Jurnal Publikasi Sistem Informasi dan Manajemen Bisnis
Volume 5, Nomor 2, Mei 2026
e-ISSN: 2808-8980; p-ISSN: 2808-9383, Hal. 576-588
DOI: https://doi.org/10.55606/jupsim.v5i2.7122
Tersedia: https://journalcenter.org/index.php/jupsim

Implementasi *High Availability* AWS pada Situs Web Berbasis *Framework CodeIgniter*

Akrim Teguh Suseno[1*], Indra Kurniawan[2], Mohammad Reza Maulana[3]
[1]Sains Data, Universitas Islam Negeri K.H. Abdurrahman Wahid Pekalongan, Indonesia
[2]Bisnis Digital, Universitas Islam Negeri K.H. Abdurrahman Wahid Pekalongan, Indonesia
[3]Informatika, Universitas Islam Negeri K.H. Abdurrahman Wahid Pekalongan, Indonesia

**Penulis Korespondensi: akrim.teguh.suseno@uingusdur.ac.id*

Abstract. *This study focuses on designing and implementing a High Availability (HA) architecture for a web application built with CodeIgniter using Amazon Web Services (AWS). The objective is to enhance system reliability while minimizing downtime. A prototype-based research method is applied, allowing iterative development, testing, and evaluation of the system. The proposed architecture utilizes an Application Load Balancer (ALB) to distribute incoming traffic efficiently, multiple Amazon EC2 instances deployed across different Availability Zones for redundancy, and Amazon RDS with replication to ensure database reliability and consistency. Performance testing includes both load testing and failure simulation scenarios. The results indicate that the system achieves an average response time between 120 and 180 milliseconds under normal conditions and remains stable when handling concurrent users. During failure testing, the system successfully performs failover in under 10 seconds without noticeable service disruption. Additionally, database replication maintains data integrity with no detected data loss, confirming improved availability, scalability, and reliability.*

Keywords: *Amazon Web Services; CodeIgniter; EC2; High Availability (HA); Load Balancer.*

Abstrak. Penelitian ini bertujuan untuk merancang dan mengimplementasikan arsitektur *High Availability* (HA) pada aplikasi web berbasis CodeIgniter dengan memanfaatkan Amazon Web Services (AWS). Tujuan utama penelitian ini adalah meningkatkan kemampuan stabilitas aplikasi serta meminimalkan *downtime*. Metode penelitian yang digunakan adalah pendekatan prototipe yang memungkinkan proses pengembangan, pengujian, dan evaluasi dilakukan secara iteratif. Arsitektur yang diusulkan memanfaatkan *Application Load Balancer (ALB)* untuk mendistribusikan lalu lintas jaringan secara efisien, beberapa *instance* Amazon EC2 yang ditempatkan pada berbagai *Availability Zone* yang bertujuan untuk redundansi aplikasi, serta Amazon RDS dengan mekanisme replikasi untuk menjamin *availibility* dan konsistensi basis data. Pengujian kinerja dilakukan melalui skenario beban kerja pada website dan simulasi kegagalan sistem dengan salah satu *instance* down. Hasil penelitian menunjukkan bahwa sistem mampu mencapai waktu respons rata-rata antara 120 hingga 180 milidetik pada kondisi normal serta tetap stabil dalam menangani pengguna secara bersamaan. Pada pengujian kegagalan sistem, proses *failover* berlangsung kurang dari 10 detik tanpa gangguan layanan yang signifikan. Selain itu, replikasi basis data mampu menjaga *availibility* data secara realtime tanpa kehilangan data yang terdeteksi, sehingga arsitektur yang diusulkan terbukti meningkatkan ketersediaan, skalabilitas, dan keandalan sistem.

Kata Kunci: Amazon Web Services; CodeIgniter; EC2*High Availability* (HA); *Load Balancer*.

1. LATAR BELAKANG

Di era digital saat ini, keberadaan aplikasi web telah menjadi elemen penting bagi banyak organisasi dan bisnis dalam menyediakan layanan, otomasi proses, serta interaksi pengguna secara daring. Salah satu framework yang banyak digunakan untuk pengembangan aplikasi web di Indonesia adalah CodeIgniter, karena bersifat ringan, mudah dipelajari, serta mendukung pengembangan aplikasi berbasis arsitektur MVC secara cepat dan terstruktur (Herdiansah, 2021; Miftakhudin & Sifaunajah, 2023). Namun disisi lain terdapat berbagai permasalahan pada implementasi dari sebuah sistem untuk tetap stabil dan memiliki ketersedian layanan yang tinggi.

Naskah Masuk: 25 Maret 2026; Revisi: 27 April 2026; Diterima: 20 Mei 2026; Terbit: 26 Mei 2026

Seperti deployment pada satu server / satu *instance* yang rentan gagal karena gangguan kecil sehingga berdampak ke seluruh aplikasi(Adiya et al., 2024a; Shadaksharappa B, 2023). Selain itu aplikasi berbasis web dan e-commerce yang sering tidak mampu menangani lonjakan pengunjung, sehingga berdampak pada akses yang lambat atau bahkan tidak bisa diakses(Bhatt, 2022).

