CLOUD COMPUTING TECHNOLOGY PRACTICUM REPORT

DOCKER

CLOUD COMPUTING TECHNOLOGY PRACTICUM

PLUG - H

By:

Nama : **Sabila Khairina Saktiwati**

NIM : **123220085**

INFORMATICS STUDY PROGRAM

DEPARTMENT OF INFORMATICS

FACULTY OF INDUSTRIAL ENGINEERING

UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”

YOGYAKARTA

2025

APPROVAL PAGE

HALAMAN PERSETUJUAN

PRACTICUM REPORT

CLOUD SQL AND CLOUD STORAGE

CLOUD COMPUTING TECHNOLOGY PRACTICUM

PLUG - H

By:

Sabila Khairina Saktiwati 123220085

Checked and approved by the Practicum Assistant of Cloud Computing Technology.

At the date of: ………………….

.

Approved by.

Practicum Assistant	**Practicum Assistant**
Muhammad Rafli	**Sayang Sani**
NIM 123210078	**NIM 123210044**

PREFACE

Praise and gratitude are expressed to the presence of God Almighty for His blessings and grace, which have enabled the completion of this practicum report. This report is prepared as part of fulfilling academic requirements and as a form of accountability for the practicum activities that have been carried out.

The author would like to express sincere appreciation to Muhammad Rafli, and Sayang Sani for their guidance and assistance throughout the practicum.

The author is fully aware that this report is still far from perfect. Therefore, constructive criticism and suggestions are highly expected for future improvements. Hopefully, this report can provide benefits to all relevant parties.

Yogyakarta, 23 March 2025

Author

TABLE OF CONTENTS

APPROVAL PAGE		ii
HALAMAN PERSETUJUAN		ii
PREFACE		3
TABLE OF CONTENTS		iv
LIST OF FIGURE		vi
CHAPTER I INTRODUCTION		7
1.1	Background	7
1.2	Problem Formulation	8
1.3	Objectives	8
1.4	Benefits	8
CHAPTER II LITERATURE REVIEW		9
2.1	Docker	9
2.2	Google Cloud Run	9
2.3	Google Cloud Artifact Registry	10
CHAPTER III METHODOLOGY		12
3.1	Problem Analysis	12
3.2	Solution Design	12
3.2.1	Clone Repository GitHub	12
3.2.2	Docker for Backend	14
3.2.3	Push and Deploy Image	15
3.2.4	Docker for Frontend	17
CHAPTER IV RESULTS AND DISCUSSION		19
4.1	Results	19

4.1.1 Deploying BE to Cloud Run 19

4.1.2 Running FE via Docker Container 20

4.2 Discussion 21

CHAPTER V CLOSING 22
5.1 Conclusion 22

5.2 Suggestions 22

BIBLIOGRAPHY 23
APPENDICES 24

LIST OF FIGURE

Figure 1. Step 3.2.1.2 13

Figure 2. Step 3.2.1.3 14

Figure 3. Step 3.2.1.4 14

Figure 4. Step 3.2.2.1 15

Figure 5. Step 3.2.2.2 15

Figure 6. Step 3.2.2.3 15

Figure 7. Step 3.2.3.1 16

Figure 8. Step 3.2.3.2 16

Figure 9. Step 3.2.3.4 16

Figure 10. Step 3.2.3.5 17

Figure 11. Step 3.2.4.1 17

Figure 12. Step 3.2.4.2 17

Figure 13. Step 3.2.4.3 17

Figure 14. Step 3.2.4.4 18

Figure 15. Step 3.2.4.5 18

Figure 16. Deploying Backend to Cloud Run.. 19

Figure 17. Running Frontend via Docker Container................................ 20

CHAPTER I
INTRODUCTION

1.1 Background

Cloud computing has revolutionized the way applications are deployed and managed. It provides scalable, on-demand computing resources without the need for maintaining physical infrastructure. One of the most efficient methods for deploying applications in a cloud environment is through containerization. Containerization allows applications to run consistently across different computing environments by packaging the application and its dependencies into a single, portable unit.

Docker, a widely used containerization tool, streamlines application development and deployment by ensuring uniformity across different platforms. It eliminates common issues related to dependency management, version conflicts, and compatibility across various development and production environments. By using Docker, developers can create lightweight, portable applications that can be seamlessly deployed across cloud platforms.

With the increasing adoption of microservices architecture, separating the Back-End (BE) and Front-End (FE) deployment has become a common industry practice. Google Cloud Run, a serverless platform, offers an efficient way to deploy and manage containerized Back-End applications without requiring manual infrastructure management. By leveraging Cloud Run, developers can deploy their applications with high availability, automatic scaling, and cost efficiency.

On the other hand, Front-End applications can also be containerized and run in an isolated environment using Docker. Running the FE container in Cloud Shell provides a controlled and reproducible development environment without the need for separate hosting infrastructure. This approach enhances application performance, scalability, and maintainability.

