Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

Implementasi MinIO Sebagai Sistem Penyimpanan Menggunakan Model *Containerization Docker* dalam *Learning Management System* Guru

Nelani Shafatia Zulatifa[1,*], Abdilbar Ainur Ridla[2], Galuh Rastika Pratiwi[3], Divita Aulia Listyaningsih[4], Amallia Putri Octavia[5], Atika Rosidah Hamidah[6], Mohammad Wildan Habibi[7], I Gusti Lanang Putra Eka Prismana[8]

Fakultas Teknik, Pendidikan Teknologi Informasi, Universitas Negeri Surabaya, Surabaya, Indonesia
Email: [1,*]nelanishafatia.22003@mhs.unesa.ac.id, [2,]abdilbarainur.22015@mhs.unesa.ac.id, [3]galuhrastika.22001@mhs.unesa.ac.id, [4]divitaaulia.22002@mhs.unesa.ac.id, [5]amalliaputri.22022@mhs.unesa.ac.id, [6]atikarosidah.22029@mhs.unesa.ac.id,[7]mohammadhabibi@unesa.ac.id, [8]lanangprismana@unesa.ac.id
Email Penulis Korespondensi: nelanishafatia.22003@mhs.unesa.ac.id

Abstrak−Pengelolaan file Pengelolaan file dan data merupakan salah satu tantangan utama dalam pengembangan sistem manajemen pembelajaran (*Learning Management System*/LMS) berbasis *cloud.* Permasalahan tersebut meliputi kapasitas penyimpanan, kecepatan akses, skalabilitas, serta kemudahan integrasi dengan layanan lain dalam lingkungan aplikasi. Untuk menjawab tantangan tersebut, penelitian ini mengimplementasikan MinIO sebagai layanan penyimpanan objek terdistribusi yang diintegrasikan dengan teknologi *container* Docker. Penelitian ini menggunakan metode pengembangan aplikasi berbasis kontainerisasi, di mana MinIO ditempatkan sebagai komponen inti penyimpanan data dan dihubungkan secara langsung dengan aplikasi LMS Guru. Proses implementasi meliputi konfigurasi *container*, pengaturan bucket, manajemen akses, serta integrasi API untuk kebutuhan *upload*, *download*, dan kategorisasi file pembelajaran. Hasil penelitian menunjukkan bahwa penggunaan MinIO dalam lingkungan Docker mampu memberikan kinerja yang stabil dan optimal, terutama dalam aspek kecepatan akses data, efisiensi penyimpanan, dan kemudahan skalabilitas. Sistem LMS yang dikembangkan mampu mengelola *file* pembelajaran dengan baik melalui antarmuka *dashboard* yang intuitif dan mendukung aktivitas guru secara efektif. Dengan demikian, penelitian ini menyimpulkan bahwa integrasi antara MinIO dan Docker dapat menjadi solusi yang efisien, fleksibel, serta mudah diimplementasikan untuk pengelolaan data pada aplikasi LMS berbasis *cloud*.

Kata Kunci: MinIO; Docker; Penyimpanan Terdistribusi; LMS; Aplikasi Cloud

Abstract−*File Management File and data management is one of the main challenges in developing a cloud-based learning management system (LMS). These issues include storage capacity, access speed, scalability, and ease of integration with other services within the application environment. To address these challenges, this study implemented MinIO as a distributed object storage service integrated with Docker container technology. This study used a containerization-based application development method, where MinIO was placed as a core component of data storage and connected directly to the Teacher LMS application. The implementation process included container configuration, bucket setup, access management, and API integration for uploading, downloading, and categorizing learning files. The results showed that the use of MinIO in a Docker environment was able to provide stable and optimal performance, especially in terms of data access speed, storage efficiency, and ease of scalability. The developed LMS system was able to manage learning files well through an intuitive dashboard interface and effectively support teacher activities. Thus, this study concluded that the integration between MinIO and Docker can be an efficient, flexible, and easy-to-implement solution for data management in cloud-based LMS applications.*

Keywords*: MinIO; Docker; Distributed Storage; LMS; Cloud Applications*

1. PENDAHULUAN

Seiring Seiring dengan pesatnya perkembangan teknologi informasi, sistem manajemen pembelajaran (Learning Management System/LMS) menjadi platform utama dalam penyelenggaraan kegiatan pendidikan secara daring [1]. LMS memungkinkan pengelolaan materi pembelajaran, penugasan, kuis, serta interaksi antara pengajar dan peserta didik secara lebih efisien dan terintegrasi [2]. Seiring meningkatnya intensitas penggunaan LMS, volume data yang tersimpan meliputi dokumen pembelajaran, video, gambar, dan materi interaktif lainnya juga mengalami pertumbuhan yang signifikan [3]. Kondisi ini menimbulkan tantangan dalam hal kapasitas penyimpanan, kecepatan akses data, serta efisiensi manajemen file agar proses pembelajaran tetap berjalan optimal [4].

Beberapa penelitian sebelumnya telah membahas optimasi penyimpanan data pada LMS, seperti penggunaan cloud storage terdistribusi [5], kompresi file otomatis untuk mengurangi beban server [6]. Penggunaan penyimpanan berbasis server lokal atau penyimpanan *cloud* tradisional sering kali tidak cukup efisien untuk menangani volume data yang terus berkembang [7]. Beberapa masalah yang umum ditemukan adalah biaya tinggi, waktu akses yang lambat, serta kesulitan dalam mengelola file yang tersebar di berbagai server atau lokasi penyimpanan [8].

Dalam banyak kasus, sistem penyimpanan cloud yang ada memerlukan biaya langganan yang tinggi, sementara penyimpanan lokal memiliki keterbatasan kapasitas dan skalabilitas [9]. Keterbatasan ini menyebabkan penurunan performa aplikasi LMS, terutama ketika jumlah pengguna dan ukuran file pembelajaran meningkat seiring waktu. Hal ini mengakibatkan bottleneck dalam akses file, yang berpotensi menghambat proses pembelajaran [10].