Salah satu pendekatan untuk mengatasi hal tersebut adalah dengan penggunaan layanan *High Availability* (HA) yang mampu memberikan ketersediaan tinggi dalam layanan suatu sistem sehingga dapat meminimalisir adanya down pada sistem ataupun lonjakan pengunjung. Penggunaan *High Availability* dapat melalui komputasi awan *(cloud computing)* yang merupakan model yang menyediakan sumber daya komputasi melalui internet secara *on-demand.* Sumber daya tersebut meliputi server, penyimpanan, basis data, jaringan, perangkat lunak, serta layanan lainnya yang dapat diakses kapan saja tanpa harus memiliki dan mengelola infrastruktur fisik sendiri(Amajuoyi et al., 2024; Sidhaarth Vishnu K.R. S, 2023; Sikka & Ojha, 2021) . Implementasi *cloud computing* memungkinkan tercapainya layanan *High Availability*, sekaligus mendukung pengelolaan aplikasi web dengan kebutuhan trafik yang besar. Oleh sebab itu implementasi *cloud computing* dapat meminimalisir downtime karena hal tersebut menjadi prioritas utama guna menjaga kontinuitas layanan dan kepercayaan pengguna (Chirumamilla, 2021; Liu et al., 2021). Namun demikian, penerapan HA pada aplikasi berbasis website terdapat beberapa tantangan seperti pengelolaan beban trafik penggunaan, lapisan aplikasi dan basis data, serta pengelolaan penyimpanan file statis terdistribusi guna mencegah hambatan kinerja (*performance bottleneck*)(Ajmi, 2025).

Amazon Web Services (AWS), sebagai salah satu penyedia layanan cloud terkemuka, menawarkan berbagai layanan yang dapat dimanfaatkan untuk membangun arsitektur dengan ketersediaan tinggi. Fitur yang disediakan meliputi desain *multi-Availability Zone* (AZ) dengan replikasi lintas AZ/region untuk mengurangi *single point of failure* dan menjaga keberlangsungan layanan saat terjadi gangguan lokal maupun regional. Layanan Elastic Load Balancing (ELB/ALB) berfungsi untuk mendistribusikan trafik ke beberapa *instance* serta menyesuaikan kapasitas secara otomatis sesuai beban (Kopparthi, 2024). Selain itu, AWS menyediakan Amazon RDS dengan fitur multi-AZ, replikasi, pencadangan otomatis, serta mekanisme *failover* yang lebih cepat untuk meningkatkan ketersediaan basis data meskipun terjadi kegagalan pada *instance* utama(Bhatt, 2022). Berbagai layanan lain seperti Amazon EC2 dan Auto Scaling juga berperan dalam meningkatkan ketersediaan tinggi pada implementasi aplikasi(Alanda et al., 2025)(Viatoire, 2025).

Oleh sebab itu penelitian ini akan melakukan implementasi website berbasis codeigniter dengan menggunakan teknologi AWS *cloud computing* dengan penerapan *High Availability* untuk meningkatkan layanan dan stabilitas performance sistem serta mencegah adanya *single point of failure*.

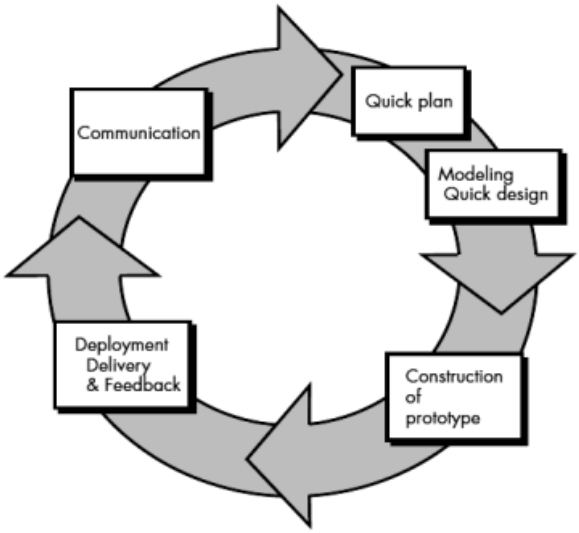
2. KAJIAN TEORITIS

Sejumlah penelitian terdahulu telah mengkaji topik yang berkaitan dengan implementasi High Availability dan cloud computing, yang menjadi landasan bagi penelitian ini. Bhatt (2022) meneliti pemanfaatan layanan AWS untuk membangun sistem High Availability dan disaster recovery pada aplikasi SAP, dengan hasil bahwa kombinasi fitur multi-AZ RDS dan Elastic Load Balancing mampu meningkatkan ketersediaan sistem secara signifikan. Chirumamilla (2021) mengeksplorasi desain arsitektur fault tolerant menggunakan multi-region AWS, dan menyimpulkan bahwa strategi multi-region memberikan perlindungan lebih komprehensif terhadap kegagalan infrastruktur berskala besar dibandingkan pendekatan single-region. Shadaksharappa B (2023) menganalisis berbagai mekanisme High Availability dan fault tolerance yang tersedia di AWS, termasuk Auto Scaling, ELB, dan RDS Multi-AZ, dan menegaskan pentingnya integrasi layanan-layanan tersebut untuk mencapai ketersediaan sistem yang optimal. Manurung et al. (2024) mengimplementasikan High Availability pada WordPress berbasis teknologi AWS dan membuktikan bahwa penggunaan ALB dan multi-AZ EC2 dapat meminimalkan downtime bahkan saat salah satu instance mengalami kegagalan

3. METODE PENELITIAN

Penelitian ini menerapkan metode prototipe, yaitu pendekatan pengembangan sistem yang menekankan pembuatan model awal (prototipe) sebagai representasi dari sistem yang akan dibangun. Metode ini memungkinkan peneliti untuk melakukan pengujian konsep secara bertahap, serta melakukan perbaikan secara iteratif hingga sistem mencapai kinerja yang diharapkan(Adiya et al., 2024b; Kasmar et al., 2025; Manurung et al., 2024). Dalam konteks teknologi jaringan dan komputasi awan, metode prototipe sangat relevan karena dapat memfasilitasi pengujian langsung arsitektur dan layanan cloud dalam lingkungan yang terkendali(Kamila et al., 2022). Pendekatan prototipe dalam penelitian ini digunakan untuk merancang dan mengimplementasikan *High Availability* (HA) berbasis Amazon Web Services (AWS) pada aplikasi yang dibangun menggunakan CodeIgniter.

