

University of Wales Trinity Saint David

LONDON

Department of Cyber Security and Digital Forensics Master's Programme

Module: ACCA7027 – Cloud Engineering Principles and Practice

Assignment Title: Cloud Computing, Docker Containerisation and Serverless Functions

Submitted by: Abdullah Al Mamun

Submitted To: Renuga Jayakumar

Student ID: 2434327

Submission Date: 18 December 2025

Table of Contents

- 1. Introduction**
- 2. Principles of Virtualisation**
 - 2.1 Virtualisation of Shared Hardware
- 3. Principles and Benefits of Containerisation**
 - 3.1 Benefits of Containerisation in the Cloud
- 4. Constructing a Docker Image Using a Dockerfile**
 - 4.1 Dockerfile and Application Setup
 - 4.2 Docker Volumes
 - 4.3 Container Networking
- 5. Load Balancing, Scalability, and Application Gateways**
 - 5.1 Load Balancing
 - 5.2 Scalability
 - 5.3 Application Gateways
- 6. Relationship Between Load Balancing and Scalability**
- 7. Benefits of Combining Load Balancing, Scalability, and Application Gateways**

- 8. Cloud Providers and Container-Based Applications**
 - 9. Summary of Key Concepts**
 - 10. Motivational Factors for Cloud Computing Growth**
 - 10.1 Cost Efficiency
 - 10.2 Flexibility and Agility
 - 10.3 Reliability and Global Reach
 - 11. Serverless Computing (Functions as a Service)**
 - 11.1 Benefits of Serverless Computing
 - 12. Comparison of Serverless Computing and Containerisation**
 - 12.1 Containerisation
 - 12.2 Serverless Computing
 - 12.3 Identifying the Driving Forces
 - 13. Challenges and Limitations of Containerisation in Cloud Environments**
 - 14. Critical Discussion**
 - 15. Dockerhub Demonstration Video Link:**
 - 16. Conclusion**
 - 17. References**
-

1. Introduction

Cloud computing is now a foundational model for internet-based computation and a core component of contemporary IT environments. Organisations increasingly use cloud platforms to host applications, store data, and deliver services in a scalable and cost-efficient manner. Traditional on-premises systems often lack the flexibility, speed, and financial benefits offered by cloud solutions, which explains the widespread shift toward cloud infrastructure.

A key innovation in this space is containerisation, particularly through Docker. Containers package an application with its dependencies, ensuring consistent operation across environments. This approach addresses issues like inconsistent deployments, poor scalability, and supports modern DevOps and continuous integration practices.

This report explores the role of containerisation in cloud computing, demonstrating its practical value through Docker examples. It begins with virtualisation fundamentals, moves to containerisation, and covers practical tasks such as Dockerfile creation, image management, volumes, and networking. These illustrate how containers enable efficient application delivery in the cloud.

2. Principles of Virtualisation

Virtualisation is the cornerstone of cloud computing, enabling abstraction of physical hardware to support multiple virtual systems on a single machine. A hypervisor manages resource allocation—CPU, memory, storage—allowing independent virtual machines (VMs) to run separate operating systems and applications. This isolation improves security and resource efficiency while enabling rapid provisioning and scaling, essential for dynamic cloud platforms.

2.1 Virtualisation of Shared Hardware

A major benefit of virtualisation is efficient sharing of hardware across VMs. Rather than dedicating a physical server per application, multiple VMs utilise the same hardware, reducing costs and improving utilisation. This supports pay-per-use models in cloud data centres. However, VMs are resource-intensive due to requiring full OS instances, which has driven interest in lighter alternatives like containers.

3. Principles and Benefits of Containerisation

Containerisation virtualises at the OS level, with containers sharing the host OS kernel while maintaining isolation. Docker is the leading platform. Containers bundle apps with libraries and configurations, ensuring consistent behaviour across environments. This eliminates compatibility concerns and supports portable, scalable deployments.

3.1 Benefits of Containerisation in the Cloud

- **Efficiency:** Lower resource use than VMs, as containers share the host OS.
- **Portability:** Easily moved across environments and cloud types.
- **Scalability:** Quick creation and removal supports dynamic scaling.
- **Consistency:** Same image runs in development, testing, and production.

These advantages make containerisation vital for cloud-native development.

4. Constructing a Docker Image Using a Dockerfile

A Dockerfile is a script with instructions to build a Docker image. Images serve as templates for containers, enabling repeatable deployments.

