IMPLEMENTASI CI/CD MENGGUNAKAN JENKINS PADA APLIKASI DI LINGKUNGAN PENGEMBANGAN FEDORA WORKSTATION 40

Rizal Muslim Sinaga, Sovantri Putra Paskah Halawa, Dedy Kiswanto
Ilmu Komputer, Universitas Negeri Medan
Jl. William Iskandar Ps. V, Kabupaten Deli Serdang, Sumatera Utara, Indonesia
sinaga.rizal456@gmail.com

ABSTRAK

Dalam pengembangan perangkat lunak, otomatisasi proses build, pengujian, dan deployment penting untuk meningkatkan efisiensi dan kualitas aplikasi. Continuous Integration (CI) dan Continuous Delivery (CD) adalah pendekatan populer untuk tujuan ini, dengan Jenkins sebagai alat utama. Penelitian ini mengimplementasikan CI/CD dengan Jenkins pada Fedora Workstation 40 untuk aplikasi web berbasis PHP. Metode eksperimen meliputi instalasi Fedora di VirtualBox, konfigurasi Jenkins dengan plugin seperti Pipeline, MySQL, dan GitHub, serta pembuatan aplikasi to-do list. Pengelolaan kode dilakukan dengan Git, di mana perubahan dilakukan secara lokal di Fedora menggunakan git add, git commit, dan git push untuk mengirim perubahan ke repositori GitHub. Setelah itu, Jenkins secara otomatis menjalankan pipeline untuk build, pengujian, dan deployment. Hasil penelitian menunjukkan waktu build rata-rata 3 detik, pengujian 5 detik, dan deployment 3 detik, dengan keberhasilan 100% pada build dan deploy, serta 95% pada pengujian. Pipeline yang diterapkan juga mendeteksi bug lebih awal, meningkatkan kualitas aplikasi. Penelitian ini memberikan kontribusi dalam penerapan CI/CD dengan Jenkins pada Fedora Workstation 40, mendukung pengembangan aplikasi PHP yang efisien dan berkelanjutan.

Kata kunci : *Continuous Integration, Continuous Deployment, Jenkins, Fedora Workstation, CI/CD, PHP*

1. PENDAHULUAN

Efisiensi dan kecepatan dalam pengembangan perangkat lunak menjadi elemen utama untuk menghadapi tantangan pasar yang terus berkembang. Industri perangkat lunak saat ini menuntut waktu pengembangan yang singkat, pengiriman yang cepat, dan kualitas yang tetap terjaga. Pendekatan tradisional yang mengandalkan pengujian manual dan deployment manual tidak lagi relevan dalam memenuhi tuntutan tersebut karena prosesnya yang memakan waktu dan rentan terhadap kesalahan manusia. Oleh karena itu, penerapan Continuous Integration (CI) dan Continuous Delivery (CD) muncul sebagai solusi utama.

CI adalah praktik di mana pengembang secara rutin menggabungkan perubahan kode mereka ke dalam repositori bersama, yang kemudian diuji secara otomatis untuk mendeteksi kesalahan lebih awal. Dengan ini, tim pengembang dapat mengidentifikasi dan memperbaiki masalah lebih cepat, mengurangi risiko bug yang mencapai tahap produksi. Sedangkan CD memungkinkan perangkat lunak yang telah diuji siap untuk dirilis kapan saja, menjadikan proses pengiriman aplikasi lebih cepat, andal, dan terprediksi [1].

Untuk mendukung implementasi CI/CD, berbagai alat dan teknologi telah dikembangkan, salah satunya adalah Jenkins. Sebagai alat open-source, Jenkins memungkinkan pengembang untuk mengotomatisasi berbagai tahap dalam siklus pengembangan perangkat lunak, mulai dari build, pengujian, hingga deployment aplikasi. Jenkins mengelola pipeline otomatisasi dengan memantau perubahan kode, menjalankan serangkaian pengujian otomatis, dan memastikan aplikasi selalu dalam kondisi yang siap dirilis [6]. Alat ini telah menjadi standar de facto dalam industri perangkat lunak karena fleksibilitasnya yang tinggi dan ekosistem plug-in yang kaya, yang memungkinkan integrasi dengan berbagai teknologi modern.

