Cloud vs. Local Development Stacks: Leveraging CDE and PaaS Platforms for Academic Web Development Labs

Raffi Ihza Zuhairnawan
Departement of Information System
Telkom University
Bandung, Indonesia
raffiihza@student.telkomuniversity.ac.id

Umar Yunan K. S. Hediyanto
Departement of Information System
Telkom University
Bandung, Indonesia
umaryunan@telkomuniversity.ac.id

Muhammad Fathinuddin
Departement of Information System
Telkom University
Bandung, Indonesia
muhammadfathinuddin@telkomuniversity.ac.id

***Abstract*—In the era of the Fourth Industrial Revolution, technical skills in web application development are increasingly crucial for information technology students. This paper presents a comparative study of cloud-based and local development stacks within the context of a Web Application Development practical course at a local academic institution. The study benchmarked four CDEs (GitHub Codespaces, CodeSandbox, Jetify Devspace, and DevZero) and four PaaS platforms (Koyeb, Render, Northflank, and Lade) across parameters including performance, network latency, booting time, and features. A final comparison contrasts these cloud systems with a traditional local setup. The findings indicate that GitHub Codespaces and Koyeb represent the most suitable combination of CDE and PaaS for this practical course due to their technical capabilities, ease of use, and instructional practicality. Furthermore, the cloud-based target system, utilizing GitHub Codespaces and Koyeb, proved superior to the existing local development system across most parameters, offering a more inclusive, efficient, and scalable solution. Future research may incorporate wider platforms and integrations.**

Keywords—Web Application Development, Cloud Development Environment, Platform-as-a-Service, GitHub Codespaces, Koyeb

I. INTRODUCTION

In the Industry 4.0 era, proficiency in web application development has become a fundamental competency for Information Technology students, underpinning a wide range of systems from academic information portals to e-commerce and social media platforms [1], [2]. Practical courses in Web Application Development, such as the one offered at the Information Systems Program in a local academic institution, aim to equip students with hands-on experience using technologies such as PHP, MySQL, and Laravel. However, the traditional requirement for each student to install and configure local toolchains introduces substantial variability and technical overhead that can detract from core learning objectives [3].

Students frequently encounter technical difficulties related to the installation and configuration of programming environments, including issues with operating system compatibility, device specifications, port conflicts, mismatched library versions, hardware constraints, and various technical errors that can impede their learning [3], [4], [5]. These issues consume valuable lab time in troubleshooting, as differences in OS, hardware, and software proficiencies lead to inconsistent learning experiences [6].

Furthermore, the conventional methods of assessing practical work, which often rely on screenshots and manual source code review, have been deemed inefficient [7]. These processes can be time-consuming for both students and instructors, potentially increasing student anxiety and obscuring the intended learning objectives [6].

To address these challenges, this paper explores the implementation and comparative analysis of Cloud Development Environments (CDE) and Platform-as-a-Service (PaaS) platforms within the context of a Web Application Development practical course. CDEs offer pre-configured, cloud-based programming environments that eliminate the need for local installations and configurations, allowing students to begin practical work more efficiently [3], [8]. PaaS platforms enable the direct deployment of web applications to the internet [9], [10], streamlining the assessment process for instructors. Together, they promise a more homogeneous, scalable, and maintainable workflow for both students and instructors. This research specifically investigates and compares four CDE services (GitHub Codespaces, CodeSandbox, Jetify Devspace, and DevZero) and four PaaS platforms (Koyeb, Render, Northflank, and Lade). The evaluation is based on parameters such as performance, network latency, booting time, load time, and features.

This study aims to determine the most suitable CDE and PaaS solutions for a Web Application Development practical course by comparing their technical capabilities, ease of use, and instructional practicality. By analysing the strengths and weaknesses of these cloud-based platforms and contrasting them with traditional local development setups, this paper seeks to provide insights into creating a more inclusive, efficient, and accessible learning environment for web application development. Ultimately, the findings will contribute to enhancing the quality of practical web development education in academic settings.