Example Dockerfile:

text

FROM nginx:alpine

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

EXPOSE 80

This uses a lightweight Alpine-based Nginx image and copies a custom HTML file to serve as the default page.

4.1 Dockerfile and Application Setup

The Dockerfile and application files must reside in the same directory. Ensure Docker Desktop is running in the correct working directory before building the image. The build process outputs detailed logs, helping verify each step and troubleshoot issues.

4.2 Docker Volumes

Volumes persist data beyond container lifecycle, preventing data loss when containers stop or are replaced. They allow data sharing between containers and simplify backups, which is crucial for stateful apps like databases.

4.3 Container Networking

Containers are isolated by default; networking must be configured to allow communication between containers, microservices, or external users. Custom Docker networks improve control, organisation, and inter-service connectivity, supporting scalable and secure cloud architectures.

5. Load Balancing, Scalability, and Application Gateways

5.1 Load Balancing

Distributes traffic across resources to prevent overload and improve availability, using managed services like AWS ELB or Azure Load Balancer.

5.2 Scalability

Enables dynamic adjustment of resources via vertical (more power per instance) or horizontal (more instances) scaling, often automated in cloud platforms.

5.3 Application Gateways

Act as intelligent entry points, providing SSL termination, routing, and firewall protection at the application layer.

6. Relationship Between Load Balancing and Scalability

Load balancing and scalability are complementary. Scaling adds resources, while load balancing distributes traffic across them. Together, they ensure high availability and performance under varying loads.

7. Benefits of Combining Load Balancing, Scalability, and Application Gateways

Integration improves availability, security, and latency management. It allows cloud providers to deliver responsive, secure services that adapt to demand.

8. Cloud Providers and Container-Based Applications

Major clouds offer managed container services like AWS ECS, Azure AKS, and Google GKE, which automate orchestration, networking, and scaling, letting developers focus on application logic.

9. Summary of Key Concepts

Containerisation integrates with cloud fundamentals—networking, deployment, load balancing, and scalability—proving essential to modern cloud architecture. The next sections discuss cloud adoption drivers and serverless computing.

10. Motivational Factors for Cloud Computing Growth

10.1 Cost Efficiency

Pay-as-you-go models reduce upfront investment and idle capacity, benefiting businesses of all sizes.

10.2 Flexibility and Agility

Rapid resource provisioning supports changing workloads and accelerates development cycles.

10.3 Reliability and Global Reach

Built-in redundancy and multi-region deployments ensure high availability and low-latency access worldwide.

11. Serverless Computing (Functions as a Service)

Serverless or FaaS (e.g., AWS Lambda, Azure Functions) lets developers write event-triggered functions without managing servers. Providers handle infrastructure, scaling, and maintenance.

11.1 Benefits of Serverless Computing

- **No server management**
- **Automatic, fine-grained scaling**
- **Cost-effective for sporadic workloads**
- **Reduced operational overhead (Forrester, 2023)**

12. Comparison of Serverless Computing and Containerisation

12.1 Containerisation

Offers greater control over the runtime environment, suitable for long-running, complex, or stateful applications. Requires orchestration tools like Kubernetes, introducing management complexity (CNCF, 2024).

12.2 Serverless Computing

Ideal for event-driven, short-lived tasks with variable load. Limitations include cold-start latency and less environmental control (IEEE Cloud Computing, 2023).

12.3 Identifying the Driving Forces

Both models support cloud adoption by improving cost efficiency, agility, and operational simplicity. Hybrid approaches are increasingly common (IDC, 2024).

13. Challenges and Limitations of Containerisation in Cloud Environments

- **Security:** Shared kernel vulnerabilities require rigorous image scanning and runtime protection (Snyk, 2024).
- **Orchestration Complexity:** Kubernetes management demands specialised skills.
- **Storage Persistence:** Requires careful design with volumes or cloud-native storage.
- **Performance Overhead:** Poorly optimised images can reduce efficiency.
- **Vendor Lock-in:** Despite portability, cloud-specific services may limit migration (TechTarget, 2023).

14. Critical Discussion

Containers and serverless represent complementary approaches: containers offer control and consistency for complex systems, while serverless provides agility for event-driven workflows. The future lies in hybrid architectures that leverage both paradigms (GigaOm, 2024).

