

Cloud Architecture and Deployment Analysis

Keval Mahendra Motivaras

Student ID- 2515743

14 December 2025

Demo video - <https://youtu.be/USSEqDJs0aM>

Table of Contents

PART A – Containerised Web Application Using Docker	4
1. Creating the Web Application	4
2. Building the Dockerfile	6
3. Building the Docker Image.....	7
4. Running the Container.....	11
5. Testing the Deployment	12
6. Reflection on Part A.....	13
PART B — Cloud Concepts.....	14
1. Introduction.....	14
2. Chosen Application Scenario.....	15
3. Explanation of Each Concept in Azure	16
3.1 Load Balancing in Azure	16
3.2 Scalability in Azure	17
3.3 Application Gateway in Azure	19
4. Why Load Balancing and Scaling Are Complementary	21
5. Benefits of Combining All Three Technologies	22
PART C – Driving Forces for Cloud Adoption.....	24
1. Growth Drivers	24
2. Serverless Functions Explained	26
3. Comparison: Serverless Functions vs Containerisation	27
Conclusion	29

References	30
Appendix	33
Appendix A — Docker Commands & Outputs	33
Appendix B — index.html Source File	37
Appendix C — Dockerfile.....	37
Appendix D — Diagram for Part B	38

PART A – Containerised Web Application Using Docker

This chapter will describe the process of constructing, packaging, and deploying a basic web application with Docker and demonstrate how containerisation and the deployment of applications on the cloud work. Part A of the test provided a background on the fundamentals of containerisation by developing and operating a lightweight web app with Docker. Containerisation has now become one of the fundamental instruments in contemporary cloud engineering since it provides a reproducible, isolated and portable environment. Because the developer places an application and all of its components into a single container image, he can be confident that the application will behave the same way regardless of the operating system, hardware, or cloud provider. This addresses the historic issues of software deployment, including setting changes, all-in-compatible dependencies, and differences between the developer environment and the live one. Each of the making and launching actions of the app in Docker is detailed below [1].

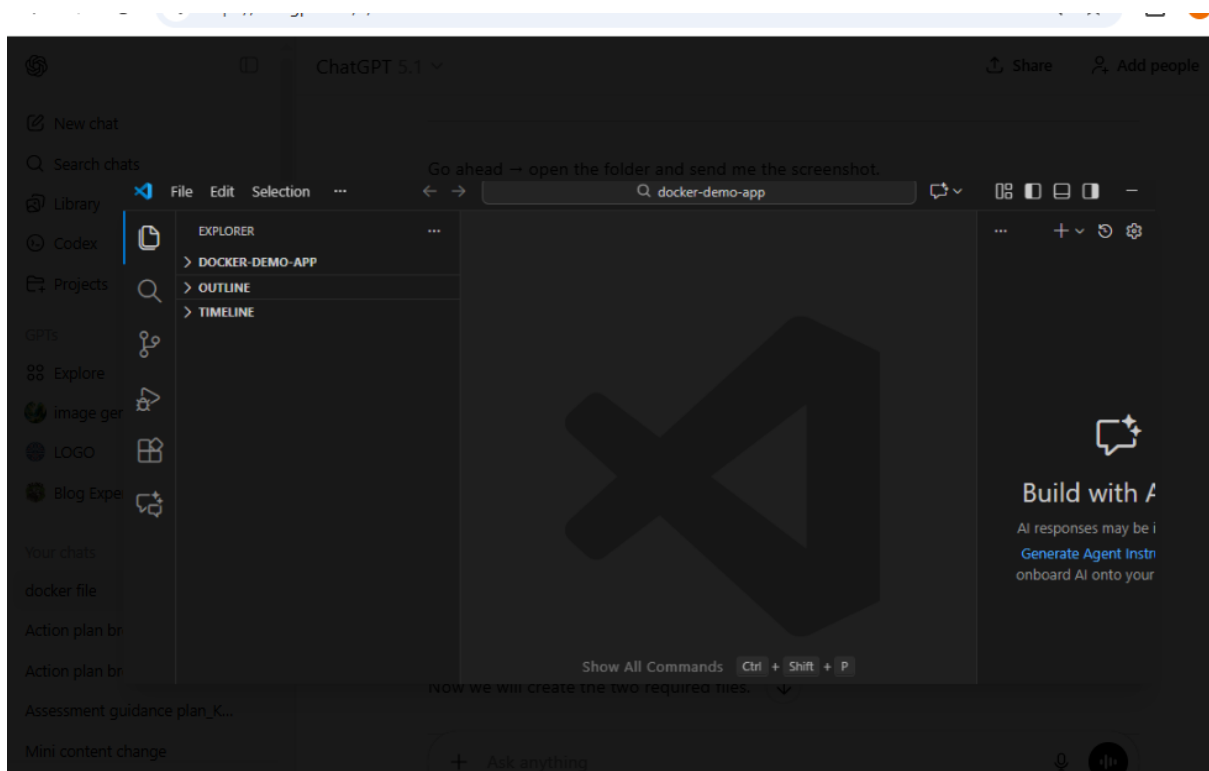
1. Creating the Web Application

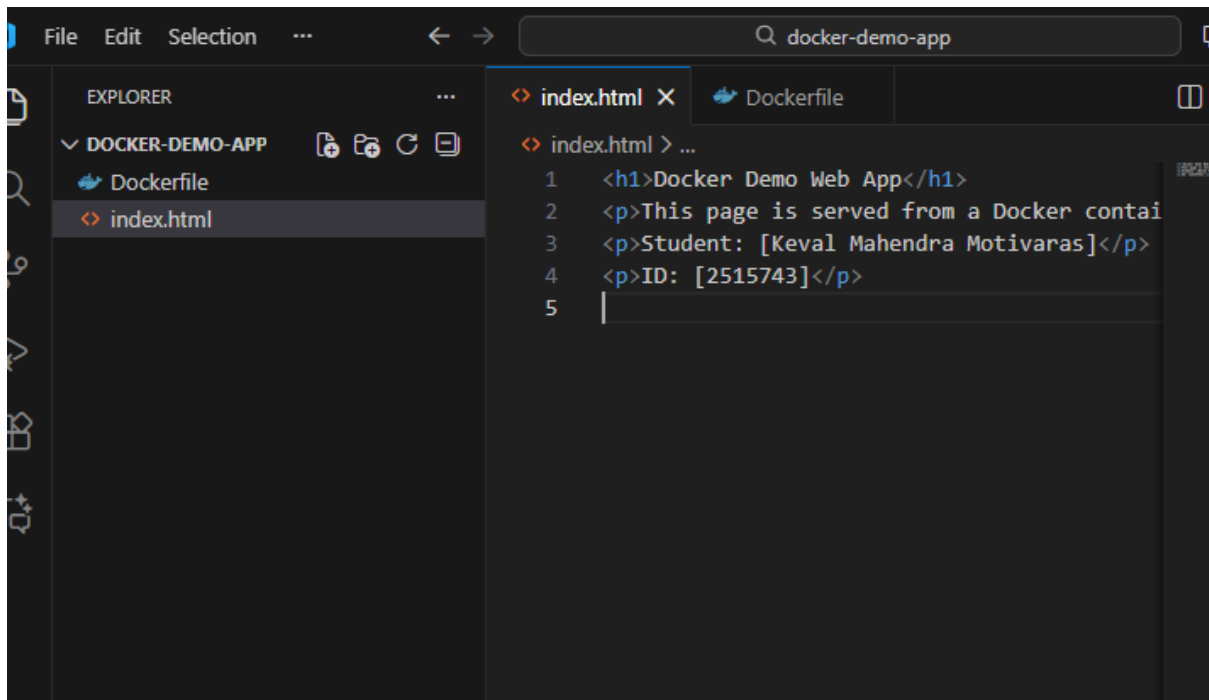
Part A: they created a simple statical HTML page. The index.html was the primary file which contained some basic text including a welcome message, name of the student and student ID. The page was small and they did not attempt to be fancy in design. Rather, they desired a file that could be bundled and executed in a containerized Docker with certain reasonable certainty [2].

This exercise is relevant in two ways using the static HTML. First, it gets rid of the complications around scripting languages, databases or frameworks and is free to concentrate fully on the mechanics of containerisation. Second, the best types of content are those that are static and only prove Docker to be powerful: even basic applications can be configured into a

container image and can be deployed by a standard process. In practice, the use of lightweight stateless content served by web servers such as Nginx is found in numerous production systems (documentation portals, marketing websites, and status pages) in the real world, so this exercise is close to practical reality [3].

The HTML file was placed on its own project folder. This prepared the way to the creation of the Dockerfile and the creation of the container.





