JURNAL DIPAKOMTI
Pusat Penelitian dan Pengabdian kepada Masyarakat (P3M) Universitas Dipa Makassar
Jl. Perintis Kemerdekaan Km. 9 Makassar

Pengembangan Performa Pos Edukasi Keuangan Bersama Menggunakan Redis Caching dan Kontainerisasi Docker

Dimas Anugerah Pratama*, A.Hasrianto , M. Syukri Mustafa, Hasyrif SY
Jurusan Teknik Informatika, Universitas Dipa Makassar, Makassar Sulawesi Selatan
e-mail: [1]dimasanugrah883@gmail.com, [2]andihasrianto08@gmail.com, [3]syukri@undipa.ac.id, [4]hasyrif@gmail.com

Abstrak

Pengembangan aplikasi website modern sering menghadapi tantangan seperti inkonsistensi lingkungan antara lokal dan produksi. Perbedaan konfigurasi seperti sistem operasi, dependensi, dan versi perangkat lunak pendukung kerap memunculkan fenomena "it works on my machine", di mana aplikasi berjalan lancar di perangkat pengembang tetapi gagal di server produksi. Selain itu, aplikasi web yang memproses data besar rentan terhadap hambatan kinerja, terutama saat kueri basis data dieksekusi bersamaan, yang menyebabkan rendering halaman lambat dan waktu respons meningkat. Masalah ini tidak hanya mengganggu pengalaman pengguna tetapi juga memperlambat proses deployment. Penelitian ini bertujuan mengatasi kedua permasalahan tersebut dengan pendekatan hybrid, yaitu menggabungkan containerization dan Redis caching di luar container untuk meningkatkan konsistensi deployment serta performa aplikasi POS Edukasi Keuangan Bersama. Metode yang digunakan adalah penerapan Redis pada layanan cloud computing Amazon EC2, sementara aplikasi dijalankan dalam kontainer Docker. Hasil pengujian menunjukkan bahwa Docker pada lingkungan cloud computing efektif menjaga konsistensi environment aplikasi sehingga terhindar dari fenomena "it works on my machine". Selain itu, pengujian performa menggunakan Postman dengan skenario 10 virtual user selama 1 menit membuktikan bahwa Redis meningkatkan performa sebesar 11,3%, dengan rata-rata waktu respons berkurang dari 443 ms menjadi 393 ms.

Kata kunci*—Cloud Computing, Redis, Docker, Caching, Kontainerisasi.*

Abstract

The development of modern web applications often encounters challenges such as inconsistencies between local and production environments. Configuration differences including operating systems, dependencies, and software versions frequently cause the common "it works on my machine" phenomenon, where applications run smoothly on a developer's device but fail on the production server. Web applications that process large data are also prone to performance bottlenecks, especially when database queries run concurrently, leading to slow page rendering and longer response times. These issues disrupt user experience and hinder deployment efficiency. This study aims to address these problems through a hybrid approach that combines containerization with Redis caching outside the container to improve deployment consistency and application performance in the POS Edukasi Keuangan Bersama system. The method involves deploying the application within a Docker container on Amazon EC2 cloud services, while Redis is implemented as a caching layer. Results show that Docker in a cloud environment effectively ensures application consistency, preventing the "it works on my machine" issue. Performance testing using Postman with 10 virtual users over one minute demonstrated that Redis improved application performance by 11.3%, reducing average response time from 443 ms to 393 ms.

Keywords—*Cloud Computing, Redis, Docker, Caching, Containerization.*

1. PENDAHULUAN

Pos Edukasi Keuangan Bersama merupakan aplikasi berbasis website yang dapat diakses pada https://pos-pintar.digitaldesa.id dan https://pos-pintar.digitaldesa.id/mobile yang berfungsi sebagai platform yang menyediakan daftar program pelatihan dari instansi-instansi mitra yang terdiri dari Badan Usaha Milik Negara (BUMN) dan beberapa perusahaan swasta yang ada di Indonesia. Terkait aplikasi ini, penulis berperan sebagai web developer intern di PT. Digital Desa Indonesia yang bertanggung jawab dalam membuat perencanaan, perancangan, dan pengembangan fitur pada sistem tersebut.