Several studies, including those by Potdar et al. (2020) and Singh & Singh (2016), highlight the advantages of using Docker over traditional virtual machines in terms of efficiency and resource utilization. Furthermore, deployment best practices documented by Google Cloud (2025) and Saha (2024) provide essential insights into containerization strategies. This assignment focuses on implementing these techniques by deploying the BE of a Notes-themed application to Cloud Run while running the FE via Docker in Cloud Shell.

1.2 Problem Formulation

1. How can a Back-End service be containerized and deployed to Cloud Run efficiently?
2. What are the steps required to run a Front-End container via Docker in Cloud Shell?

1.3 Objectives

1. To create Docker images for both the Back-End and Front-End of the Notes Application.
2. To deploy the Back-End service to Cloud Run for a scalable and serverless solution.
3. To run the Front-End container in Cloud Shell without deploying it to Cloud Run.

1.4 Benefits

1. Users can access the Back-End service deployed on Cloud Run from any device without managing servers.
2. Users experience a seamless and scalable application performance with optimized resource utilization.
3. Users benefit from enhanced portability, ensuring consistent functionality across different environments.
4. Users gain a reliable and efficient application with minimal downtime through cloud-based deployment strategies.

CHAPTER II

LITERATURE REVIEW

2.1 Docker

Docker is a leading containerization technology that enables developers to package applications and their dependencies into standardized units called containers. According to Potdar et al. (2020), Docker provides significant performance improvements over traditional virtual machines by reducing resource overhead and enhancing application portability. Singh & Singh (2016) further emphasize the role of Docker in modern cloud environments, highlighting its ability to facilitate rapid development, testing, and deployment cycles. Additionally, Saha (2024) discusses how Docker can be used effectively for frontend development environments, ensuring consistency across different stages of the software development lifecycle.

Docker provides various advantages, including improved application isolation, scalability, and resource efficiency. Compared to virtual machines, Docker containers consume fewer resources since they share the host operating system rather than requiring a separate OS instance. Furthermore, Docker enables CI/CD (Continuous Integration/Continuous Deployment) pipelines, allowing faster and more efficient software releases. The ability to create and manage multi-container applications using Docker Compose further enhances its appeal for complex applications.

2.2 Google Cloud Run

Google Cloud Run is a fully managed serverless computing platform designed for running containerized applications. It allows developers to deploy and scale applications without managing infrastructure. According to Google Cloud (2025), Cloud Run provides automatic scaling, high availability, and cost-efficient pricing models, making it an attractive option for cloud-based application deployment. Achsan & Affandi (2023) also discuss the benefits of serverless platforms like Cloud Run, noting that they reduce operational complexity while enhancing scalability and performance.

One of the key advantages of Cloud Run is its ability to scale applications dynamically based on demand. When no requests are being processed, Cloud Run scales down to zero, reducing operational costs. This feature makes it an ideal choice for applications with variable workloads. Additionally, Cloud Run integrates seamlessly with other Google Cloud services, such as Cloud SQL, Cloud Storage, and Cloud Logging, enabling a more robust cloud-native ecosystem. The security features of Cloud Run, including built-in IAM (Identity and Access Management) controls and automatic TLS (Transport Layer Security) encryption, further enhance the reliability of deployed applications.

2.3 Google Cloud Artifact Registry

Google Cloud Artifact Registry is a managed storage solution for container images and software packages. It enables secure and efficient management of container artifacts. According to Google Cloud (2025), Artifact Registry integrates seamlessly with Cloud Run and Kubernetes, allowing for streamlined deployment workflows. Potdar et al. (2020) also highlight the importance of artifact management in containerized environments, emphasizing the need for secure and reliable storage of container images to ensure smooth deployment and version control.

Artifact Registry supports multiple artifact formats, including Docker images, npm packages, Maven repositories, and Python packages, making it a versatile tool for developers. It offers advanced security features such as vulnerability scanning, ensuring that container images stored in the registry are free from known security threats. Additionally, it supports automated synchronization with CI/CD pipelines, enabling seamless updates to containerized applications. By integrating Artifact Registry with Google Kubernetes Engine (GKE) and Cloud Build, developers can further optimize their cloud development and deployment workflows.

Another notable feature of Artifact Registry is its regional replication capability, which enhances reliability and access speed. Organizations can store container images in multiple locations, ensuring fast retrieval and reducing latency for global deployments. By using Artifact Registry, developers and DevOps teams can maintain a structured and secure container image repository, improving overall software management and deployment efficiency.