2. Building the Dockerfile

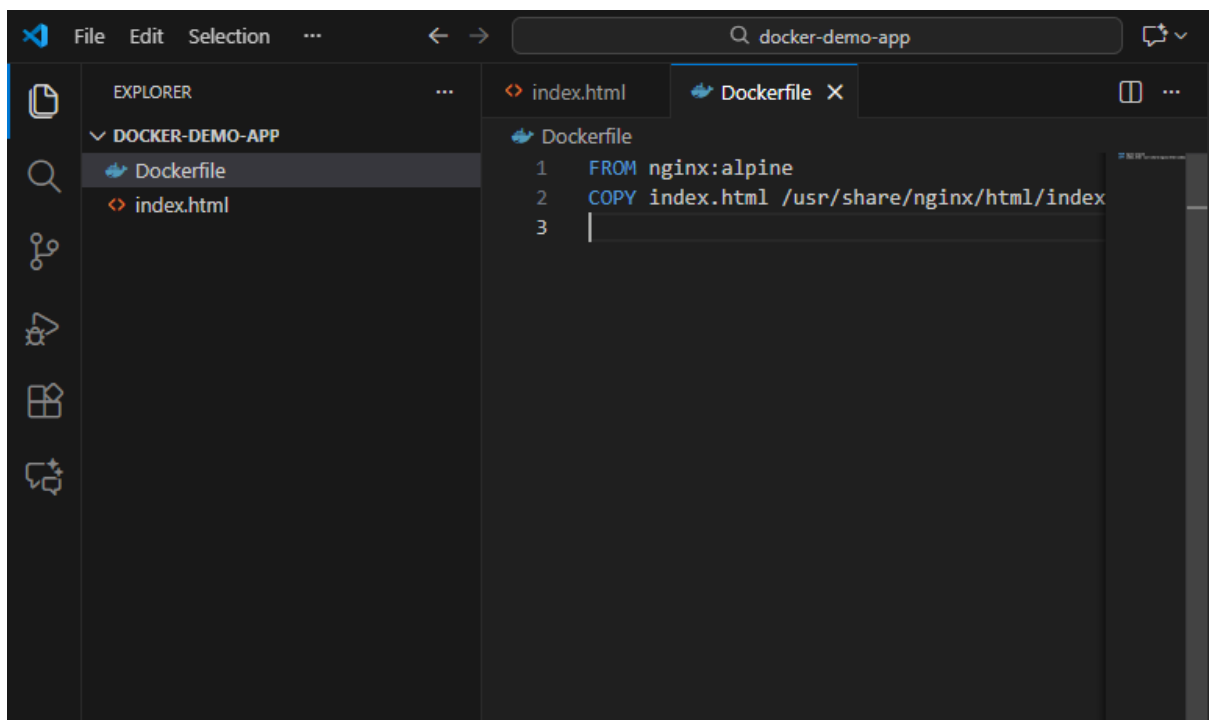
To install the environment in which the web app would be running, they developed a Dockerfile. A Dockerfile is simply a script that contains commands with instructions to Docker on how to create an image. There were two main commands in the Dockerfile in this project:

```
FROM nginx:alpine
COPY index.html /usr/share/nginx/html/index.html
```

The initial command, `nginx:alpine`, instructs Docker to use the official image of Nginx web server that is built on the foundation of the Alpine Linux, as the base image. The popularity of this base image is that it is extremely small; most of the times it is less than 10 MB and is based on security and speed. Alpine Linux is also established as lightweight and with a simple package system, and therefore can be recommended when cloud apps require to utilize resources effectively. Through the choice of a trusted base image, developers do not need to install Nginx or configure the operating system [4].

The second command, `COPY index.html /usr/share/nginx/html`, will transfer the `index.html` file of the app into the default directory where Nginx searches files. This implies that Nginx will automatically display the page when the container is initially started when one accesses the exposed port of the container.

These two lines are demonstrating the simplicity of Docker: it takes a few commands to package a functional web-server. Dockerfile also provides reproducibility, anyone who uses the same Dockerfile will receive the same image, regardless of the OS or settings of his or her computer [5].



3. Building the Docker Image

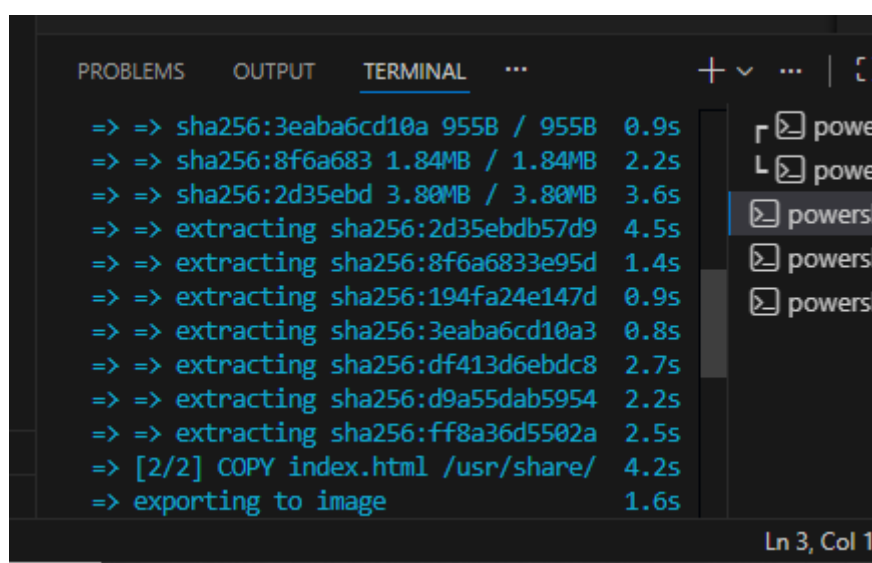
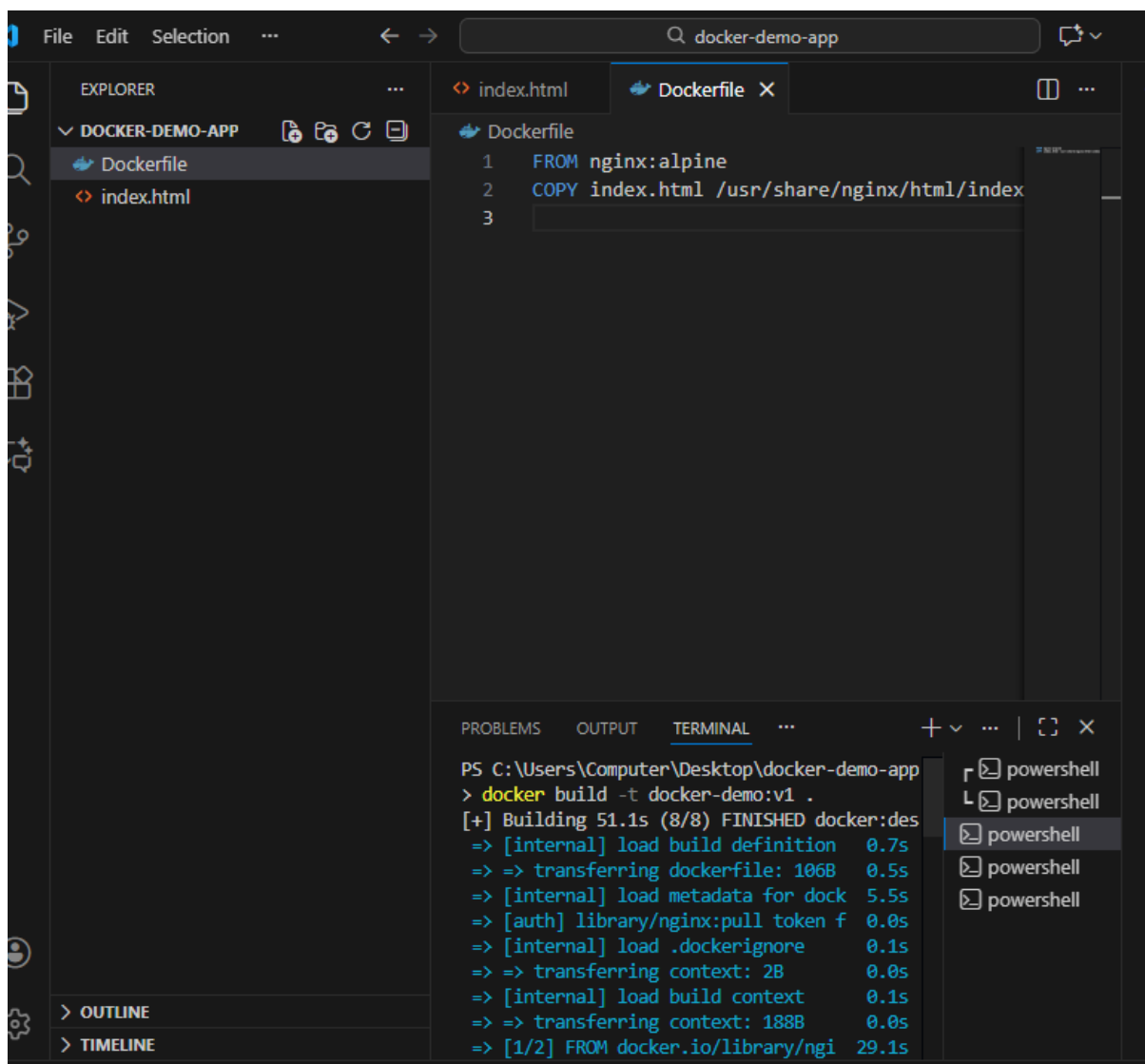
Once they created the Dockerfile, they created the Docker image. The tools were Docker Desktop and Visual Studio Code terminal. Here is the command they ran:

```
docker build -t docker-demo:v1 .
```

This instruction makes Docker create an image based on the present folder (.) and assign it the tag `docker-demo:v1`. Docker will read the Dockerfile, and download the base Alpine version of Nginx in case it is not already on the machine, and will move the HTML file into the correct folder and will assemble all the layers of the final image [6].