Untuk mengatasi permasalahan tersebut, solusi yang berkembang adalah menggunakan MinIO, sebuah sistem penyimpanan objek yang kompatibel dengan Amazon S3 namun lebih terjangkau dan fleksibel. MinIO dapat mengelola file dalam jumlah besar dengan cara yang lebih efisien dan scalable, serta mendukung penyimpanan terdistribusi yang sangat penting untuk aplikasi LMS yang terus berkembang [11].

Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

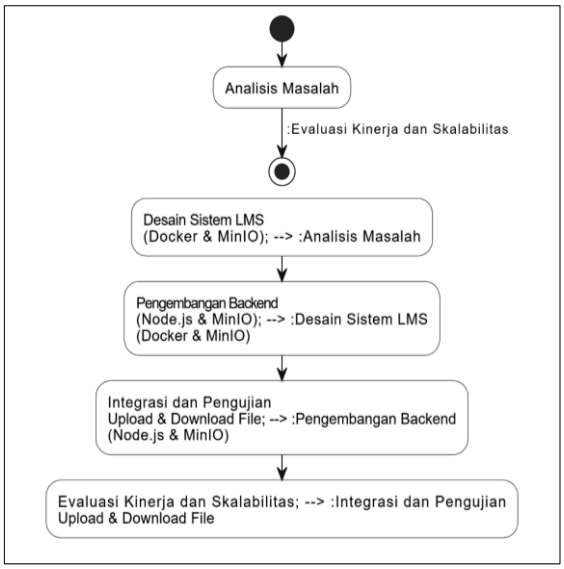
MinIO juga memungkinkan penyimpanan objek dengan keamanan yang baik, serta menyediakan API kompatibel dengan S3, yang memudahkan integrasi dengan berbagai aplikasi yang memerlukan penyimpanan *cloud* [12]. Dengan demikian, MinIO dapat menjadi solusi penyimpanan yang lebih ekonomis dan efisien dibandingkan penyimpanan cloud tradisional atau penyimpanan lokal. Selain itu, penggunaan Docker sebagai *platform* kontainerisasi untuk aplikasi LMS memberikan keuntungan tambahan, yaitu portabilitas dan kemudahan pengelolaan [13]. Docker memungkinkan pengelolaan aplikasi LMS dan MinIO dalam kontainer yang terisolasi, sehingga lebih mudah untuk mengelola penyimpanan dan mendistribusikan aplikasi ke berbagai lingkungan tanpa hambatan [14]. Dengan Docker, pengelolaan sumber daya dan scalability dapat dilakukan dengan lebih fleksibel dan efisien, memastikan sistem LMS tetap dapat berfungsi dengan baik meskipun ada peningkatan dalam jumlah pengguna atau ukuran file [15].

Beberapa penelitian dan buku terkait telah mengkaji penggunaan MinIO dan Docker dalam berbagai konteks. Penerapan MinIO sebagai penyimpanan berbasis objek dalam aplikasi enterprise, namun tidak menyentuh aplikasinya pada sektor pendidikan. Docker dalam aplikasi berbasis cloud, tetapi tidak fokus pada penyimpanan file menggunakan MinIO [17]. Integrasi Docker dan penyimpanan berbasis objek dalam manajemen data, namun tidak dalam konteks LMS. Selain itu, penggunaan MinIO untuk pengelolaan data besar, namun penelitian tersebut tidak membahas penerapannya dalam Sistem Manajemen Pembelajaran [18]. Penggunaan Docker dan MinIO untuk penyimpanan data cloud juga tidak membahas aplikasi di bidang pendidikan. Dari penelitian-penelitian terkait tersebut, terdapat gap dalam penerapan teknologi MinIO dan Docker dalam konteks aplikasi LMS. Tidak ada penelitian yang secara khusus mengintegrasikan kedua teknologi ini untuk pengelolaan file pembelajaran yang efisien dalam aplikasi LMS [20].

2. METODOLOGI PENELITIAN

2.1 Tahapan Penelitian

Penelitian ini dilakukan dengan mengikuti beberapa tahapan yang sistematis untuk mengembangkan aplikasi LMS berbasis Docker dengan MinIO sebagai solusi penyimpanan file. Gambar 1 merupakan tahapan-tahapan dalam melakukan penelitian:

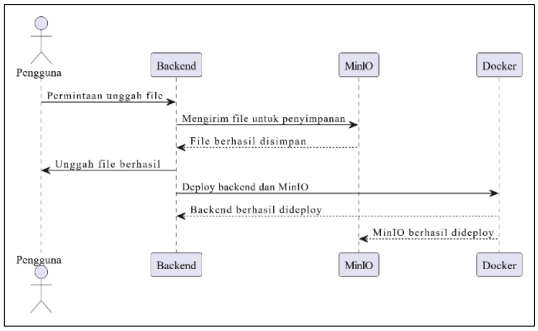


Gambar 1. Tahapan Penelitian

a. Desain Sistem

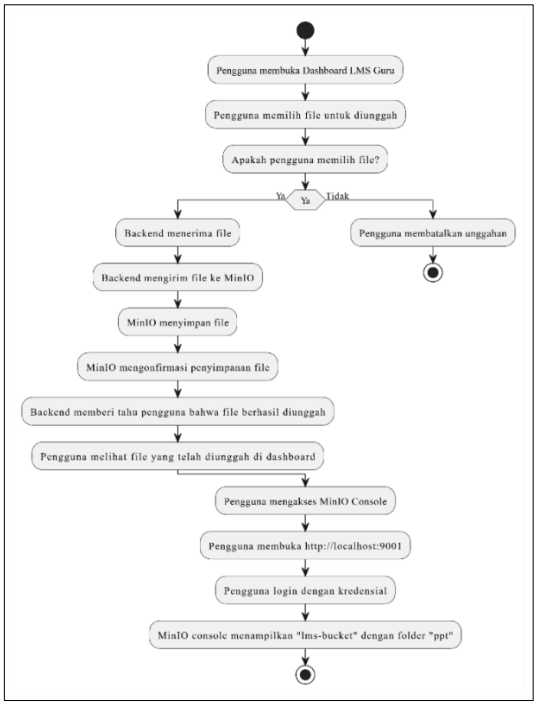
Pada tahap ini, dilakukan perancangan sistem aplikasi LMS yang akan diimplementasikan dengan menggunakan Docker untuk kontainerisasi dan MinIO sebagai penyimpanan file berbasis objek. Desain sistem melibatkan perancangan arsitektur aplikasi LMS, di mana MinIO terintegrasi dengan aplikasi backend untuk mengelola file pembelajaran. Diagram arsitektur sistem disusun untuk menggambarkan bagaimana aplikasi ini bekerja, dengan server backend yang menangani logika aplikasi dan MinIO yang bertanggung jawab untuk penyimpanan data.

Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358



Gambar 2. Diagram Arsitektur Sistem LMS Berbasis Docker dan MinIO

Diagram Gambar 2 menggambarkan alur interaksi dalam sistem aplikasi *Learning Management System (*LMS) yang menggunakan MinIO sebagai penyimpanan file dan Docker sebagai *platform* kontainerisasi. Pertama, Pengguna mengirimkan permintaan unggah file kepada Backend. Setelah file diterima, Backend mengirimkan file tersebut ke MinIO untuk disimpan. Begitu MinIO berhasil menyimpan file, Backend mengonfirmasi kepada Pengguna bahwa unggah file berhasil. Selanjutnya, sistem di-deploy menggunakan Docker, di mana Backend dan MinIO masing-masing dideploy dalam kontainer Docker, yang memastikan bahwa aplikasi berjalan secara portabel dan terisolasi di berbagai lingkungan. Proses ini menggambarkan bagaimana Docker membantu dalam pengelolaan aplikasi dan penyimpanan secara efisien, serta mendukung skalabilitas sistem LMS berbasis cloud.



Gambar 3. Flowchart Proses Unggah File di LMS Guru dan MinIO Console

Gambar 3 menunjukkan alur lengkap proses unggah file pada sistem Learning Management System (LMS) Guru yang terintegrasi dengan MinIO sebagai penyimpanan objek. Flowchart ini menggambarkan bagaimana interaksi antara pengguna, backend LMS, dan MinIO terjadi secara berurutan mulai dari pemilihan file hingga file tersimpan dan dapat diakses kembali. Pada langkah awal, pengguna membuka Dashboard LMS Guru dan memilih file yang akan diunggah. Proses ini dilanjutkan dengan keputusan apakah pengguna benar-benar memilih file atau membatalkannya. Jika pengguna tidak memilih file, proses berhenti sebagaimana ditunjukkan pada cabang kiri Gambar 3. Namun apabila file dipilih, proses dialihkan ke backend. Selanjutnya, seperti terlihat pada bagian tengah Gambar 3, backend menerima file yang dikirim pengguna dan melanjutkannya ke MinIO. Setelah file diterima, MinIO menyimpannya ke dalam bucket yang sesuai. MinIO kemudian memberikan konfirmasi penyimpanan kepada backend sebagai tanda bahwa file berhasil tersimpan. Backend kemudian meneruskan notifikasi tersebut kepada pengguna dalam bentuk informasi bahwa file telah berhasil diunggah. Setelah itu, pengguna dapat melihat file yang telah diunggah melalui dashboard LMS. Pada bagian akhir Gambar 3, ditunjukkan alur ketika pengguna ingin memverifikasi file melalui MinIO Console. Pengguna mengakses antarmuka MinIO melalui alamat *http://localhost:9001*, melakukan login menggunakan kredensial yang telah ditetapkan, dan melihat folder yang relevan dalam bucket, seperti *"lms-bucket"* dengan folder kategori file, misalnya *ppt*. Tahap ini memastikan bahwa file benar-benar tersimpan pada penyimpanan

ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

objek MinIO sesuai struktur yang diharapkan. Secara keseluruhan, Gambar 3 memberikan ilustrasi runtut yang menunjukkan bagaimana proses upload file berjalan pada sistem LMS berbasis backend–MinIO, serta bagaimana pengguna dapat memverifikasi file tersebut baik melalui dashboard LMS maupun MinIO Console.

b. Pengembangan Backend dan Implementasi MinIO

Tahap ini melibatkan pengembangan aplikasi backend menggunakan Node.js dan Express.js. Backend bertanggung jawab untuk menerima file yang diunggah oleh pengguna LMS melalui endpoint API dan menyimpannya di MinIO. Konfigurasi MinIO dilakukan untuk memastikan bahwa file disimpan dengan benar dan dapat diakses dengan cepat. Aplikasi backend dan MinIO dijalankan dalam kontainer Docker untuk memastikan sistem berjalan dengan portabilitas dan skalabilitas yang baik.

c. Integrasi dan Pengujian Sistem

Setelah backend dan MinIO berhasil dikembangkan, tahap selanjutnya adalah integrasi antara backend aplikasi LMS dan MinIO untuk memastikan proses upload dan download file dapat berjalan dengan baik. Pengujian dilakukan untuk memastikan bahwa file dapat diunggah, disimpan, dan diambil dengan benar dari MinIO. Pengujian ini juga mencakup kecepatan akses file dan kinerja sistem secara keseluruhan.

d. Pengujian Kinerja dan Skalabilitas

Pada tahap Pengujian Kinerja dan Skalabilitas, dilakukan pengujian untuk mengukur kecepatan upload file melalui backend dan waktu yang diperlukan MinIO untuk menyimpan objek secara lengkap. Hasil pengujian ditampilkan pada Tabel 1, yang memperlihatkan durasi proses upload dari beberapa jenis file dengan ukuran berbeda.