Dengan metode ini, solusi yang dikembangkan tidak hanya bersifat konseptual, tetapi juga dapat diuji secara praktis untuk menilai efektivitas dan keandalannya. Tahapan penelitian yang dilakukan meliputi langkah-langkah yang ditunjukan pada gambar 1 berikut:



Gambar 1. Tahapan Metode Prototipe.

Komunikasi *(Communication)*

Pada tahap ini, pengembang sistem berkomunikasi secara intensif dengan pengguna atau *stakeholder* untuk memahami masalah bisnis, alur kerja, serta kebutuhan fungsional dan non-fungsional sistem. Tujuannya adalah mengumpulkan kebutuhan awal yang jelas sebelum merancang prototipe(Adiya et al., 2024a).

Perencanaan Cepat *(Quick Plan)*

Berdasarkan hasil komunikasi, dikembangkan perencanaan cepat yang mencakup ruang lingkup fitur, target pengguna, serta alokasi tugas dan jadwal iterasi. Tahap ini menyiapkan kerangka pengembangan yang tetap ringan dan fleksibel terhadap perubahan. Desain difokuskan pada pemilihan layanan AWS yang mendukung konsep *High Availability*, seperti *Load Balancer*, *instance* server, basis data terkelola, serta layanan penyimpanan dan distribusi konten. Tahap ini menyiapkan kerangka pengembangan yang tetap ringan dan fleksibel terhadap perubahan(Mewengkang et al., 2022; Rawis et al., 2023).

Pemodelan Desain Cepat *(Modelling Quick Design)*

Tahap ini menghasilkan desain awal antarmuka dan struktur sistem, yang biasanya berupa diagram, sketsa layar, dan tata letak basis data. Desain ini masih bersifat kasar namun cukup untuk memvisualisasikan cara kerja sistem, sehingga memudahkan diskusi dengan pengguna sebelum implementasi penuh.

Konstruksi Prototipe *Cconstruction of Prototype)*

Tahap konstruksi prototipe dilaksanakan dengan mengimplementasikan desain arsitektur yang telah dibuat sebelumnya. Pada tahap ini, peneliti mulai mengonfigurasi layanan AWS, menerapkan aplikasi, dan mengintegrasikan komponen-komponen sistem sesuai dengan desain yang telah direncanakan. Prototipe dikembangkan untuk merepresentasikan sistem *High Availability* CodeIgniter secara fungsional, sehingga dapat digunakan untuk menguji konsep dasar, alur kerja sistem, dan keterkaitan antarlayanan cloud yang digunakan(Mewengkang et al., 2022; Rawis et al., 2023).

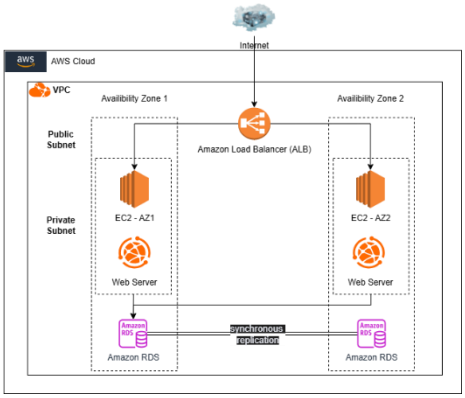
Penyerahan & Umpan Balik *(Deployment Delivery and Feedback)*

Setelah prototipe berhasil dibangun, dilakukan fase evaluasi untuk menilai kinerja dan efektivitas sistem. Evaluasi ini mencakup pengujian fungsional, pengujian beban, serta pemantauan stabilitas dan ketersediaan sistem saat menerima permintaan pengguna. Tujuan dari fase ini adalah mengidentifikasi kelemahan, kendala teknis, dan potensi perbaikan yang diperlukan guna memastikan sistem beroperasi secara optimal sesuai dengan tujuan penelitian(Irawan & Muarif, 2024).

4. HASIL DAN PEMBAHASAN

Pada bagian hasil, pengumpulan data dilakukan terhadap kebutuhan server non-fungsional, seperti dua *Availability Zone*, *Load Balancer*, subnet publik dan privat, serta replikasi basis data. Kebutuhan-kebutuhan ini digunakan untuk memenuhi layanan *High Availability* yang akan diterapkan dalam penelitian ini. Tahap selanjutnya adalah perancangan dan implementasi prototipe, khususnya melalui implementasi VPC, Amazon *Load Balancer* (ALB), RDS Database, dan website berbasis CodeIgniter. Berikut adalah deskripsi rinci dari setiap tahapan:

Perancangan Topologi Infrastruktur *High Availability* (HA)