CHAPTER III
METHODOLOGY

3.1 Problem Analysis

Deploying containerized applications to the cloud presents several key challenges, particularly in terms of infrastructure configuration, security, and resource optimization. One of the primary difficulties is ensuring that container images are properly built and stored in a secure environment, such as Google Cloud Artifact Registry, before deployment. Additionally, managing the networking and scaling configurations in Google Cloud Run requires careful consideration to balance performance and cost-effectiveness. Another challenge is ensuring smooth integration between the Back-End and Front-End components, as the BE will be hosted on Cloud Run while the FE runs within a Docker container in Cloud Shell. Security concerns such as image vulnerability scanning, access controls, and IAM (Identity and Access Management) policies must also be addressed to protect application data and prevent unauthorized access. Furthermore, troubleshooting and monitoring containerized applications in a cloud environment require expertise in logging, debugging, and performance analysis tools provided by Google Cloud. Overcoming these challenges is essential for achieving a smooth, scalable, and efficient deployment process.

3.2 Solution Design

3.2.1 Clone Repository GitHub

1. Go to Link https://github.com/stroberinanas/123220085-TCCPract.git
2. Open Google Cloud. Choose a Project

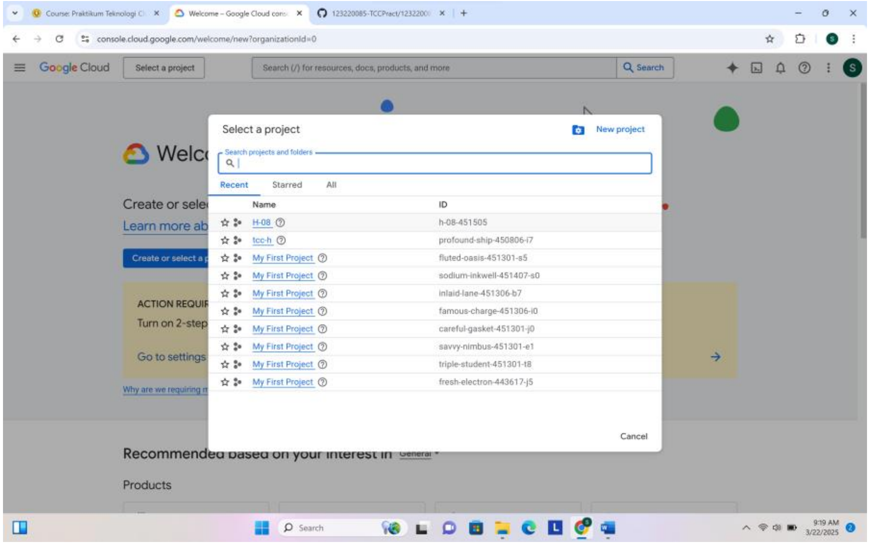
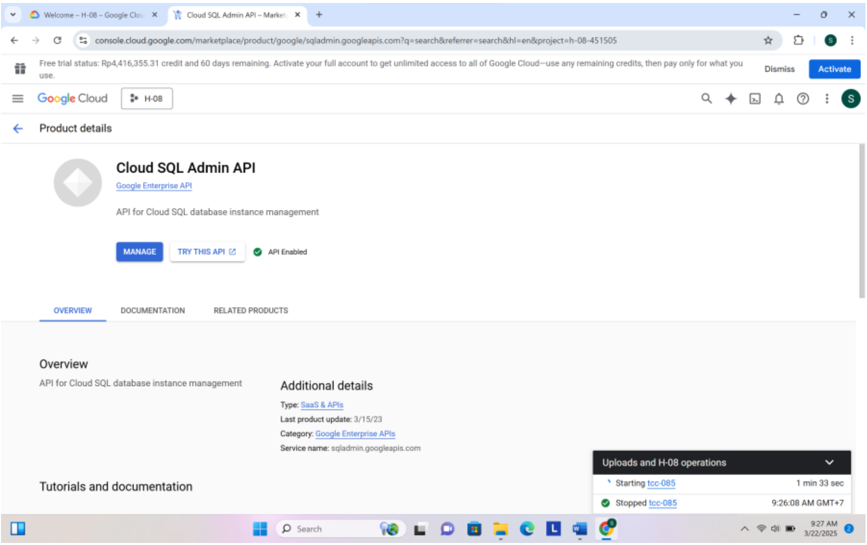