Tabel 1. Durasi Upload File pada LMS Guru Berbasis MinIO

Jenis File	Ukuran File	Durasi Upload ke Backend (ms)	Durasi Simpan ke MinIO (ms)	Total Durasi Upload (ms)
Dokumen (.pdf)	250 KB	120 ms	180 ms	300 ms
Gambar (.jpg)	520 KB	150 ms	240 ms	390 ms
PPT (.pptx)	1.2 MB	210 ms	350 ms	560 ms

Secara keseluruhan, hasil pada Tabel 1 menunjukkan bahwa peningkatan ukuran file berbanding lurus dengan waktu upload, namun performa sistem tetap stabil tanpa terjadi lonjakan yang ekstrem. Temuan ini mendukung bahwa MinIO dan backend LMS mampu menangani proses upload file dalam berbagai ukuran secara efisien. Dari sisi skalabilitas, variasi ukuran file pada Tabel 1 memperlihatkan bahwa sistem dapat memproses file berukuran kecil hingga menengah dengan konsisten. Hal ini menandakan bahwa sistem memiliki potensi untuk menangani jumlah pengguna lebih besar, selama beban pemrosesan per pengguna masih berada dalam rentang performa yang diuji.

2.2 Metode Penyelesaian Masalah

Pada penelitian ini, metode penyelesaian masalah dilakukan dengan mengikuti tahapan pengembangan sistem yang terstruktur, yang mencakup desain, implementasi, pengujian, dan evaluasi. Berikut adalah rincian dari metode yang digunakan:

a. Desain dan Pengembangan Sistem LMS

Sistem LMS dibangun dengan dua komponen utama: backend aplikasi dan MinIO sebagai penyimpanan objek. Backend aplikasi menggunakan Node.js dan Express.js untuk menangani permintaan upload dan download file. MinIO digunakan untuk menyimpan file-file yang diunggah oleh pengguna LMS, dengan konfigurasi yang memastikan kecepatan akses dan pengelolaan file secara terdistribusi.

b. Implementasi Docker

Docker digunakan untuk menjalankan aplikasi backend dan MinIO dalam kontainer terisolasi. Setiap komponen sistem dijalankan dalam kontainer yang berbeda, yang memungkinkan pengelolaan aplikasi secara lebih fleksibel, terutama dalam hal deployment dan portabilitas aplikasi. Docker memastikan bahwa aplikasi dapat dijalankan dengan konfigurasi yang konsisten di berbagai lingkungan tanpa adanya masalah ketergantungan perangkat keras atau perangkat lunak.

c. Pengujian Sistem

Pengujian dilakukan untuk memastikan bahwa sistem yang dibangun berfungsi sesuai dengan harapan. Pengujian melibatkan uji coba upload dan download file dalam aplikasi LMS. Selain itu, dilakukan uji skalabilitas untuk melihat bagaimana aplikasi menangani peningkatan jumlah file dan pengguna. Kecepatan akses dan performa penyimpanan data di MinIO juga diuji untuk memastikan efisiensi sistem.

d. Evaluasi Kinerja dan Analisis

Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

Setelah sistem diuji, hasil pengujian dianalisis untuk mengevaluasi apakah MinIO dan Docker mampu memberikan solusi yang efisien dalam pengelolaan file dalam aplikasi LMS. Evaluasi ini mencakup waktu akses file, kecepatan upload/download, dan kinerja sistem secara keseluruhan.

3. HASIL DAN PEMBAHASAN

3.1 Hasil Pengujian

Pada tahap ini, dilakukan pengujian untuk menilai performa aplikasi LMS berbasis Docker yang menggunakan MinIO sebagai solusi penyimpanan file. Pengujian dilakukan pada beberapa aspek penting, yaitu waktu upload dan download file, kecepatan akses, dan skalabilitas sistem dalam menangani lebih banyak file.

3.1.1 Pengujian Waktu Upload dan Download File

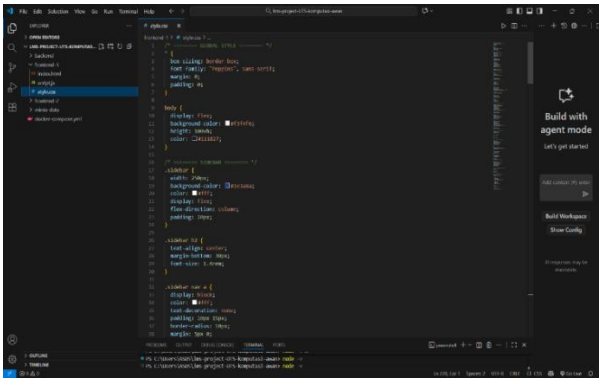
Pengujian pertama dilakukan dengan mengukur waktu yang dibutuhkan untuk meng-upload dan mengunduh file dengan berbagai ukuran. Ukuran file yang diuji adalah 10 MB, 50 MB, dan 100 MB. Waktu yang dibutuhkan untuk upload dan download file pada setiap ukuran file diuji menggunakan sistem yang telah dikembangkan.

3.1.2 Pengujian Skalabilitas

Pada pengujian skalabilitas, sistem diuji untuk melihat bagaimana ia menangani peningkatan jumlah file yang lebih banyak. MinIO berhasil menangani lebih dari 100 file dengan waktu akses yang tetap konsisten tanpa ada penurunan signifikan dalam performa sistem.