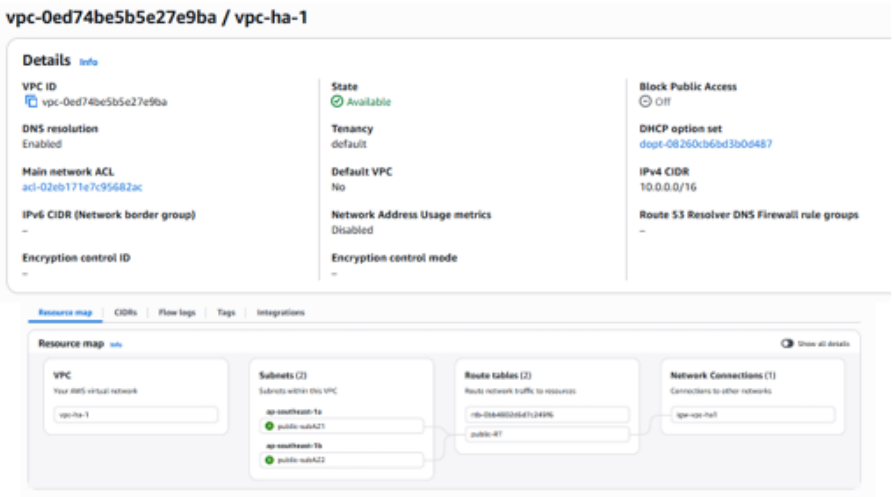
Gambar 2. Perancangan Topologi Infrastruktur HA

Perancangan infrastruktur *High Availability* pada Gambar 2 memanfaatkan layanan cloud AWS untuk menjaga ketersediaan dan keandalan aplikasi. Pengguna mengakses aplikasi melalui internet, yang kemudian diteruskan ke komponen *Load Balancer* menggunakan layanan Application Load Balancing (ALB). ALB berfungsi sebagai *Load Balancer* yang beroperasi pada Layer 7 (Application Layer) dari model OSI. ALB dirancang khusus untuk mendistribusikan lalu lintas HTTP dan HTTPS secara cerdas ke berbagai target, seperti *instance* EC2.

Pada level komputasi dan basis data, sistem memanfaatkan beberapa *instance* Amazon EC2 yang berlokasi di dua *Availability Zone* berbeda untuk meningkatkan toleransi kesalahan (fault tolerance). Sementara itu, penyimpanan data dikelola menggunakan Amazon RDS yang dikonfigurasi dengan replikasi sinkron antar zona, sehingga data pada basis data utama secara otomatis direplikasi ke basis data cadangan. Konfigurasi ini memungkinkan failover yang cepat apabila terjadi gangguan pada salah satu komponen, sehingga ketersediaan layanan terjaga dan risiko kehilangan data dapat diminimalkan.

Implementasi *Virtual Private Cloud* (VPC)

Jaringan diimplementasikan dengan membangun Virtual Private Cloud (VPC) sebagai fondasi infrastruktur layanan AWS. Arsitektur ini dirancang menggunakan pendekatan multi-tier yang memisahkan lapisan jaringan dengan membagi alamat IP ke dalam dua jenis subnet: subnet publik dan subnet privat, yang tersebar di dua *Availability Zone* (AZ). Implementasi VPC ditunjukkan pada Gambar 3.

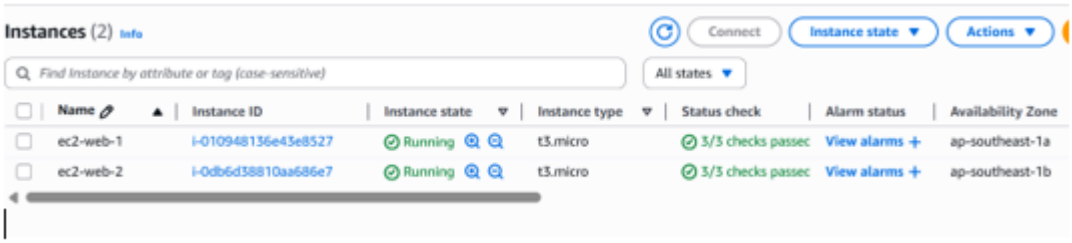


Gambar 3. Implementasi Infrastruktur Jaringan dengan VPC.

Pada Gambar 3, VPC ID adalah vpc-0ed74be5b5e27e9ba dengan tag atau inisial "vpc-ha-1" dan alamat IP 10.0.0.0/16. Terdapat dua subnet publik: public-subAZ1 dan public-subAZ2, dengan alamat IP masing-masing 10.0.1.0/24 dan 10.0.2.0/24. Strategi penempatan di dua AZ ini bertujuan untuk meminimalkan risiko kegagalan sistem secara total jika salah satu website mengalami gangguan, sehingga mendukung prinsip *High Availability*. Kedua subnet publik kemudian dirutekan ke gateway "igw-vpc-ha1" untuk akses internet.

Implementasi *Instance*

Implementasi website menggunakan *instance* AWS EC2, yaitu server virtual yang digunakan untuk menginstal website berbasis CodeIgniter. Dua *instance* digunakan: satu sebagai *instance* utama dan yang lainnya berfungsi sebagai *instance* cadangan atau sekunder. Jika satu *instance* mengalami gangguan, *instance* lainnya dapat secara otomatis menggantikannya. Berikut adalah implementasi EC2 untuk *instance-instance* tersebut yang ditunjukkan pada Gambar 4.



