Journal of Information System and Application Development
Volume 1 Number 2 September 2023 page 138-147
P-ISSN: 2988-5698 | E-ISSN: 2988-4721
https://jurnal.unmer.ac.id/index.php/jisad

Pemanfaatan layanan *cloud computing* dan *docker container* untuk meningkatkan kinerja aplikasi web

Utilization of cloud computing services and Docker containers to improve web application performance

Achsan Rizky Ekaputra*, Arif Saivul Affandi
Program Studi Sistem Informasi, Fakultas Teknologi Informasi, Universitas Merdeka Malang, Indonesia

E-mail: *achsan.ekaputra@student.unmer.ac.id

Abstract. *This research aims to analyze and implement the utilization of Content Delivery Networks (CDN) and cloud computing services in web applications using Nginx Proxy Manager and Docker containers. This research uses experimental methods and case studies to analyze the performance of web applications before and after using CDN services and implementing the application in a cloud computing environment using Docker containers. Next, Nginx Proxy Manager is used to provide a user-friendly interface for configuring and managing Nginx proxy servers. Docker containers are used to manage web applications and proxy servers. In the CDN and cloud computing environment, Cloudflare & Google Cloud Platform will be used as service providers. The results of this research indicate a significant improvement in the security and reliability of web applications after implementing CDN services and adopting a cloud computing environment. Additionally, this research provides insight into how to use Nginx Proxy Manager and Docker container to manage web applications efficiently and effectively in a cloud computing environment.*

Keywords: *Content Delivery Networks, Cloud Computing, Docker Container, Web Application Performance*

Abstrak. Penelitian ini bertujuan untuk menganalisis dan mengimplementasikan pemanfaatan layanan *Content Delivery Network* (CDN) dan *cloud computing* pada aplikasi web menggunakan Nginx Proxy Manager dan *docker container*. Penelitian ini menggunakan metode eksperimental dan studi kasus untuk menganalisis kinerja aplikasi web sebelum dan setelah menggunakan layanan CDN serta mengimplementasikan aplikasi pada lingkungan *cloud computing* dengan menggunakan *docker container*. Selanjutnya, Nginx Proxy Manager digunakan untuk menyediakan antarmuka yang mudah digunakan untuk mengonfigurasi dan mengelola *server proxy* Nginx. *Docker container* digunakan untuk mengelola aplikasi web dan *server proxy*. Pada lingkungan CDN dan *cloud computing* akan digunakan Cloudflare & Google Cloud Platform sebagai penyedia layanan. Hasil dari penelitian menunjukkan bahwa adanya peningkatan signifikan dalam keamanan dan keandalan aplikasi web setelah menerapkan layanan CDN dan mengadopsi lingkungan *cloud computing*. Selain itu, penelitian ini memberikan wawasan tentang cara menggunakan Nginx Proxy Manager dan *docker container* untuk mengelola aplikasi web dengan efisien dan efektif dalam lingkungan *cloud computing*.

Kata kunci: *Content Delivery Networks, Cloud Computing, Docker Container*, Kinerja Aplikasi Web

Submitted: 19-07-2023 | ***Accepted:*** 05-09-2023 | ***Published:*** 30-09-2023

How to Cite:
A. R. Ekaputra and A. S. Affandi, "Pemanfaatan layanan *cloud computing* dan *docker container* untuk meningkatkan kinerja aplikasi web," *Journal of Information System and Application Development*, vol.1, no. 2, September 2023.

DOI: 10.26905/jisad.v1i2.11084

PENDAHULUAN

Dalam era kemajuan teknologi yang terus berkembang, aplikasi berbasis web telah menjadi salah satu elemen terpenting dalam ekosistem digital global. Aplikasi web saat ini memainkan peran utama dalam berbagai bidang, termasuk bisnis, pendidikan, hiburan, dan interaksi sosial. Keberhasilan aplikasi web memberikan dampak positif yang signifikan dalam hal komunikasi, konektivitas global, dan efisiensi operasional. Namun, di balik kesuksesan tersebut, terdapat sejumlah tantangan yang perlu diatasi yang mempengaruhi efisiensi, performa, dan ketersediaan aplikasi berbasis web [1].

Salah satu permasalahan yang mendominasi adalah performa aplikasi web yang kurang optimal. Seringnya, pengguna menghadapi pengalaman yang kurang memuaskan, seperti waktu pemuatan halaman yang tinggi atau respon *server* yang lambat. Permasalahan distribusi konten juga menonjol sebagai isu yang signifikan. Aplikasi web modern sangat bergantung pada penyediaan konten statis seperti HTML, CSS, JavaScript, gambar, dan video [2]. Memastikan distribusi konten dengan cepat dan efisien kepada pengguna yang tersebar di seluruh dunia merupakan hal yang tidak mudah. Tantangan lain yang perlu diatasi adalah ketersediaan dan skalabilitas aplikasi. Aplikasi web harus mampu mengatasi lonjakan beban kerja yang tiba-tiba, baik yang disebabkan oleh peningkatan penggunaan harian atau peristiwa khusus [3]. Memastikan ketersediaan dan performa yang stabil dalam kondisi ini menjadi sangat penting. Keamanan juga merupakan perhatian utama dalam mengelola aplikasi web. Ancaman terhadap keamanan siber semakin kompleks. Aplikasi perlu dilindungi dari berbagai serangan siber yang mungkin terjadi, seperti serangan DDoS, SQL *Injection*, serta eksploitasi kerentanan [4].