Namun, penerapan CI/CD tidak dapat berdiri sendiri tanpa lingkungan pengujian yang memadai. Dalam hal ini, virtualisasi memainkan peran penting, khususnya dalam menciptakan lingkungan pengujian yang konsisten, terisolasi, dan fleksibel. VirtualBox, sebagai salah satu perangkat lunak virtualisasi yang banyak digunakan, memungkinkan pengembang untuk menciptakan mesin virtual (VM) yang mampu meniru server produksi. Dengan VirtualBox, pengembang dapat menguji aplikasi di berbagai sistem operasi tanpa mempengaruhi sistem utama, memastikan aplikasi berjalan optimal dalam berbagai kondisi [7].

Salah satu distribusi Linux yang mendukung praktik ini adalah Fedora Workstation. Dikenal dengan stabilitasnya dan dukungannya terhadap teknologi terbaru, Fedora Workstation 40 menjadi pilihan ideal untuk menciptakan lingkungan pengembangan modern. Fedora Workstation mendukung kebutuhan pengembang dengan menyediakan perangkat lunak terkini dan integrasi yang mudah dengan alat-alat pengembangan seperti Jenkins.

Latar belakang implementasi ini juga didasari oleh kebutuhan untuk mempraktikkan CI/CD pada aplikasi web berbasis PHP sederhana. Aplikasi seperti to-do list yang terhubung dengan database MySQL dipilih sebagai studi kasus karena sifatnya yang sederhana namun cukup representatif untuk

mencerminkan tantangan dalam siklus pengembangan perangkat lunak.

Penelitian ini bertujuan untuk mengimplementasikan CI/CD menggunakan Jenkins pada aplikasi berbasis PHP di lingkungan pengembangan Fedora Workstation 40 yang berjalan di VirtualBox. Dengan kombinasi teknologi ini, diharapkan pengembang dapat memanfaatkan keunggulan CI/CD, virtualisasi, dan distribusi Linux secara bersamaan untuk meningkatkan efisiensi, konsistensi, dan kualitas aplikasi perangkat lunak.

2. TINJAUAN PUSTAKA

2.1. Continuous Integration (CI) dan Continuous Delivery(CD)

Continuous Integration (CI) adalah teknik pengembangan perangkat lunak di mana kode yang dikembangkan secara teratur digabungkan ke dalam repositori pusat. Setiap perubahan kode yang dilakukan akan diuji secara otomatis untuk mendeteksi kesalahan lebih cepat dan mengurangi konflik antara pengembang. Dengan menerapkan CI, tim pengembang dapat memperbaiki bug lebih awal, sehingga meningkatkan kualitas perangkat lunak secara keseluruhan. Implementasi Continuous Delivery (CD) adalah langkah lebih lanjut setelah CI, yang memastikan aplikasi yang telah diuji siap untuk dirilis ke lingkungan produksi secara otomatis tanpa memerlukan intervensi manual [1].

2.2. Jenkins Sebagai Alat CI/CD

Jenkins adalah alat open-source yang banyak digunakan untuk otomatisasi pengujian dan deployment dalam praktik CI/CD. Jenkins dapat digunakan untuk mengotomatisasi seluruh siklus pengembangan perangkat lunak, mulai dari pengambilan kode sumber, pengujian otomatis, hingga deployment ke server produksi. Dalam implementasinya, Jenkins dapat diintegrasikan dengan berbagai alat dan sistem lain seperti Docker, Git, dan PHP untuk memastikan aplikasi selalu dalam kondisi siap deploy [6]. Jenkins bekerja dengan menggunakan pipeline yang memungkinkan pengembang mendefinisikan urutan tahapan dalam proses CI/CD, sehingga memungkinkan aplikasi untuk diuji dan dirilis dengan cepat dan efisien [6].