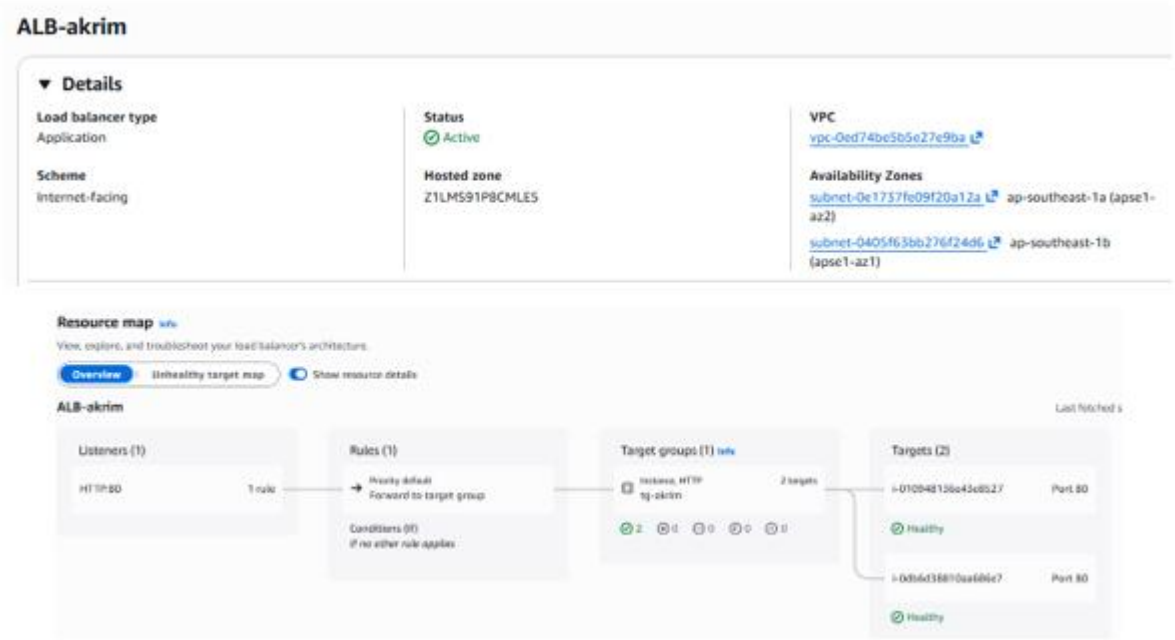
Gambar 4. Implementasi *Instance* EC2.

Pada Gambar 3, *instance* pertama diberi nama ec2-web-1, dan *instance* kedua diberi nama ec2-web-2. Kedua *instance* menggunakan sistem operasi AWS Linux berbasis Fedora Linux untuk konfigurasinya. Tipe *instance* yang digunakan adalah t3.micro, yang memiliki 2 vCPU dan 1 GB RAM, serta prosesor Intel Xeon Skylake. Tipe ini dipilih berdasarkan kapasitas website yang akan dipasang dan lebih efisien dari segi biaya. *Availability Zone* yang digunakan adalah server di kawasan Asia Pasifik Singapura, dengan kode ap-southeast-1a untuk ec2-web-1 dan ap-southeast-1b untuk ec2-web-2.

Implementasi *Load Balancer*

Load Balancer diimplementasikan menggunakan layanan Application *Load Balancer* (ALB) yang disediakan oleh AWS. Berbeda dengan *Load Balancer* tradisional, ALB dapat membuat keputusan perutean berdasarkan konten permintaan, seperti path URL (path-based routing), nama host (host-based routing), bahkan parameter dalam header HTTP. Hal ini menjadikannya solusi ideal untuk arsitektur aplikasi modern. Dalam penelitian ini, ALB digunakan untuk memastikan bahwa jika satu *instance* mengalami gangguan.

Instance lainnya dapat langsung mengambil alih (failover) agar website tetap berjalan lancar melalui perutean berbasis parameter HTTP. Dengan komponen ini, sistem dapat menghindari kelebihan beban pada satu server, sehingga meningkatkan kinerja dan ketersediaan layanan aplikasi. Berikut adalah implementasi ALB pada AWS yang ditunjukkan pada Gambar 5.



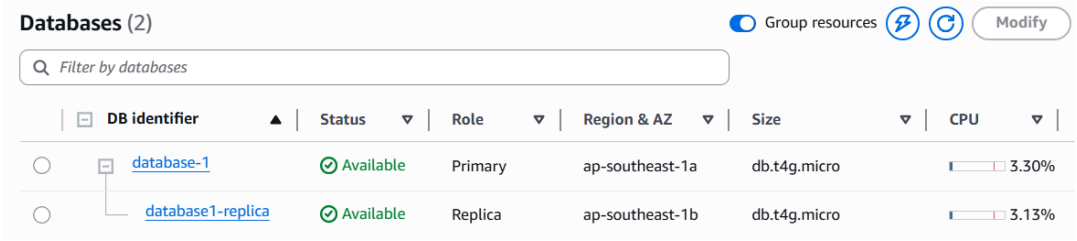
Gambar 5. Implementasi Application *Load Balancer* (ALB) pada AWS.

Selama tahap implementasi, *Load Balancer* Ditempatkan dalam VPC dengan subnet publik di dua *Availability Zone* (AZ): ap-southeast-1a dan ap-southeast-1b, dengan lokasi server di Singapura. *Load Balancer* ini masih menggunakan listener HTTP pada port 80, yang meneruskan permintaan pengguna ke grup target yang berisi dua *instance* EC2, ec2-web-1 dan ec2-web-2. *Instances* tersebut kemudian menerima lalu lintas terdistribusi dari *Load Balancer*. AWS kemudian secara otomatis melakukan pemeriksaan kesehatan pada setiap *instance* untuk memastikan server aktif dan berjalan serta mampu melayani permintaan pengguna. Jika satu *instance* mengalami masalah atau gagal merespons pemeriksaan kesehatan, *Load Balancer* secara otomatis mengalihkan lalu lintas ke *instance* lain yang masih berjalan normal.