Oleh karena itu diperlukan sebuah teknologi yang dapat menjadi solusi terhadap beberapa permasalahan tersebut, diantaranya yaitu *Content Delivery Network* (CDN), *cloud computing*, Nginx Proxy Manager, dan *docker container*. CDN adalah jaringan *server* terdistribusi secara geografis yang bertujuan untuk menyampaikan konten web dengan cepat dan efisien kepada pengguna di seluruh dunia. Melalui penyimpanan salinan konten di *server* terdekat dengan lokasi pengguna, CDN dapat mengurangi latensi dan mempercepat waktu respons dalam mengakses konten web. Hal ini memberikan pengalaman yang lebih baik bagi pengguna, terutama dalam hal kecepatan akses dan pengiriman konten [5]. Sementara itu, *cloud computing* telah menjadi dasar utama dalam pengembangan dan penyediaan aplikasi web yang dapat ditingkatkan kapasitasnya dan dapat diandalkan. Dengan menggunakan infrastruktur komputasi awan, pengembang dapat dengan mudah meningkatkan kapasitas aplikasi sesuai dengan kebutuhan pengguna [6], [7].

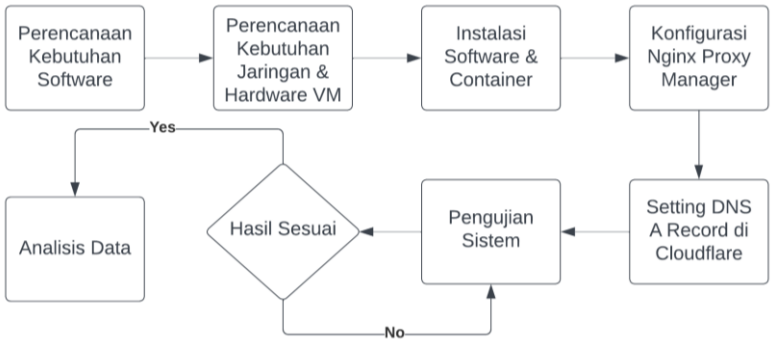
Fleksibilitas yang disajikan oleh komputasi awan memungkinkan skalabilitas yang mudah dan efisien, yang memungkinkan aplikasi web untuk mengatasi lonjakan lalu lintas dan meningkatkan kinerjanya. Dalam konteks ini, Nginx Proxy Manager dan *docker container* adalah dua teknologi yang dapat digunakan untuk memperkuat pemanfaatan layanan CDN dan *cloud computing* terhadap pengembangan aplikasi web. Nginx Proxy Manager adalah alat manajemen yang menggunakan Nginx sebagai *reverse proxy*, membantu mengoptimalkan lalu lintas aplikasi web dan meningkatkan kecepatan penyampaian konten. *Docker container*, di sisi lain, menyediakan *platform* untuk mengemas dan menjalankan aplikasi beserta dependensinya dalam wadah yang terisolasi, memungkinkan portabilitas dan konsistensi dalam lingkungan yang berbeda [8].

Penelitian ini bertujuan untuk mengevaluasi kinerja aplikasi web dengan memanfaatkan layanan CDN dan *cloud computing*, mempelajari peran Nginx Proxy Manager dalam manajemen lalu lintas, dan mengeksplorasi penggunaan *docker container* dalam mengoptimalkan infrastruktur aplikasi. Dengan melakukan penelitian ini, diharapkan dapat ditemukan cara yang lebih efisien untuk meningkatkan kecepatan penyampaian konten, meningkatkan skalabilitas aplikasi, sekaligus memastikan pengalaman pengguna yang baik dalam aplikasi web. Selain itu, penelitian ini juga diharapkan dapat memberikan wawasan tentang pemanfaatan layanan CDN dan *cloud computing*, serta penggunaan Nginx Proxy Manager dan *docker container* dalam konteks aplikasi web. Hasil implementasi teknologi ini dapat memberikan panduan praktis bagi pengembang aplikasi berbasis web untuk meningkatkan kinerja dan efisiensi aplikasi mereka.