II. METHODOLOGY

This research employs a case study methodology to analyse and compare the implementation of CDE and PaaS platforms within the context of a Web Application Development practical course at the Information Systems Program.

A. Case Study Method

According to [11], the case study approach allows for an in-depth exploration and understanding of a specific phenomenon in its real-world context. This method is particularly suitable for analysing the effectiveness of technology implementation in an educational setting [12]. The research followed the four stages of the case study method as described by [13]. These stages consist of Foundation Phase, Prefield Phase, Field Phase, and Reporting Phase.

1) Foundation Phase

This initial phase involved a thorough identification and understanding of the challenges associated with the existing practical Web Application Development course, including issues related to the installation and configuration of programming environments, variations in student setups, and the inefficiencies of the traditional assessment methods.

2) Prefield Phase

This phase involved the preparation of operational protocols, the selection of candidate CDE and PaaS platforms, and the design of detailed data-collection instruments such as benchmarks and latency commands.

3) Field Phase

This phase involved the execution the implementations (provision of four CDEs and four PaaS instances) and systematically collect data on each platform's behavior under simulated lab workloads.

4) Reporting Phase

This final stage encompassed the testing and analysis of the implemented CDE and PaaS platforms. The findings were then synthesised into a comprehensive research report.

B. Research Device

All tests were conducted on a system equipped with an Intel® Celeron® N4020 dual-core processor (1.1 GHz base frequency, up to 2.8 GHz burst frequency), 4 GB of RAM, and the Windows 11 operating system to ensure stability and consistency throughout the testing process.

C. Test Scenarios

Data analysis focused on comparing four CDE services, four PaaS, and two systems (existing local system and target cloud-based system). The testing was based on a set of predefined technical and functional parameters, categorized as follows:

1. CDEs: usage quota, performance, build time, network latency, booting time, and features.
2. PaaS: performance, build time, network latency, load time, uptime, and features.
3. Systems: total install time, performance, network latency, booting time, and features.

III. Results and Discussion

This section details a comprehensive testing of selected platforms, divided into three comparative analyses, which are CDEs, PaaS providers, and a comparison between local and cloud-based systems.

A. Comparison Results of CDE Platforms

This section presents the results of the comparative analysis of the four CDE services respectively GitHub Codespaces, CodeSandbox, Jetify Devspace, and DevZero. The comparison is based on the parameters outlined in the methodology which are usage quota, performance, build time, network latency, booting time, and features.

1) Usage Quota

The available usage quotas for the free tiers of each CDE service were compared. The lowest available specifications were selected, as they were sufficient to support the needs of the practical course while offering the maximum possible usage duration. All four CDEs provide ample free-tier usage for a typical lab schedule (±15 hours total over five 3-hour modules), but quotas vary significantly.

TABLE I. CDE Usage Quota Comparison

No	Name	vCPU (core)	RAM (GB)	Storage (GB)	Quota (per month)
1	GitHub Codespaces	2	8	15 per month	60
2	CodeSandbox	1	2	20 per VM	57.14
3	Jetify Devspace	4	16	20 per month	25
4	DevZero	1	4	5 per VM	58.33

As shown in Table I, GitHub Codespaces provides the most extensive usage quota, allowing for prolonged utilization.

2) Performance

Performance was evaluated using single-core and multi-core CPU benchmark tests on their highest free-tier specifications with Geekbench 6 for five runs.

TABLE II. CDE Performance Comparison

No	Name	vCPU (core)	RAM (GB)	Avg Single core	Avg Multi core
1	GitHub Codespaces	4	16	1444	3229.2
2	CodeSandbox	4	8	1226.2	3631.2
3	Jetify Devspace	4	16	716.4	1454.2
4	DevZero	1	4	-	-

In Table II, GitHub Codespaces achieved the highest single-core score and strong multi-core performance. CodeSandbox's extra cores yield the best aggregate multi-core throughput. DevZero failed to complete benchmarks on its 1 vCPU/4 GB instance.

3) Build Time

Build time, the duration required to initialise or build the CDE instance [14], was measured.