Figure 1. Step 3.2.1.2

3. Enable Cloud SQL Admin API and Artifact Registry API



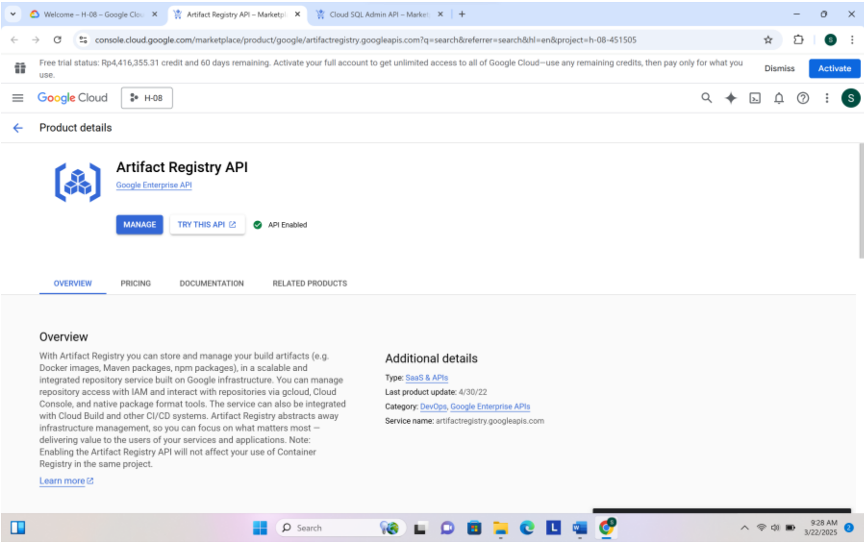


Figure 2. Step 3.2.1.3

4. Open Cloud shell, Authorize Access, and Clone Repository in Google Cloud Shell

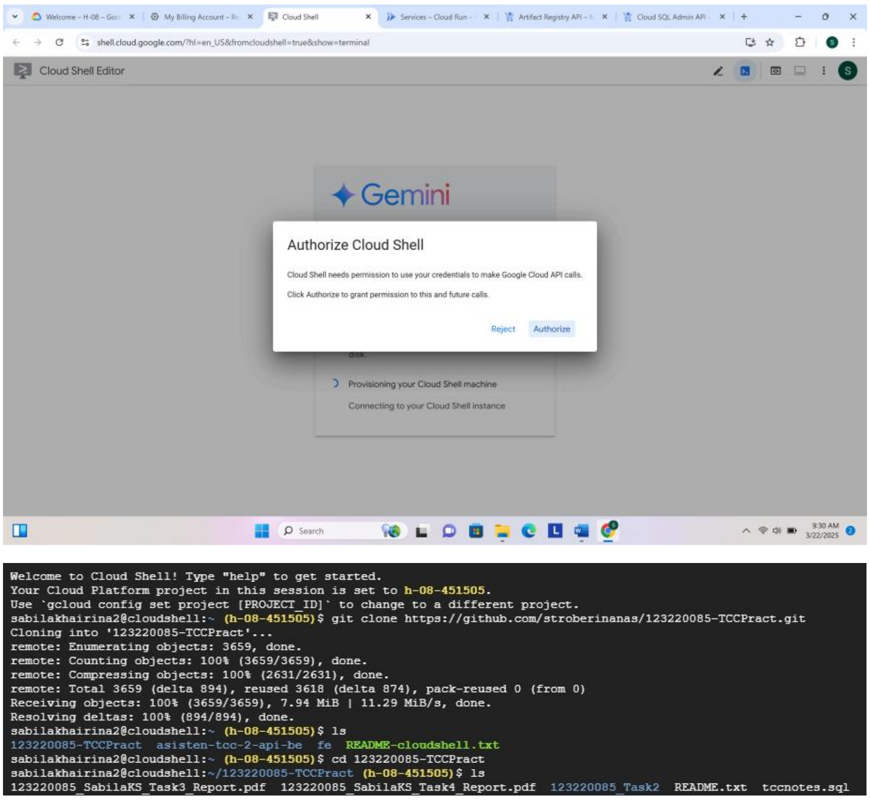


Figure 3. Step 3.2.1.4

3.2.2 Docker for Backend

1. Create a Dockerfile in the folder

Figure 4. Step 3.2.2.1

2. Build and Run the Dockerfile with the appropriate port settings

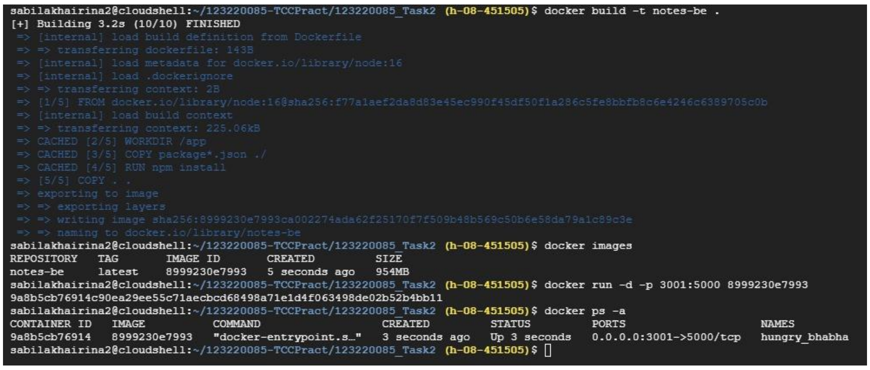


Figure 5. Step 3.2.2.2

3. Preview the web and modify the port

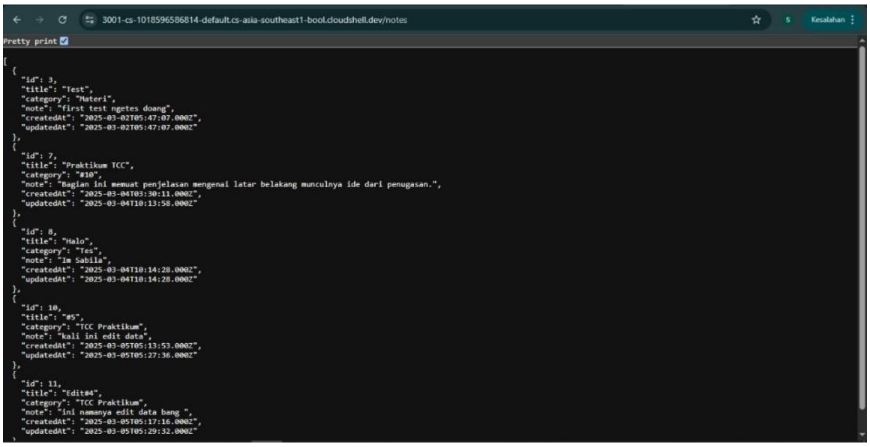


Figure 6. Step 3.2.2.3

3.2.3 Push and Deploy Image

1. Open Artifact Registry. Select repository

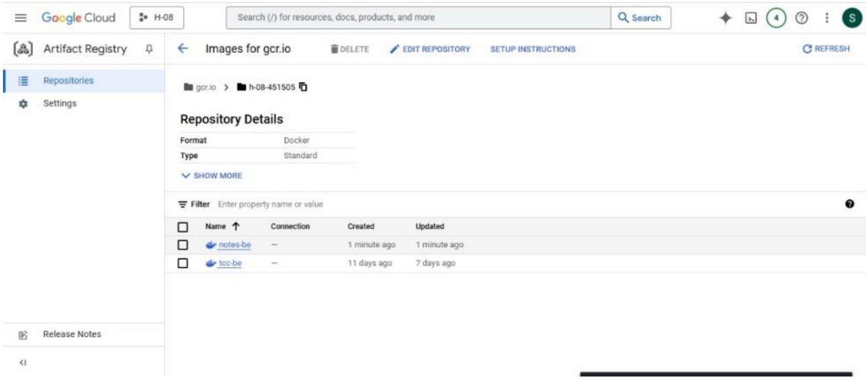


Figure 7. Step 3.2.3.1

2. Push the Docker image to google Container based on project ID in artifact registry

```
sabilakhairina2@cloudshell:~/123220085-TCCPract/123220085_Task2 (h-08-451505)$ docker images
REPOSITORY   TAG      IMAGE ID       CREATED         SIZE
notes-be     latest   8999230e7993   7 minutes ago   954MB