METODE

Desain Penelitian

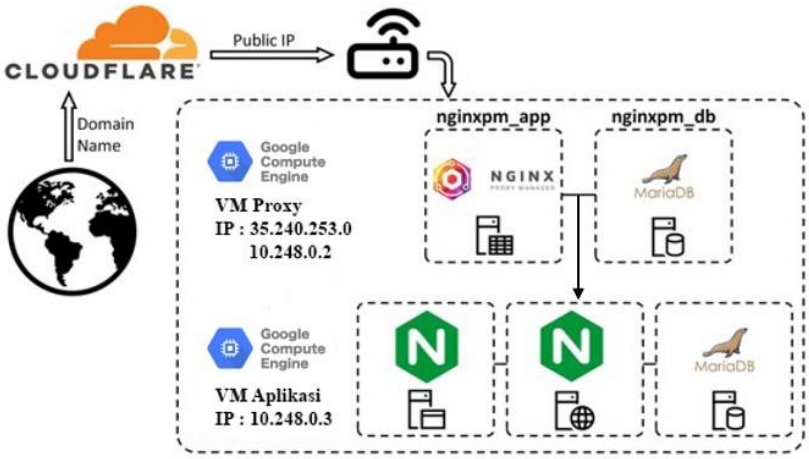
Desain penelitian yang dilakukan adalah penelitian eksperimental. Pada desain penelitian eksperimental, sampel dibagi menjadi dua kelompok secara acak yaitu kelompok kontrol dan kelompok perlakuan. Kelompok kontrol akan digunakan sebagai pembanding dan menjalankan kondisi normal atau standar, sedangkan kelompok perlakuan akan diberikan perlakuan tertentu untuk melihat pengaruhnya terhadap hasil pengukuran. Dalam penelitian ini, kelompok kontrol akan menjalankan aplikasi web dengan pendekatan tradisional, sedangkan kelompok perlakuan akan menjalankan aplikasi web dengan *docker container* dan diintegrasikan dengan Nginx Proxy Manager sebagai *reverse proxy* dan CDN dari Cloudflare. Alur langkah penelitian diperlihatkan pada Gambar 1.



Gambar 1. *Flowchart* Alur Penelitian

Perancangan Sistem

Perancangan topologi arsitektur sistem merujuk pada proses merencanakan struktur jaringan atau sistem komputer untuk memenuhi kebutuhan bisnis atau teknis tertentu. Hal ini merupakan langkah penting dalam pengembangan sistem yang efisien dan handal. Pada Gambar 2 diperlihatkan topologi arsitektur sistem yang akan digunakan di dalam penelitian.



Gambar 2. Topologi Arsitektur Sistem

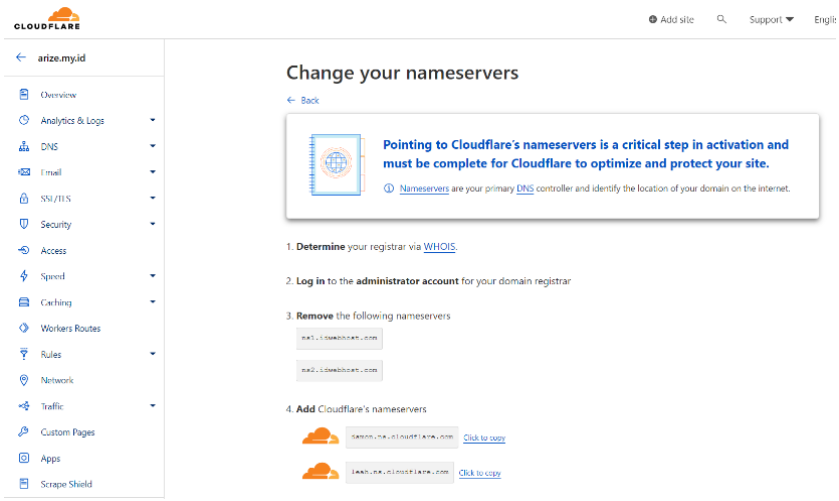
Di dalam gambar dijelaskan bahwa ketika seorang pengguna mencoba mengakses aplikasi web melalui internet, langkah awal yang dilakukan adalah mengakses *domain* arize.my.id yang telah dikonfigurasi dengan *name server* dan layanan CDN dari Cloudflare [9]. Kemudian, Cloudflare akan mengarahkan permintaan dari pengguna tersebut ke *server proxy* Nginx yang memiliki alamat IP publik 35.240.253.0. Langkah ini memungkinkan layanan Cloudflare untuk dapat berkomunikasi dengan *server proxy* tersebut.

Selanjutnya, *server proxy* Nginx akan meneruskan permintaan *website* tersebut ke *server* aplikasi yang menjalankan aplikasi web di dalam *docker container* pada jaringan *private cloud*, yang mana hanya dapat diakses dari dalam jaringan tersebut. Meskipun demikian, melalui perantara *server proxy* Nginx yang terhubung ke internet, pengguna dari luar dapat mengakses aplikasi tersebut dengan aman menggunakan protokol HTTPS. Arsitektur ini memungkinkan pengguna eksternal untuk dapat mengakses aplikasi di dalam jaringan *private cloud* melalui CDN Cloudflare, *server proxy* Nginx, dan *docker container*, dengan tetap menjaga keamanan dan privasi akses ke aplikasi tersebut.