Basis Data Amazon RDS

Implementasi basis data menggunakan Amazon RDS, sebuah fitur AWS yang efektif untuk *High Availability* karena kemampuannya dalam mereplikasi basis data, sehingga menyediakan cadangan apabila terjadi kegagalan pada data utama.

Implementasi ini ditunjukkan pada gambar berikut:



Gambar 6. Implementasi Basis Data Amazon RDS pada AWS.

Gambar 6 menunjukkan konfigurasi layanan basis data dengan dua *instance* basis data: basis data utama bernama database-1 dan basis data replika bernama database1-replica. Kedua *instance* berstatus available, yang mengindikasikan bahwa keduanya berjalan secara normal. Konfigurasi ini mendemonstrasikan implementasi mekanisme replikasi basis data untuk meningkatkan ketersediaan dan keandalan sistem penyimpanan data.

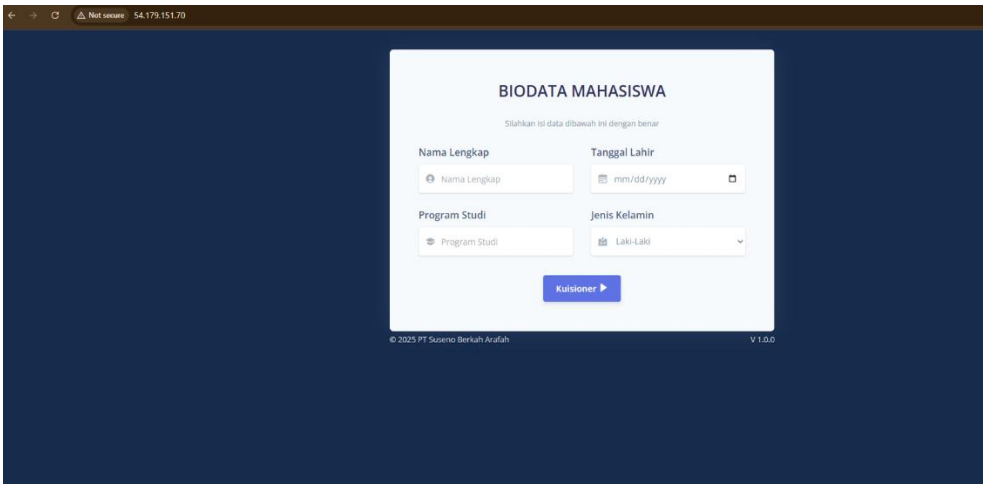
Instance basis data utama, database-1, berlokasi di *Availability Zone* ap-southeast-1a, sementara *instance* basis data replika, database1-replica, berlokasi di *Availability Zone* ap-southeast-1b. Kedua zona berada dalam region yang sama, yaitu Asia Pasifik (Singapura), namun terletak di pusat data yang berbeda. Penempatan basis data di dua *Availability Zone* yang berbeda meningkatkan ketahanan terhadap kegagalan infrastruktur. Apabila terjadi gangguan pada satu zona, sistem masih dapat mengandalkan basis data di zona lainnya, sehingga kelangsungan layanan tetap terjaga.

Selanjutnya, kedua *instance* basis data menggunakan tipe komputasi db.t4g.micro, yaitu kelas *instance* dengan sumber daya komputasi yang ringan dan efisien untuk beban kerja skala kecil hingga menengah. Kelas ini memiliki 2 vCPU, 1 GB memori, dan jaringan hingga 5 Gbps dengan karakteristik kinerja yang dapat ditingkatkan secara dinamis (*burstable*). Informasi utilisasi CPU yang ditampilkan sekitar 3% mengindikasikan bahwa beban kerja basis data relatif rendah dan sistem berjalan secara stabil. Dengan mekanisme replikasi antara basis data utama dan replika dalam layanan Amazon RDS, data dari basis data utama secara otomatis disalin ke basis data replika, sehingga memungkinkan pemulihan cepat dan failover apabila terjadi kegagalan pada basis data utama. Konfigurasi ini merupakan praktik terbaik dalam membangun arsitektur sistem yang mendukung *High Availability* di lingkungan komputasi awan.

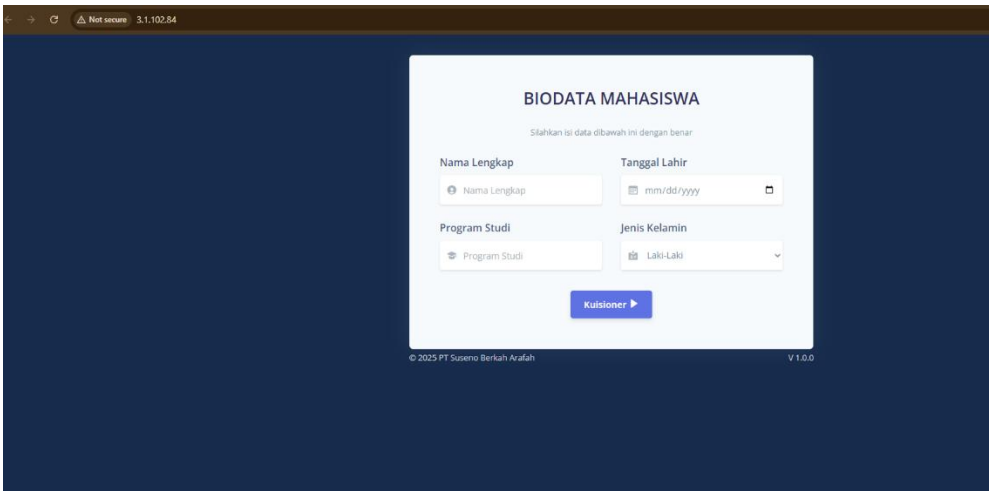
Pengujian

Selama fase pengujian, dilakukan simulasi pada arsitektur *High Availability* (HA) yang dibangun dalam lingkungan Amazon Web Services untuk memastikan sistem beroperasi secara optimal dalam berbagai kondisi.