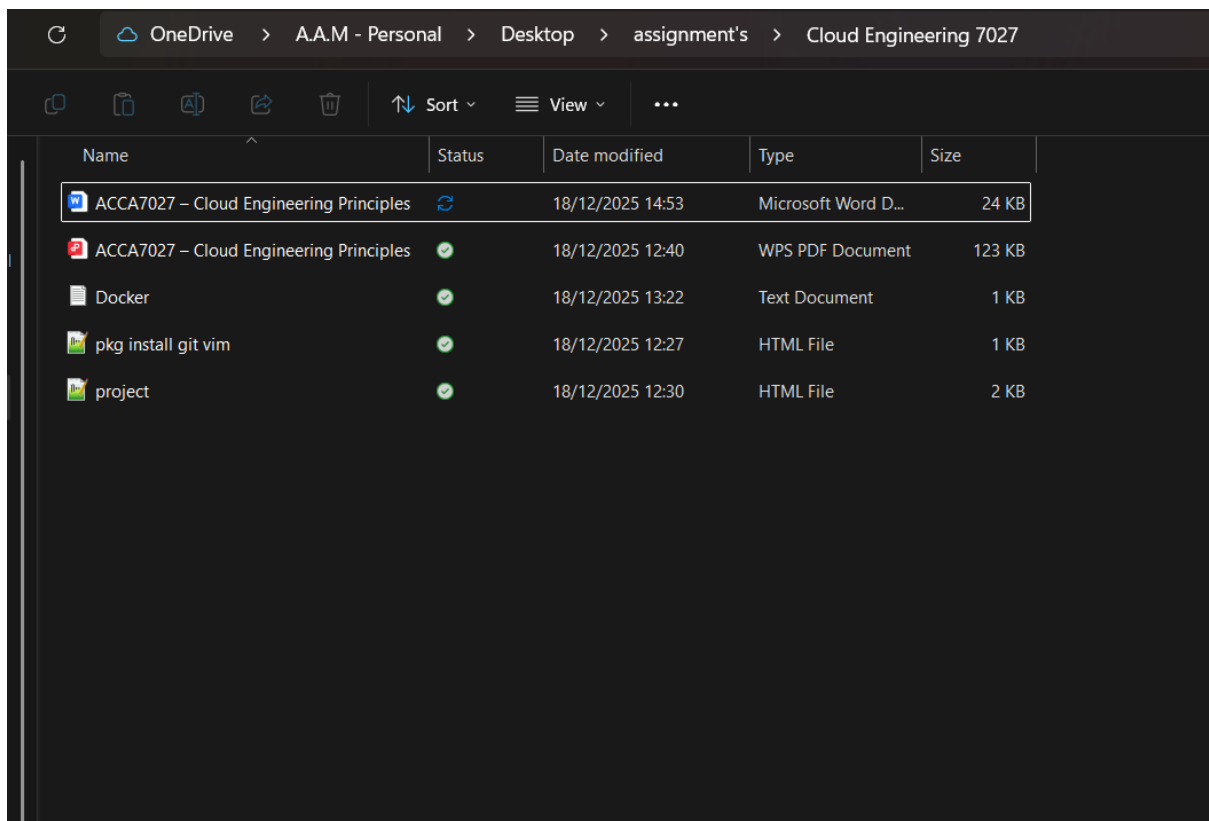
15. Dockerhub Demonstration Video Link

www.youtube.com/

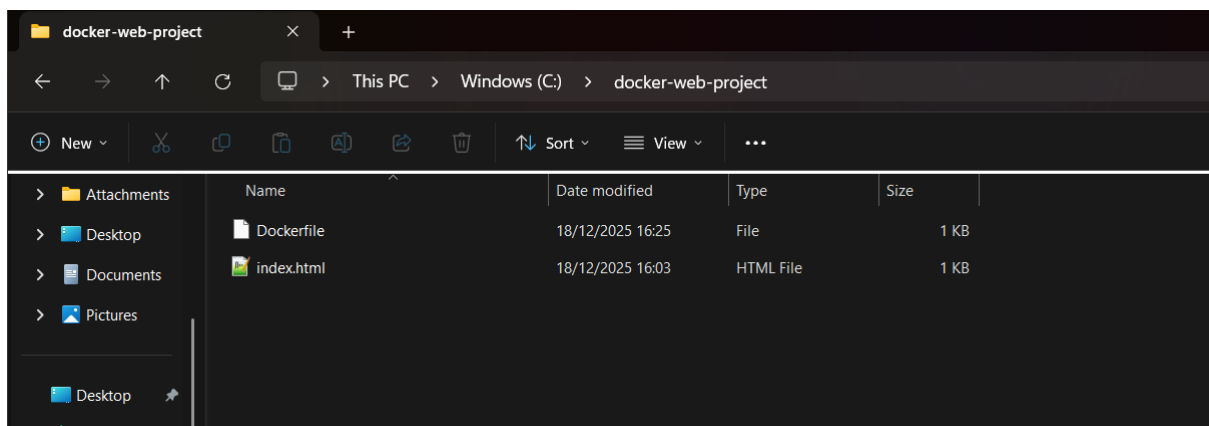
16. Conclusion

This report has explored containerisation's pivotal role in cloud computing, demonstrating Docker-based implementation through image creation, networking, and storage. It further examined load balancing, scalability, serverless computing, and cloud adoption drivers. Containerisation remains integral to scalable, portable, and efficient cloud-native development, especially when integrated with complementary cloud technologies.

17. Docker Screenshots



Docker and html file



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\sagor> ^C
PS C:\Users\sagor> cd C:\docker-web-project
PS C:\docker-web-project> docker build -t my-web-image .
[+] Building 1.7s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 212B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 296B
=> [1/2] FROM docker.io/library/nginx:latest@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486
=> => resolve docker.io/library/nginx:latest@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486
=> [2/2] COPY index.html /usr/share/nginx/html/index.html
=> => exporting to image
=> => exporting layers
=> => exporting manifest sha256:934450d0e253e1787092be22a68641c30a349fb33961c60a3d203112aab4d73a
=> => exporting config sha256:907a95cce332f7701cb328610b7a5fb860c428fbef480841e49afef8b714df6d
=> => exporting attestation manifest sha256:b47825205473ed444aa41628d683018eaface224857fbcf2450a36d076ba3697
=> => exporting manifest list sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af
=> => naming to docker.io/library/my-web-image:latest
=> => unpacking to docker.io/library/my-web-image:latest
PS C:\docker-web-project>