Namun pada saat development penulis mengalami fenomena yang umum terjadi dalam pengembangan perangkat lunak yaitu kondisi yang dikenal sebagai "It works on my machine". Istilah ini menggambarkan situasi dimana sebuah program berhasil dijalankan pada komputer pengembang namun mengalami kendala atau kegagalan saat dijalankan pada komputer lain contohnya pada server production sehingga berdampak pada penundaan waktu deployment, membutuhkan waktu yang tidak dapat diduga untuk melakukan troubleshooting, hingga menurunnya user experience [1]. Fenomena ini disebabkan oleh perbedaan konfigurasi (environment, dependency, sistem operasi, ataupun versi perangkat lunak pendukung) antara environment development (perangkat developer) dan production (sistem pada server) sehingga menjadi masalah yang dihadapi penulis selama proses development dan deployment [2].

Terkait dengan proses deployment, penulis menyadari bahwa proses dilakukan dengan menjalankan tiap langkah secara satu persatu yang dilakukan juga rentan terhadap kesalahan dan membutuhkan waktu yang signifikan, menghambat alur pengembangan yang cepat. Kondisi ini menyoroti kebutuhan akan sebuah sistem yang dapat mengotomatisasi proses deployment secara konsisten dan terintegrasi. Masalah berikutnya adalah terjadinya bottleneck (kecepatan proses data yang lambat) saat melakukan pengujian dengan mengakses website tersebut menggunakan tools postman memerlukan waktu yang kurang cepat hingga dapat mencapai 8 detik sampai halaman dapat dimuat. Ketika sebuah halaman web memuat banyak data sekaligus, setiap permintaan ke database dapat menyebabkan waktu respon yang kurang cepat. Permasalahan ini umumnya disebabkan oleh beberapa faktor seperti keterbatasan kapasitas server, lambatnya performa query database, atau arsitektur aplikasi yang belum optimal.

Meskipun sudah banyak penelitian yang secara terpisah menyoroti manfaat dari masing-masing teknologi—seperti Docker yang berperan dalam menjaga konsistensi lingkungan pengembangan dan produksi, Redis yang terbukti efektif meningkatkan performa aplikasi melalui mekanisme caching, maupun CI/CD yang berfokus pada otomatisasi proses integrasi serta deployment [3]—namun sebagian besar studi tersebut masih membahasnya dalam lingkup yang terfragmentasi. Dengan kata lain, penelitian-penelitian sebelumnya cenderung melihat teknologi ini sebagai solusi tunggal untuk permasalahan spesifik, bukan sebagai sebuah ekosistem yang saling melengkapi. Padahal, dalam praktik nyata pengembangan perangkat lunak modern, tantangan yang dihadapi seringkali bersifat kompleks dan saling terkait, sehingga membutuhkan pendekatan yang lebih menyeluruh. Hingga saat ini, masih relatif jarang ditemukan penelitian yang secara komprehensif mengkaji penerapan Docker, CI/CD, dan Redis secara terpadu sebagai sebuah kerangka end-to-end. Pendekatan integratif ini diyakini mampu tidak hanya menjamin konsistensi lingkungan, tetapi juga menghadirkan otomatisasi proses deployment yang lebih andal serta meningkatkan performa aplikasi secara signifikan.

Sebagai upaya dalam mengatasi masalah perbedaan environment penulis akan menawarkan penerapan sistem container base menggunakan docker. Dengan menggunakan docker, developer dimungkinkan untuk bisa melakukan pengemasan terhadap aplikasi dan

seluruh dependensi yang dibutuhkan untuk bisa dioprasikan secara optimal dan meminimalisir resiko dari perbedaan environment [4]. Menurut [5] dengan menggunakan teknologi docker meningkatkan efisiensi pengembangan dan kemudahan deployment aplikasi dapat ditingkatkan. Kemudian, penulis mengusulkan implementasi CI/CD (Continuous Integration/Continuous Deployment). Dengan membangun pipeline otomatis, proses deployment dapat dilakukan secara terintegrasi dan konsisten, mengurangi potensi kesalahan manusia dan mempercepat siklus pengembangan. Docker akan menjadi fondasi utama dalam pipeline ini, memastikan setiap deployment berjalan di lingkungan yang sama [6].

Selain itu, penulis juga menawarkan penggunaan teknologi redis sebagai caching tools dengan tujuan untuk meningkatkan kecepatan respons aplikasi. Redis bekerja dengan menyimpan data sementara ke dalam memori, sehingga dengan kemampuan baca dan tulis yang lebih cepat diharapkan dapat meningkatkan waktu akses dan performa aplikasi secara keseluruhan. Pada penelitian [7] dijelaskan bahwa penerapan caching dapat secara signifikan mengurangi beban server dan meningkatkan pengalaman pengguna.