TABLE III. CDE Build Time Comparison

No	Name	Build Time
1	GitHub Codespaces	2 min 3 s
2	CodeSandbox	2 min 50 s
3	Jetify Devspace	2 min 19 s
4	DevZero	6 min 15 s

According to Table III, GitHub Codespaces boots the fastest. Codespaces' near-region (Singapore) infrastructure likely contributes to its fast container provisioning. A faster build time contributes to a more seamless user experience, allowing students to commence practical work promptly [14].

4) Network Latency

Network latency, crucial for the responsiveness of the browser-based IDE [15], was assessed by measuring the average latency with HTTP requests (curl) to a local endpoint from within each CDE for five runs.

TABLE IV. CDE Network Latency Comparison

No	Name	Nearest Region	Avg Latency (ms)
1	GitHub Codespaces	Singapore	19.4888
2	CodeSandbox	Germany	227.0240
3	Jetify Devspace	Virginia, US	229.6950
4	DevZero	Portland, US	216.8794

In Table IV, GitHub Codespaces exhibited the lowest average network latency, benefiting from a server region in

Singapore. Only Codespaces remains well under the 100 ms interactivity threshold [16], minimizing typing and UI lag. Low latency is essential for a fluid coding experience, minimising delays during interaction with the CDE [15].

5) Booting Time

Booting time, the time from activating the CDE until it is ready for use [17], was measured.

TABLE V. CDE BOOTING TIME COMPARISON

No	Name	Booting Time
1	GitHub Codespaces	32 s
2	CodeSandbox	27 s
3	Jetify Devspace	58 s
4	DevZero	1 min 16 s

According to Table V, CodeSandbox was the fastest for boot time. CodeSandbox displayed the fastest booting time. CodeSandbox's AWS Firecracker micro-VMs explain its rapid spin-up [18]. A quicker booting time enhances the efficiency and convenience of using the CDE at the start of a practical session [17].

6) Features

The features offered by each CDE service were compared based on their relevance to the web application development practicum course.

TABLE VI. CDE FEATURES COMPARISON

No	Feature	Availability			
		Code Spaces	Code Sandbox	Jetify Devspace	DevZero
1	Predefined environment	x	x	x	x
2	VS Code in browser	x	x	x	
3	Remote VS Code to local device	x	x	x	x
4	Git version control	x	x	x	x
5	Extension marketplace	x	x	x	x
6	Live web-app preview	x	x	x	x
7	PHP and its extensions support	x	x	x	x
8	Apache support	x	x	x	x
9	MySQL support	x	x	x	x

In Table VI, all platforms meet the practicum's baseline needs. DevZero's requirement for a local VS Code and CLI install and lack of browser VS Code introduce extra complexity.

B. Discussion of CDE Comparison

GitHub Codespaces emerges as the strongest free-tier CDE. It combines the largest usage quota, top single-core performance, fast build/boot times, low latency, and a full feature set closely integrated with GitHub. CodeSandbox offers rapid resume and solid multi-core throughput but suffers from higher latency. Jetify Devspace's limited quota and middling performance make it less suitable for sustained use. DevZero's nominal quota advantage is outweighed by its lengthy build/boot process and cumbersome setup. For academic web-dev labs prioritizing consistency, speed, and minimal setup, GitHub Codespaces is the clear choice.

While the free tier is sufficient for academic lab use and provides ample usage quota, a potential issue arises if students heavily utilize GitHub Codespaces for other projects. Should this occur, students can apply for the GitHub Student Pack to obtain additional Codespaces usage, or create a new, separate account for the academic practicum. Both options effectively circumvent the issue while maintaining free access.

C. Comparison Results of PaaS Platforms

This section presents the results of the comparative analysis of the four PaaS platforms that consist of Koyeb, Render, Northflank, and Lade. The comparison is based on the parameters outlined in the methodology which are performance, build time, network latency, load time, uptime, and features.

1) Performance

Performance was evaluated using 1-thread CPU benchmark tests on each platform's highest-free-tier instance for five runs.