```

1-docker-web-project

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

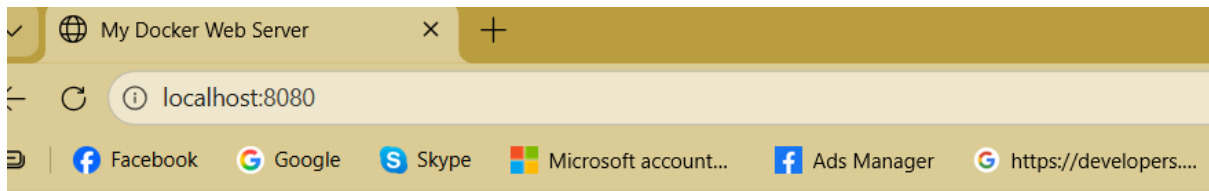
PS C:\Users\sagor> ^C
PS C:\Users\sagor> cd C:\docker-web-project
PS C:\docker-web-project> docker build -t my-web-image .
[+] Building 1.7s (7/7) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 212B
=> [internal] load metadata for docker.io/library/nginx:latest
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build context
=> => transferring context: 296B
=> [1/2] FROM docker.io/library/nginx:latest@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486
=> => resolve docker.io/library/nginx:latest@sha256:3b7732505933ca591ce4a6d860cb713ad96a3176b82f7979a8dfa9973486
=> [2/2] COPY index.html /usr/share/nginx/html/index.html
=> => exporting to image
=> => exporting layers
=> => exporting manifest sha256:934450d0e253e1787092be22a68641c30a349fb33961c60a3d203112aab4d73a
=> => exporting config sha256:907a95cce332f7701cb328610b7a5fb860c428fbef480841e49afef8b714df6d
=> => exporting attestation manifest sha256:b47825205473ed444aa41628d683018eaface224857fbcf2450a36d076ba3697
=> => exporting manifest list sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af
=> => naming to docker.io/library/my-web-image:latest
=> => unpacking to docker.io/library/my-web-image:latest
PS C:\docker-web-project> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my-web-image        latest             8c73b39f0ab3       About a minute ago 236MB
habib0007/calculator-app latest            004bfa3fa0f5       7 weeks ago        183MB
nginx               latest            3b7732505933       2 months ago       236MB
ubuntu             latest            66460d557b25       2 months ago       117MB
PS C:\docker-web-project>
```