Upon the full construction of Docker, it displayed a special ID and a few messages. A verification with the command `docker images` ensured that the image was present in computer and was approximately 79MB. This compactness indicates one of the main benefits of Alpine images that will take lower storage place and are quicker to download, making them usable in distributed system deployment much more efficient [7].

The image is a portable package and it has all the contents to start the application, therefore, it can be deployed in a container.



4. Running the Container

In order to execute the app the new image was launched using the following command:

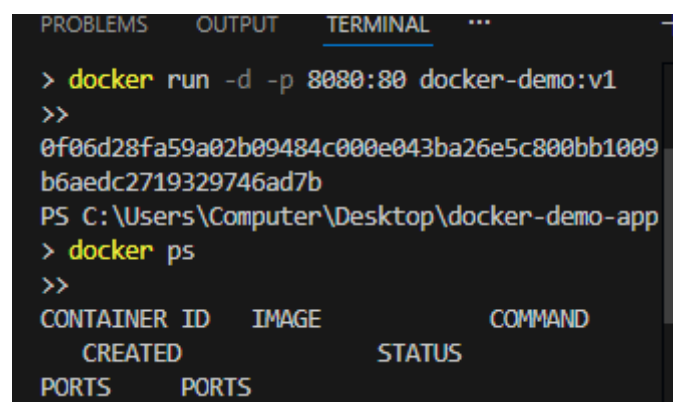
```
docker run -d -p 8080:80 docker-demo:v1
```

There are a few things that this command does:

- **-d (detached mode):** The container works as a background service.
- **-p 8080:80 (port mapping):** It links the port 8080 of the computer to the port 80 of the container.
- **docker-demo:v1** It indicates what image is to be used to open the container.

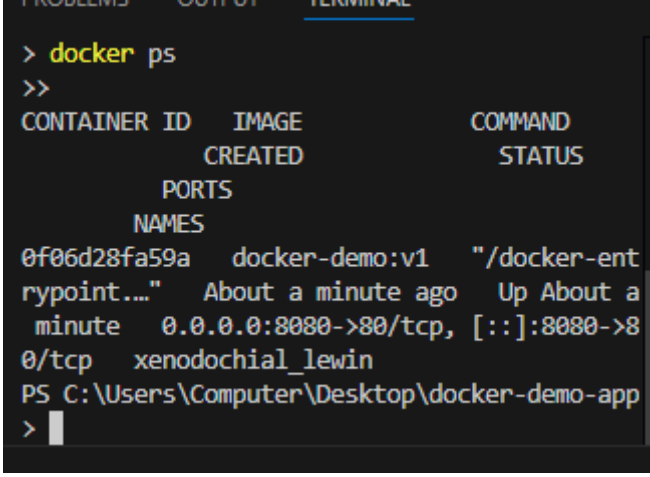
The reason why port mapping is significant when deploying containers is that containers operate in isolated spaces. The services within the container are accessible to outside users or local browsers by mapping ports [8].

The command of `docker ps` revealed an active container with its ID, status, port mapping, and name (assigned by Docker). This was to ensure that the container was started and Nginx had started.



```

PROBLEMS  OUTPUT  TERMINAL  ...
> docker run -d -p 8080:80 docker-demo:v1
>>
0f06d28fa59a02b09484c000e043ba26e5c800bb1009
b6aedc2719329746ad7b
PS C:\Users\Computer\Desktop\docker-demo-app
> docker ps
>>
CONTAINER ID   IMAGE      COMMAND
CREATED              STATUS
PORTS              PORTS
  
```



```

> docker ps
>>
CONTAINER ID   IMAGE          COMMAND                  STATUS
              CREATED
              PORTS
              NAMES
0f06d28fa59a   docker-demo:v1 "/docker-entrypoint..." Up About a
minute        0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
              xenodochial_lewin
PS C:\Users\Computer\Desktop\docker-demo-app
>

```

5. Testing the Deployment

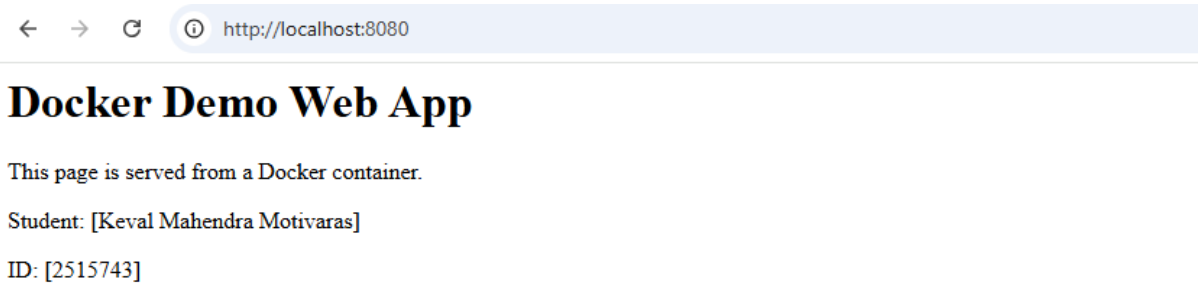
The final procedure was to test its deployment on a web browser by accessing:

http://localhost:8080

The webpage loaded immediately, and it was displaying the bare HTML that the Nginx was serving within the container. This was an indication that everything was done in the right way.

- The application file was transferred to the container.
- The Nginx server works.
- The container runs on its own.
- Port forwarding between the container and the host works.

This entire process indicates that when an app is placed in a container, it can be reliably used on any platform that is able to run Docker- Windows, MacOS, Linux or a cloud platform [9].



6. Reflection on Part A

Part A demonstrates the utility of Docker in the current software deployment. It develops a basic web application, which runs in a container, to describe how Docker corrects the environment differences and simplifies the deployment process. Applications can be made the same everywhere in containers, causing less time to be spent debugging, and making the development pipelines more predictable [10].

The building block of this exercise is the higher principles of cloud engineering in the report, including auto-scaling, load balancing, and serverless computing. It is imperative to have good knowledge of containerization in order to work in the cloud-native ecosystems. Part A provides such a foundation through a practical demonstration.

PART B — Cloud Concepts

1. Introduction

In this section, cloud computing is introduced and the role of load balancing, scalability and application gate ways is discussed to allow a current e-commerce application to run on Microsoft Azure.

Most of the digital services are based on cloud computing today. It allows businesses to execute applications on flexible on-demand infrastructure. Businesses do not maintain their own servers but rather they can access computing power, storage, and networking capability installed in the cloud and which increases as the demand increases. According to Gartner, the amount of money will exceed 600 billion dollars spent in global cloud services by 2025. This indicates the general trend towards cloud native services in areas such as retail, finance and logistics [11].

In cloud-based architecture, three architectural principles are important in performance and reliability assurance and they are load balancing, scalability, and application gateways. Load balancing is the distribution of traffic among more than one server so as to avoid the overloading of a single server. Scalability enables systems to expand or contract to the need. Application gateways provide secure smart application level routing. These are necessary in customer interfaces which handle the unpredictable traffic patterns [12].

To explain these ideas, this report takes an online shopping site as an example. E-commerce is one of the areas where the traffic increases at an alarming rate during sales or holidays, and it leads to spikes. These variations are managed through cloud services, which are specifically deployed by big retailers such as Amazon and Flipkart. This report is about Microsoft Azure,

one of the leading providers of clouds. Azure has important services including the Azure Load Balancer, Virtual machine Scale sets and the Azure Application Gateway.

2. Chosen Application Scenario

In this section, there is an actual e-commerce scenario that reveals the relevance of load balancing scalability, and application gateways in the cloud.

One of the bright examples of the applications that are based on cloud load balancing, scaling and gateways is the e-commerce business. There is no predictability in the traffic in online stores particularly when it comes to sales, holidays or special offers. As an example, the sale of Big Billion Days at Flipkart issued millions of user requests at the same time in a few minutes, indicating how demand may fluctuate drastically. Such traffic bursts may overwhelm normal on-premise systems and thus it is important to utilize the cloud to ensure that the site is operational [13].

An online shop of today relies on a number of services, which are interconnected: browsing products, placing them in the cart, checking out, and logging on. All these services must be fast and safe in the transmission of data. They also require good security since they deal with personal information of customers and credit-card numbers. Failure of one of its components might result in loss of money, reputation or even customers to the business.

These problems can be addressed with the tools provided by Microsoft Azure. Reliability, scalability, and high security are ensured by the combination of the Azure Load Balancer, Azure Virtual Machine Scale Sets (VMSS) and Azure Application Gateway to ensure that the shopping experience remains seamless [14].