2.3. VirtualBox Sebagai Alat Virtualisasi

Penerapan CI/CD tidak dapat berdiri sendiri tanpa lingkungan pengujian yang memadai. VirtualBox, sebagai salah satu contoh hosted hypervisor versi gratis, memainkan peran penting dalam menyediakan lingkungan pengujian yang konsisten, fleksibel, dan terisolasi. Hosted hypervisor seperti VirtualBox bekerja di atas sistem operasi host, seperti Windows atau Linux, dan memungkinkan pengguna untuk membuat serta menjalankan mesin virtual. Dengan VirtualBox, pengguna dapat mengalokasikan sumber daya seperti CPU, RAM, dan penyimpanan untuk menjalankan lebih dari satu sistem operasi secara bersamaan pada satu perangkat keras. Solusi virtualisasi ini sangat cocok untuk keperluan eksperimen, pengembangan, atau pengujian perangkat lunak tanpa memengaruhi sistem utama. Dalam implementasi ini, Fedora Workstation 40 digunakan di dalam VirtualBox untuk menciptakan lingkungan pengujian yang stabil dan dapat disesuaikan untuk berbagai sistem operasi serta konfigurasi [7].

2.4. Fedora Workstation sebagai Sistem Operasi Pengembangan

Fedora merupakan distribusi Linux yang banyak digunakan di kalangan pengembang perangkat lunak karena sifatnya yang open-source dan memiliki dukungan keamanan yang kuat. Fedora dilengkapi dengan berbagai alat dan fitur yang memungkinkan pengguna untuk mengelola dan mengamankan sistem mereka dengan lebih efektif. Selain itu, Fedora juga menyediakan pembaruan keamanan secara rutin dan memiliki komunitas yang aktif dalam mengembangkan fitur-fitur terbaru, menjadikannya pilihan yang baik bagi pengembang yang membutuhkan lingkungan yang stabil dan aman untuk bekerja. Dengan Fedora Linux, pengembang dapat memanfaatkan berbagai tool dan sistem manajemen yang membantu dalam menciptakan aplikasi yang tidak hanya efisien, tetapi juga aman, terutama dalam hal melindungi data dan mencegah ancaman dari luar [3].

2.5. Pengembangan Aplikasi Web PHP dan Database MySQL

Pengembangan aplikasi web menggunakan PHP dan MySQL merupakan solusi yang efektif untuk menciptakan aplikasi berbasis internet yang dinamis dan dapat diakses secara online. PHP, sebagai bahasa pemrograman yang populer, digunakan untuk membangun logika aplikasi di sisi server, sementara MySQL digunakan sebagai sistem manajemen database untuk menyimpan dan mengelola data aplikasi. PHP memungkinkan pembuatan halaman web interaktif yang dapat berkomunikasi dengan database MySQL untuk menyimpan data pengguna, hasil ujian, atau data lainnya secara efisien. Penggunaan kombinasi PHP dan MySQL ini memungkinkan aplikasi web untuk berjalan dengan optimal, memberikan kemudahan dalam pengelolaan data, serta memungkinkan pengguna untuk mengakses aplikasi secara fleksibel melalui internet kapan saja dan di mana saja[8][2][10].

2.6. Studi Kasus Implementasi CI/CD pada Aplikasi Web

Beberapa studi menunjukkan bahwa penerapan CI/CD pada aplikasi web berbasis PHP dapat meningkatkan kecepatan dan kualitas pengembangan. Misalnya, dalam penelitian yang dilakukan oleh Enda et al. (2022), penerapan Continuous Integration (CI) pada aplikasi web Karang Taruna Kabupaten Bengkalis berbasis PHP menunjukkan dampak positif

terhadap kecepatan pengembangan dan deteksi bug yang lebih awal. Setiap perubahan kode yang diajukan langsung diuji secara otomatis, yang mempercepat proses rilis dan meningkatkan kualitas aplikasi. Implementasi CI/CD ini juga membantu dalam memperbaiki kolaborasi antara anggota tim dan mempermudah pemeliharaan aplikasi [4].

2.7. Keuntungan dan Tantangan dalam Implementasi CI/CD

Implementasi CI/CD, meskipun menawarkan berbagai keuntungan seperti otomatisasi dan peningkatan kualitas kode, juga menghadirkan beberapa tantangan. Salah satu tantangan utama adalah kompleksitas dalam mengelola berbagai alat CI/CD seperti Jenkins, GitLab CI, dan Travis CI, terutama saat digunakan dalam aplikasi berbasis PHP yang memiliki ketergantungan kompleks [5]. Tantangan lainnya adalah pengelolaan dependensi yang tidak terkelola dengan baik, yang dapat menyebabkan kesalahan pada proses build atau deployment. Untuk mengurangi masalah ini, penggunaan manajer paket seperti Composer untuk PHP serta pengaturan lingkungan yang konsisten sangat diperlukan [4][5].