TABLE VII. PAAS PERFORMANCE COMPARISON

No	Name	vCPU (core)	RAM (MB)	Avg CPU benchmark
1	Koyeb	0.1	512	329.438
2	Render	0.1	512	513.184
3	Northflank	0.2	512	645.684
4	Lade	1	128	332.464

Table VII shows that in Sysbench CPU benchmarks (events/sec), Northflank delivers the strongest free-tier compute despite having fewer cores than Lade. It is noted that the application used for the practicum was small, allowing it to run even on Lade's limited RAM and Koyeb's lower benchmark score.

2) Build Time

Build time, the duration required to build and deploy an application, was measured [14].

TABLE VIII. PAAS BUILD TIME COMPARISON

No	Name	Build Time
1	Koyeb	1 m 44 s
2	Render	2 m 15 s
3	Northflank	1 m 17 s
4	Lade	48 s

In Table VIII, Lade deploys the fastest. Lade's streamlined pipeline likely underlies its rapid rollout.

3) Network Latency

Network latency, crucial for the responsiveness of the deployed web application [15], was assessed using the curl command for each PaaS instead of ping command because not every platform supports ping command.

TABLE IX. PAAS NETWORK LATENCY COMPARISON

No	Name	Nearest Region	Avg Latency (ms)
1	Koyeb	Frankfurt, Germany	34.5964
2	Render	Singapore	34.597
3	Northflank	Europe	192.7666
4	Lade	Singapore	34.4818

Measured via time_connect metric for five runs, as shown in Table IX, Lade had the lowest latency. Both Render and Lade, owing to their servers in Singapore, demonstrated lower average network latencies. Koyeb's built-in CDN and edge network keep its European origin nearly as responsive.

4) Load Time

Load time, the time taken for the web application to fully load in a browser [19], was also measured.

TABLE X. PAAS LOAD TIME COMPARISON

No	Name	Avg Load Time (ms)
1	Koyeb	629
2	Render	508
3	Northflank	969
4	Lade	437

Measured via SiteSpeed for five runs, as shown in Table X, server proximity gives Lade and Render an edge once again, while Koyeb's CDN narrows the gap despite a Europe host.

5) Uptime

Uptime, the percentage of time the deployed application was accessible [20], was monitored over three days.

TABLE XI. PAAS UPTIME COMPARISON

No	Name	Uptime Percentage	Total Downtime
1	Koyeb	100%	0 s
2	Render	99.77%	9 min 55 s
3	Northflank	100%	0 s
4	Lade	100%	0 s

Table XI shows that over a continuous 72-hour probe (5-minute polls via UptimeRobot), only Render recorded an uptime percentage below 100%. Despite this, all platforms met the basic reliability needs for non-mission-critical academic web apps.

6) Features

The features offered by each PaaS platform were compared based on their relevance to web application deployment and the goals of the research.

TABLE XII. PAAS FEATURES COMPARISON

No	Feature	Availability			
		Koyeb	Render	Northflank	Lade
1	Web dashboard	x	x	x	
2	Docker deployment	x	x	x	x
3	CLI access	x	x	x	x
4	No credit-card verification	x			x
5	Deployment on push	x	x	x	
6	Live web-app preview	x	x	x	x
7	Free-tier multiple instance allowance	1 running app	Unlimited (limited to total 720 hours per month)	2	3

According to Table XII, all platforms meet the practicum's baseline needs. A key differentiator was the requirement for credit or debit card verification for the free tier, which was required by Render and Northflank but not by Koyeb and Lade. Lade's requirement for CLI install and lack of web dashboard introduce extra complexity.

D. Discussion of PaaS Comparison

Koyeb emerges as a particularly suitable option for the Web Application Development practical course due to its combination of factors. While not consistently leading in all performance metrics, Koyeb offered good performance, low network latency and load times (benefiting from its built-in CDN and edge network despite the European server), and perfect uptime during testing. A significant advantage of Koyeb is that it does not require credit or debit card verification for its free tier, making it more accessible to students. A limitation of Koyeb is the restriction to one free-tier application instance, which is nonetheless sufficient for the practical course requirements. Students can circumvent this limitation for other projects by registering separate accounts.