sabilakhairina2@cloudshell:~/123220085-TCCPract/123220085_Task2 (h-08-451505)$ docker tag notes-be gcr.io/h-08-451505/notes-be
sabilakhairina2@cloudshell:~/123220085-TCCPract/123220085_Task2 (h-08-451505)$ docker push gcr.io/h-08-451505/notes-be
Using default tag: latest
The push refers to repository [gcr.io/h-08-451505/notes-be]
104a3a443f06: Pushed
de7c155cb128: Pushed
ac054a85d9f4: Pushed
5d3560be3124: Pushed
be322b479aee: Layer already exists
d41bcd3a037b: Layer already exists
fe0d845e767b: Layer already exists
f25ec1d93a58: Layer already exists
794ce8b1b516: Layer already exists
3220beed9b06: Layer already exists
684f82921421: Layer already exists
9af5f53e8f62: Layer already exists
latest: digest: sha256:5e99e95bb2c0bcb3ecd6e03ee6492d03586f623461c56f8cb92bde07b31321db size: 2841
sabilakhairina2@cloudshell:~/123220085-TCCPract/123220085_Task2 (h-08-451505)$
```

Figure 8. Step 3.2.3.2

3. Open Google Cloud Run and Create a Service
4. Configure settings, including select the container image from artifact registry that have pushed