Dengan demikian, penelitian ini bertujuan untuk meningkatkan performa dan stabilitas aplikasi sehingga juga berdampak pada kualitas penyebaran informasi kepada pengguna. Diharapkan, hasil dari penelitian ini dapat memberikan kontribusi positif bagi pengembangan aplikasi menggunakan teknologi yang penulis gunakan.

2. METODE PENELITIAN

Penelitian ini dilakukan di PT. Digital Desa Indonesia yang berlokasi di Ruko Bisnis I-Walk J/10 Lantai 1, Ciputra CitraLand Celebes, Jl. Tun Abdul Razak, Tombolo, Kec. Somba Opu, Kabupaten Gowa, Sulawesi Selatan dengan perkiraan waktu sekitar 3 bulan dalam kurun waktu Juni 2025 – Agustus 2025. Data penelitian diperoleh melalui dua sumber utama. Pertama, data eksperimen berupa hasil pengukuran performa sistem, khususnya waktu yang diperlukan dalam memproses permintaan (response time) pada aplikasi yang telah dikembangkan.

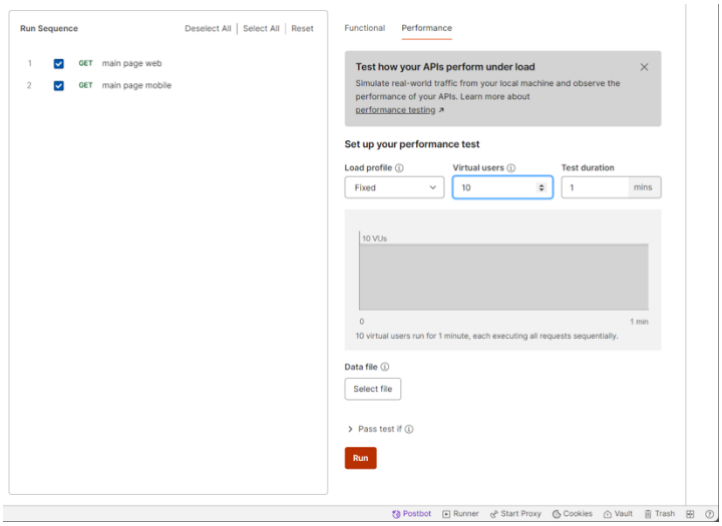
Jenis penelitian yang digunakan pada penelitian ini adalah penelitian eksperimental dengan pendekatan kuantitatif yang berfokus pada pengujian pengembangan sistemmenggunakan teknologi yang akan digunakan dalam hal ini adalah docker dan redis dengan tujuan meningkatkan kualitas deployment dan user experience terhadap sistem [8].

Kedua, data diperoleh dari objek penelitian, yaitu PT Digital Desa Indonesia, berupa data SQL dari sistem website Pos Edukasi Keuangan Bersama yang digunakan sebagai basis pengujian performa aplikasi. Jenis penelitian yang digunakan pada penelitian ini adalah penelitian eksperimental dengan pendekatan kuantitatif yang berfokus pada pengujian pengembangan sistem menggunakan teknologi yang akan digunakan dalam hal ini adalah docker dan redis dengan tujuan meningkatkan kualitas deployment dan user experience terhadap sistem.

Sistem yang telah dibangun akan melewati prosesi testing dengan tujuan memastikan seluruh fungsionalitas bekerja dan dapat berfungsi sebagaimana yang diharapkan. Metode pengujian yang dapat digunakan untuk bisa menjawab pokok permasalahan pada penelitian ini adalah dengan menggunakan load testing yang akan menguji fungsionalitas dan kecepatan pemrosesan tiap fungsi yang ada[9]. Diperlukan melakukan pengujian menggunakan alat dengan kemampuan load testing Application Programming Interface (API) untuk bisa mengukur kemampuan aplikasi dalam memproses data[10]. Pada penelitian ini penulis menggunakan postman sebagai tools yang digunakan sebagai alat uji dengan beberapa konfigurasi seperti peak hours (jumlah pengguna tinggi) dan off-peak hours (jumlah pengguna rendah).