Another limitation is that Koyeb doesn't offer separate free MySQL databases, which are needed for this practicum. While deploying MySQL within the application container seems plausible, Koyeb's local storage is ephemeral, and its persistent volume feature isn't free. To circumvent this, Aiven was used. Aiven is an external database service that offers a free MySQL instance. This integration allows students to meet the database requirements of the practicum without incurring additional costs or compromising data persistence.

E. Comparison of Existing and Target Systems

This section presents a comparative analysis of the existing system, which involves local development using VSCode, Composer, Git, and XAMPP, and the target system, which utilizes GitHub Codespaces and Koyeb. The comparison is based on the parameters outlined in the methodology and relevant to the practical Web Application Development course.

1) Total Install Time

The total install time, representing the time taken for students to prepare their development environment, was compared.

TABLE XIII. SYSTEMS TOTAL INSTALL TIME COMPARISON

No	Name	Total Install Time
1	Existing system with local stacks	14 min 8 s
2	Target system with cloud stacks	7 min 45 s

In Tabel XIII, installing all local dependencies took was around 45% longer than spinning up GitHub Codespaces, deploying to Koyeb, and provisioning Aiven's database. The target system offers a pre-configured environment, drastically reducing the time needed to start practical work.

2) Performance

Performance was evaluated using single-core and multi-core CPU benchmark tests with Geekbench 6 for five runs.

TABLE XIV. SYSTEMS PERFORMANCE COMPARISON

No	Name	vCPU (core)	RAM (GB)	Avg Single core	Avg Multi core
1	Existing system with local stacks	2	4	382	651.8
2	Target system with cloud stacks	4	8	1420.4	1733

In Tabel XIV, the target system, using GitHub Codespaces demonstrated superior single-core and multi-core performance compared to the existing system on the research device. This suggests that the target system provides a more powerful environment for running development tasks, potentially benefiting students with lower-specification local devices.

3) Network Latency

Network latency, crucial for the responsiveness of the development and deployed environments, was measured for five runs.

TABLE XV. SYSTEMS NETWORK LATENCY COMPARISON

No	Name	Region Used	Avg Latency (ms)
1	Existing system with local stacks	No region (local machine)	0.0000

2	Target system with cloud stacks	Singapore	19.4888

According to Tabel XV, Local VS Code requires no internet, effectively 0 ms, versus 19.49 ms for Codespaces, which is still imperceptible to users. The deployed application on Koyeb, also benefiting from a Singapore server, similarly experienced manageable latency. The reliance on an internet connection is a key difference, as the existing system can function offline.

4) Booting Time

Booting time, the time taken to start the development environment, was also compared.

TABLE XVI. SYSTEMS BOOTING TIME COMPARISON

No	Name	Booting Time (s)
1	Existing system with local stacks	13
2	Target system with cloud stacks	27

According to Table XVI, The existing system had a slightly faster average booting time compared to the target system. While the difference exists, the booting time of the target system is still considered relatively quick and unlikely to significantly disrupt the practical workflow.

5) Features

The features offered by each system were compared based on their relevance to the web application development practicum course.

TABLE XVII. SYSTEMS FEATURES COMPARISON

No	Feature	Availability	
		Existing System	Target System
1	Predefined environment	x	x
2	VS Code support	x	x
3	Offline support	x	
4	GitHub repository integration	x	x
5	Extension marketplace	x	x
6	Live web-app preview	x	x
7	PHP and its extensions support	x	x
8	Apache support	x	x
9	MySQL support	x	x
10	Grading method	Screenshot and source code	Live application URL
11	Portability (device-independence)		x
12	Quota limits	Unlimited (local resources only)	60h per month for Codespaces

In Table XVII, both systems meet the practicum's baseline needs. However, the target system offers a predefined programming environment, device-independence, and a more streamlined assessment process via hosted application URLs. On the other hand, existing system offers the advantages of offline capabilities and unlimited runtime flexibility, rendering it more suitable for environments with constrained network connectivity or quota limitations.