HASIL DAN PEMBAHASAN

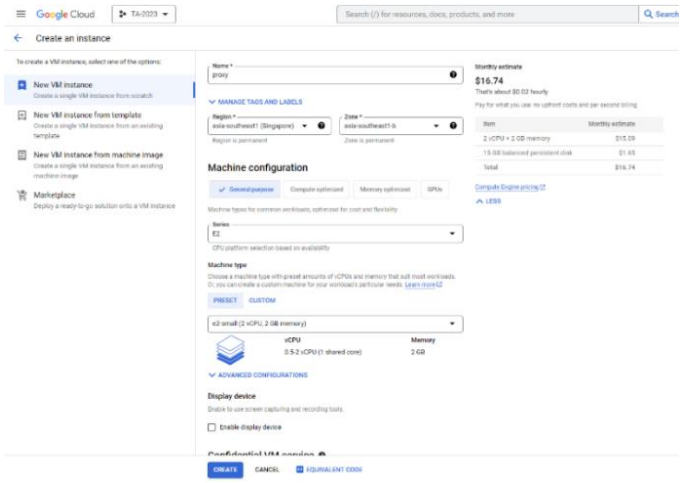
Konfigurasi *Domain* dan *Virtual Machine* (VM)

Pada tahap ini dilakukan pembelian *domain* melalui salah satu *registrar* Domain ID. Nama *domain* yang akan dibeli dan didaftarkan yaitu arize.my.id. Setelah *domain name* didaftarkan, langkah selanjutnya yaitu mendaftar akun Cloudflare dan mendaftarkan *domain name* ke Cloudflare dengan cara mengubah *name server default* dari penyedia *domain* ke *name server* Cloudflare. Untuk mengubah *name server* lama ke *name server* baru diperlukan waktu setidaknya maksimal 24 jam. Konfigurasi *name server* pada Clouflare diperlihatkan pada Gambar 3.



Gambar 3. Konfigurasi *Name Server* pada Cloudflare

Kemudian tahap selanjutnya adalah mempersiapkan VM yang akan digunakan sebagai *server proxy* dan *server* aplikasi. Dalam hal ini, digunakan layanan dari Google Cloud Platform (GCP) yang bernama *compute engine* untuk membuat VM *instance* [10], seperti yang ditunjukkan pada Gambar 4.



Gambar 4. Konfigurasi VM pada GCP

Konfigurasi VM untuk *server proxy* memilki IP *public static* dan digunakan untuk menjalankan Nginx Proxy Manager sebagai *reverse proxy* Nginx. Sedangkan VM untuk *server* aplikasi dirancang untuk internal aplikasi yang hanya memiliki alamat IP internal dan nantinya akan menjalankan aplikasi berbasis web menggunakan *docker container*. Aplikasi dibuat sesuai dengan kebutuhan dan spesifikasi yang sudah ditentukan sebelumnya. Setelah konfigurasi selesai dan VM berhasil dibuat, maka GCP akan menyediakan alokasi sumber daya untuk *server* yang akan digunakan. Melalui GCP, pengguna juga dapat melihat biaya untuk menjalankan VM setiap bulan.

Instalasi Nginx Proxy Manager dengan Docker Compose

Langkah pertama yang dilakukan yaitu masuk ke *server proxy* yang sudah terpasang Docker Compose. *Server* VM dikendalikan melalui SSH yang berada di Google Cloud Console. Untuk instalasi Nginx Proxy Manager diperlukan direktori baru menggunakan perintah berikut.

```
mkdir nginx-proxy-manager
cd nginx-proxy-manager
```

Kemudian *file* docker-compose.yml dibuat dengan menggunakan editor teks melalui perintah `nano`. Untuk menambahkan konten ke dalam *file* docker-compose.yml, Nginx Proxy Manager perlu dijalankan menggunakan Docker Compose dengan perintah berikut.

```
nano docker-compose.yml
docker-compose up -d
```

```
version: '3.8'
services:
  app:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: unless-stopped
    ports:

      - '80:80'
      - '443:443'
      - '81:81'

    volumes:
      - ./data:/data
      - ./letsencrypt:/etc/letsencrypt
```

Setelah *container* selesai berjalan, Nginx Proxy Manager dapat diakses melalui *browser* dengan mengunjungi alamat IP *public static* yang dipesan sebelumnya, yaitu pada alamat URL http://35.240.253.0:81. Selanjutnya dilakukan perubahan *username default* (admin@example.com) dan *password (*changeme).

Pembuatan *Docker Images* dan *Deploy Website* Statis

Tahap ini merupakan proses *deploy* aplikasi web yang menggunakan Nginx Sebagai *Web Server*. Hal yang harus dipersiapkan sebelumnya yaitu *source code* dari halaman *website* berserta Dockerfile. Dockerfile adalah *file* konfigurasi yang digunakan untuk membangun *docker image* [11]. Dalam Dockerfile, ditentukan komponen apa yang harus dimasukkan ke dalam *image*. Berikut perintah yang digunakan untuk membuat *image* web Nginx dalam Dockerfile.

