Hub that the repository now contains 1 tag.

kaland01 / my-nginx

Description

This repository does not have a description 

 Last pushed: a minute ago

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 latest		Image	---	a minute ago

[See all](#)

[Go to Advanced Image Management](#)

This is how you can publish an image to Docker Hub.

1.6.4 Providing a Web Server that Displays an HTML Page

The final part of the overall process is to provide a web server that displays an HTML page. The first step is to create the HTML file that you want to display in the same folder as the dockerfile.

 dockerfile

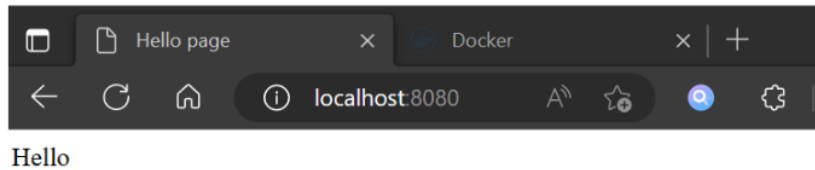
 index.html

```
1 <html>
2 <head>
3   <title>Hello page</title>
4 </head>
5 <body>
6   Hello
7 </body>
8 </html>
9
```

You then need to enter the following command. You need to specify a bind mount using the current host directory. You also need to use 'var/www/html' as the mounting point in the container.

```
C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx>docker run -it -p 8080:80 --name nginx-container-four --volume "C:\Users\kalan\OneDrive\Documents\Uni\Year 4\cloud\my-nginx":/var/www/html my-nginx
```

You can test to see if it working by going to 'localhost:8080' from the browser on the host. This shows that the web server displays the HTML page correctly.



This is how you can provide a web server that displays an HTML page.

This concludes the section that demonstrates the overall process.

1.7 Drawbacks of Docker

This section is going to discuss some of the drawbacks of Docker. The first disadvantage of Docker is that it is difficult to manage a large number of containers [13]. This can make it difficult for the users to get work done as they may find it difficult to manage multiple containers. Another disadvantage of Docker is that it does not provide cross-platform compatibility which means that an application that is designed to run on Windows, will not run on a Linux Docker container [13]. This means that the users would have to develop multiple containers for different platforms. The third disadvantage of Docker is that it does not provide any solution for data backup and recovery [13]. This means that the user will have to create their own solution for data backup and recovery. These are some of the drawbacks of Docker.

1.8 Conclusion

In conclusion, this section of the report has demonstrated with answers and examples the importance of Docker for running applications effectively in the cloud. The report has talked about some of the important characteristics of Docker which shows why it is an efficient way of running applications in the cloud. This concludes the Docker report.

Task 2: Video Demonstration

Video Link:

<https://youtu.be/auV8ytX8UtY>

Task 3: Cloud Computing Concepts

This section of the report is going to contain information regarding the three main concepts in cloud computing. This section is also going to contain a brief description of a cloud provider's approach to implementing the three concepts. There will be a section that discusses why load balancing and scaling are complementary concepts. Finally, an example will be provided which demonstrates the benefits of combining all three concepts together in providing service to a customer.

3.1 Load Balancing

The first main concept of cloud computing is load balancing. Load balancing is a way of distributing workloads and traffic in cloud computing to ensure that no machine or server is overloaded, under-loaded, or idle [14]. Load balancing can help to improve the overall cloud performance by optimizing a variety of constrained parameters such as response time, execution time, and system stability. The goal of load balancing is to minimise the response time for application users and maximise the organisational resources [14]. There are a variety of different types of load-balancing algorithms in cloud computing. Some of these are IP Hash, Round Robin, Least Bandwidth, and Least Response Time [14]. There are a variety of different benefits for load balancing. The first benefit is that it allows for improved flexibility [14]. This is because it allows for traffic to be temporarily routed to alternative servers which would allow for developers to perform updates, maintenance, and tests in a production environment [14]. Another benefit of load balancing is that it is cost-effective [14]. This is because there are accessible to a variety of different business sizes as they run on the cloud, or they can be paid for as a service [14]. This is what load balancing is in cloud computing and the benefits of implementing it.