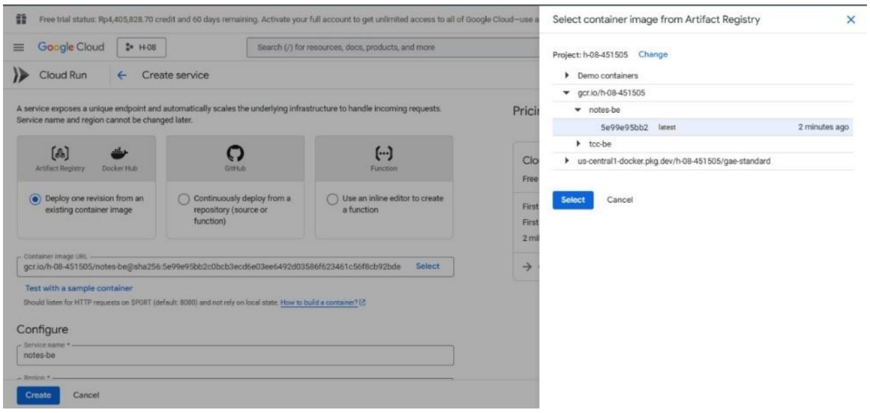


Figure 9. Step 3.2.3.4

5. Click Create to launch the service

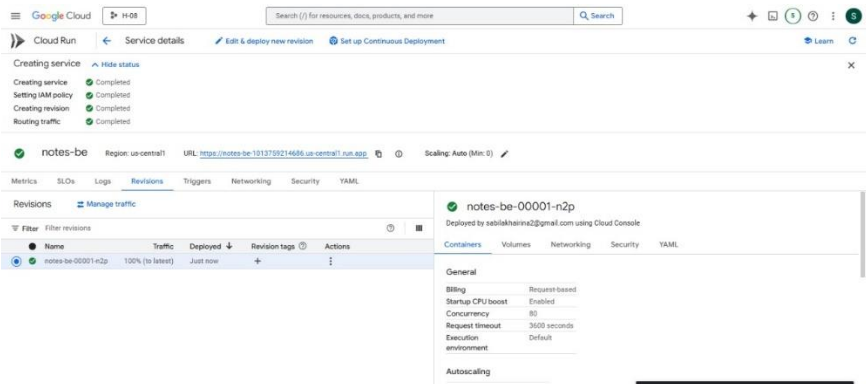


Figure 10. Step 3.2.3.5

3.2.4 Docker for Frontend

1. Open Cloud Shell, create file `utils.js`

Figure 11. Step 3.2.4.1

2. Paste url from the Cloud run deployment that have made before

Figure 12. Step 3.2.4.2

3. Create a Dockerfile in frontend folder

Figure 13. Step 3.2.4.3

4. Build and Run the frontend Dockerfile

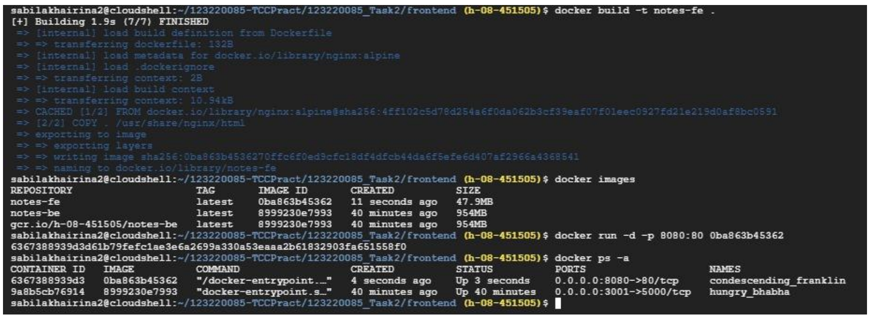


Figure 14. Step 3.2.4.4

5. Preview Web and change port that suitable with that config

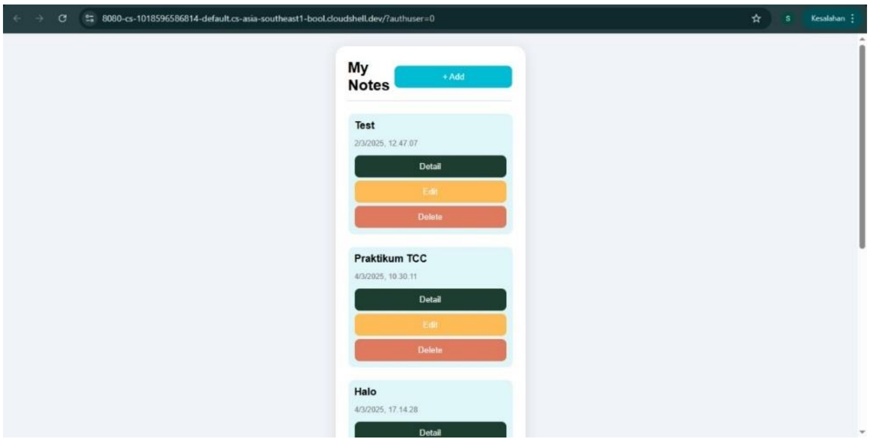


Figure 15. Step 3.2.4.5

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Results

4.1.1 Deploying BE to Cloud Run

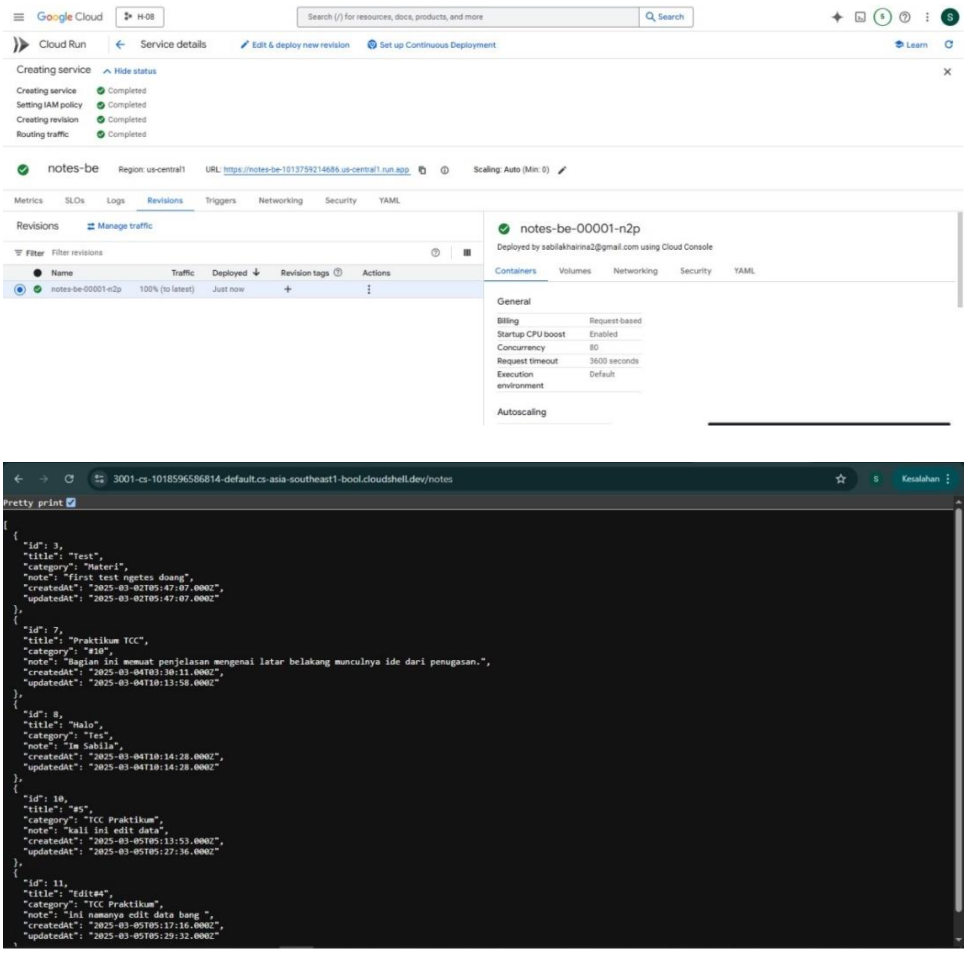


Figure 16. Deploying Backend to Cloud Run

The deployment of the Back-End (BE) to Google Cloud Run was successfully completed. The Cloud Run service details indicate that all setup processes, including service creation, IAM policy configuration, revision creation, and traffic routing, have been marked as "Completed." Additionally, the deployed service "notes-be" is running with request-based billing, concurrency settings, and an enabled startup CPU boost, confirming its operational status. Next image validates this successful deployment, showing the API response from the backend running on Cloud Shell, which correctly retrieves and displays stored notes data in a structured JSON format. This confirms that the backend is functioning as intended and is accessible via the deployed Cloud Run service.