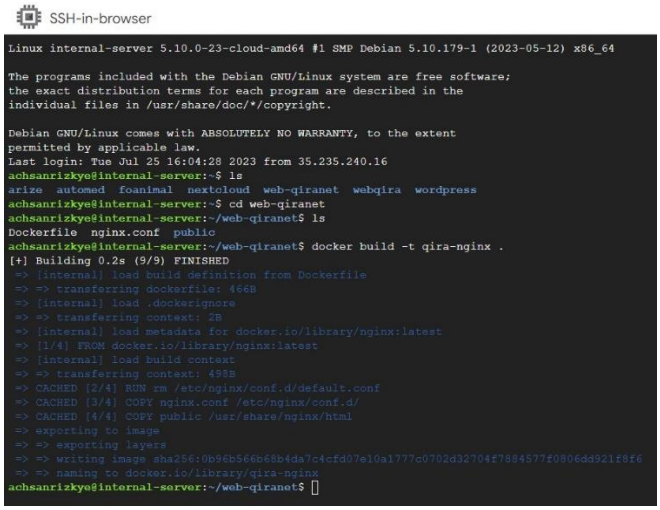
```
FROM nginx:latest

RUN rm /etc/nginx/conf.d/default.conf

COPY nginx.conf /etc/nginx/conf.d/

COPY public /usr/share/nginx/html

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Selanjutnya, semua *source code* dikompres menjadi satu ke dalam format ZIP. *File* tersebut kemudian diunggah ke *server* aplikasi melalui SSH dengan fitur *upload file* pada SSH Console. Setelah *file source code* diekstrak, *file* dimasukkan ke dalam direktori `webqira`. *Docker images* dibuat dengan perintah berikut.

```
unzip web-qiranet.zip
cd web-qiranet
docker build -t qira-nginx
```

Setelah *docker images* berhasil dibuat, langkah berikutnya adalah *deploy image* tersebut ke dalam *docker container* melalui perintah berikut. *Port* yang digunakan antara lain *port* 8888 sebagai *port* yang diakses dari luar jaringan, dan *port* 80 sebagai *port* aplikasi yang berjalan pada jaringan internal *docker*. Pada tahap ini, *deploy docker container* sudah berhasil dan sudah berjalan [12]. Gambar 5 memperlihatkan tampilan SSH Console pada pembuatan *docker images*.

