

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi dalam dunia pendidikan telah mendorong berbagai inovasi berbasis digital, salah satunya adalah *Learning Management System* (LMS). LMS dirancang untuk meningkatkan fleksibilitas dalam pembelajaran serta memberikan solusi untuk penyampaian materi secara asinkron maupun sinkron. Salah satu jenis LMS yang banyak digunakan secara global adalah *Modular Object-Oriented Dynamic Learning Environment* (Moodle), platform *open source* yang memungkinkan institusi pendidikan untuk menyesuaikan sistem dengan kebutuhan pembelajaran (Listiani, 2021). Hingga Maret 2025, Moodle telah digunakan oleh 147.785 situs dengan 433.866.383 pengguna di 236 negara (Moodle Statistik, 2025).

Seiring dengan meningkatnya jumlah pengguna dan kompleksitas layanan pendidikan, Moodle menghadapi tantangan dalam hal skalabilitas (Rani dkk., 2023). Hal ini disebabkan oleh peningkatan kompleksitas karena kode yang semakin rumit dan ketergantungan antar modul dalam sistem (Hassan, 2024). Kendala skalabilitas dapat dioptimalkan melalui dua pendekatan, yaitu dengan memperbesar skala arsitektur monolitik dengan meningkatkan kapasitas *server* atau melakukan transisi ke arsitektur *microservices* (Chippagiri & Kassetty, 2025). Secara *default* Moodle memiliki arsitektur monolitik tetapi pendekatan *microservices* dapat diimplementasikan melalui fitur *web service* dari Moodle (Dahri dkk., 2022).

Meskipun kedua pendekatan tersebut dapat mengatasi tantangan skalabilitas pada kedua arsitektur, tantangan lain yang masih dapat terjadi adalah kegagalan *deployment*. Kompleksitas Moodle dapat menyebabkan proses *deployment* yang lama sehingga dapat meningkatkan risiko *downtime* dan kegagalan *deployment*. Oleh karena itu, diperlukan solusi yang efektif untuk mengatasi masalah tersebut.

Salah satu strategi yang dapat diterapkan adalah *Continuous Integration and Continuous Deployment (CI/CD) pipeline*, yang memungkinkan mengotomatiskan proses *deployment*, baik pada arsitektur monolitik maupun *microservices* (Kolawole & Fakokunde, 2025).

Dengan penerapan *CI/CD pipeline*, proses *deployment* dapat disesuaikan dengan struktur masing-masing arsitektur. Pada arsitektur monolitik, *CI/CD pipeline* hanya dibuat satu untuk mengelola seluruh proses *deployment*, dikarenakan seluruh sistem dibangun dalam satu kode sumber. *Pipeline* ini berperan dalam pengujian menyeluruh (*end-to-end testing*) sebelum *deployment* dan *rollback* otomatis untuk mencegah *downtime*. Sementara itu, dalam arsitektur *microservices*, setiap layanan memiliki *CI/CD pipeline* masing-masing sehingga proses *deployment* lebih fleksibel dan independen. *Pipeline* ini harus mendukung *deployment* paralel serta memungkinkan *rollback* selektif tanpa mengganggu layanan lainnya (Kolawole & Fakokunde, 2025).

Berdasarkan penjelasan tersebut, penelitian ini berfokus pada strategi pengembangan *CI/CD pipeline* yang mampu mengatasi kegagalan *deployment* pada arsitektur monolitik dan *microservices*, khususnya yang disebabkan oleh kesalahan dependensi yang sering terjadi saat proses *deployment*. Penelitian ini juga bertujuan untuk meningkatkan keandalan sistem dengan penerapan *CI/CD pipeline* pada aplikasi Moodle dengan Jenkins. Untuk mengevaluasi keandalan sistem tersebut, penelitian ini akan menguji beberapa parameter utama, yaitu *downtime*, waktu *rollback*, serta keberhasilan *deployment*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, penelitian ini berfokus pada pertanyaan-pertanyaan berikut:

1. Bagaimana Implementasi *CI/CD pipeline* pada arsitektur monolitik dan *microservices*?

2. Bagaimana perbandingan penerapan *CI/CD pipeline* pada arsitektur monolitik dan *microservices* berdasarkan parameter *downtime*, waktu *rollback* dan keberhasilan *deployment*?
3. Bagaimana korelasi antara kesalahan dependensi dan kegagalan *deployment* pada arsitektur monolitik dan *microservices*?