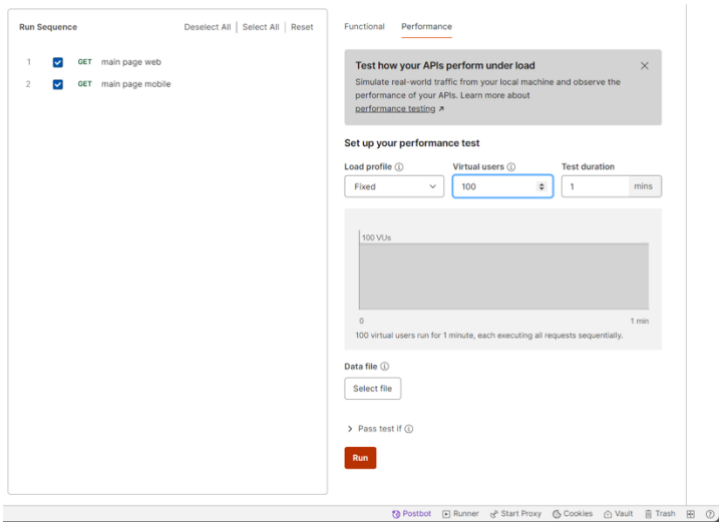
Berdasarkan informasi yang diperoleh dari team produk web pos edukasi keuangan bersama jumlah user yang mengakses secara bersamaan dalam keadaan peak hours sekitar 100 hingga 120 user dan dalam 10 hingga 15 user dalam keadaan off-peak hours sehingga hal tersebut yang menjadi landasan penulis dalam menentukan jumlah virtual user pada proses pengujian. Pada gambar 1 yang merupakan dengan skenario aplikasi berada pada jam saat jumlah user yang

sedang mengakses aplikasi sedang sedikit atau biasa disebut off-peak hours, penulis melakukan skema pengujian selama satu menit dengan 10 virtual user.



Gambar 1. Konfigurasi Pengujian Off-Peak Hours

Dan gambar 2 merupakan gambar yang merupakan skenario aplikasi berada pada jam saat jumlah user yang sedang mengakses aplikasi sedang banyak atau ramai yang biasanya disebut peak hours, penulis melakukan skema pengujian selama 1 menit dengan 100 virual user. Selain itu, dengan menggunakan postman data error rate juga penulis dapat menguji apakah ada error yang dihasilkan oleh aplikasi web pada tiap tahap pengembangan maupun produksi sehingga bisa menjadi data perbandingan untuk tiap tahap pengembangan apakah konsistensi environment tetap terjaga.



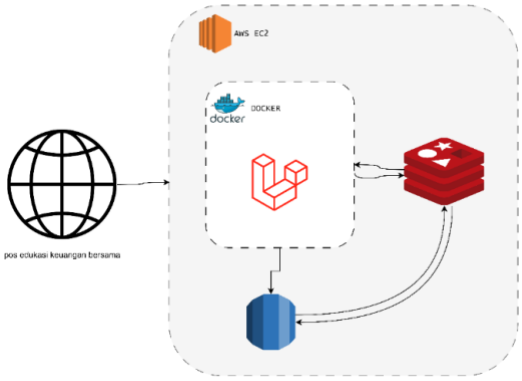
Gambar 2. Konfigurasi Peak Hours

3. HASIL DAN PEMBAHASAN

3.1 Desain Solusi

Pada gambar 3 digambarkan arsitektur dari solusi yang dikembangkan pada penelitian ini. Aplikasi diakses melalui alamat domain pos edukasi keuangan bersama yang terhubung pada pada public Internet Protocol (IP) cloud-based services dalam hal ini adalah Amazon Web Service (AWS) Elastic Compute Cloud (EC2) yang menyimpan dan menjalankan berbagai teknologi antara lain seperti docker, redis dan Relational Database Services (RDS).

Dalam docker container terdapat aplikasi berbasis web yang dibuat dengan teknologi framework laravel. Aplikasi akan berkomunikasi dengan redis untuk mengecek apakah data yang diperlukan ada pada memori redis, jika tidak maka aplikasi akan mengakses data ke database yang menggunakan service dari AWS yaitu RDS.