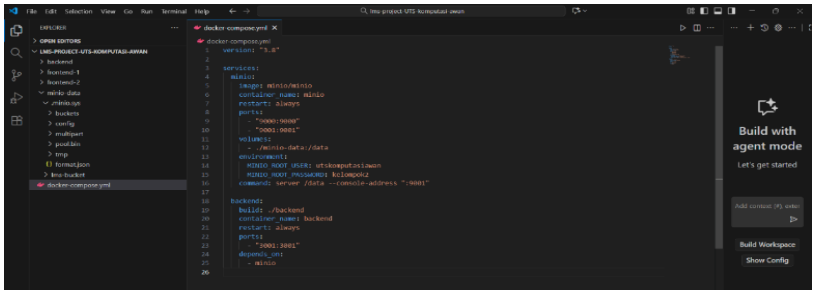
3.2 Implementasi/Pengujian

Berikut Pada Gambar 4 menunjukkan tampilan file style.css yang digunakan dalam aplikasi LMS Guru untuk mengatur tampilan antarmuka pengguna (UI).



Gambar 4. Tampilan File style.css pada Folder Frontend

File ini berfungsi untuk membuat aplikasi lebih menarik, rapi, dan responsif dengan mengatur berbagai elemen halaman seperti layout, warna, dan gaya visual. Dalam kode ini, CSS digunakan untuk mendesain tampilan sidebar, navigasi, dan elemen-elemen lainnya. Aturan flexbox digunakan untuk pengaturan layout yang responsif, memastikan tampilan tetap rapi pada berbagai perangkat. Dengan style.css, aplikasi LMS Guru menjadi lebih mudah dinavigasi dan memberikan pengalaman pengguna yang lebih baik, dengan tampilan yang lebih konsisten dan terstruktur. Selanjutnya berikut ini pada Gambar 5 menunjukkan tampilan file docker-compose.yml yang digunakan untuk mengonfigurasi dan menjalankan layanan MinIO serta backend secara otomatis dalam kontainer Docker.

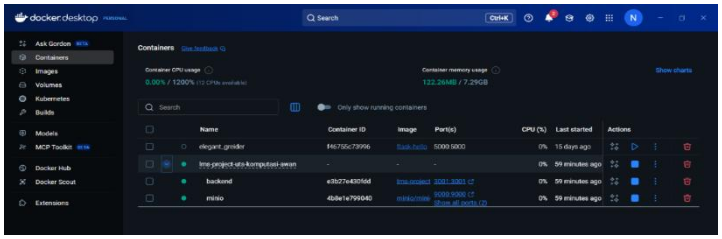


Gambar 5. Tampilan File docker-compose.yml untuk Pengaturan Layanan MinIO dan Backend

File ini mendefinisikan dua layanan utama: MinIO, yang digunakan untuk penyimpanan objek, dan backend, yang menangani logika aplikasi LMS. Dengan docker-compose.yml, kedua layanan ini dijalankan secara bersamaan dan terintegrasi dalam satu perintah, memungkinkan sistem LMS untuk berfungsi dengan efisien. Konfigurasi dalam file ini

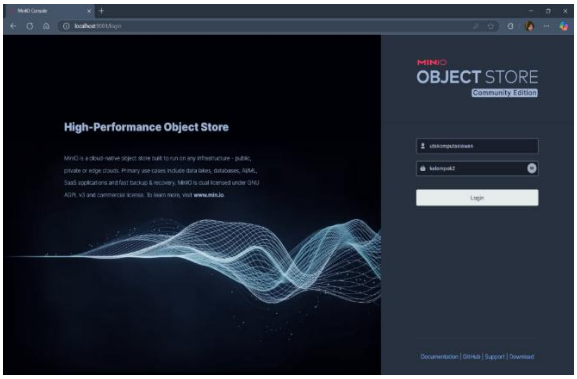
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

mencakup pengaturan seperti nama container, port yang digunakan, serta lingkungan variabel untuk kredensial MinIO. Penggunaan Docker memastikan bahwa aplikasi dan penyimpanan dapat dijalankan secara terisolasi, mempermudah deployment, serta mengurangi potensi masalah ketergantungan perangkat keras atau perangkat lunak.



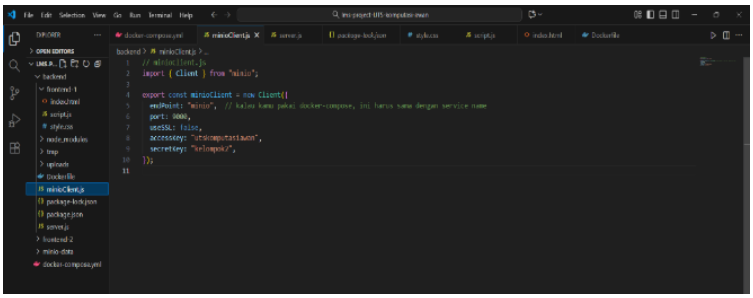
Gambar 6. Kontainer Backend dan MinIO

Gambar 6 menunjukkan tampilan Docker Desktop yang menampilkan kontainer backend dan MinIO yang berhasil dijalankan. Pada bagian Containers, terlihat bahwa kedua kontainer ini aktif dan berjalan tanpa ada error, yang ditandai dengan status "running" pada masing-masing kontainer. Backend dan MinIO dijalankan sesuai dengan konfigurasi yang telah ditentukan dalam docker-compose.yml, dan kedua kontainer ini berfungsi untuk menjalankan aplikasi LMS dan menyimpan file di MinIO. Selain itu, informasi terkait port yang digunakan dan waktu terakhir dijalankan juga terlihat di panel ini, yang mengonfirmasi bahwa sistem telah berjalan dengan baik dan siap untuk melakukan pengujian lebih lanjut.