3.2 Scalability

The second main concept of cloud computing is scalability. Scalability is a way of increasing or decreasing IT resources to meet changing demands [15]. The existing cloud computing infrastructure can be used to scale the processing power, the data storage capacity, and the networking [15]. Scaling can be done with little to no disruption as it can be done quickly and easily. This is because third-party cloud providers already have the infrastructure in place [15]. The reason why a scalable cloud architecture is possible is because of virtualisation [15]. Virtual machines can be easily scaled up or down as they are highly flexible, unlike physical machines whose resources and performance are relatively set [15]. Cloud scalability should be employed in businesses to allow them to grow quickly and meet the changing demands [15]. There are a variety of different benefits for scalability. The first benefit is that allows for improved flexibility and speed [15]. This is because scalability allows businesses to grow and adapt to changing needs which can help to make them more flexible, and they will be able to carry out work faster. Another benefit of scalability is the cost savings [15]. This is because businesses can avoid the upfront cost of purchasing new expensive equipment as they would only have to pay cloud providers for what they use which can help to save money [15]. This is what scalability is in cloud computing and the benefits of implementing it.

3.3 Application Gateways

The third main concept of cloud computing is application gateways. Application gateways are a way of letting users gain access to traditional web applications that are hosted in corporate data centres by using the same methods and login credentials that they would use to access cloud services and mobile apps [16]. Application gateways are an important concept as they allow businesses to improve user satisfaction by unifying and simplifying remote access security [16]. They also help to reduce the expenses and the complexity of IT operations as application gateways eliminate the need for expensive and complex VPNs [16]. Application gateways are easy to install and set up unlike most VPN appliances [16]. The application gateways run on industry-standard servers as they are software-based solutions [16]. Application gateways allow users to sign on via a secure web portal which means that no special endpoint software is required [16]. There are a variety of different benefits for application gateways. The first benefit is that it allows for a better user experience [16]. This is because users are able to access all of their services and application from any device or location by using a single set of login credentials [16]. Another benefit is that it simplifies operations [16]. This is because there is no need for complex VPNs, endpoint software support, or any help desk interactions. This is what application gateways are in cloud computing and the benefits of implementing them.

3.4 Cloud Providers Approach (Kubernetes)

This section is going to give a brief description of a cloud provider's approach to implementing load balancing, scalability, and application gateways. The approach that is going to be talked about is Kubernetes. Kubernetes is an approach that would allow cloud providers to implement load balancing. This is because Kubernetes offers a type of load balancing that is called load distribution [17]. Kubernetes uses two different methods of load distribution [17]. Both of the methods are easy to implement at the dispatch level and they operate through the kube-proxy feature. Kubernetes is an approach that would allow for scalability. This is because Kubernetes allows users to horizontally scale the total containers that are used based on the requirements or needs which could change over time [18]. The total containers that are being used can be easily changed via the command line or the Horizontal Pod Autoscaler [18]. Kubernetes is an approach that would allow for application gateways. This is because the Application Gateway Ingress Controller is a Kubernetes application which allows the Azure Kubernetes Service users to leverage Azure's native Application Gateway L7 load-balancer to expose cloud software to the Internet [19]. This is how cloud providers can use Kubernetes as an approach to implement load balancing, scalability, and application gateways.

3.5 Complementary Concepts

This section is going to talk about why load balancing and scaling are complementary concepts. One reason why they are complementary concepts is that applications are typically scaled based on the load-balancing serving capacity [20]. Another reason why they are complementary concepts is that they both reduce backend tasks like managing the traffic load of servers, monitoring the health of servers, and increasing or reducing servers based on requirements [20]. The final reason why they are complementary concepts is that scalability is the main advantage of load balancing [21]. These are some of the reasons why load balancing and scalability are complementary concepts.