Gambar 3. Rancangan Solusi

Pada service AWS EC2 terdapat teknologi redis sebagai caching tools yang terhubung pada RDS yang merupakan database service dari AWS dan aplikasi berbasis web menggunakan teknologi framework Laravel yang berada pada container docker. Redis akan memberikan data yang diminta oleh aplikasi. Kemudian databases service dari AWS yaitu RDS yang berfungsi sebagai teknologi database yang menyimpan seluruh data secara permanen pada cloud-based services[11]. Jika data tidak tersedia di Redis, aplikasi akan mengambil data tersebut dari RDS. Data yang diperoleh kemudian disimpan ke Redis sebagai cache yang kemudian nantinya dapat diakses oleh aplikasi[12].

3.2 Analisis Penggunaan Docker

Penggunaan Docker pada pengembangan aplikasi ini bertujuan untuk menciptakan lingkungan kerja yang konsisten, terisolasi, dan mudah didistribusikan. Implementasi Docker terbukti mampu menjaga konsistensi aplikasi pada setiap tahapan pengembangan, termasuk pada berbagai platform. Konsistensi ini tercipta karena penggunaan Dockerfile yang sama, sehingga mengatasi masalah umum seperti perbedaan versi PHP, ekstensi, atau konfigurasi sistem yang sering menjadi penyebab error[13].

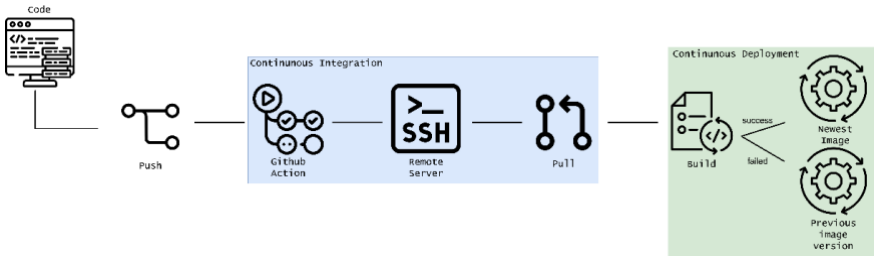
Berdasarkan hasil pengujian dengan menggunakan tools postman yang disajikan pada tabel 1, error rate menjadi parameter perbandingan antara lingkungan produksi dan pengembangan. Hasil pengujian menggunakan tools postman menunjukkan bahwa error rate yang didapatkan pada dua lingkungan pengembangan yang berbeda memberikan hasil yang identik. Hal ini membuktikan bahwa kedua lingkungan tersebut telah konsisten, sehingga meminimalkan potensi bug yang disebabkan oleh perbedaan konfigurasi.

Tabel 1 Persentase Error

	Development	*Production*
Persentase error	0%	0%

Selain itu, dengan menggunakan docker pengembang dapat mereduksi konfigurasi manual. Dengan menggunakan image Docker yang telah dikonfigurasi sebelumnya, proses setup aplikasi menjadi lebih cepat dan minim kesalahan. Deploy aplikasi cukup dilakukan dengan satu perintah, tanpa perlu instalasi PHP, Composer, atau ekstensi lain secara manual di server[14]. Sebagaimana alur pada gambar 4 menjelaskan bahwa dengan penggunaan docker proses deployment dapat menjadi lebih sederhana, durasi rendah serta memberikan kemudahan dalam

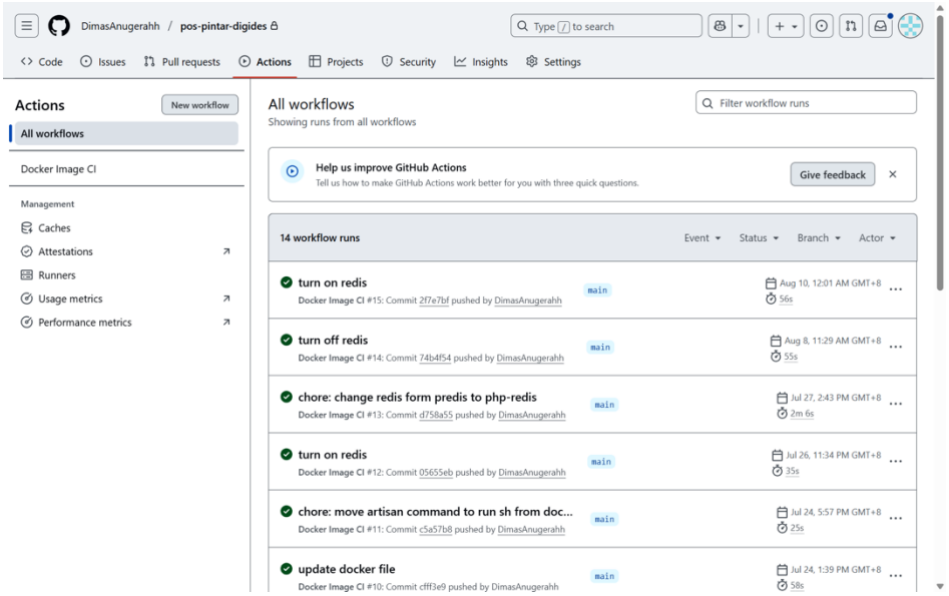
penanganan error dengan melakukan rollback ke versi sebelumnya secara otomatis dengan durasi waktu yang lebih cepat menjadi 19 detik hingga 5 menit seperti pada gambar 5 dan tabel 2 [15].