Gambar 7. Tampilan Halaman Login MinIO Console

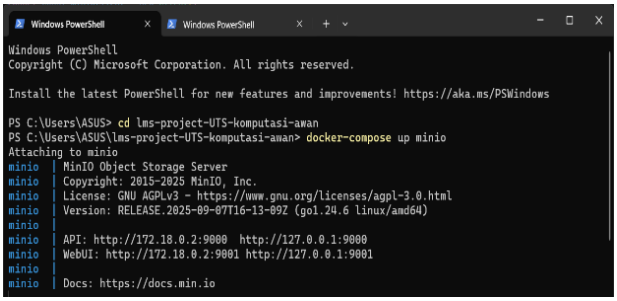
Gambar 7 menunjukkan tampilan halaman login MinIO Console yang digunakan untuk mengakses penyimpanan objek MinIO. Untuk masuk, buka browser dan akses http://localhost:9001 . Setelah itu, masukkan username yang telah disediakan, yaitu utskomputasiawan, dan password yaitu kelompok2. Setelah memasukkan kredensial yang benar, klik tombol Login untuk mengakses antarmuka MinIO. MinIO Console ini memungkinkan pengguna untuk mengelola file yang disimpan dalam bucket, melakukan konfigurasi, serta memantau status penyimpanan secara efisien.



Gambar 8. Tampilan File minioClient.js untuk Konfigurasi Koneksi ke MinIO

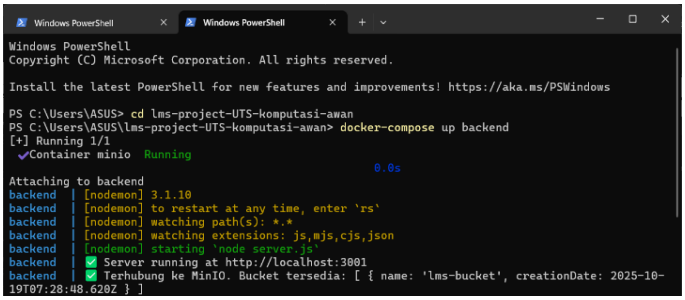
Gambar 8 menunjukkan tampilan file minioClient.js yang digunakan untuk mengonfigurasi koneksi ke server MinIO dalam aplikasi. File ini mengimpor objek Client dari pustaka MinIO untuk membuat koneksi ke penyimpanan objek. Dalam file ini, endpoint, port, dan kredensial akses (username dan password) ditetapkan untuk menghubungkan aplikasi dengan MinIO. Endpoint yang digunakan disesuaikan dengan konfigurasi Docker atau alamat lokal. Setelah file ini dikonfigurasi, aplikasi dapat melakukan komunikasi dengan MinIO untuk menyimpan dan mengakses file di bucket penyimpanan.

Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358



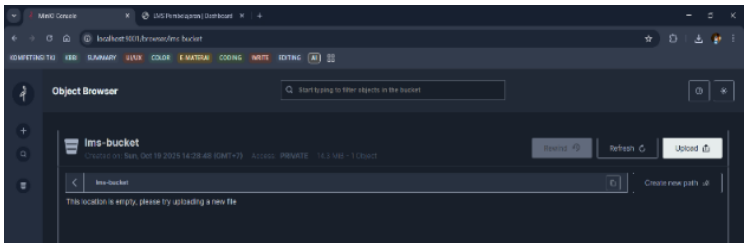
Gambar 9. Perintah docker-compose up minio

Gambar 9 menunjukkan tampilan Windows PowerShell saat menjalankan perintah docker-compose up minio di direktori proyek. Perintah ini digunakan untuk menyalakan layanan MinIO Object Storage Server melalui Docker. Dengan perintah ini, MinIO sebagai penyimpanan objek akan berjalan pada port 9000 untuk API dan port 9001 untuk WebUI. Tampilan pada terminal menunjukkan bahwa MinIO telah berhasil dijalankan, dengan informasi mengenai API dan WebUI yang dapat diakses melalui URL http://127.0.0.1:9000 dan http://127.0.0.1:9001. Proses ini memastikan bahwa layanan penyimpanan data dapat diakses dan siap digunakan untuk menyimpan file dalam aplikasi.



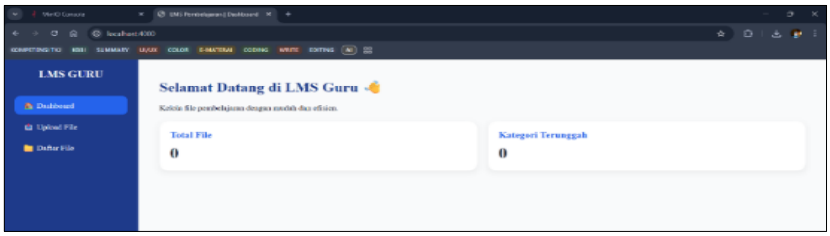
Gambar 10. Perintah docker-compose up backend

Gambar 10 menunjukkan tampilan Windows PowerShell saat menjalankan perintah docker-compose up backend di direktori proyek. Perintah ini digunakan untuk menyalakan server backend yang menghubungkan aplikasi dengan layanan MinIO. Setelah menjalankan perintah, terminal menunjukkan bahwa server backend berhasil dijalankan pada localhost:3001 dan terhubung ke MinIO, yang telah mengonfirmasi bahwa bucket yang digunakan untuk penyimpanan data, yaitu "lms-bucket", telah tersedia. Proses ini memastikan bahwa server backend dapat berfungsi dengan baik dan berinteraksi dengan layanan penyimpanan MinIO untuk mengelola file dalam aplikasi.