```
docker run -d -p 8888:80 --restart=unless-stopped qira-nginx
```



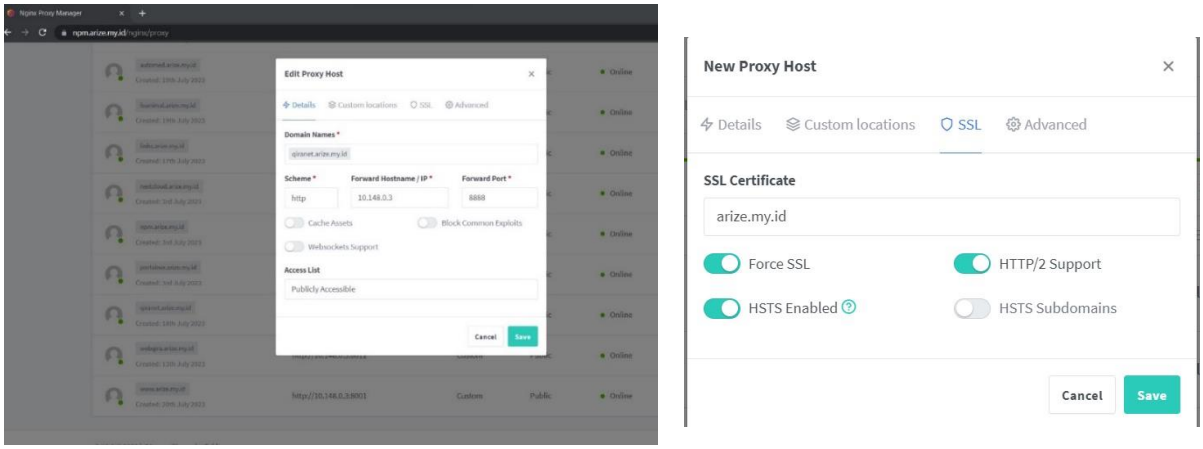
Gambar 5. Tampilan SSH Console pada Pembuatan *Docker Images*

Konfigurasi Nginx Proxy Manager sebagai *Reverse Proxy*

Setelah *docker container* dikonfigurasi dan dapat berfungsi dengan baik, maka dilakukan konfigurasi Nginx Proxy Manager sebagai *reverse proxy* [13]. Dalam hal ini dibutuhkan akses *admin user interface* dengan memasukkan *username* dan *password*. Di dalam tampilan Web UI, navigasi diarahkan untuk menambahkan *proxy host* dan mengisi beberapa detail informasi. Langkah selanjutnya dilakukan konfigurasi *domain name* yang akan digunakan dengan *subdomain* qiranet.arize.my.id. Pada opsi *scheme* dipilih HTTP, karena *server backend* aplikasi menggunakan protokol tersebut. Kemudian diisikan alamat IP internal *server* aplikasi, yaitu 10.148.0.3. *Port* aplikasi yang digunakan untuk mengekspos aplikasi web yaitu *port* 8888.

Pada opsi menu SSL, sertifikat SSL dapat ditambahkan dengan menggunakan fitur yang telah disediakan oleh Nginx Proxy Manager untuk menghasilkan sertifikat SSL dari Let's Encrypt. Jika sudah memiliki SSL *certificate* sendiri, maka sertifikat dapat diunggah langsung ke Nginx Proxy Manager. Setelah menambahkan sertifikat SSL, opsi *Force* SSL digunakan agar pengguna selalu diarahkan untuk mengakses situs web melalui koneksi HTTPS (SSL/TLS) dan mencegah akses menggunakan protokol HTTP yang tidak aman. Selain itu, terdapat opsi HSTS *Enabled* yaitu mekanisme keamanan yang memberitahu *browser* untuk selalu menggunakan koneksi HTTPS saat berkomunikasi dengan situs web [14].

Konfigurasi lainnya yaitu HTTP/2 *Support* untuk mengaktifkan dukungan terhadap HTTP/2. Dengan mengaktifkan dukungan ini, Nginx Proxy Manager dapat memfasilitasi komunikasi antara *client* dan *server* menggunakan protokol HTTP/2 saat melewati *reverse proxy*. Konfigurasi *reverse proxy* dan SSL pada Nginx Proxy Manager masing-masing diperlihatkan pada Gambar 6(a) dan 6(b).

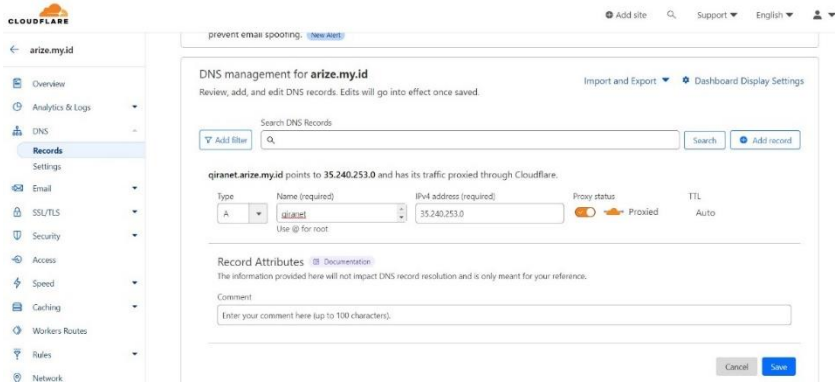


(a) (b)

Gambar 6. Konfigurasi *Reverse Proxy* (a) dan SSL (b) pada Nginx Proxy Manager

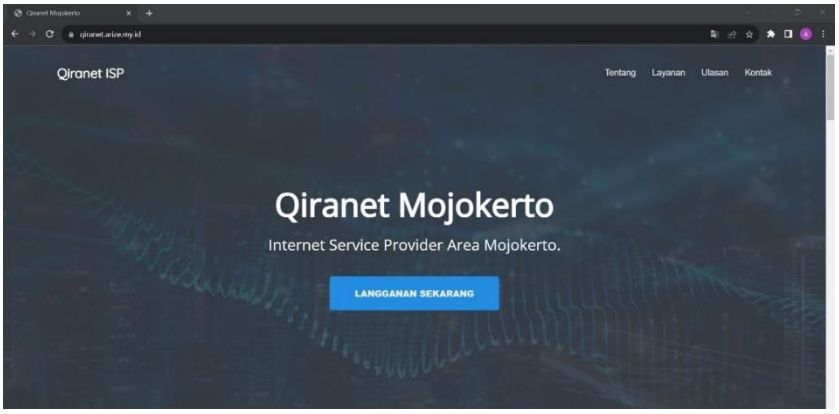
Konfigurasi CDN dan DNS *Record* arize.my.id pada Cloudflare

Konfigurasi dilakukan dengan mengakses akun Cloudflare, kemudian pada menu DNS *Record* ditambahkan *record* dengan memilih tipe *record* A. Kemudian diberikan nama *domain* qiranet.arize.my.id dan alamat IP *Public* 35.240.253.0 yang merupakan informasi IP dari VM *proxy*. Fitur *proxy* diaktifkan agar *website* dilayani oleh CDN Cloudflare [15]. Tampilan dari hasil penambahan DNS *Record* ditunjukkan pada Gambar 7.



Gambar 7. Penambahan DNS *Record* pada Cloudflare

Setelah menjalankan seluruh rangkaian proses *deploy* dan konfigurasi, maka halaman *website* dapat diakses melalui internet dengan membuka *browser* dan memasukkan alamat URL https://qiranet.arize.my.id. Tampilan halaman website yang dihasilkan diperlihatkan pada Gambar 8.



Gambar 8. Hasil tampilan halaman *website*

Pengujian Sistem dengan Blackbox Testing

Pengujian sistem bertujuan untuk memastikan bahwa aplikasi *website* statis yang diintegrasikan dengan CDN, *cloud computing*, dan *docker container* dapat berfungsi dengan baik dalam hal fungsionalitas, kinerja, keamanan, serta integritas integrasi teknologi yang digunakan. Pengujian dilakukan menggunakan metode *blackbox* pada aplikasi *website* statis yang di-*host* dalam *docker container*, menggunakan manajemen *proxy* pada Nginx Proxy Manager, serta terintegrasi dengan layanan CDN Cloudflare dan Google Cloud Compute Engine. Pengujian ini melibatkan pengujian fungsionalitas, pengujian kinerja, pengujian keamanan dasar, dan pengujian integrasi. Hasil pengujian diuraikan pada Tabel 1.

Tabel 1. Hasil Pengujian Menggunakan *Blackbox Testing*