3. Explanation of Each Concept in Azure

This section describes the way in which Azure balances traffic, scales resources, and makes use of application gateways with cloud-hosted apps.

3.1 Load Balancing in Azure

In this section, the author describes how Azure distributes the incoming traffic and maintains services by balancing the load.

One of the characteristics of cloud services is load balancing, which prevents congestion of a single server. In internet stores, traffic may get spike high and hence it is required to spread the work properly so that the site remains fast with no crashes. There are two major load balancing tools of Azure, which are Azure Load Balancer and Azure Application Gateway. Both operate at a different level of the network [15].

Azure Load Balancer operates on Layer 4 (Transport Layer) and supports TCP and UDP traffic. It is high speed and low delay, hence is commonly used to distribute web traffic or API calls across a large pool of virtual machines. The most important component of this service is the backend pool which is a collection of virtual machines that gathers the incoming requests collectively. Azure also periodically runs health probes to all the machines to verify the endpoints of the machines. When a machine fails a probe, it is automatically put out of the backend pool to ensure that no traffic is sent to a bad or unresponsive machine. This automatic backup is used to maintain the service in case of a maintenance or other unforeseen issues.

In Azure Load Balancer, the hash-based technique is applied and packets are relayed to the servers and hence the load is evenly distributed. This prevents overloading any VM in instances whereby the application is being utilized by a large number of users. Similar load-balancing

tricks are employed by such companies as Walmart or Amazon to maintain the performance steady during peak hours, like Black Friday, and it demonstrates the importance of this feature in real retail life [16].

Azure Load Balancer transmits traffic on the network layer. Azure Application Gateway operates at the Layer 7 (Application Layer) and is able to inspect the contents of the traffic. This can be handy with complex web applications requiring such features as URL-based routing, host-based routing, or ensuring that a user is on the same server (session affinity). Indicatively, Application Gateway can make /checkout requests to payment-ready servers, and /product-images can be made to fast caching servers. This is an exact control that enhances performance and maintains important jobs operating smoothly particularly sensitive jobs such as payment.

One of the major advantages of Application Gateway is that it may complete the process of the SSL/TLS prior to the delivery of the traffic to the back-end servers. It implies that the servers are not forced to do the encryption processing, which consumes CPU and makes it quicker. Application Gateway would also be compatible with the Web Application Firewall (WAF). The WAF prevents attacks such as SQL injection and cross-site scripting that is frequently used to attack e-commerce sites [17].

Azure Load Balancer and Application Gateway used together assist high-traffic applications to be fast, safe, and reliable even when traffic varies in unpredictable ways.

3.2 Scalability in Azure

This part describes the ability of Azure to add and reduce resources according to the demand.

Cloud computing also offers scalability, particularly to online stores that have to cope with traffic fluctuation during online store holidays or sales. Azure also allows them to expand by providing the virtual machine with additional power or scale out by adding machines. Growing is more appropriate to web apps that can run on several servers since it enables them to collaborate and overcome the restrictions of a single machine [18].

Azure gives them the opportunity to add new virtual machines concurrently. This is referred to as Virtual machine Scale set (VMSS). When using a VMSS, it becomes possible to create and provision numerous identical VMs simultaneously. It also possesses inbuilt autoscaling. It implies that the system would be able to automatically switch on or off more machines depending on the traffic of the servers. They instruct it on what to monitor such as CPU usage, memory usage, or the number of requests in a given second. As an example, an online store might have a policy that allows CPU usage to exceed 70 percent more than five minutes later, VMSS will add additional machines to ensure the site is responsive and will not crash.

Automatic scaling is particularly useful in stores that all of a sudden increase the amount of traffic to the store. Traffic can increase several times in a few minutes with the help of flash sales or holiday campaigns. Amazon claims that it applies autoscaling on its Prime Day because there is a reason to deal with the rush. Under VMSS, additional servers are installed within a short period, and this serves to ensure that shoppers do not leave carts because the shoppage is slow or it is out of service [19].

Azure pairs VMSS and Azure Monitor to monitor the performance of the app and the utilization of the resources. Azure monitor collects information and notifies them thus scaling is done basing on real time and accurate data. This scaling only on demand and therefore saves money and maintains the appropriate instances.

VMSS enhances fault tolerance, other than scaling. In case one of the instances fails, VMSS will replace it with a healthy one, thus, ensuring that traffic flows as expected. This is critical to online retailing where even a couple of minutes of unavailability will damage sales [20].

Lastly, the scaling solutions provided by Azure enables online retailers to add and decrease resources within a short time to keep customers happy and business processes running smoothly.

3.3 Application Gateway in Azure

In this part, they will demonstrate the use of Azure application gateway in securing and intelligent routing of internet-based applications.

Azure Application Gateway is a layer 7 reverse proxy, which handles traffic of application applications like e-commerce websites. It is not just like a normal load balancer as it reads through the traffic of the HTTP/S and can redirect it following URL paths, host headers or session rules. This allows them to manage the way and location in which requests are processed [21].

URL-based routing is considered to be one of its key features. It directs the traffic to the right backend server group by the URL path. An example with this would be to /api/products requesting a server pool that will be efficient at retrieving data whereas /checkout request will be directed to a more secure pool that is faster and has added resources. It also possesses host-based routing, which allows a single-gateway to process other sites such as shop.example.com or payment.example.com.

The gateway has the ability of session affinity to retain a customer in the same backend server when he/she is making a checkout. This minimizes disruptions and multi-step processes like paying and logging-ins are more dependable.

There is security enhancement that includes Web Application Firewall, or WAF. It defends against typical web threats that include those in the OWASP top 10, including bot traffic, injection attacks, and credential stuffing. The rule sets of the WAF of Azure are updated frequently in order to protect against emerging global threats and secure retail websites.

SSL offloading is also another notable characteristic. The encryption and decryption is carried out by the gateway rather than the back end servers. This reduces the workload on the VMs and allows them to concentrate on the key application duties, increasing the overall speed.

Large retailers such as Walmart or Alibaba deploy the same type of application-layer gateways to maintain user experiences with security, speed and consistency regardless of the number of visitors. Azure Application Gateway moves those enterprise best practices to a native cloud environment [22].

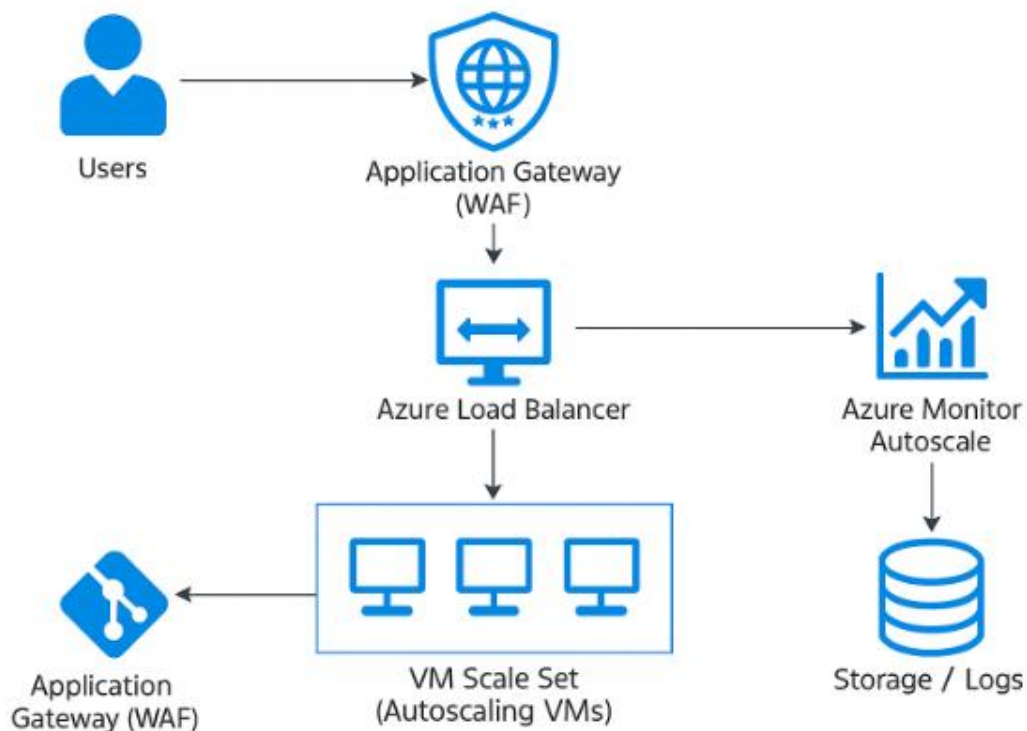


Figure 1 – Azure Architecture Diagram

4. Why Load Balancing and Scaling Are Complementary

This part describes the approach of load balancing and scaling by Azure that provides flexibility and robust service. Scaling and load balancing complement each other in the cloud as they address the various aspects of system behavior. Load balancing distributes the traffic to a large number of servers. Scaling is used to adjust the demand of the number of servers that are running. It is a combination of a flexible system that can sustain the work of apps in the case of changes in the load, such as online stores [23].