4.1.2 Running FE via Docker Container

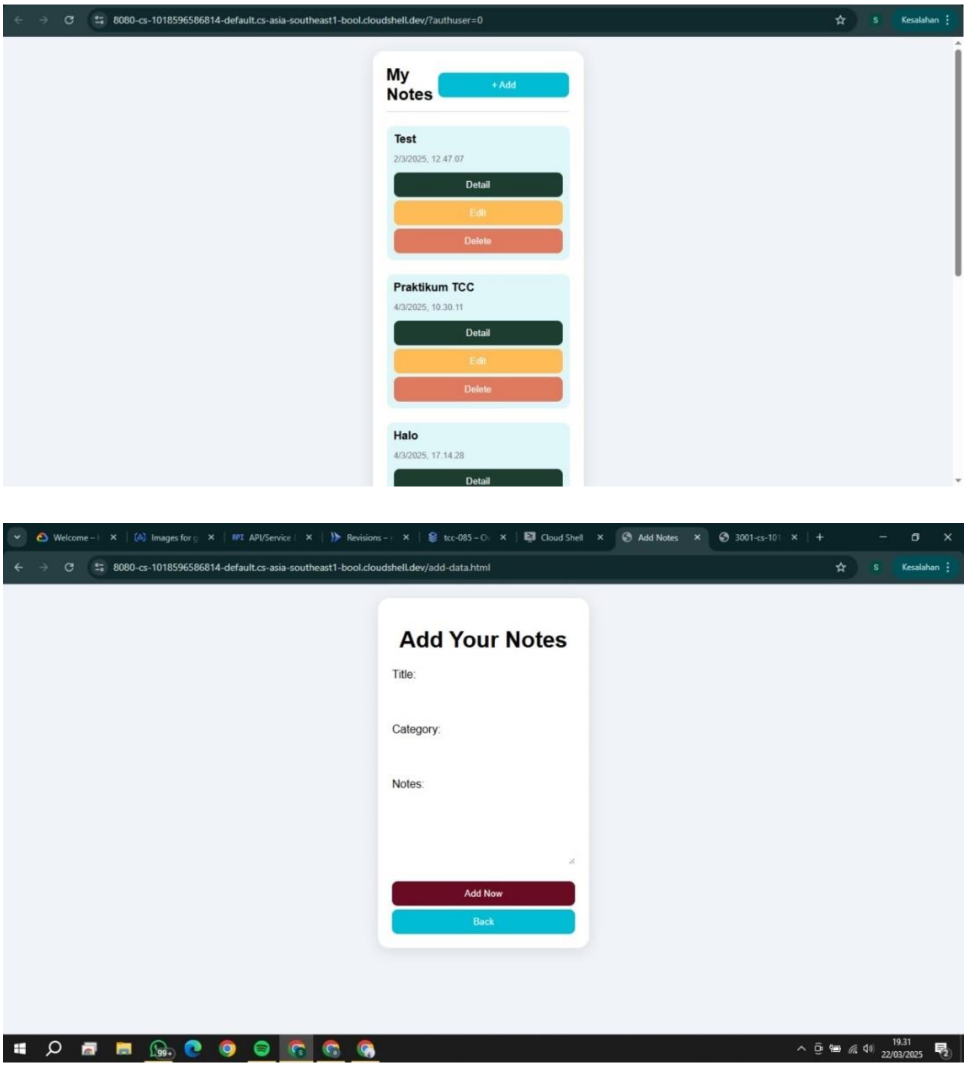


Figure 17. Running Frontend via Docker Container

The web application is running on a Cloud Shell environment with a functional user interface displaying stored notes along with options to add, edit, and delete entries. The correct retrieval and rendering of notes from the backend confirm that the FE container is effectively communicating with the deployed backend service on Cloud Run. Additionally, the structured layout and interactive buttons indicate that the FE Docker container has been correctly built and executed, ensuring seamless functionality within the cloud environment.

4.2 Discussion

The deployment process of the Back-End (BE) to Cloud Run and the Front-End (FE) via Docker Container was successfully executed, demonstrating the effectiveness of containerization in cloud-based applications. Deploying the BE to Cloud Run provided automatic scaling, high availability, and efficient resource management, as seen from the successful service creation and request handling. Running the FE via Docker ensured a controlled and reproducible environment, with the application correctly displaying stored data. However, several challenges were encountered, including configuring port mappings, ensuring smooth communication between BE and FE, and managing dependencies within containers. Debugging network-related issues between Cloud Run and the FE container required careful attention to API endpoint configurations and security settings.

CHAPTER V
CLOSING

5.1 Conclusion

Overall, the deployment process successfully leveraged cloud and containerization technologies to build a scalable and portable application. Cloud Run facilitated efficient BE hosting without the need for infrastructure management, while Docker provided consistency in the FE environment. Despite the challenges faced, the final deployment ensures smooth application performance and proper interaction between BE and FE.

.

5.2 Suggestions

To enhance the deployment further, it is recommended to implement CI/CD pipelines for automated updates, enforce stronger security policies to protect API endpoints, and integrate monitoring tools like Google Cloud Logging for performance tracking. These improvements will help streamline future updates, improve security, and maintain application stability at scale.

BIBLIOGRAPHY

[1] Potdar, A. M., G, N. D., Shivaraj Kengond, & Mulla, M. M. (2020). Performance Evaluation of Docker Container and Virtual Machine. *Procedia Computer Science*, *171*, 1419–1428. https://doi.org/10.1016/j.procs.2020.04.152

[2] Singh, S., & Singh, N. (2016). *Containers & Docker: Emerging roles & future of Cloud technology*. https://doi.org/10.1109/icatcct.2016.7912109

[3] *Deploying container images to Cloud Run*. (2025). Google Cloud. https://cloud.google.com/run/docs/deploying

[4] Saha, M. (2024, August 13). *How to Use Docker for Frontend Development Environments*. PixelFreeStudio Blog -. https://blog.pixelfreestudio.com/how-to-use-docker-for-frontend-development-environments/

[5] Achsan Rizky Ekaputra, & Arif Saivul Affandi. (2023). Pemanfaatan layanan cloud computing dan docker container untuk meningkatkan kinerja aplikasi web. *Journal of Information System and Application Development*, *1*(2), 138–147. https://doi.org/10.26905/jisad.v1i2.11084

[6] *LinkedIn. (2025). Linkedin.com. https://www.linkedin.com/pulse/deploying-ai-docker-cloudrun-amar-kasbe-sxohf/*

APPENDICES

[1] https://github.com/stroberinanas/123220085-TCCPract.git