3. METODE PENELITIAN

Metode penelitian yang digunakan dalam studi ini adalah eksperimen untuk menguji dan mengevaluasi implementasi CI/CD dengan Jenkins pada aplikasi web dalam lingkungan pengembangan Fedora Workstation. Penelitian ini dilakukan melalui serangkaian tahapan yang meliputi instalasi, konfigurasi, implementasi pipeline CI/CD, serta pengujian langsung pada lingkungan yang telah disiapkan.

3.1. Persiapan Lingkungan

Untuk mendukung pengembangan dan pengujian aplikasi menggunakan CI/CD, perangkat keras yang digunakan perlu memenuhi spesifikasi tertentu.

Tabel 1. Spesifikasi Laptop sebagai Perangkat Keras Lingkungan Uji

Komponen	Spesifikasi
Model Laptop	Acer Aspire A515-45
Prosesor	AMD Ryzen 5 5500U
RAM	8 GB
Sistem Operasi	Windows 11 Home

Berikut merupakan penjelasan dari Tabel 1:

a. Perangkat keras yang digunakan adalah Acer Aspire A515-45, sebuah perangkat yang cocok untuk pengembangan perangkat lunak.
b. Prosesor AMD Ryzen 5 5500U memberikan kecepatan pemrosesan yang andal untuk mendukung virtualisasi.
c. Kapasitas RAM sebesar 8 GB cukup untuk menjalankan aplikasi berat, termasuk VirtualBox.
d. Sistem operasi Windows 11 Home digunakan sebagai sistem utama, memberikan kompatibilitas dengan berbagai alat pengembangan.

Virtualisasi digunakan untuk menyediakan lingkungan pengembangan yang terisolasi dan konsisten. Fedora Workstation 40 diinstal pada VirtualBox dengan alokasi sumber daya tertentu.

Tabel 2. Spesifikasi yang Diberikan kepada Fedora pada VirtualBox

Komponen	Spesifikasi
Sistem Operasi	Fedora Workstation 40
Alokasi Core Prosesor	2 core
Alokasi RAM	4090 MB
Virtualisasi	VirtualBox

Berikut penjelasan dari Tabel 2:

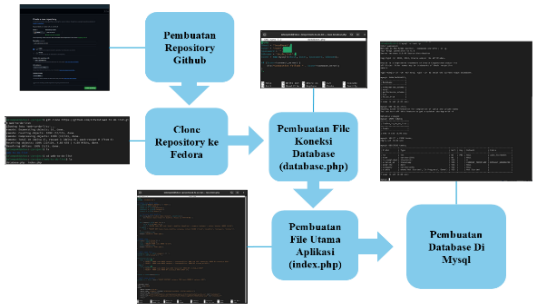
a. Sistem operasi yang digunakan dalam VirtualBox adalah Fedora Workstation 40, yang dirancang untuk mendukung pengembangan perangkat lunak.
b. Virtualisasi dilakukan menggunakan VirtualBox, memberikan fleksibilitas dalam menciptakan lingkungan pengujian yang terisolasi.
c. Alokasi 2 core prosesor dan 4090 MB RAM cukup untuk menjalankan Fedora Workstation dengan lancar, memastikan kinerja optimal selama proses pengujian.

3.2. Instalasi Perangkat Lunak

Setelah lingkungan dasar disiapkan, berbagai perangkat lunak dan plugin diinstal untuk mendukung pengembangan dan otomatisasi CI/CD. MySQL digunakan untuk manajemen basis data, PHP sebagai bahasa pemrograman server-side, dan Git untuk sistem kontrol versi. Composer mengatur dependensi eksternal dalam aplikasi PHP. Jenkins beserta plugin seperti Pipeline Plugin, MySQL Plugin, GitHub & Git Plugins, dan SSH Plugin membantu mengotomatiskan proses build, pengujian, dan deployment. Dengan perangkat lunak ini, pengembangan, pengujian, dan otomatisasi CI/CD berjalan efisien di lingkungan Fedora Workstation 40.