Azure Virtual Machine Scale Sets (VMSS) and Azure Load Balancer work together to ensure that the application is quick. VMSS will automatically add VM machines when there is an

increase in traffic such as a large sale. This is because the Load Balancer runs health probes on each new VM and adds it to the pool. Thereupon the traffic is evenly distributed to all well machines, and conferences no delays occur.

In case of traffic reduction, VMSS eliminates additional machines. The Load Balancer immediately ceases to send the traffic to the said machines and redirects the requests to the rest of the healthy ones. This is economical and maintenance of the service goes on.

The two tools combined render the system elastic and thus it expands or contracts accordingly, when required and the two also ensure that the app remains available in case one of the machines failed. This assists actual shopping places to prevent downtime, prevent overloading and enable the customers to shop or make purchases with ease. Strong cloud services consist of load balancing and scaling [24].

5. Benefits of Combining All Three Technologies

In this section, it is demonstrated that when load balancing, scaling, and gateways are used in combination a powerful, secure and customer friendly cloud service is obtained.

Once load balancing, auto-scaling, and gateways are combined, they form a complete cloud system that is capable of executing resource intensive applications such as online stores. The greatest concession is that the system remains online. Azure load balancer directs traffic to functioning computers only and VMSS gives automatic replacement to failed computers. This reduces the downtime and cost is saved. Amazon claims that one minute of downtime on a high-shopping day can result in millions of dollars lost, which proves its need to maintain a 24-hour uptime.

There is also improved fault tolerance. When a computer fails, it will be spotted by probes and removed off the traffic list and another one will automatically be launched. It is that self-healing that prevents people from fixing the service because it self-heals. Retailers such as Walmart have setups of such nature to ensure that their sales proceed on major shopping days [25].

The other advantage is the quicker response time. The system distributes the traffic to a large number of servers and introduces more servers whenever the demand is high, consequently, pages are loaded fast even during a high degree of demand. Customers abandon shopping carts as a result of slow page loads, which has been a challenge in e-commerce.

Azure Application Gateway is better than the built-in Web Application Firewall (WAF) in improving security. The WAF prevents such common attacks as SQL injection, cross-site scripting and bot attacks prevalent in online retail. The end of the encryption process in the gateway will eliminate the load on the back-end server, which is faster and safer. Azure maintains WAF rules that contain new threats and, therefore, protection is up to date.

Another benefit is cost effectiveness. VMSS also lowers the cost of servers during light traffic hours, hence the businesses are only charged on the amount they utilize. This renders the operations of clouds far more affordable in comparison to fixed on-premises hardware.

Lastly, the three technologies create an enjoyable experience to their customers. Rapid pages, safe payments and 24 / 7 access enhance customer satisfaction and loyalty. On the whole, the use of load balancing, auto-scaling, and application gateway creates a stable architecture that adheres to the best practices employed by major retailers across the world [26].

PART C – Driving Forces for Cloud Adoption

In the section, the primary reasons why individuals use cloud computing, the meaning of serverless functions, and a comparison of serverless computing and containerisation will be discussed in detail.

1. Growth Drivers

Cloud computing has developed rapidly due to the desire of firms to get cheaper, more accommodative, and global infrastructure. According to Gartner, as of 2023, global public cloud expenditure exceeded USD 600 billion, which is why cloud services can be regarded as one of the most rapid segments of business IT. There are a number of important reasons behind this trend [27].

Cost Efficiency

One of the primary reasons why companies are adopting the cloud is the shift between the high initial expenditure (CAPEX) and recurrent expenditure (OPEX). Large preliminary investments in machines, cooling, and space, and workers are required in old data centers. Cloud services eliminate such a requirement since they can only pay on usage. Amazon claims that businesses that migrate to AWS can reduce the cost of infrastructure by up to 60 percent, particularly in the case of those using reserved or spot instances. This saving is significant to such stores as a retail, logistics, or fintech when the margins are low, and the workload may vary.

Elasticity

Elasticity implies that cloud systems can be able to add or delete resources automatically. This is significant to firms whose workload varies. To illustrate the point, when Flipkart has their

Big Billion Days sale, millions of users visit simultaneously; it is only cloud elasticity that is capable of managing these spikes. The cloud services automatically provide the additional compute, database or storage to maintain the application running at sudden peaks.

Global Reach

Cloud providers operate data centers in locations all over the globe in order that companies can keep their apps close to their clients. Azure is the cloud provider with the most regions worldwide, at over 60. This international availability reduces the waiting time, makes the app faster and assists in compliance with the local data regulations. Global audience is significant to streaming platforms, online retailers, and large global companies.

Reliability and High Availability

Cloud providers provide backup and fail largely options that on-prem systems can hardly match. Most services offer a guarantee of 99.9 to 99.99 percent of uptime by Azure. In case of the failure of one of the zones, the workloads will be handled by a different zone. This is critical to life-need services like banking, healthcare and transportation [28].

Reduced Operational Burden

It is easier with cloud platforms since there is no need to maintain hardware, to patch servers, to upgrade systems or keep networks running. Their engineers can then work on emerging ideas and products by companies. This is one of the reasons why a lot of work is saved using managed databases, serverless computing, and PaaS, with which no management of the infrastructure is needed at all.

2. Serverless Functions Explained

The idea of serverless computing, which is available in the form of AWS Lambda, Azure Functions and Google Cloud Functions, is quite different to the traditional server deployment pattern. Serverless functions do not reveal the infrastructure that is running beneath, like a container or a virtual machine [29].

Event-Driven Compute

Serverless systems execute code on demand, based on an event such as an API request, database update, upload or a message queue. Netflix encodes videos in real time when uploading new content with the help of AWS Lambda, as an example. Due to the event-based nature, serverless is highly efficient in automation, real-time data processing, and backend API processes.

Zero-Admin Approach

Serverless platforms also do not require any configuration of servers as well as how to repair or expand them. The developers add the code that performs a given task and the cloud provider takes care of the rest. It is a no-admin method and is fast to develop and makes the system easier to manage hence particularly useful with start-ups and small engineering teams.

Auto Scaling

Serverless functions are easily and rapidly growing. In cases of large volume of requests, the provider will automatically start more copies of the functions. An example of AWS Lambda can launch thousands of copies simultaneously in a matter of seconds [66].

Pay-Per-Execution Model

The use of serverless billing is based on the number of times a function is executed and the duration required. AWS charges on a 1-millisecond increment, and hence, they only pay when the function is actually operating. In case of light or inconsistent workloads, this has the capability of saving a significant amount of money compared to running containers or virtual machines on a permanent basis [30].

3. Comparison: Serverless Functions vs Containerisation

This part of the comparison of serverless and container approaches pays close attention to instances where containers remain superior.

Comparison Table

Criteria	Serverless (Lambda / Azure Functions)	Containers (Docker / AKS)
Cost Model	Pay-per-invocation, billed per millisecond	Pay for compute runtime even when idle
Control	Low – provider manages OS/runtime	High – full control over OS, dependencies
Scaling	Fully automatic & instant	Defined scaling rules; may require Kubernetes
Use Case	Event-driven logic, automation, lightweight APIs	Microservices, long-running apps, custom runtimes

Strengths of Serverless

Serverless is ideal when they want to execute short bursty stateless programs. It is good for:

- Lightweight APIs

- Data transformation tasks
- Image or log processing
- Catboats, IoT triggers and automation flows.

Fintech and streaming are examples of companies that are utilizing serverless increasingly to reduce the cost of operation and introduce their products quicker into the market.

Where Containerisation Is Superior (Critical Evaluation)

Although serverless has numerous advantages, it is not suitable in every situation and a lot of practice requires the usage of containerisation to be more flexible and controlled.

1. Long-Running Processes

Serverless functions have time restrictions, and some have time limits such as AWS Lambda, which has a maximum of 15 minutes. Containers are more appropriate, since the applications such as fraud detector, machine learning training or data pipeline may require longer.

2. Complex Microservices

Microservices of Big firms such as Uber and Airbnb are based on libraries, frameworks, and long-lived connections. Containers allow them to customize the runtime environment to the fullest, which is not the case with serverless.

3. Predictable High-Load Systems

When an application has high traffic continuously, such as the site of a large bank, then deploying containers on Kubernetes is less expensive than a serverless environment, where they pay every time the app is accessed.

4. Legacy Application Migration

Containers are also common in companies that migrate old systems to the cloud since they are able to reproduce the current system without significant rewrites.

Conclusion

Containers and serverless computing can assist individuals implement cloud since they maintain the expenses low, adaptable enough, can expand and contract swiftly, and run without difficulties. Serverless is suitable to the jobs that require triggering on demand and a rapid completion. Containers are best suited in long running jobs, complex jobs, or those demanding significant custom settings. Being able to use both methods is satisfying a great number of needs and retains more customers using the cloud [31].