Skenario Pengujian	Deskripsi Pengujian	Hasil Pengujian	Kesimpulan
Pengujian Fungsionalitas	Pengujian ini akan memastikan bahwa semua elemen pada *website* ditampilkan dengan benar dan tautan antar halaman berfungsi dengan baik. Hal ini mencakup: • Memeriksa apakah semua halaman dapat diakses. • Memastikan gambar, teks, dan elemen lainnya ditampilkan dengan benar. • Verifikasi tautan antar halaman.	• *Website* berhasil diakses melalui Nginx Proxy Manager. Semua halaman *website* dapat diakses tanpa kesalahan. • Semua elemen konten *website* termasuk gambar, teks, dan elemen lainnya, ditampilkan dengan benar. • Semua tautan antar halaman berfungsi seperti yang diharapkan.	Valid
Pengujian Kinerja	Pengujian ini akan mengevaluasi bagaimana *website* menangani lalu lintas tinggi dan apakah waktu muat halaman tetap cepat. Hal ini mencakup: • Pengujian beban tinggi pada *website*. • Pemantauan waktu muat halaman pada kondisi normal dan beban tinggi.	• Pengujian beban menggunakan ApacheBench sebanyak 10000 koneksi dan 1000 *user* bersamaan menunjukkan bahwa *website* tetap responsif dan tidak mengalami penurunan kinerja yang signifikan saat diakses oleh banyak pengguna secara bersamaan. • Waktu muat halaman tetap dalam batas yang dapat diterima bahkan pada lalu lintas tinggi. Performa *website* dalam kondisi beban tinggi memenuhi harapan.	Valid
Pengujian Keamanan Dasar	Pengujian ini akan menguji tingkat dasar keamanan, termasuk perlindungan terhadap serangan *Denial of Service* dan injeksi kode. Hal ini mencakup: • Penggunaan SSL/TLS pada *website*. • Pengujian penetrasi untuk mencari kerentanan. • Simulasi serangan *Denial of Service*. • Uji coba injeksi kode.	• *Website* sudah menggunakan SSL/TLS sehingga proses pertukaran data dapat terenkripsi. • Pengujian penetrasi dasar tidak mengungkapkan kerentanan yang signifikan. • Upaya serangan *Denial of Service* menggunakan SlowHTTPTest berhasil ditangkal. • Uji coba injeksi kode tidak berhasil merusak integritas *server*.	Valid

Pengujian Integrasi CDN	Pengujian ini akan memastikan bahwa konten *website* disimpan pada *cache* oleh Cloudflare CDN dan perubahan konten juga disimpan pada *cache*. Hal ini mencakup: • Pemantauan penggunaan *cache* CDN. • Perubahan pada konten *website* dan pemantauan apakah perubahan tersebut tersimpan pada *cache*.	• Pengujian menunjukkan bahwa konten disimpan pada *cache* oleh Cloudflare CDN dan terjadi pengurangan beban pada *server* utama. • Saat dilakukan perubahan konten pada *website*, perubahan tersebut secara otomatis tersimpan pada *cache* CDN. • Integrasi antara *website*, Nginx Proxy Manager, dan CDN Cloudflare berjalan dengan lancar.	Valid
Pengujian Integrasi dengan VM Compute Engine	Pengujian ini akan memeriksa apakah *website* dapat di-*host* dengan sukses pada Google Cloud Compute Engine dan apakah koneksi antara Compute Engine dan Nginx Proxy Manager stabil. Hal ini mencakup: • Pembuatan *instance* Compute Engine, konfigurasi, dan pengujian koneksi antara Compute Engine dan Nginx proxy. • Verifikasi ketersediaan *website*.	• Website berhasil di-*host* pada Google Cloud Compute Engine di dalam *docker container* dan dapat diakses melalui alamat IP dan *port* yang dikonfigurasi. • Koneksi antara VM Compute Engine dan Nginx Proxy Manager stabil selama pengujian. • VM Compute Engine memiliki *uptime* yang tinggi sebesar 99.99% sehingga *website* hampir selalu tetap tersedia.	Valid

SIMPULAN DAN SARAN

Pemanfaatan layanan CDN dan *cloud computing* secara keseluruhan dapat meningkatkan performa dan keamanan pada aplikasi berbasis *website*. Hasil pengujian dengan metode *blackbox* membuktikan manfaat implementasi CDN dan *cloud computing* dalam aplikasi berbasis web. Penggunaan Nginx Proxy Manager sebagai *reverse proxy* memberikan kemudahan dalam mengatur dan mengelola *proxy server*. Sementara itu, pemanfaatan *docker container* pada aplikasi berbasis web memberikan isolasi, portabilitas, skalabilitas, efisiensi, dan pengelolaan aplikasi secara efisien dan responsif.