2-docker images

```
PS C:\docker-web-project> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
my-web-image        latest             8c73b39f0ab3       About a minute ago 236MB
habib0007/calculator-app latest            004bfa3fa0f5       7 weeks ago        183MB
nginx               latest            3b7732505933       2 months ago       236MB
ubuntu             latest            66460d557b25       2 months ago       117MB
PS C:\docker-web-project> docker run -d -p 8080:80 --name my-web-container my-web-image
d93458f4b2ed1841861999cb74b60ccb57604deb1bef3d3a321503fd37b85622
PS C:\docker-web-project> |
```

3-docker my web container

4-localhost8080 running .png



My name is MD ABDULLAH AL MAMUN

This is my Docker Project

This web page is running inside a Docker container.

4-localhost8080 running

```
PS C:\docker-web-project> docker run -d -p 8080:80 --name my-web-container my-web-image
d93458f4b2ed1841861999cb74b60ccb57604deb1bef3d3a321503fd37b85622
PS C:\docker-web-project> docker volume create web-volume
web-volume
PS C:\docker-web-project> |
```

5-docker run 8080

```
web-volume
PS C:\docker-web-project> docker run -d -p 8081:80 `
>> -v web-volume:/usr/share/nginx/html `
>> --name web-volume-container `
>> my-web-image
4268bea7c60b2b77399cbe3a1c7bd55ef91d9dd5e40bd0261b1e86f1c881ca91
PS C:\docker-web-project> |
```

6-docker run 8081.80.

```
4268bea7c60b2b77399cbe3a1c7bd55ef91d9dd5e40bd0261b1e86f1c881ca91
PS C:\docker-web-project> docker volume ls
DRIVER    VOLUME NAME
local     web-volume
PS C:\docker-web-project> |
```

7-docker ls

```
PS C:\docker-web-project> docker volume ls
DRIVER    VOLUME NAME
local     web-volume
PS C:\docker-web-project> docker network create web-network
f05fa99e1230594714621f7090323d06c7f5d236374fb6de497747b73aba0a23
PS C:\docker-web-project> |
```

8-docker web network

```
PS C:\docker-web-project> docker network create web-network
f05fa99e1230594714621f7090323d06c7f5d236374fb6de497747b73aba0a23
PS C:\docker-web-project> docker run -d -p 8082:80 `
>> --network web-network `
>> --name web-network-container `
>> my-web-image
ac9b2aad9e7e0862b0a1128050e2f51507667319eb6f0a3f26d73b9d2476eb57
PS C:\docker-web-project> |
```

9-docker run 8082.80

```
my-web-image
ac9b2aad9e7e0862b0a1128050e2f51507667319eb6f0a3f26d73b9d2476eb57
PS C:\docker-web-project> docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
b3ee41e5d9d1    bridge    bridge      local
f298b7528597    host      host        local
eff9cf9d5d0a    none      null        local
f05fa99e1230    web-network bridge      local
PS C:\docker-web-project> |
```

10- network ls

```
>> my-web-image
ac9b2aad9e7e0862b0a1128050e2f51507667319eb6f0a3f26d73b9d2476eb57
PS C:\docker-web-project> docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
b3ee41e5d9d1    bridge    bridge      local
f298b7528597    host      host        local
eff9cf9d5d0a    none      null        local
f05fa99e1230    web-network bridge      local
PS C:\docker-web-project> docker stop my-web-container
my-web-container
PS C:\docker-web-project> docker rm my-web-container
my-web-container
PS C:\docker-web-project> |
```

11-docker stop & rm web container

```
my-web-container
PS C:\docker-web-project> docker login
Authenticating with existing credentials... [Username: mamun997]

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

Login Succeeded
PS C:\docker-web-project> |
```

12- docker user login succeeded

```
Login Succeeded
PS C:\docker-web-project> docker tag my-web-image yourusername/my-web-image:latest
PS C:\docker-web-project> |
```

13-docker my web page latest

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

```

Login Succeeded
PS C:\docker-web-project> docker tag my-web-image yourusername/my-web-image:latest
PS C:\docker-web-project> docker push yourusername/my-web-image:latest
The push refers to repository [docker.io/yourusername/my-web-image]
8da8ed3552af: Waiting
54e822d8ee0c: Waiting
58d144c4badd: Waiting
8c7716127147: Waiting
60bae0537f18: Waiting
5d8ea9f4c626: Waiting
33a62c4ed0a3: Waiting
b459da543435: Waiting
250b90fb2b9a: Waiting
push access denied, repository does not exist or may require authorization: server message: insufficient_scope: authorization failed
PS C:\docker-web-project> |