References

- [1]
N. Bangar, “Containerization in cloud computing: comparing Docker and Kubernetes for scalable web applications,” *International Journal of Science and Research Archive*, vol. 13, no. 1, pp. 3302–3315, Oct. 2024, doi: <https://doi.org/10.30574/ijsra.2024.13.1.2035>.
- [2]
M. A. Khder, “Web Scraping or Web Crawling: State of Art, Techniques, Approaches and Application,” *International Journal of Advances in Soft Computing and its Applications*, vol. 13, no. 3, pp. 145–168, Nov. 2021, doi: <https://doi.org/10.15849/ijasca.211128.11>.
- [3]
None Gogulakrishnan Thiyagarajan and N. P. Nayak, “Docker under Siege: Securing Containers in the Modern Era,” *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, vol. 11, no. 1, pp. 3674–3719, Feb. 2025, doi: <https://doi.org/10.32628/cseit25112773>.
- [4]
S. Bhat, *Practical Docker with Python*. 2021. doi: <https://doi.org/10.1007/978-1-4842-7815-4>.
- [5]
P. Muzumdar, A. Bhosale, G. P. Basyal, and G. Kurian, “Navigating the Docker Ecosystem: A Comprehensive Taxonomy and Survey,” *Asian Journal of Research in Computer Science*, vol. 17, no. 1, pp. 42–61, Jan. 2024, doi: <https://doi.org/10.9734/ajrcos/2024/v17i1411>.
- [6]
G. Rosa, S. Scalabrino, G. Bavota, and R. Oliveto, “What Quality Aspects Influence the Adoption of Docker Images?,” *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, pp. 1–30, Sep. 2023, doi: <https://doi.org/10.1145/3603111>.
- [7]
H. Shi, L. Ying, L. Chen, H. Duan, M. Liu, and Z. Xue, “Dr. Docker: A Large-Scale Security Measurement of Docker Image Ecosystem,” pp. 2813–2823, Apr. 2025, doi: <https://doi.org/10.1145/3696410.3714653>.
- [8]
R. Queiroz, T. Cruz, J. Mendes, P. Sousa, and Paulo Simões, “Container-based Virtualization for Real-Time Industrial Systems – A Systematic Review,” *ACM Computing Surveys*, Aug. 2023, doi: <https://doi.org/10.1145/3617591>.
- [9]
R. Berta, F. Bellotti, A. De Gloria, and L. Lazzaroni, “Assessing Versatility of a Generic End-to-End Platform for IoT Ecosystem Applications,” *Sensors*, vol. 22, no. 3, p. 713, Jan. 2022, doi: <https://doi.org/10.3390/s22030713>.
- [10]
J. Mäkinen, “Improving Software Deployment for Mobile Machinery: A Case Study and Proof of Concept Implementation,” *Aalto.fi*, Jan. 22, 2024. <https://aaltodoc.aalto.fi/items/3f011f56-31b0-4b27-8426-eb3feabfa206> (accessed Dec. 03, 2025).
- [11]
E. Dritisas and M. Trigka, “A Survey on the Applications of Cloud Computing in the Industrial Internet of Things,” *Big Data and Cognitive Computing*, vol. 9, no. 2, pp. 44–44, Feb. 2025, doi: <https://doi.org/10.3390/bdcc9020044>.
- [12]
D. Kisina, Oyinomomo-emi Emmanuel Akpe, S. Owoade, Bright Chibunna Ubanadu, Toluwase Peter Gbenle, and Oluwasanmi Segun Adanigbo, “Advances in CI/CD Pipeline

Resilience for Airline Reservation and Customer Experience Systems,” *International Journal of Multidisciplinary Research and Growth Evaluation*, vol. 4, no. 1, pp. 1254–1261, Jan. 2023, doi: <https://doi.org/10.54660/ijmrge.2023.3.1.1254-1261>.

[13]

P. Kusuma, “A Holistic Framework for Designing Secure, Scalable, and Cost-Effective Cloud-Based E-Commerce Platforms,” *Journal of Advances in Cybersecurity Science, Threat Intelligence, and Countermeasures*, vol. 6, no. 12, pp. 7–16, 2022, Available: <http://polarpublishations.com/index.php/JACSTIC/article/view/2022-12-07>

[14]

Florin Olariu and Lenuța Alboaie, “Challenges In Optimizing Migration Costs From On-Premises To Microsoft Azure,” *Procedia Computer Science*, vol. 225, pp. 3649–3659, Jan. 2023, doi: <https://doi.org/10.1016/j.procs.2023.10.360>.

[15]

Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, “Load Balancing in Cloud Environment: A State-of-the-Art Review,” *IEEE Access*, vol. 11, pp. 134517–134530, 2023, doi: <https://doi.org/10.1109/access.2023.3337146>.

[16]

F. Rahman, “Hardening A Web Server Infrastructure : An Applied Study of TLS, Reverse Proxy Security, and Attack Simulations,” *Theseus.fi*, 2025, doi: <http://www.theseus.fi/handle/10024/894672>.

[17]

R. P. Shermy and N. Saranya, “Cloud-Based Big Data Architecture and Infrastructure,” *Resilient Community Microgrids*, pp. 131–188, Mar. 2025, doi: <https://doi.org/10.1002/9781394272549.ch6>.

[18]

N. Pala, “Understanding Event-Driven Architecture: A Framework for Scalable and Resilient Systems,” *International Journal on Science and Technology (IJSAT) IJSAT25012921*, vol. 16, no. 1, 2025, Available: <https://www.ijst.org/papers/2025/1/2921.pdf>

[19]

K. Acharya, “Assessing the Resilience of Adaptive Intrusion Prevention Systems in SaaS-Driven E-Retail Ecosystems,” *Journal of Emerging Cloud Technologies and Cross-Platform Integration Paradigms*, vol. 6, no. 12, pp. 1–11, 2022, Available: <http://hammingate.com/index.php/JECTCIP/article/view/2022-12-04>

[20]

M. Saadoon, S. H. Ab. Hamid, H. Sofian, H. H. M. Altarturi, Z. H. Azizul, and N. Nasuha, “Fault tolerance in big data storage and processing systems: A review on challenges and solutions,” *Ain Shams Engineering Journal*, vol. 13, no. 2, p. 101538, Mar. 2022, doi: <https://doi.org/10.1016/j.asej.2021.06.024>.

[21]

M. Pûlpán, “Development and deployment of SaaS solution for logistics automation in e-commerce,” *Cuni.cz*, Dec. 2027, doi: <http://hdl.handle.net/20.500.11956/190692>.

[22]

B. Pismenny *et al.*, “Securitization of cloud, edge and IoT communications through hardware accelerations/offloadings,” *Data Archiving and Networked Services (DANS)*, Sep. 2021, doi: <https://doi.org/10.1109/meditcom49071.2021.9647500>.

[23]

Farida Siddiqi Prity and Md. Maruf Hossain, “A comprehensive examination of load balancing algorithms in cloud environments: a systematic literature review, comparative analysis, taxonomy, open challenges, and future trends,” *Iran Journal of Computer Science*, Apr. 2024, doi: <https://doi.org/10.1007/s42044-024-00183-y>.

[24]

K. S. Sahoo, M. Tiwary, A. Kr. Luhach, A. Nayyar, K.-K. R. Choo, and M. Bilal, “Demand-Supply Based Economic Model for Resource Provisioning in Industrial IoT Traffic,” *IEEE Internet of Things Journal*, pp. 1–1, 2021, doi: <https://doi.org/10.1109/jiot.2021.3122255>.

[25]

J. Yang, Z. Jia, R. Su, X. Wu, and J. Qin, “Improved Fault-Tolerant Consensus Based on the PBFT Algorithm,” *IEEE Access*, vol. 10, pp. 30274–30283, Jan. 2022, doi: <https://doi.org/10.1109/access.2022.3153701>.

[26]

M. M. Islam, “THE IMPACT OF DATA-DRIVEN WEB FRAMEWORKS ON PERFORMANCE AND SCALABILITY OF U.S. ENTERPRISE APPLICATIONS,” vol. 05, no. 03, pp. 523–558, Sep. 2025, doi: <https://doi.org/10.63125/f07n4p12>.

[27]

A. Dalal, “Exploring Emerging Trends in Cloud Computing and Their Impact on Enterprise Innovation,” *SSRN Electronic Journal*, Jan. 2025, doi: <https://doi.org/10.2139/ssrn.5268114>.

[28]

V. Mohammadian, N. J. Navimipour, M. Hosseinzadeh, and A. Darwesh, “Fault-Tolerant Load Balancing in Cloud Computing: A Systematic Literature Review,” *IEEE Access*, vol. 10, pp. 12714–12731, 2022, doi: <https://doi.org/10.1109/access.2021.3139730>.

[29]

M. Amman, “Serverless vs. traditional computing : a multi-dimensional comparative analysis,” *Lutpub.lut.fi*, 2025, doi: <https://lutpub.lut.fi/handle/10024/170251>.