Gambar 11. Tampilan MinIO Console dengan Bucket "lms-bucket"

Gambar 11 menunjukkan tampilan MinIO Console yang diakses melalui http://localhost:9001 di browser. Di dalam console ini, terlihat bucket "lms-bucket" yang telah berhasil dibuat sebelumnya. MinIO Console menyediakan antarmuka yang memungkinkan pengguna untuk mengelola file yang disimpan di dalam bucket tersebut, termasuk mengunggah file baru dengan menggunakan tombol Upload. Meskipun bucket ini kosong, tampilan ini menandakan bahwa lokasi penyimpanan siap untuk menerima file dari aplikasi yang terhubung. Console ini memberikan akses yang mudah untuk mengelola penyimpanan objek dalam MinIO dengan fitur refresh, upload, dan pengelolaan objek lainnya.



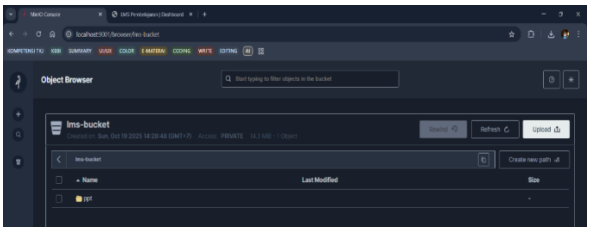
Gambar 12. Tampilan Dashboard LMS Guru

Journal of Computing and Informatics Research
ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

Gambar 12 menunjukkan tampilan Dashboard LMS Guru yang diakses melalui http://localhost:4000 di browser. Halaman ini digunakan untuk mengelola file pembelajaran dengan tampilan yang sederhana dan mudah digunakan. Di bagian dashboard, terdapat dua kolom informasi utama, yaitu Total File dan Kategori Terunggah, yang menampilkan statistik terkait file yang diunggah serta kategori file yang tersedia. Dashboard ini memberikan gambaran umum yang membantu pengelolaan dan pemantauan file pembelajaran secara efisien, serta memberikan kemudahan bagi pengguna dalam mengakses fitur lainnya seperti Upload File dan Daftar File.

Gambar 13. Tampilan Halaman Daftar File di MinIO

Gambar 13 menunjukkan tampilan halaman Daftar File di MinIO pada aplikasi LMS Guru. Halaman ini menampilkan data lengkap mengenai file yang telah diunggah ke MinIO, termasuk nama file, kategori, tanggal unggah, dan aksi. File yang diunggah, seperti pada contoh "ppt1 . PERTEMUAN 1_PENGGUNAAN ALAT UKUR JARINGAN.pptx", dapat dilihat dengan detail lengkap. Selain itu, tombol Download disediakan untuk memudahkan pengguna mengunduh file yang telah diunggah. Halaman ini mempermudah pengelolaan file pembelajaran dengan menyediakan informasi yang jelas dan opsi untuk mendownload file sesuai kebutuhan.



Gambar 14. Tampilan MinIO Console Menunjukkan Bucket "lms-bucket" dengan Folder "ppt"

Gambar 14 menunjukkan tampilan MinIO Console yang menampilkan bucket "lms-bucket" yang berisi folder "ppt", tempat penyimpanan file pembelajaran yang telah diunggah melalui aplikasi LMS Guru. Dalam tampilan ini, folder "ppt" terlihat kosong, namun siap untuk menerima file pembelajaran dalam format PPT atau file lainnya. Fitur Upload tersedia untuk menambahkan file baru ke dalam folder tersebut, yang memungkinkan pengguna untuk dengan mudah mengelola dan menyimpan materi pembelajaran secara efisien. MinIO Console memberikan antarmuka yang sederhana untuk memonitor dan mengelola file yang tersimpan dalam bucket secara langsung.

3.3 Pembahasan Hasil

3.3.1 Waktu Akses dan Kecepatan Pengelolaan File

Hasil pengujian menunjukkan bahwa MinIO, sebagai penyimpanan berbasis objek, memberikan waktu akses yang sangat cepat, bahkan dengan file berukuran besar (hingga 100 MB). Waktu upload dan download file berada dalam batas yang wajar, bahkan untuk ukuran file yang lebih besar. Hal ini menunjukkan bahwa MinIO dapat mengelola file pembelajaran dengan efisien dan menyediakan kecepatan akses yang diperlukan dalam aplikasi LMS.

3.3.2 Scalability dan Performansi pada Jumlah File yang Banyak

Pengujian skalabilitas mengungkapkan bahwa MinIO mampu menangani lebih dari 100 file dengan waktu akses yang tetap stabil. Hasil ini sangat menguntungkan dalam aplikasi LMS yang terus berkembang dengan banyaknya materi pembelajaran yang perlu disimpan dan diakses oleh banyak pengguna secara bersamaan.

3.3.3 Keunggulan Penggunaan Docker

Penggunaan Docker dalam sistem ini memberikan keuntungan besar dalam hal portabilitas dan kemudahan deployment. Dengan menggunakan Docker, aplikasi LMS beserta MinIO sebagai storage dapat dijalankan dengan konfigurasi yang konsisten di berbagai platform tanpa masalah ketergantungan perangkat keras. Hal ini memastikan bahwa aplikasi dapat di-deploy dengan lebih cepat dan mudah, serta memudahkan pengelolaan lingkungan sistem.

ISSN 2808-375X (Media Online)
Vol 5, No 1, November 2025
Hal 401−409
DOI 10.47065/comforch.v5i1.2358

4. KESIMPULAN

Penelitian ini berhasil mengimplementasikan MinIO sebagai storage dalam aplikasi LMS berbasis Docker, yang berhasil mengatasi masalah penyimpanan file pembelajaran yang efisien dan skalabel. Pengujian yang dilakukan menunjukkan bahwa MinIO mampu menangani file dengan berbagai ukuran (hingga 100 MB) dengan waktu akses yang cepat untuk upload dan download, serta kinerja yang stabil meskipun jumlah file meningkat. Penggunaan Docker sebagai platform kontainerisasi juga meningkatkan fleksibilitas dan portabilitas sistem, memudahkan deployment aplikasi LMS di berbagai lingkungan, serta menjaga konsistensi konfigurasi tanpa bergantung pada perangkat keras tertentu. Meskipun demikian, penelitian ini memiliki beberapa keterbatasan, seperti pengujian yang hanya dilakukan pada ukuran file tertentu dan jumlah pengguna terbatas. Penelitian selanjutnya perlu menguji sistem dengan ukuran file lebih besar dan lebih banyak pengguna aktif untuk menilai skalabilitas dan performa sistem secara menyeluruh. Selain itu, penelitian lebih lanjut dapat difokuskan pada pengembangan keamanan data, seperti enkripsi file dan kontrol akses berbasis peran (role-based access control), untuk meningkatkan perlindungan data dalam sistem LMS berbasis cloud. Penelitian ini menunjukkan bahwa MinIO dan Docker dapat menjadi solusi efisien untuk pengelolaan penyimpanan file dalam aplikasi pendidikan digital.