Berikut adalah implementasi website menggunakan framework CodeIgniter, beserta pengujian untuk memastikan website berjalan secara normal, sebagaimana ditunjukkan pada Gambar 7 dan 8.



Gambar 7. Pengujian pada ec2-web-1.



Gambar 8. Pengujian pada ec2-web-2.

Gambar 7 dan 8 mendemonstrasikan bagaimana Application *Load Balancer* mendistribusikan lalu lintas ke dua *instance* server Amazon EC2 dengan mengakses aplikasi secara bersamaan dari beberapa klien. Hasil implementasi menunjukkan bahwa kedua *instance* memiliki alamat IP publik: ec2-web-1 (54.179.151.70) dan ec2-web-2 (3.1.102.84). Hasil pengujian menunjukkan bahwa permintaan pengguna berhasil dirutekan secara bergantian ke kedua alamat IP tersebut, mengindikasikan bahwa proses load balancing berjalan dengan lancar. Respons aplikasi tetap stabil dengan waktu akses yang konsisten, sehingga mencegah terjadinya bottleneck pada salah satu server. Pengujian berikutnya berfokus pada pengujian kegagalan untuk menilai ketahanan sistem. Satu *instance* EC2, baik pada IP 54.179.151.70 maupun 3.1.102.84, sengaja dihentikan untuk mensimulasikan downtime server.

Hasilnya menunjukkan bahwa Application *Load Balancer* mampu mendeteksi *instance* yang tidak aktif melalui mekanisme health check dan secara otomatis mengalihkan seluruh lalu lintas ke *instance* yang masih berjalan. Proses ini berlangsung tanpa downtime yang signifikan, sehingga pengguna tetap dapat mengakses aplikasi secara normal. Hal ini membuktikan bahwa sistem memiliki kemampuan failover yang efektif pada lapisan aplikasi dan mampu menjaga kelangsungan layanan. Untuk mengevaluasi kinerja sistem, dilakukan pengujian beban dengan mensimulasikan pengguna bersamaan yang mengakses aplikasi. Hasilnya dirangkum dalam Tabel 1.

Tabel 1. Pengujian Beban Aplikasi.

Parameter	Hasil
Rata-rata Waktu Respons	120–180 ms
Throughput	250–400 requests/second
Tingkat Kesalahan	< 1%
Waktu Failover	< 10 seconds
Ketersediaan Sistem	~99.9%

Hasil pada Tabel 1 menunjukkan bahwa sistem beroperasi secara efisien di bawah beban yang bervariasi. Waktu respons tetap stabil dan tingkat kesalahan sangat minimal. Selama simulasi kegagalan, sistem berhasil mengalihkan lalu lintas ke *instance* yang sehat dalam hitungan detik, sehingga mendemonstrasikan kemampuan failover yang efektif.

Selanjutnya, pengujian pada lapisan basis data menggunakan Amazon RDS menunjukkan bahwa mekanisme replikasi antar *Availability Zone* berjalan dengan baik. Data yang tersimpan dalam basis data utama secara konsisten direplikasi ke basis data cadangan secara real time, sehingga menjaga integritas data. Dalam simulasi pemadaman basis data utama, sistem mempertahankan ketersediaan data melalui basis data replika. Secara keseluruhan, hasil pengujian ini menunjukkan bahwa implementasi *High Availability* tidak hanya berhasil mendistribusikan beban lalu lintas, tetapi juga menjaga keandalan sistem dan meminimalkan risiko downtime pada aplikasi berbasis cloud.

5. KESIMPULAN DAN SARAN

Penelitian ini berhasil merancang dan mengimplementasikan arsitektur *High Availability* (HA) untuk aplikasi web berbasis CodeIgniter menggunakan Amazon Web Services. Integrasi Application Load Balancer, instans EC2 di berbagai *Availability Zone*, dan replikasi Amazon RDS secara signifikan meningkatkan ketersediaan dan keandalan sistem. Hasil penelitian menunjukkan bahwa sistem mempertahankan kinerja yang stabil dengan waktu respons yang rendah dan tingkat kesalahan minimal dalam berbagai kondisi beban.

Mekanisme *failover* beroperasi secara efektif, dengan waktu pemulihan yang berlangsung dalam waktu kurang dari 10 detik, sehingga memastikan layanan yang tidak terputus. Meskipun demikian, penelitian ini masih terbatas pada penerapan satu region dan pengujian dalam skala yang relatif kecil. Penelitian di masa mendatang dapat mengeksplorasi arsitektur multi-region, penerapan berbasis kontainer menggunakan Kubernetes, serta pengujian stres yang lebih ekstensif untuk lebih meningkatkan skalabilitas dan ketahanan sistem.