Gambar 4. Deployment Flow Setelah Menggunakan Docker

Tabel 2 Data Pengujian Kecepatan Deployment

	Sebelum	Sesudah
Rentang Waktu	5 - 12 Menit	19 detik - 5 menit



Gambar 5. GIT Workflow

3.3 Analisis Penggunaan Redis

Penggunaan redis digunakan sebagai teknologi caching. Teknologi ini berfungsi untuk menyimpan data secara sementara di dalam memori, yang bertujuan mempercepat operasi baca dan tulis data[16]. Dan dari pengujian yang dilakukan pada penelitian ini ditemukan bahwa redis berdampak pada peningkatan performa seperti yang ada pada tabel 3 setelah melakukan pengujian latensi menggunakan aplikasi postman dengan skema off-peak hours (saat jumlah pengguna yang mengakses aplikasi rendah secara bersamaan atau 10 orang) selama 1 menit berhasil menjalankan request sebanyak 594 kali atau 8.78 request perdetik dan diperoleh rata-rata waktu yang diperlukan adalah selama 393ms per-request.

Tabel 3 Load Testing Off-Peak Hours

Kecepatan Rata-rata	Total Request	Total Request per-detik	Jumlah Error	Kecepatan Tertinggi	Kecepatan Terendah
393ms	590	8.78	0	mobile: 226ms desktop: 171ms	mobile: 895ms desktop:974ms

Kemudian, pada tabel 4 merupakan hasil dari pengujian latensi menggunakan aplikasi postman dengan skema peak hours (saat jumlah pengguna yang mengakses aplikasi tinggi secara bersamaan atau 100 orang) selama 1 menit berhasil menjalankan request sebanyak 750 kali atau 11,14 request perdetik dan diperoleh rata-rata waktu yang diperlukan adalah selama 7114ms per-request.

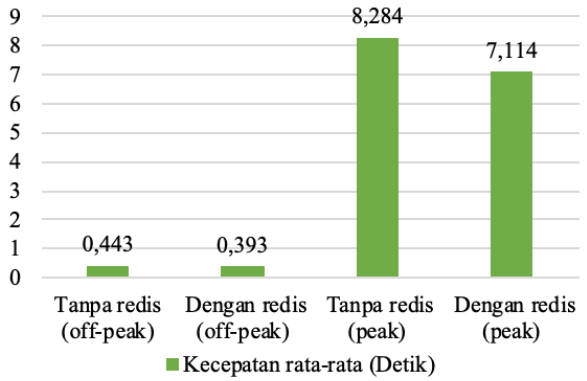
Tabel 4 Load Testing Peak Hours

Kecepatan Rata-rata	Total Request	Total Request per-detik	Jumlah Error	Kecepatan Tertinggi	Kecepatan Terendah
7114ms	750	11.14	0	mobile: 1245ms desktop: 361ms	mobile: 8320ms desktop:8181ms

Berdasarkan hasil rekapitulasi data pengujian menggunakan tools postman tersebut seperti yang dapat dilihat pada tabel 5 dan gambar 6 diperoleh bahwa performa aplikasi web dalam keadaan peak hours meningkat dari yang awalnya hanya berhasil melakukan 647 request dalam satu menit atau 9,61 request perdetik dan dengan rata-rata waktu yang diperlukan 8284ms (8,284 detik) per-request. Selain itu, hasil pengujian dengan menggunakan postman dalam keadaan off-peak hours performa meningkat dari yang awalnya hanya berhasil melakukan 560 request dalam satu menit atau 8.34 request perdetik dan dengan rata-rata waktu yang diperlukan 443ms (0,443 detik) per-request.