[30]

C. Lin, Y. Ma, and M. Shahrad, “Demystifying Serverless Costs on Public Platforms: Bridging Billing, Architecture, and OS Scheduling.” Accessed: Dec. 03, 2025. [Online]. Available: https://cirrus.ece.ubc.ca/papers/eurosys26_lin.pdf

[31]

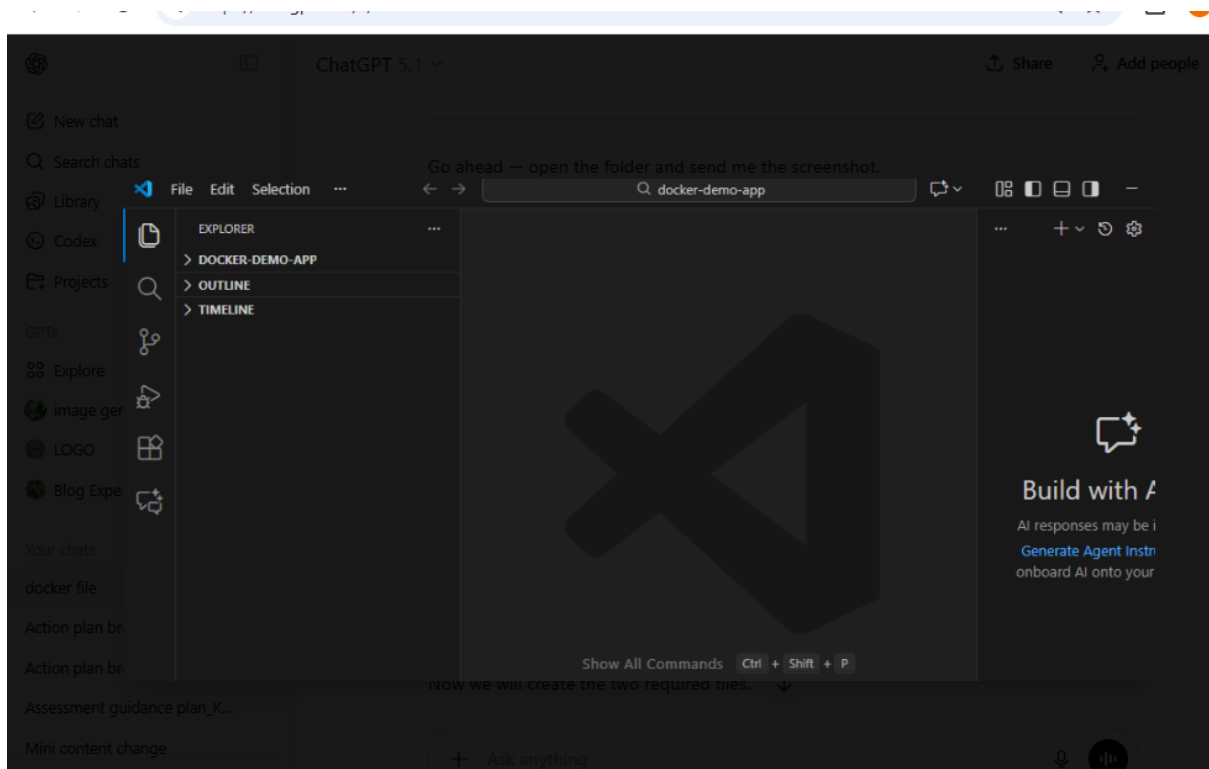
Sai, “Containerization,” *Advances in systems analysis, software engineering, and high performance computing book series*, pp. 98–122, Apr. 2024, doi: <https://doi.org/10.4018/979-8-3693-1682-5.ch006>.

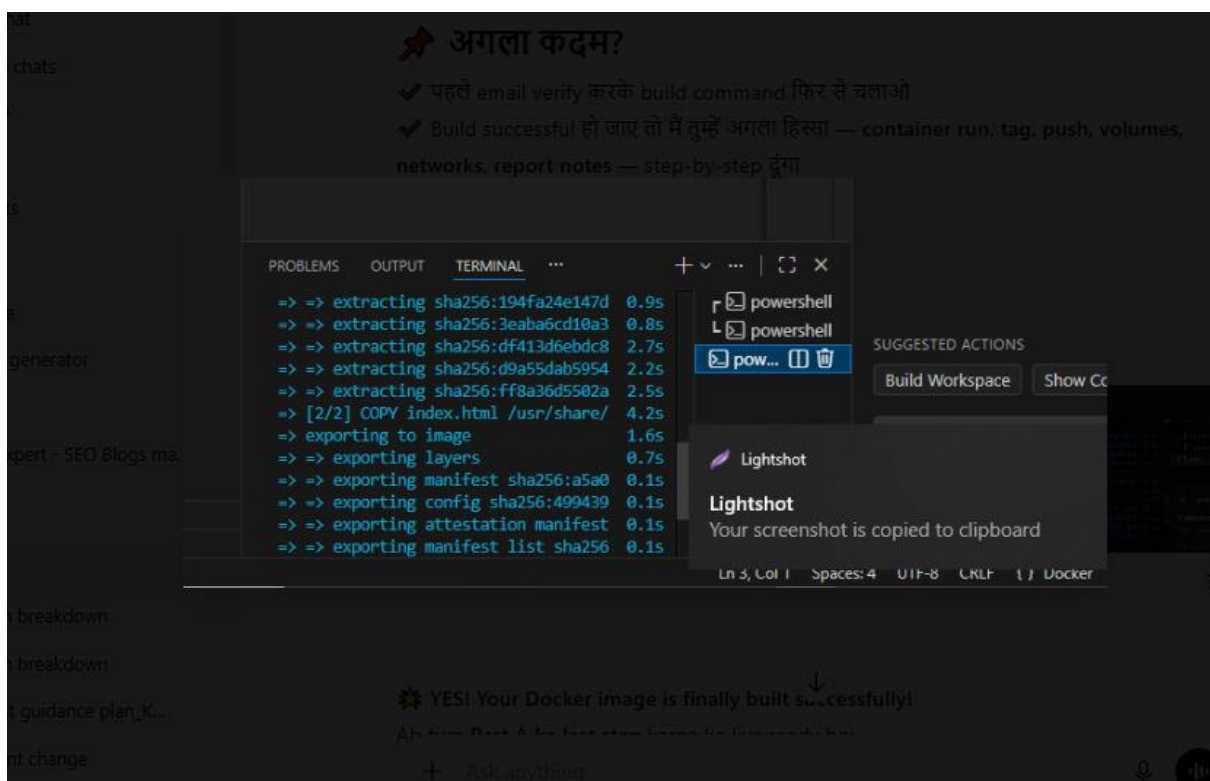
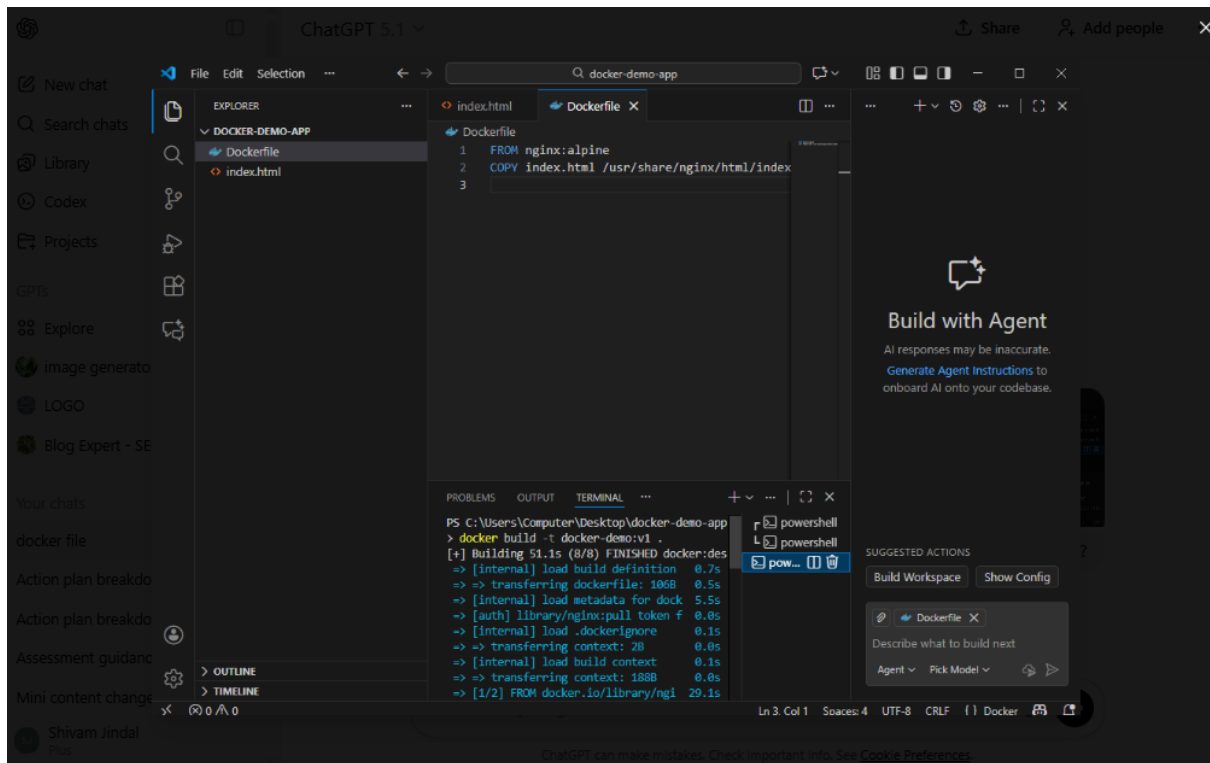
Appendix



docker-demo-app.zip

Appendix A — Docker Commands & Outputs





PROBLEMS OUTPUT **TERMINAL** ... + v ... | [] X

```
PS C:\Users\Computer\Desktop\docker-demo-app
> docker images
>>
```