1.3 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Merancang *CI/CD pipeline* pada arsitektur monolitik dan *microservices* untuk meningkatkan keandalan sistem.
2. Menganalisis perbandingan *CI/CD pipeline* pada arsitektur monolitik dan *microservices* berdasarkan parameter *downtime*, waktu *rollback*, dan tingkat keberhasilan *deployment*, guna memahami keandalan sistem dengan penerapan *CI/CD pipeline*.
3. Mengidentifikasi korelasi antara kesalahan dependensi dan kegagalan *deployment* pada arsitektur monolitik dan *microservices*.

1.4 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1.4.1 Secara Teoritis

Penelitian ini diharapkan dapat memberikan manfaat teoritis berupa:

1. Menambah wawasan dalam *deployment* perangkat lunak menggunakan *CI/CD pipeline*, terutama dalam menangani kegagalan *deployment* akibat kesalahan dependensi.
2. Memberikan pemahaman lebih mendalam mengenai tantangan dan perbedaan penerapan *CI/CD pipeline* pada arsitektur monolitik dan *microservices*.
3. Menjadi referensi dalam studi tentang optimasi *deployment* pada sistem LMS berbasis *cloud*, khususnya dalam konteks Moodle.

1.4.2 Secara Praktis

Bagi Pengembang Perangkat Lunak:

Tita Rismawati, 2025

ANALISIS KEANDALAN *CI/CD PIPELINE* PADA ARSITEKTUR MONOLITIK DAN *MICROSERVICES* DALAM MENANGANI KEGAGALAN *DEPLOYMENT*

Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

1. Memberikan wawasan mengenai keunggulan, tantangan, dan strategi penerapan *CI/CD pipeline* pada arsitektur monolitik dan *microservices*.
2. Menyediakan panduan mengenai praktik terbaik dalam menangani kesalahan dependensi dalam proses *deployment*.
3. Memberikan strategi peningkatan skabilitas pada arsitektur monolitik dan *microservices* pada aplikasi Moodle.
4. Memberikan pemahaman mengenai peralihan arsitektur monolitik ke *microservices* pada aplikasi Moodle.

Bagi Institusi Pendidikan dan Pengguna Moodle:

1. Membantu administrator sistem dalam memilih arsitektur sesuai untuk implementasi Moodle guna meningkatkan skalabilitas dan keandalan sistem.
2. Mengurangi risiko *downtime* serta kegagalan *deployment* yang disebabkan oleh kesalahan dependensi, sehingga meningkatkan efisiensi operasional.

Bagi Peneliti Selanjutnya:

1. Menjadi dasar penelitian lanjutan tentang optimasi *deployment* dan pengelolaan *server* berbasis *CI/CD pipeline* dalam LMS dan aplikasi berbasis *cloud*.
2. Memberikan referensi dalam studi kasus penerapan *CI/CD* dalam LMS menggunakan Moodle untuk meningkatkan keandalan sistem.
3. Mendorong penelitian lanjutan mengenai strategi peralihan arsitektur monolitik ke *microservices* serta peran *CI/CD pipeline* dalam mendukung transisi tersebut.

1.5 Ruang Lingkup Penelitian

Ruang lingkup penelitian ini mencakup aspek-aspek berikut untuk menjaga fokus dan kejelasan studi.

1. Penelitian ini hanya berfokus pada analisis keandalan *CI/CD pipeline* pada arsitektur monolitik dan *microservices*, tanpa membahas aspek performa lainnya.
2. Migrasi Moodle ke arsitektur *microservices* masih berada pada tahap awal pengembangan, sehingga dalam penelitian ini digunakan pendekatan *web service* sebagai representasi proses integrasi layanan.
3. Penelitian ini berfokus pada kegagalan *deployment* yang disebabkan oleh kesalahan dependensi.
4. Penelitian dilakukan pada aplikasi Moodle, sehingga hasil penelitian lebih relevan untuk sistem LMS atau aplikasi dengan modularitas tinggi.
5. Parameter yang diukur yaitu *downtime*, waktu *rollback*, dan keberhasilan *deployment*.
6. Pengujian dilakukan dalam lingkungan simulasi menggunakan Jenkins sebagai alat *CI/CD* untuk meniru skenario *deployment* di lingkungan *production*.
7. Seluruh proses mulai dari konfigurasi, implementasi *pipeline*, hingga pelaksanaan pengujian dilakukan secara individual tanpa melibatkan tim pengembang, sehingga kolaborasi *multi-developer* tidak menjadi variabel dalam penelitian ini.
8. Pengujian dilakukan pada lingkungan virtual menggunakan *Windows Subsystem for Linux* (WSL) dan Ubuntu 24.04.