3.3. Pengaturan Repository GitHub dan Pembuatan Aplikasi Web di Fedora

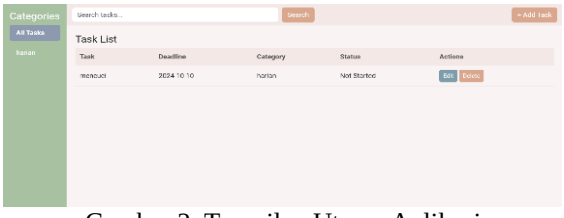
Setelah Jenkins dikonfigurasi, langkah berikutnya adalah membuat repository GitHub untuk aplikasi web. Repository ini kemudian di-*clone* ke Fedora menggunakan Git, dan semua pengembangan kode dilakukan langsung di Fedora. Proses ini mencakup langkah-langkah berikut:



Gambar 1. Pengaturan Repository GitHub dan Pembuatan Aplikasi Web di Fedora

a. Pembuatan repository dilakukan melalui antarmuka GitHub, di mana URL repository yang dihasilkan akan digunakan untuk meng-clone repository ke mesin pengembangan di Fedora. Repository harus terkonfigurasi dengan benar dan siap digunakan untuk menyimpan kode aplikasi.
b. Setelah repository GitHub dibuat, lakukan cloning repository ke mesin pengembangan Fedora. Proses cloning dilakukan dengan membuka terminal di Fedora dan menggunakan perintah Git untuk menyalin repository ke direktori lokal.
c. Setelah repository berhasil di-clone, pengembangan kode aplikasi dimulai dengan pembuatan file utama index.php. File ini berfungsi sebagai antarmuka pengguna (UI) aplikasi to-do list. Struktur HTML dasar dibuat untuk memungkinkan pengguna menambahkan, melihat, dan menghapus tugas.
d. Langkah berikutnya adalah pembuatan file database.php, yang bertugas mengatur koneksi aplikasi dengan database MySQL. File ini memungkinkan aplikasi untuk berinteraksi dengan database dan melakukan operasi CRUD pada data tugas.
e. Selanjutnya, dibuat sebuah tabel tasks dalam database MySQL untuk menyimpan data tugas aplikasi to-do list. Tabel ini mencatat informasi tugas, termasuk deskripsi, status penyelesaian, tanggal pembuatan, deadline, kategori, dan status saat ini.

Setelah semua kode dan struktur database selesai disiapkan, langkah berikutnya adalah melakukan pengujian aplikasi secara lokal untuk memastikan bahwa aplikasi berfungsi sesuai dengan yang diharapkan. Pengujian ini dilakukan dengan menjalankan aplikasi menggunakan PHP built-in server di Fedora.



Gambar 2. Tampilan Utama Aplikasi

3.4. Pengelolaan Kode Secara Lokal di Fedora

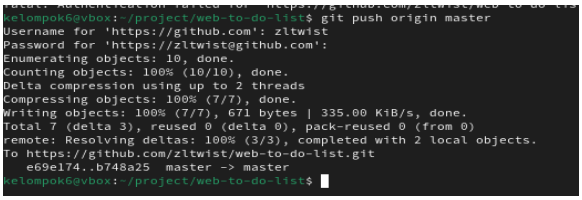
Pengelolaan kode dengan Git memungkinkan pelacakan perubahan, pengelolaan versi, dan kolaborasi tim. Di Fedora, kode dikelola secara lokal sebelum di-*push* ke repository seperti GitHub. Setelah file dimodifikasi, langkah pertama adalah memasukkannya ke *staging area*, tempat sementara sebelum disimpan ke repository lokal.

Gambar 3. Penambahan file index.php dan database.php yang sudah dibuat ke dalam staging area

Commit dilakukan untuk menyimpan perubahan di repository lokal dengan pesan deskriptif sebagai dokumentasi.

Gambar 4. Menyimpan perubahan ke dalam lokal Git

Selanjutnya, perubahan di-*push* ke repository jarak jauh seperti GitHub. Ini memungkinkan tim mengakses perubahan sekaligus memicu pipeline CI/CD di Jenkins untuk membangun, menguji, dan mendepoy aplikasi.