3.6 Benefits of Combining All Three Technologies

This section is going to provide an example that demonstrates the benefits of combining all three technologies together in providing a service to a customer. An example that demonstrates the benefits is if a customer had a website that experiences high traffic at random periods. By providing load balancing and scalability to the customers' website, the website will be able to automatically allocate more resources to deal with the high traffic. This will help to prevent the website from being slow which can help to ensure that the website continues to be fast even when there is a lot of traffic. By providing an application gateway to the customers' website, users will be able to get access to different versions of the website depending on the device that they are using. The benefit of combining all three technologies within the customers' website is that the website will run faster regardless of traffic, and it will also result in a better user experience.

3.7 Conclusion

In conclusion, load balancing, scalability, and application gateways are very important concepts which complement each other. Cloud providers can use Kubernetes as an approach to implement these concepts when providing a customer with a service to improve the quality of the service.

Task 4: Driving Forces of Cloud Computing

This section of the report is going to contain information regarding the driving forces leading to the expansion of cloud computing. This section will also talk about serverless functions, and they will be compared and contrasted with containerisation in identifying the driving forces.

4.1 Driving Forces

This section is going to discuss some of the driving forces leading to the expansion of cloud computing. The first driving force is cost savings [22]. This is because you do not have to pay for storage, hardware, and network devices like an on-premises data centre as the cloud model means that you only have to pay for what you use which can help to save a lot of money [22]. The second driving force is flexibility and scalability [22]. This is because the cloud allows for more flexibility and scalability due to the fact that you can purchase more storage and you will have instant access to expanded capabilities [22]. The third driving force is rapid recovery [22]. This is because if you were to store your data on-premises, you would need to have backups of the data in another location to ensure that the data is not lost in the event of a disaster. Whereas with cloud backup, the data is stored across multiple storage centres so even if one of the centres went through a disaster, the data will still be able to be accessed through the other backups and storage centres [22]. The fourth driving force is security [22]. This is because cloud computing can help to make businesses more secure from cyber-attacks [22]. This means that by using cloud computing, the business will be able to protect its data more easily than if they do not use cloud computing. The fifth driving force is increased convenience [22]. This is because users are able to easily access the necessary files regardless of where they are stored [22]. This is useful as it can save the users a lot of time as they do not have to be on the premises to gain access to the file. The sixth driving force is efficiency [22]. This is because cloud computing can help to remove any unnecessary steps to help streamline the process [22]. This can help to improve the users' productivity which can result in the customers receiving their service faster. These are some of the driving forces that are leading to the expansion of cloud computing.

4.2 Serverless Functions

This section is going to explain serverless functions. A serverless function is a programmatic function that is written by a software developer for a single purpose [23]. Cloud computing companies then use their infrastructure to host and maintain the serverless function [23]. This allows developers to deploy new code easier and faster because the cloud companies take care of the code maintenance and execution [23]. Serverless functions are good as it means that the users do not have to maintain and scale the servers to fit their needs because the companies automatically add server capacity when you need it, and they remove it when you do not need it [23]. One benefit of serverless functions is that it is cost-effective [23]. This is because most companies offer a pay-as-you-go service. This means

that you only have to pay for the resources that you use which can help to save a lot of money overall [23]. This is what serverless functions are and how they are beneficial.

4.3 Serverless Functions & Containerisation Comparison

This section is going to compare and contrast serverless functions and containerisation in identifying the driving forces. One difference between serverless functions and containerisation is the cost which is one of the driving forces [24]. This is because containers are always running which means that the cloud providers need to charge for the server space even if they are not being used. Whereas serverless does not have any continued expenses as the application code does not run unless called [24]. This means that developers are only charged for the server capacity that their application uses [24]. Another difference between them is scalability which is also one of the driving forces [24]. This is because the number of containers that are being deployed is determined in advance. Whereas the backend in a serverless architecture inherently and automatically scales up to meet the needs [24]. Another difference between them is maintenance [24]. This is because developers have to update and manage each container they deploy as the cloud providers do not maintain or update them. Whereas in a serverless architecture, there is no backend to manage. This is because the providers manage the maintenance and updates for the servers that run the code [24]. One similarity between them is that they are both cloud-based which means that they both do a good job of reducing the infrastructure overhead which is one of the driving forces [24]. Another similarity is that they both break down and deploy applications into smaller components [24]. These are some similarities and differences between serverless functions and containerisation in identifying the driving forces.