F. Discussion of System Comparison

The target system, leveraging GitHub Codespaces and Koyeb, offers substantial advantages over the existing local development environment for the Web Application Development practical course. It significantly reduces setup time and vastly improves compute performance, outweighing modest increases in latency and workspace startup time. While the local system provides lower network latency and offline capability, the target system's latency is acceptable, and reliable campus Wi-Fi mitigates internet dependency concerns. This cloud-based approach effectively alleviates technical challenges for students and instructors, enabling a greater focus on core web development concepts. Overall, for a controlled academic lab, this cloud stack delivers superior consistency, scalability, and instructor convenience, with the local stack serving as a viable fallback.

The proposed target system offers significant advantages, yet its capabilities can be further expanded. CI/CD pipeline, like GitHub Actions, would enable automated testing, thus enhancing code quality. Seamless integration with GitHub Classroom's built-in code runner could also provide automated grading and streamlined assessment and feedback. These additions would further optimize the development workflow and enrich the learning experience.

However, reliance on free-tier usage introduces certain implications. While the current curriculum's needs are met by the free tiers, the practice of using multiple accounts to circumvent rare overages, such as for other practicums or projects, is a workaround. This method is contingent on platform allowances and carries the risk of account suspension due to alleged abuse. Should these free options become unsustainable, transitioning to other free platforms (each with its own limitations) offers a temporary solution. Ultimately, ensuring the most consistent and stable experience may necessitate considering paid-tier services.

IV. CONCLUSION

The shift to cloud-native tools, particularly the combination of GitHub Codespaces for development and Koyeb for deployment, has proven to not only reduce the initial friction associated with setting up local environments, but also to introduce a higher degree of consistency and scalability across student experiences. From drastically reduced installation times to improved compute benchmarks and a vastly more efficient workflow for instructors assessing student work, the cloud stack demonstrated a clear edge in almost every measurable category.

While local development still holds value, especially in contexts where internet access is limited or consistent availability of cloud quotas cannot be guaranteed, it is increasingly difficult to justify its continued dominance in educational settings. The local approach demands more manual configuration, introduces environmental inconsistencies, and creates a heavier workload for teaching assistants during assessment.

What makes the cloud stack particularly compelling is not just its technical superiority, but also its alignment with modern software development practices. Students become familiar with CI/CD pipelines, containerized environments, and remote repositories from day one, which are skills that are directly transferable to industry scenarios.

Looking forward, the success of this case study invites further exploration into wider platforms, deeper integration with learning management systems, and the incorporation of more advanced use cases such as auto-grading or analytics. However, even in its current form, the GitHub Codespaces and Koyeb pairing already represents a robust and practical answer to the pedagogical challenges of running modern web application labs.

REFERENCES

[1] A. P. M. Dela Rosa, "Development of a Web Application for Learning Basic Mandarin Chinese," *International Journal of Emerging Technologies in Learning*, vol. 18, no. 4, 2023, doi: 10.3991/ijet.v18i04.37121.

[2] N. Phumeechanya and S. Soonthara, "The Development of Engineering Design Process on Web Application Learning Model to Enhance Web Programming Skills for Computer Education Students," *International Journal of Information and Education Technology*, vol. 13, no. 10, 2023, doi: 10.18178/ijiet.2023.13.10.1964.

[3] D. J. Malan, J. Carter, R. Liu, and C. Zenke, "Providing Students with Standardized, Cloud-Based Programming Environments at Term's Start (for Free)," in *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, 2023. doi: 10.1145/3545947.3569611.

[4] D. J. Malan, "Standardizing Students' Programming Environments with Docker Containers: Using Visual Studio Code in the Cloud with GitHub Codespaces," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 2022. doi: 10.1145/3502717.3532164.

[5] D. J. Malan, "Containerizing CS50: Standardizing Students' Programming Environments," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, Association for Computing Machinery, Jul. 2024, pp. 534–540. doi: 10.1145/3649217.3653567.