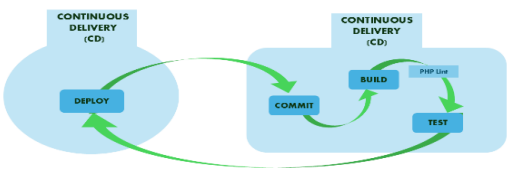


Gambar 5. Push perubahan ke dalam Repository GitHub

Untuk mencegah konflik, pengembang bekerja di cabang terpisah saat menambahkan fitur atau memperbaiki bug sebelum digabungkan ke cabang utama.

3.5. Konfigurasi Jenkins dan Pipeline

Setelah aplikasi siap di Fedora, Jenkins digunakan untuk mengelola siklus pengembangan melalui pembuatan job pipeline yang menghubungkan aplikasi dengan repository GitHub. Jenkins kemudian dikonfigurasi untuk menjalankan pipeline yang mencakup langkah-langkah seperti build, test, dan deploy secara otomatis. Pipeline CI/CD yang diimplementasikan ditunjukkan pada Gambar, yang menggambarkan alur kerja otomatis dari proses development hingga deployment aplikasi.



Gambar 6. Alur Pipeline CI/CD pada Jenkins

a. Commit: Developer melakukan push kode ke GitHub setelah melakukan git commit secara lokal di Fedora.
b. Build: merupakan proses pembangunan aplikasi setelah kode terbaru diambil. Dalam tahap ini, Jenkins melakukan pemeriksaan sintaks PHP (PHP Lint) untuk memastikan tidak ada kesalahan syntax dalam kode aplikasi.
c. Test: merupakan tahap pengujian dimana aplikasi yang telah di-build akan melalui proses testing. Dalam implementasi ini, testing dilakukan dengan PHP Lint untuk memverifikasi kualitas kode dan memastikan tidak ada error dalam sintaks PHP.
d. Deploy: merupakan tahap akhir dalam pipeline dimana aplikasi yang telah lulus testing akan dideploy ke server. Pada tahap ini, Jenkins menjalankan PHP built-in server di port 8020 untuk menghosting aplikasi web To Do List.
e. Pipeline ini berjalan dalam dua loop utama yang saling terhubung: Loop Continuous Integration (CI) yang mencakup proses commit, build, dan test.; Loop Continuous Delivery (CD) yang fokus pada proses deployment aplikasi.

3.6. Proses Deployment dengan PHP Built-in Server

Setelah aplikasi melewati tahapan build dan test, PHP built-in server digunakan untuk melakukan deployment aplikasi ke server lokal. Jenkins secara otomatis memulai server PHP setelah aplikasi lulus pengujian dan siap untuk di-deploy, memastikan aplikasi dapat diakses di lingkungan pengujian lokal tanpa intervensi manual.

3.7. Metode Pengumpulan Data

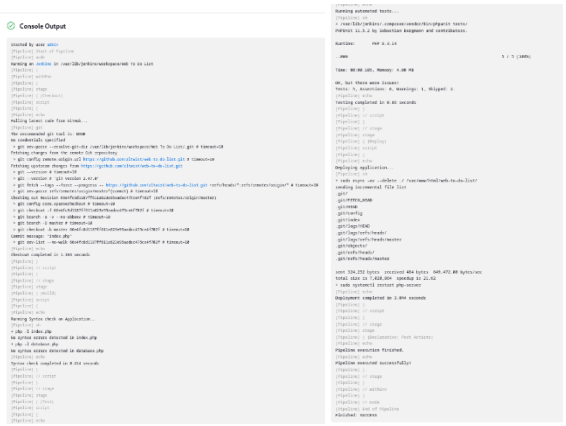
Pengujian kinerja CI/CD dilakukan dengan memantau waktu yang dibutuhkan pada setiap tahap pipeline Jenkins, termasuk waktu build, waktu test, dan waktu deploy. Data yang dikumpulkan mencakup durasi tiap tahap serta tingkat keberhasilan proses otomatisasi, untuk mengevaluasi efisiensi pipeline CI/CD dalam mendukung pengembangan aplikasi web berbasis PHP. Selanjutnya, evaluasi keberhasilan implementasi CI/CD pada Fedora Workstation dilakukan dengan menguji kelancaran proses, mulai dari pengambilan kode di GitHub hingga deployment aplikasi menggunakan PHP built-in server. Proses ini bertujuan untuk memastikan bahwa Fedora mampu mendukung otomatisasi yang efektif. Selain itu, monitoring dilakukan selama pengujian untuk mencatat bug atau error yang muncul selama proses build dan pengujian. Hasil monitoring ini digunakan untuk menganalisis efektivitas pipeline Jenkins dalam mendeteksi kesalahan secara cepat dibandingkan dengan pengujian manual.