4.4 Conclusion

in conclusion, there are a variety of different driving forces that are leading to the expansion of cloud computing. There are also a variety of similarities and differences between serverless functions and containerisation in identifying these driving forces.

Conclusion

In conclusion, there are many different reasons why Docker is an important approach for running applications effectively in the cloud. The three main concepts in cloud computing are also important and they are complementary. Cloud providers should implement the concepts into customers' websites to improve the user experience. There are many different driving forces leading to the expansion of cloud computing. This concludes the report regarding cloud computing.

References

- [1 "What is Containerization? What are the Benefits?," [Online]. Available:
] <https://www.veritas.com/en/uk/information-center/containerization>. [Accessed 17 April 2023].
- [2 R. Powell, "circleci.com," 6 April 2023. [Online]. Available: <https://circleci.com/blog/benefits-of-containerization/>. [Accessed 17 April 2023].
- [3 Veritis, "7 Key Containerization Benefits for Your IT Business," [Online]. Available:
] <https://www.veritis.com/blog/7-key-containerization-benefits-for-your-it-business/>. [Accessed 17 April 2023].
- [4 "Docker overview," [Online]. Available: <https://docs.docker.com/get-started/overview/#:~:text=Docker's%20container%2Dbased%20platform%20allows,in%20a%20mixture%20of%20environments..> [Accessed 17 April 2023].
- [5 A. S. Gillis, "Docker image," May 2021. [Online]. Available:
] <https://www.techtarget.com/searchitoperations/definition/Docker-image>. [Accessed 17 April 2023].
- [6 E. Kisller, "A Beginner's Guide to Understanding and Building Docker Images," 22 January 2023.
] [Online]. Available: <https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/>. [Accessed 17 April 2023].
- [7 "Overview," [Online]. Available: <https://docs.docker.com/docker-hub/>. [Accessed 17 April 2023].
]
- [8 "Volumes," [Online]. Available: <https://docs.docker.com/storage/volumes/>. [Accessed 17 April 2023].
]
- [9 A. Frieze, "Guide to Docker Volumes," 3 August 2022. [Online]. Available:
] <https://www.baeldung.com/ops/docker-volumes>. [Accessed 17 April 2023].
- [1 A. Choudhary, "Understanding Docker Networking," 17 April 2023. [Online]. Available:
0] <https://earthly.dev/blog/docker-networking/>. [Accessed 17 April 2023].
- [1 S. Afreen, "Introduction To Docker Networking: Advantages and Working," 1 February 2023.
1] [Online]. Available: <https://www.simplilearn.com/tutorials/docker-tutorial/docker-networking>.
[Accessed 17 April 2023].
- [1 "Docker Networking," [Online]. Available: <https://www.aquasec.com/cloud-native-academy/docker-container/docker-networking/>. [Accessed 17 April 2023].
2]
- [1 D. Jamil, "Docker-An Overview/Pros and Cons," 23 April 2022. [Online]. Available:
3] <https://www.linkedin.com/pulse/docker-an-overviewpros-cons-danish-jamil?trk=pulse-article>.
[Accessed 18 April 2023].
- [1 "Load Balancing In Cloud Computing," [Online]. Available:
4] [https://avinetworks.com/glossary/load-balancing-in-cloud-computing/#:~:text=Load%20balancing%20in%20cloud%20computing%20distributes%20traffic%](https://avinetworks.com/glossary/load-balancing-in-cloud-computing/#:~:text=Load%20balancing%20in%20cloud%20computing%20distributes%20traffic%20)

20and%20workloads%20to,to%20improve%20overall%20cloud%20performance.. [Accessed 19 April 2023].

[1 "What is Cloud Scalability?," [Online]. Available:

5] <https://www.vmware.com/topics/glossary/content/cloud-scalability.html#:~:text=Cloud%20scalability%20in%20cloud%20computing,its%20exploding%20popularity%20with%20businesses..> [Accessed 19 April 2023].