Tabel 5 Tabel Rekapitulasi Data Hasil Pengujian Menggunakan Postman

	Sebelum (off-peak)	Sesudah (off-peak)	Sebelum (peak)	Sesudah (peak)
Kecepatan Rata-rata	443ms	393ms	8284ms	7114ms
Total Request	560	647	647	750
Total Request per-detik	8,34	9,61	9,61	11,14
Jumlah Error	0	0	0	0
Kecepatan Tertinggi	mobile: 200ms desktop: 183ms	mobile: 226ms desktop: 171ms	mobile: 5664ms desktop: 586ms	mobile: 1245ms desktop: 361ms
Kecepatan Terendah	mobile: 1059ms desktop:1271ms	mobile: 895ms desktop:974ms	mobile: 11075ms desktop:10371ms	mobile: 8320ms desktop:8181ms



Gambar 6 Grafik Data Kecepatan rata-rata

4. KESIMPULAN

Penggunaan teknologi kontainerisasi Docker dan Redis terbukti memberikan dampak positif terhadap peningkatan performa aplikasi web Pos Edukasi Keuangan Bersama, baik dari

sisi kecepatan akses aplikasi meningkat sebesar 11.3% maupun efektivitas dan kualitas deployment, sehingga proses deployment menjadi lebih efisien. Kombinasi Redis Caching dan Docker Containerization merupakan pendekatan yang efektif untuk meningkatkan performa, efisiensi, dan keandalan aplikasi berbasis web, khususnya pada sistem pelatihan berbasis platform seperti POS Edukasi Keuangan Bersama.

4.1 Implementasi Redis Caching Pada Aplikasi Laravel

a. Menurunkan rata-rata waktu respon dari 0,443 detik menjadi 0,393 detik per-request pada kondisi off-peak hours (setara dari 443ms menjadi 393ms). Dan 8,284 detik menjadi 7,144 detik per-request pada kondisi peak hours.
b. Meningkatkan jumlah permintaan yang dapat ditangani per detik dari 8,34 menjadi 8,78 request pada kondisi off-peak hours. Dan yang awalnya dapat melakukan request sebanyak 9,61 request menjadi 11,14 request perdetik dalam kondisi peak hours, hal ini menunjukkan peningkatan throughput sistem.
c. Menyimpan data yang sering diakses di memori sehingga mengurangi beban langsung ke database MySQL dan meningkatkan efisiensi sistem secara keseluruhan.

4.2 Penerapan Docker Containerization

a. Mempercepat proses deployment dari yang sebelumnya 5–12 menit menjadi hanya 19 detik–5 menit.
b. Menjaga konsistensi environment antara development dan production, mengurangi risiko error akibat perbedaan konfigurasi server.
c. Memungkinkan penerapan proses CI/CD melalui GitHub Actions dan skrip otomatis pada Docker yang meningkatkan efisiensi kerja tim pengembang.

4.3 Desain Arsitektur Sistem

a. Redis dijalankan langsung pada EC2, sedangkan Laravel berjalan di dalam Docker container, namun keduanya tetap terintegrasi dengan baik melalui konfigurasi jaringan dan environment.
b. Redis berfungsi sebagai middleware caching antara aplikasi dan database, mempercepat akses data dan menjaga kestabilan performa aplikasi..

5. SARAN

5.1 Saran Untuk Penelitian Selanjutnya

a. Lakukan analisis perbandingan biaya antara service-service caching yang ada pada cloud provider.
b. Lakukan analisis penerapan redis dan docker pada contoh kasus aplikasi yang berbeda.

5.2 Saran Untuk Pengembangan Aplikasi

a. Terapkan arsitektur yang lebih kompleks seperti orkestrasi kontainer menggunakan teknologi seperti kubernetes.
b. Lakukan analisis penggunaan docker dan redis pada service cloud berbeda..

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih sebesar-besarnya kepada PT. Digital Desa Indonesia yang telah memberikan kesempatan dan izin kepada penulis untuk bisa melakukan penelitian ini, tanpa dukungan dan bantuan tersebut tentunya penelitian ini tidak dapat terselesaikan. Penulis

juga mengucapkan terima kasih sebesar-besarnya kepada bapak M. Syukri Mustafa, S.Si., M.M.S.I. dan bapak Hasyrif S Y, S.Kom., M.T. untuk bimbingan dan arahan yang diberikan kepada penulis yang sangat membantu penulis dalam menyusun dan menyelesaikan penelitian ini.