DAFTAR REFERENSI

Adiya, A. Z. D. N., Anggraeni, D. L., & Albana, I. (2024a). Analisa Perbandingan Penggunaan Metodologi Pengembangan Perangkat Lunak (Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD)). *Merkurius : Jurnal Riset Sistem Informasi Dan Teknik Informatika*. https://doi.org/10.61132/merkurius.v2i4.148

Adiya, A. Z. D. N., Anggraeni, D. L., & Albana, I. (2024b). Analisa Perbandingan Penggunaan Metodologi Pengembangan Perangkat Lunak (Waterfall, Prototype, Iterative, Spiral, Rapid Application Development (RAD)). *Merkurius : Jurnal Riset Sistem Informasi Dan Teknik Informatika*. https://doi.org/10.61132/merkurius.v2i4.148

Ajmi, S. Q. (2025). High availability strategies in cloud infrastructure management. *International Journal of Cloud Computing and Database Management*. https://doi.org/10.33545/27075907.2025.v6.i1a.85

Alanda, A., Mooduto, H., Amnur, H., & Fadhel, M. (2025). Scaling Kubernetes Architectures for High Availability in Cloud. *2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI)*, 818–823. https://doi.org/10.1109/icocseti63724.2025.11020084

Amajuoyi, C. P., Nwobodo, L. K., & Adegbola, M. D. (2024). Transforming business scalability and operational flexibility with advanced cloud computing technologies. *Computer Science & IT Research Journal*. https://doi.org/10.51594/csitrj.v5i6.1248

Bhatt, S. (2022). Leveraging AWS Tools for High Availability and Disaster Recovery in SAP Applications. *International Journal of Scientific Research in Science, Engineering and Technology*. https://doi.org/10.32628/ijsrset2072122

Chirumamilla, S. K. (2021). Designing with High Availability: Achieving Fault Tolerance in Software Engineering through AWS Multi-Region Architectures. *International Journal For Multidisciplinary Research*. https://doi.org/10.36948/ijfmr.2021.v03i06.40677

Herdiansah, A. (2021). System Development for Learning Process Monitoring in Private Lesson Institution Using Codeigniter Framework. *JISA(Jurnal Informatika Dan Sains)*. https://doi.org/10.31326/jisa.v4i1.861

Irawan, M. D., & Muarif, R. (2024). E-Letter Design Using Prototype System Development Methodology. *Bigint Computing Journal*. https://doi.org/10.55537/bigint.v2i1.806

Kamila, N., Frnda, J., Pani, S., Das, R., Islam, S., Bharti, P., & Muduli, K. (2022). Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. *J. King Saud Univ. Comput. Inf. Sci.*, *34*, 9991–10009. https://doi.org/10.1016/j.jksuci.2022.10.001

Kasmar, A. F., Wahyuna, W., Sukma, F., & Amalia, S. (2025). IMPLEMENTASI SISTEM KEAMANAN DAN HIGH AVAILABILITY PADA CLOUD SERVER MENGGUNAKAN AMAZON WEB SERVICES (AWS). *Jurnal Teknoif Teknik Informatika Institut Teknologi Padang*. https://doi.org/10.21063/jtif.2025.v13.1.40-47

Kopparthi, V. J. R. (2024). Architecture and Implementation of Cloud-Based Disaster Recovery. *International Journal For Multidisciplinary Research*. https://doi.org/10.36948/ijfmr.2024.v06i06.33420

Liu, Z., Fan, G., Yu, H., & Chen, L. (2021). An Approach to Modeling and Analyzing Reliability for Microservice-Oriented Cloud Applications. *Wirel. Commun. Mob. Comput.*, *2021*, 5750646. https://doi.org/10.1155/2021/5750646

Manurung, M. G., Lubis, A., & Hafni. (2024). Implementasi High-Availability WordPress Deployment Berbasis Teknologi AWS. *Bulletin of Computer Science Research*. https://doi.org/10.47065/bulletincsr.v4i2.333

Mewengkang, A., Sengkey, M., Lengkong, J., & Rotty, V. (2022). Design and Implementation of Web-Based Archive Management Information System. *International Journal of Information Technology and Education*. https://doi.org/10.62711/ijite.v1i4.80

Miftakhudin, M., & Sifaunajah, A. (2023). Use of the Codeigniter Framework in the Design and Development of the Balance Sheet-Based "Sikmajo" Financial Information System. *NEWTON: Networking and Information Technology*. https://doi.org/10.32764/newton.v3i1.3965

Rawis, J., Lengkong, J., Dotulung, R., Wuwungan, H., Rambitan, M., & Rattu, O. (2023). Developing a Web-Based Information System for Sub-Districts in North Sulawesi. *International Journal of Information Technology and Education*. https://doi.org/10.62711/ijite.v2i4.154

Shadaksharappa B. (2023). High Availability and Fault Tolerance in AWS. *International Journal of Innovative Research in Information Security*. https://doi.org/10.26562/ijiris.2023.v0903.03

SIDHAARTH VISHNU K.R. S. (2023). A world with Cloud Computing. *International Scientific Journal of Engineering and Management*. https://doi.org/10.55041/isjem00279

Sikka, R., & Ojha, M. (2021). An Overview of Cloud Computing. *International Journal of Innovative Research in Computer Science & Technology*. https://doi.org/10.55524/ijircst.2021.9.6.31

Viatoire, Dr. T. A. (2025). Dynamic Auto-Scaling and Load-Balanced Web Application Deployment in AWS. *INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. https://doi.org/10.55041/ijsrem49936