[6] K. Fernalld, T. J. Oconnor, S. Sudhakaran, and N. Nur, "Lightweight Symphony: Towards Reducing Computer Science Student Anxiety with Standardized Docker Environments," in *SIGITE 2023 - Proceedings of the 24th Annual Conference on Information Technology Education*, 2023. doi: 10.1145/3585059.3611432.

[7] M. Borowski, J. Zagermann, C. N. Klokmose, H. Reiterer, and R. Radle, "Exploring the benefits and barriers of using computational notebooks for collaborative programming assignments," in *SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020. doi: 10.1145/3328778.3366887.

[8] A. D. Snowberger and K. You, "Creating a Standardized Environment for Efficient Learning Management using GitHub Codespaces and GitHub Classroom," *Journal of Practical Engineering Education*, vol. 16, no. 3, pp. 267–274, 2024, doi: 10.14702/JPEE.2024.267.

[9] N. M. Ghazaly, "Experimental Study of PaaS, Its Implementation Methods and Advantages and Challenges," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 9, no. 11, 2021, doi: 10.17762/ijritcc.v9i11.5510.

[10] A. Srivastava, A. Ojha, A. Shaji, A. Sharma, and R. Pandey, "A Review of Cloud Computing Service Models," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 2023, doi: 10.32628/cseit2390386.

[11] D. D. K. Putra, S. P. Hari, H. Panduwiyasa, U. Y. K. S. Hediyanto, R. R. Saedudin, and A. Y. Mubarok, "Comparative Study: Open-Source Cloud Computing Performance for Small Business with ISO/IEC 25010:2011," in *AIP Conference Proceedings*, 2022. doi: 10.1063/5.0107575.

[12] W. A. R. Wan Mohd Isa, A. I. H. Suhaimi, N. Noordin, A. F. Harun, J. Ismail, and R. A. Teh, "Factors influencing cloud computing adoption in higher education institution," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 17, no. 1, 2019, doi: 10.11591/ijeecs.v17.i1.pp412-419.

[13] Y. Rashid, A. Rashid, M. A. Warraich, S. S. Sabir, and A. Waseem, "Case Study Method: A Step-by-Step Guide for Business Researchers," *Int J Qual Methods*, vol. 18, 2019, doi: 10.1177/1609406919862424.

[14] R. Priedhorsky *et al.*, "Charliecloud's layer-free, Git-based container build cache," in *ACM International Conference Proceeding Series*, 2023. doi: 10.1145/3624062.3624585.

[15] Y. Fu, D. Guo, Q. Li, L. Liu, S. Qu, and W. Xiang, "Digital Twin Based Network Latency Prediction in Vehicular Networks," *Electronics (Switzerland)*, vol. 11, no. 14, 2022, doi: 10.3390/electronics11142217.

[16] H. Mohammed, Z. Wei, E. Wu, and R. Netravali, "Continuous prefetch for interactive data applications," *Proceedings of the VLDB Endowment*, vol. 13, no. 11, 2020, doi: 10.14778/3407790.3407826.

[17] K. Lee, G. Lee, and T. Song, "Enhanced Configurable Snapshot: Snapshot-based Fast Booting on NAND Flash with Lifetime Control," in *Proceedings of the ACM Symposium on Applied Computing*, 2022. doi: 10.1145/3477314.3507061.

[18] B. Holmes, J. Waterman, and D. Williams, "KASLR in the age of MicroVMs," in *EuroSys 2022 - Proceedings of the 17th European Conference on Computer Systems*, 2022. doi: 10.1145/3492321.3519578.

[19] E. B. Setiawan and A. Setiyadi, "Comparative Analysis of Web Hosting Server Performance," *International Journal of Engineering, Transactions A: Basics*, vol. 36, no. 3, 2023, doi: 10.5829/ije.2023.36.03c.16.

[20] A. Behera, C. R. Panigrahi, S. Behera, R. Patel, and S. Bera, "trACE - Anomaly Correlation Engine for Tracing the Root Cause on Cloud Based Microservice Architecture," *Computacion y Sistemas*, vol. 27, no. 3, 2023, doi: 10.13053/CyS-27-3-4498.