```

14- docker push

```

PS C:\docker-web-project> docker push mamun997/my-web-image:latest
The push refers to repository [docker.io/mamun997/my-web-image]
tag does not exist: mamun997/my-web-image:latest
PS C:\docker-web-project> docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
my-web-image        latest      8c73b39f0ab3  16 minutes ago 236MB
yourusername/my-web-image latest      8c73b39f0ab3  16 minutes ago 236MB
habib0007/calculator-app latest      004bfa3fa0f5  7 weeks ago   183MB
nginx               latest      3b7732505933  2 months ago  236MB
ubuntu             latest      66460d557b25  2 months ago  117MB
PS C:\docker-web-project> docker tag my-web-image:latest mamun997/my-web-image:latest
PS C:\docker-web-project> docker push mamun997/my-web-image:latest
The push refers to repository [docker.io/mamun997/my-web-image]
60bae0537f18: Pushed
33a62c4ed0a3: Pushed
5d8ea9f4c626: Mounted from library/nginx
b459da543435: Mounted from library/nginx
54e822d8ee0c: Mounted from library/nginx
250b90fb2b9a: Mounted from library/nginx
8da8ed3552af: Mounted from library/nginx
58d144c4badd: Mounted from library/nginx
8c7716127147: Mounted from library/nginx
latest: digest: sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af size: 856
PS C:\docker-web-project> |

```

15-docker images succeeded

The screenshot shows the Docker Hub web interface. On the left, the 'mamun997' namespace is selected. The main area displays a table of repositories. The table has columns: Name, Last Pushed, Contains, Visibility, and Scout. One repository is listed: 'mamun997/my-web-image', pushed 4 minutes ago, containing an 'IMAGE', with 'Public' visibility and an 'Inactive' scout status.

Name	Last Pushed	Contains	Visibility	Scout
mamun997/my-web-image	4 minutes ago	IMAGE	Public	Inactive

16- docker hub web view

```

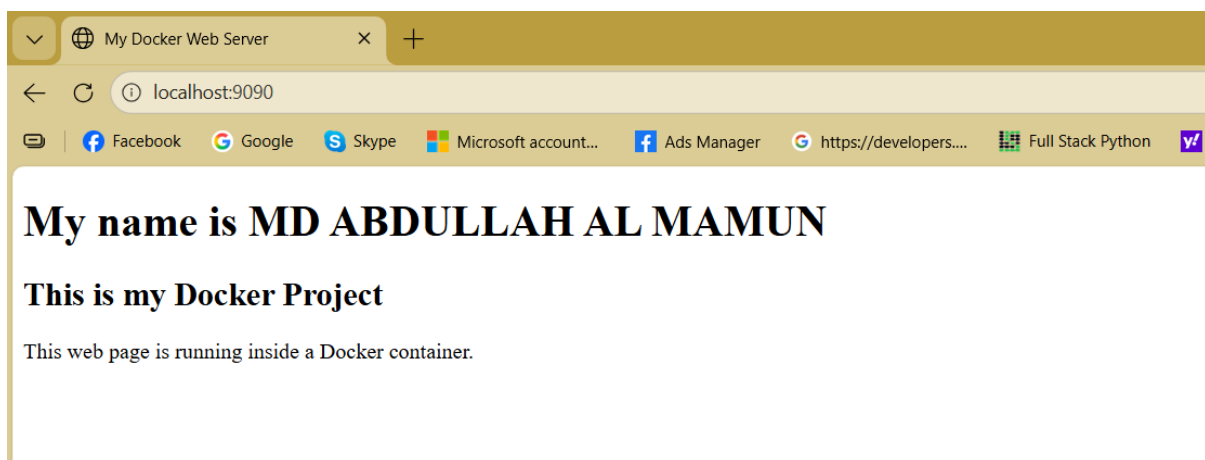
Digest: sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af
Status: Image is up to date for mamun997/my-web-image:latest
docker.io/mamun997/my-web-image:latest
PS C:\docker-web-project> docker run -d -p 9090:80 mamun997/my-web-image:latest
e2cf8482d5ccf8c58f285545fcaf17d4e57208a3909ab879cae41f919360b843
PS C:\docker-web-project> |