DAFTAR PUSTAKA

[1] V. R. Gudelli, "Containerization Technologies : ECR and Docker for Microservices Architecture," no. June 2023, 2025, doi: 10.37082/IJIRMPS.v11.i3.232165.

[2] M. Nazario, R. Bonifacio, and G. Pinto, Mitigating Configuration Differences Between Development and Production Environments: A Catalog of Strategies, vol. 1, no. 1. Association for Computing Machinery, 2025.

[3] A. Gogineni, "Automated deployment and rollback strategies for docker containers in continuous integration/continuous deployment (CI/CD) pipelines," International Journal of Multidisciplinary Research and Growth Evaluation, vol. 1, no. 5, pp. 125–130, 2020, doi: 10.54660/.ijmrge.2020.1.5.125-130.

[4] S. Dwiyatno, E. Rachmat, A. P. Sari, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer, vol. 7, no. 2, pp. 165–175, 2020, doi: 10.30656/prosisko.v7i2.2520.

[5] M. Nazário, R. Bonifácio, and G. Pinto, "Works on My Machine! Managing Configuration Differences between Development and Production Environments | Request PDF," 2023.

[6] K. M. Pitchikala, "Enhancing Software Development with CI/CD: Best Practices and Strategies," Journal of Scientific and Engineering Research, vol. 2022, no. 12, pp. 172–176, [Online]. Available: www.jsaer.com

[7] R. Ridhalri, "Pemanfaatan Caching System Menggunakan Redis Untuk Sistem Pengelolaan Informasi Ambalan Ashabul Kahfi Berbasis Web," Jurnal DIALOGIKA : Manajemen dan Administrasi, vol. 4, no. 1, pp. 39–56, 2022, doi: 10.31949/dialogika.v4i1.3750.

[8] Y. Zhang and X. Chen, "Results: Empirical Analysis," in Application-Oriented Higher Education: A Comparative Study between Germany and China, Y. Zhang and X. Chen, Eds., Singapore: Springer Nature Singapore, 2022, pp. 111–140. doi: 10.1007/978-981-19-2647-1_7.

[9] G. Mohan, Full Stack Testing. " O'Reilly Media, Inc.," 2022.

[10] M. Hendayun, A. Ginanjar, and Y. Ihsan, "ANALYSIS OF APPLICATION PERFORMANCE TESTING USING LOAD TESTING AND STRESS TESTING METHODS IN API SERVICE," JURNAL SISFOTEK GLOBAL, vol. 13, no. 1, p. 28, Mar. 2023, doi: 10.38101/sisfotek.v13i1.2656.

[11] M. Zadka, "Amazon Web Services," in DevOps in Python: Infrastructure as Python, M. Zadka, Ed., Berkeley, CA: Apress, 2022, pp. 187–200. doi: 10.1007/978-1-4842-7996-0_13.

[12] P. Späth, "Caching," in Pro Jakarta EE 10: Open Source Enterprise Java-based Cloud-native Applications Development, P. Späth, Ed., Berkeley, CA: Apress, 2023, pp. 373–379. doi: 10.1007/978-1-4842-8214-4_29.

[13] D. Saxena and N. Sharma, "Analysis of Docker Performance in Cloud Environment," 2020, pp. 9–18. doi: 10.1007/978-981-15-5421-6_2.

[14] S. Bhavsar, J. Rangras, and K. Modi, "Automating Container Deployments Using CI/CD," 2021, pp. 423–429. doi: 10.1007/978-981-15-4474-3_47.

[15] N. Islavath, "The Power of Docker: Containerization for Efficient Software Development and Deployment," International Journal of Science and Research (IJSR), vol. 9, no. 11, pp. 1748–1751, Nov. 2020, doi: 10.21275/sr201226085354.

[16] G. Anna, V. Ekaterina, K. Vladislav, R. Alena, and P. Nikolay, "Memcached's Redis

Repository Benchmark for CRUD Operations," in 2024 4th International Conference on Intelligent Technologies (CONIT), 2024, pp. 1–4. doi: 10.1109/CONIT61985.2024.10627484.