Saran untuk penelitian atau pengembangan serupa antara lain melakukan *upgrade* ke layanan CDN Cloudflare yang berbayar guna mendapatkan manfaat tambahan seperti akses ke fitur-fitur lebih lanjut dan peningkatan kecepatan. Selain itu, dapat dipertimbangkan penggunaan Traefik sebagai alternatif Nginx Proxy Manager, ataupun penggunaan arsitektur *microservices* dengan alat orkestrasi *container* seperti Kubernetes. Untuk meningkatkan kehandalan dan keamanan infrastruktur yang diimplementasikan, disarankan untuk melakukan pengujian keamanan dan performa yang lebih mendalam.

DAFTAR PUSTAKA

[1] Ivan, Firmansyah. (2021). PEMANFAATAN GOOGLE CLOUD DAN TEKNIK LOAD BALANCING UNTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB.
[2] Jh, D. E., Umar, R., & Riadi, I. (2019). Implementation of Cloudflare Hosting for Speeds and Protection on The Website.
[3] Shi, F., Fan, L., Lai, X., Chen, Y., & Lin, W. (2021). A hierarchical caching strategy in content delivery network. Computer Communications, 179, 92-101. https://doi.org/10.1016/j.comcom.2021.07.029.
[4] Nadeem, M., Arshad, A., Riaz, S., Zahra, S. W., Rashid, M., Band, S. S., & Mosavi, A. (2023). Preventing Cloud Network from Spamming Attacks Using Cloudflare and KNN. Computers, Materials and Continua, 74(2), 2641–2659. https://doi.org/10.32604/cmc.2023.028796.
[5] M. Ghaznavi, E. Jalaipour, M. A. Salahuddin and R. Boutaba, "Content Delivery Network Security : A Survey," IEEE Communications Surveys & Tutorials, vol. 23, no. 4, pp. 1-3, 2021.

[6] Aparna, Shashikant, Joshi., Rishabh, Cambo., Yashika, arora., Ashi, Gupta., Dr., Manjot, Kaur, Bhatia. (2022). Cloud Computing. International Journal For Science Technology And Engineering, 10(12):758-761. doi: 10.22214/ijraset.2022.48010.
[7] Rumetna, M. S. (2018). PEMANFAATAN CLOUD COMPUTING PADA DUNIA BISNIS: STUDI LITERATUR. Jurnal Teknologi Informasi Dan Ilmu Komputer, 5(3), 305. https://doi.org/10.25126/jtiik.201853595.
[8] Balatamoghna, B., Jaganath, A., Vaideeshwaran, S., Subramanian, A., & Suganthi., K. (2021). Integrated balancing approach for hosting services with optimal efficiency - Self Hosting with Docker. Materials Today: Proceedings, 62, 4612-4619. https://doi.org/10.1016/j.matpr.2022.03.078.
[9] Tuara, H. A., Maridyah, N., Khaerudin, K., Artikel, I., Abstrak, P., Studi, -Program, Elektro, T., & Kunci, K. (2021). Implementasi CDN(Content Delivery Network) menggunakan Cloudflare terintegrasi dengan Docker Countainer. Journal of Mechatronic and Electrical Engineering, 1(1), 42–51. https://doi.org/10.22219.
[10] Shah, J., Dubaria, D.: Building Modern Clouds: Using Docker, Kubernetes & Google Cloud Platform. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2019, pp. 0184–0189. https://doi.org/10.1109/CCWC.2019.8666479.
[11] Kropp, A., & Torre, R. (2019). Docker: Containerize your application. Computing in Communication Networks, 231-244. https://doi.org/10.1016/B978-0-12-820488-7.00026-8.
[12] Dwiyatno, S., Rakhmat, E., & Gustiawan, O. (2020). IMPLEMENTASI VIRTUALISASI SERVER BERBASIS DOCKER CONTAINER. 7(2).
[13] Lei, Z., Zhou, H., Ye, S., Hu, W., & Liu, G. (2019). Cost-Effective Server-side Re-deployment for Web-based Online Laboratories Using NGINX Reverse Proxy. IFAC-PapersOnLine, 53(2), 17204-17209. https://doi.org/10.1016/j.ifacol.2020.12.1748.
[14] Dastres R, & Soori M. (2020) "Secure Socket Layer (SSL) in the Network and Web Security". International Journal of Computer and Information Engineering, 14(10), 331-333.
[15] Laksmiati, Dewi. (2022). PENGUJIAN OPTIMASI PERFORMA WEBSITE MENGGUNAKAN CLOUDFLARE DENGAN METODE STRESS TEST (Vol. 7).