REFERENCES

[1] N. Saenko, A. Zekiy, dan E. Klochko, “Learning Management Systems in Academic and Corporate Distance Education,” *Int. J. Emerg. Technol. Learn.*, vol. 16, no. 11, hal. 121–139, 2021.
[2] N. A. Supiani, Dedy Achmad Kurniady, Tjutju Yuniarsih, “View of Evaluating Learning Management System (LMS) Effectiveness_ An LPOMR Model Approach.pdf,” *Pedagog. J. Ilm. Pendidik.*, vol. 16, hal. 71–77, 2024.
[3] A. Gurjar, D. Arse, dan A. Upadhayay, “Learning Management System,” *Int. J. Multidiscip. Res.*, vol. 7, no. 2, hal. 1–6, 2025.
[4] C. T. X. Lien, “Benefits and challenges of using LMS in blended learning: Views from EFL teachers and students at a Vietnamese public university,” *Int. J. TESOL Educ.*, vol. 3, no. 3, hal. 78–100, 2023.
[5] M. Ünver, “Design of a DFS to Manage Big Data in Distance Education Environments,” *J. Univers. Comput. Sci.*, vol. 28, no. 2, hal. 202–224, 2022.
[6] U. Rahardja, M. A. Ngadi, R. Budiarto, Q. Aini, dan M. Hardini, “Education Exchange Storage Protocol : Transformation Into Decentralized Learning Platform,” *Educ. Exch. Storage Protoc. (EESP*, vol. 6, no. December, hal. 1–14, 2021.
[7] I. A. Loebis, M. Mustofa, S. Arifin, M. Angglena, dan Y. Salim, “The Role of Cloud Computing Technology in Supporting Educational Accessibility and Scalability,” *JILTECH J. Int. Ling. Technol.*, vol. 3, no. 2, hal. 469–483, 2024.
[8] A. Quddus, K. Mihhail, M. Radu, P. Christoph, D. Roman, dan A. Soylu, “Cloud storage cost : a taxonomy and survey,” *World Wide Web*, vol. 123, hal. 1–54, 2024.
[9] S. Gore dan P. A. Dhamal, “Evaluating the Benefits of Cloud Storage Over Local Storage,” *Int. J. Res. Publ. Rev.*, vol. 6, no. 4, hal. 2730–2735, 2025.
[10] A. El Koshiry, E. Eliwa, dan T. A. El-hafeez, “Effectiveness of a Cloud Learning Management System in Developing the Digital Transformation Skills of Blind Graduate Students,” *Societies*, hal. 1–23, 2024.
[11] R. Chan, R. H. Dhaifullah, H. Saragih, N. Lediwara, R. I. Adha, dan R. Chan, “The development of a data lakehouse system for the integration and management of cyber threat intelligence data in XYZ unit,” *J. Intell. Decis. Support Syst.*, vol. 8, no. 1, hal. 44–53, 2025.
[12] S. Sumarsono dan I. I. Lawanda, “Kapabilitas preservasi arsip elektronik pada infrastruktur pembelajaran Kemenkeu Learning Center,” *Al-Kuttab J. Kaji. Perpustakaan, Inf. dan Kearsipan*, vol. 6, no. 2, hal. 14–26, 2024.
[13] S. A. Suryanto, S. Syaifuddin, dan D. Rizqiwati, “Development of Access Mechanism of E-learning Linux Container-Based,” *ITSMART J. Teknol. dan Inf.*, vol. 7, no. 2, hal. 88–93, 2019.
[14] A. Baehaqi, M. S. Basit, R. E. Indrajit, dan R. D. Kurniawan, “Pengembangan Front End Learning Management System Menggunakan Framework Nextjs,” *J. Tek. Inform.*, vol. 4, no. 4, hal. 899–911, 2023.
[15] S. Suryayusra, N. Destarina, E. S. Negara, E. Supratman, dan M. Ulfa, “Implementation Docker and Kubernetes Scaling Using Horizontal Scaler Method for Wordpress Services,” *Sink. J. Penelit. Tek. Inform.*, vol. 8, no. 4, hal. 2192–2196, 2024.
[16] A. R. Ekaputra dan A. S. Affandi, “Pemanfaatan layanan cloud computing dan docker container untuk meningkatkan kinerja aplikasi web,” *J. Inf. Syst. Appl. Dev.*, vol. 1, no. 2, hal. 138–147, 2023.
[17] A. O. P. Dewi, “Big Data di Perpustakaan dengan Memanfaatkan Data Mining,” *Anuva J. Kaji. Budaya, Perpustakaan, dan Inf.*, vol. 4, no. 2, hal. 223–230, 2020.
[18] Kusnadi Diza Aulia, Iqbal Muhammad, dan Prabowo Bramantyo, “Perancangan dan Implementasi Protokol OAuth 2.0 pada Pengembangan Produk Coofis Verse di PT ARM Solusi,” *e-Proceeding Appl. Sci.*, vol. 10, no. 6, hal. 1472, 2024.