```

17- docker run 9090.80

```
PS C:\docker-web-project> docker tag my-web-image:latest mamun997/my-web-image:latest
PS C:\docker-web-project> docker push mamun997/my-web-image:latest
The push refers to repository [docker.io/mamun997/my-web-image]
60bae0537f18: Pushed
33a62c4ed0a3: Pushed
5d8ea9f4c626: Mounted from library/nginx
b459da543435: Mounted from library/nginx
54e822d8ee0c: Mounted from library/nginx
250b90fb2b9a: Mounted from library/nginx
8da8ed3552af: Mounted from library/nginx
58d144c4badd: Mounted from library/nginx
8c7716127147: Mounted from library/nginx
latest: digest: sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af size: 856
PS C:\docker-web-project> docker pull mamun997/my-web-image:latest
latest: Pulling from mamun997/my-web-image
Digest: sha256:8c73b39f0ab3220aa68d7d661529113beb0e9d8fba77c301f84a0355530aa6af
Status: Image is up to date for mamun997/my-web-image:latest
docker.io/mamun997/my-web-image:latest
PS C:\docker-web-project> |
```

18-docker pull



19-Localhost 9090 running

References:

AWS. (2023). *AWS Auto Scaling*. Retrieved

from <https://docs.aws.amazon.com/autoscaling/>

AWS. (2024). *What is Virtualisation?* Retrieved

from <https://aws.amazon.com/virtualisation/>

Azure. (2024). *Azure Application Gateway*. Retrieved

from <https://azure.microsoft.com/en-gb/products/application-gateway/>

Cisco. (2023). *Load Balancing and Scalability in the Cloud*. Retrieved

from <https://www.cisco.com/c/en/us/solutions/cloud/load-balancing-scalability.html>

Cloud Security Alliance. (2023). *Cloud Reliability and Resilience*. Retrieved

from <https://cloudsecurityalliance.org/>

CNCF. (2024). *Cloud Native Landscape*. Retrieved from <https://landscape.cncf.io/>

Datadog. (2024). *State of Serverless Report*. Retrieved from <https://www.datadoghq.com/state-of-serverless/>

Docker. (2024). *Docker Overview*. Retrieved from <https://docs.docker.com/get-started/overview/>

Flexera. (2024). *State of the Cloud Report*. Retrieved from <https://www.flexera.com/cloud-report>

Forrester. (2023). *The Forrester Wave: Serverless Platforms*. Retrieved from <https://www.forrester.com/>

Gartner. (2023). *Market Guide for Container Management*. Retrieved from <https://www.gartner.com/>

GigaOm. (2024). *Hybrid Cloud and Container Strategies*. Retrieved from <https://gigaom.com/>

Google Cloud. (2024). *Cloud Load Balancing*. Retrieved from <https://cloud.google.com/load-balancing>

IBM Cloud. (2023). *Container Networking Explained*. Retrieved from <https://www.ibm.com/cloud/learn/container-networking>

IDC. (2024). *Future of Cloud Infrastructure*. Retrieved from <https://www.idc.com/>

IEEE Cloud Computing. (2023). *Serverless vs. Containers: A Performance Study*. Retrieved from <https://www.computer.org/>

Kubernetes Documentation. (2024). *Volumes*. Retrieved from <https://kubernetes.io/docs/concepts/storage/volumes/>

McKinsey. (2023). *Cloud Agility and Business Innovation*. Retrieved from <https://www.mckinsey.com/>

Microsoft. (2023). *Dockerfile Best Practices*. Retrieved from <https://docs.microsoft.com/en-us/dotnet/architecture/>

Oracle Cloud. (2024). *High Availability Architectures*. Retrieved from <https://www.oracle.com/cloud/>

Red Hat. (2023). *What are Containers?* Retrieved from <https://www.redhat.com/en/topics/containers>

Snyk. (2024). *Container Security Report*. Retrieved from <https://snyk.io/reports/>

TechTarget. (2023). *Challenges of Containerisation*. Retrieved from <https://www.techtarget.com/>

Zhang, Y., Liu, Q., & Li, M. (2023). *Virtualisation Technologies in Cloud Computing: A Survey*. *Journal of Cloud Computing*, 12(1), 45–58.