[1 "Application (App) Gateway," [Online]. Available: [https://www.cyberark.com/what-is/app-6\] gateway/](https://www.cyberark.com/what-is/app-6] gateway/). [Accessed 19 April 2023].

[1 "Kubernetes Load Balancer Definition," [Online]. Available:

7] <https://avinetworks.com/glossary/kubernetes-load-balancer/#:~:text=In%20other%20words%2C%20Kubernetes%20services,through%20the%20kub e%2Dproxy%20feature..> [Accessed 19 April 2023].

[1 B. Graves, "Creating Scalable and Fault-Tolerant Applications using Kubernetes," [Online].

8] Available: <https://www.cprime.com/resources/blog/creating-scalable-and-fault-tolerant-applications-using-kubernetes/#:~:text=Kubernetes%20allows%20users%20to%20horizontally,which%20may%20ch ange%20over%20time..> [Accessed 19 April 2023].

[1 "What is Application Gateway Ingress Controller?," 16 March 2023. [Online]. Available:

9] <https://learn.microsoft.com/en-us/azure/application-gateway/ingress-controller-overview>. [Accessed 19 April 2023].

[2 "Auto Scaling," [Online]. Available: [https://avinetworks.com/glossary/auto-](https://avinetworks.com/glossary/auto-scaling/#:~:text=Auto%20scaling%20and%20load%20balancing,shapes%20the%20auto%20scal in g%20policy..)

0] [scaling/#:~:text=Auto%20scaling%20and%20load%20balancing,shapes%20the%20auto%20scal in g%20policy..](https://avinetworks.com/glossary/auto-scaling/#:~:text=Auto%20scaling%20and%20load%20balancing,shapes%20the%20auto%20scal in g%20policy..) [Accessed 19 April 2023].

[2 M. Conran, "Load Balancer Scaling," 26 February 2015. [Online]. Available: [https://network-](https://network-1] insight.net/2015/02/26/load-balancing-and-scale-out-architectures/#:~:text=Scalability%20is%20the%20main%20advantage,stations%20based%20on %20an%20algorithm..)

1] [insight.net/2015/02/26/load-balancing-and-scale-out-architectures/#:~:text=Scalability%20is%20the%20main%20advantage,stations%20based%20on %20an%20algorithm..](https://network-insight.net/2015/02/26/load-balancing-and-scale-out-architectures/#:~:text=Scalability%20is%20the%20main%20advantage,stations%20based%20on %20an%20algorithm..) [Accessed 19 April 2023].

[2 F. Tech, "Top 10 Business Drivers for Cloud Transformation," 29 March 2022. [Online]. Available:

2] <https://www.flentas.com/Top-10-Business-Drivers-for-Cloud-Transformation>. [Accessed 20 April 2023].

[2 A. Fitzgerald, "Serverless Functions: Your Website's New Best Friend," 27 October 2021. [Online].

3] Available: <https://blog.hubspot.com/website/serverless-functions>. [Accessed 21 April 2023].

[2 "Serverless computing vs. containers | How to choose," [Online]. Available:

4] <https://www.cloudflare.com/learning/serverless/serverless-vs-containers/>. [Accessed 21 April 2023].

Kalan Final Assignment

GRADEMARK REPORT

FINAL GRADE

84/100

GENERAL COMMENTS

Instructor

Docker Overview + Benefits (20): 17

Docker Examples (30): 28

Video (5): 5

Cloud Comp tech (25): 21

Cloud Comp functions (20): 13

Feedback:

=====

Very good discussion of docker benefits with good examples. An overview of docker would have helped illustrate the benefits. Would have liked one of the examples to have been a little more interesting.

Quite interesting discussion of App Gateway, but one of the main benefits is that it hides the internal infrastructure from the public by providing a single point of entry. An example of how this technology could be used, would have been nice.

The factors behind the expansion of the cloud could have been elaborated upon a bit more. Also, sometimes containerisation is more appropriate than serverless, and I would have liked some discussion around when each approach is more appropriate.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23