IMAGE	ID	DISK USAGE
docker-demo:v1	aad2b4f585d4	79.8MB

```
PS C:\Users\Computer\Desktop\docker-demo-app
>
```

Ln 3, Col 1 Spaces: 4 UTF-8

```
PS C:\Users\Computer\Desktop\docker-demo-app
> docker run -d -p 8080:80 docker-demo:v1
>>
0f06d28fa59a02b09484c000e043ba26e5c800bb1009
b6aedc2719329746ad7b
PS C:\Users\Computer\Desktop\docker-demo-app
>
```

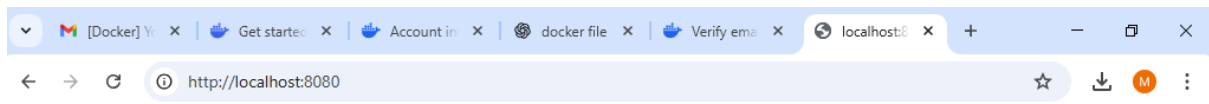
Ln 3, Col 1 Spaces:

```
> docker ps
>>
```

CONTAINER ID	IMAGE	COMMAND	STATUS
0f06d28fa59a	docker-demo:v1	"/docker-ent rypoint...."	About a minute ago Up About a minute

```
PS C:\Users\Computer\Desktop\docker-demo-app
>
```

Ln 3, Col 1 Spaces: 4 UTF-8 CRL



Docker Demo Web App

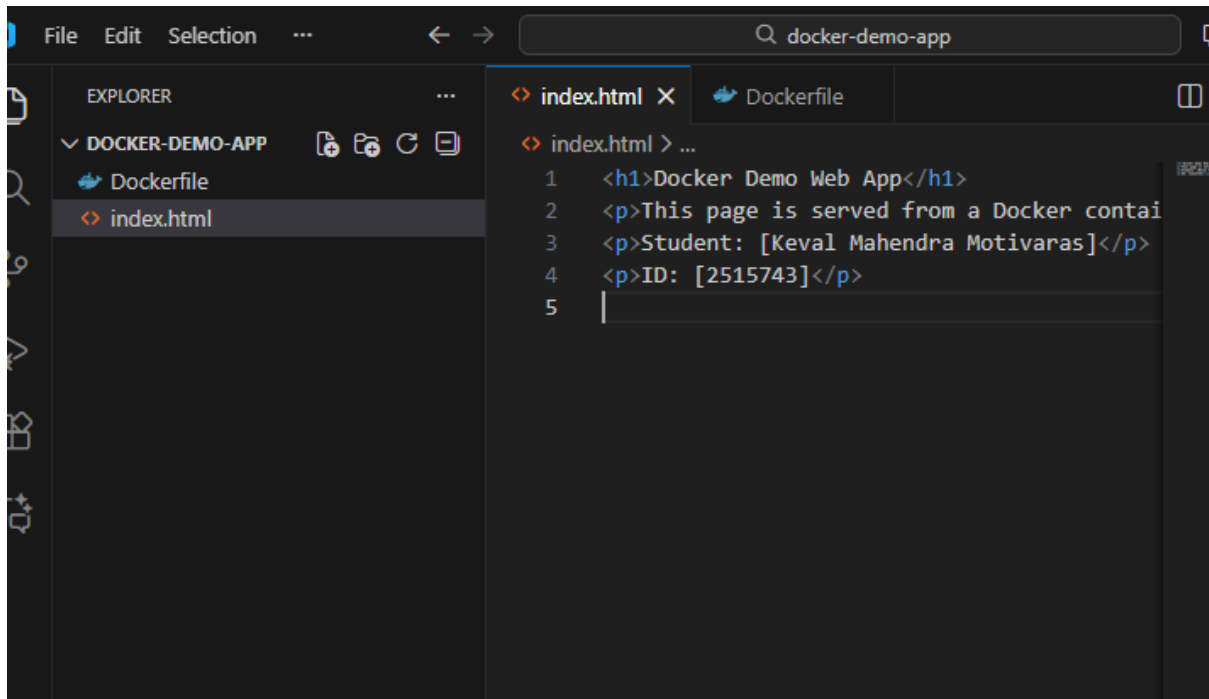
This page is served from a Docker container.

Student: [Keval Mahendra Motivaras]

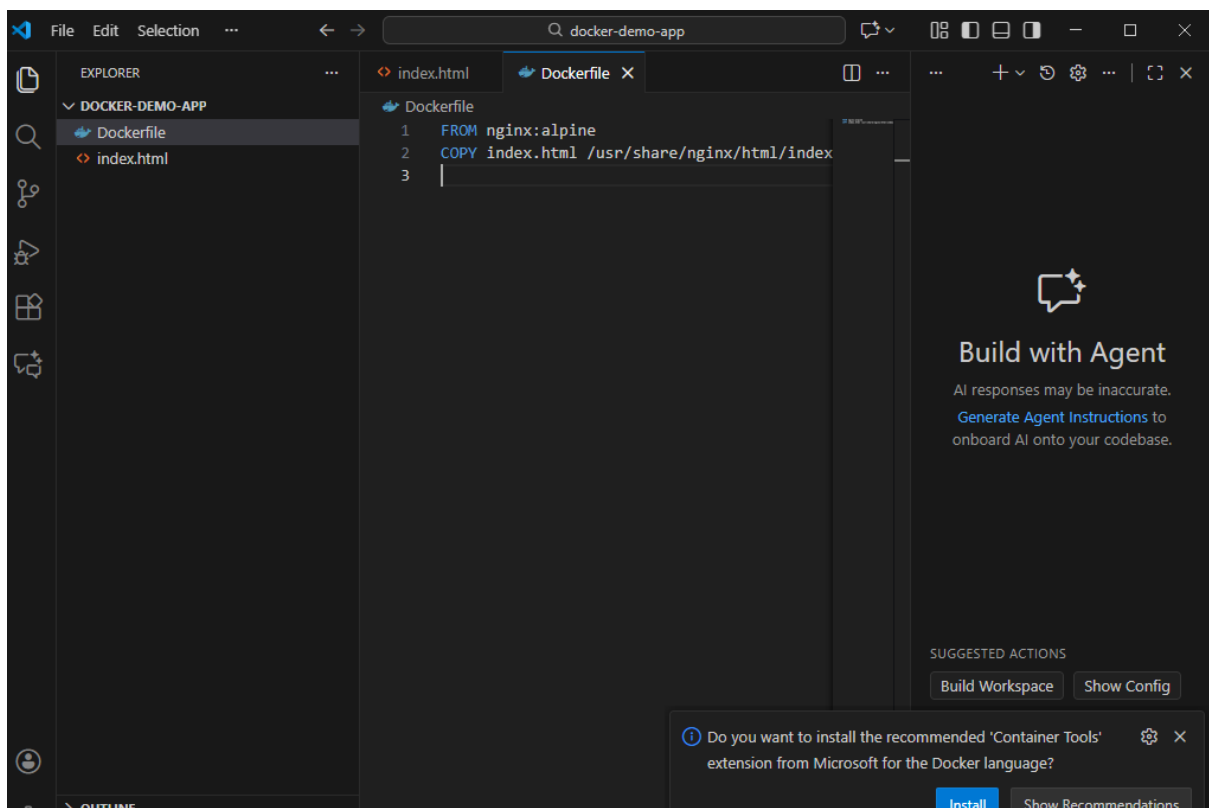
ID: [2515743]

A screenshot of the Docker Desktop application interface. The left sidebar contains a navigation menu with items: 'Ask Gordon BETA', 'Containers', 'Images' (selected), 'Volumes', 'Kubernetes', 'Builds', 'Models', 'MCP Toolkit BETA', 'Docker Hub', 'Docker Scout', and 'Extensions'. The main panel is titled 'Images' and has a sub-tab 'Local' selected. It shows '57.21 MB / 0 Bytes in use' and '1 images'. Below this is a search bar and a table of images. The table has columns: 'Name', 'Tag', 'Image ID', 'Created', 'Size', and 'Actions'. One image is listed: 'docker v1' with ID 'aad2b4f585d4' and size '79.84 MB'. Below the table, there is a 'Walkthroughs' section with two cards: 'How do I run a container?' (6 mins) and 'Run Docker Hub images' (5 mins). A link 'View more in the Learning center' is at the bottom.

Appendix B — index.html Source File



Appendix C — Dockerfile



Appendix D — Diagram for Part B