4. HASIL DAN PEMBAHASAN

4.1. Hasil Pengujian Kinerja CI/CD

Dalam penelitian ini, pipeline Jenkins diimplementasikan untuk mengotomatisasi siklus pengembangan dan deployment aplikasi web to-do list berbasis PHP. Setiap tahapan dalam pipeline build, test, dan deploy dijalankan secara otomatis setelah aplikasi diperbarui di GitHub repository.

Pada tahap pengujian, setiap proses menghasilkan log yang mencatat status dan hasil setiap langkah pipeline. Berikut adalah screenshot yang menunjukkan log pipeline Jenkins, mencakup proses build, test, dan deploy, hingga aplikasi berhasil dijalankan di server lokal:

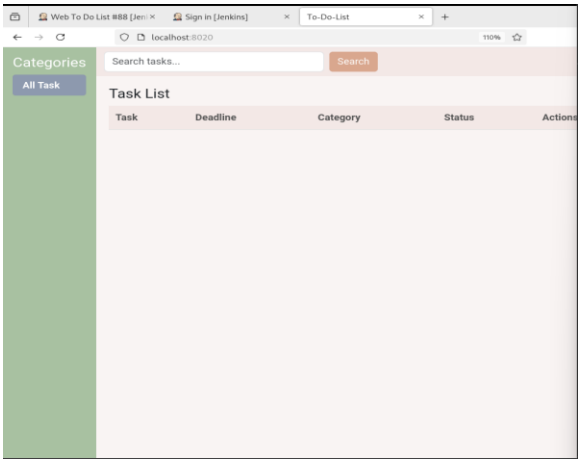


Gambar 7. Hasil Build, Test, dan Deploy di Jenkins

Berikut Penjelasan terkait Gambar 7:

a. Build: Tahap build berjalan sukses, di mana Jenkins berhasil melakukan pemeriksaan sintaks PHP (php -l index.php). Tidak ada kesalahan yang terdeteksi, menunjukkan bahwa kode aplikasi telah disiapkan dengan baik. Proses ini berjalan cepat, mengurangi waktu yang dibutuhkan dibandingkan proses build manual.
b. Test: Pada tahap ini, pengujian otomatis dilakukan, termasuk validasi logika aplikasi dan interaksinya dengan database MySQL. Meskipun tidak ada error dalam pengujian sintaks, pengujian lebih lanjut dapat ditingkatkan untuk memeriksa validasi data dan logika aplikasi.
c. Deploy: Jenkins menjalankan PHP built-in server untuk mendukung proses deployment. Server berjalan pada port 8020, memungkinkan aplikasi dapat langsung diakses di lingkungan pengujian lokal melalui URL http://0.0.0.0:8020. Tahap ini menunjukkan tingkat otomatisasi tinggi, di mana aplikasi siap digunakan tanpa memerlukan intervensi manual.

Setelah semua tahapan pipeline berhasil, aplikasi dideploy ke server lokal menggunakan PHP built-in server pada port 8020. Gambar berikut memperlihatkan aplikasi yang berhasil dijalankan dan diakses melalui browser di alamat http://0.0.0.0:8020. Aplikasi menampilkan fitur to-do list sesuai dengan spesifikasi yang diimplementasikan.



Gambar 8. Tampilan aplikasi yang berhasil dijalankan di browser

Jika terdapat kesalahan sintaks dalam kode program, pipeline Jenkins akan berhenti pada tahap build. Gambar berikut menunjukkan log kesalahan yang terjadi akibat error sintaks, seperti tanda kurung yang hilang atau variabel yang tidak dideklarasikan. Log ini memberikan informasi detail, seperti lokasi baris kode yang bermasalah dan jenis kesalahan, sehingga memudahkan pengembang dalam memperbaikinya.

```
Running syntax check on PHP files...
[Pipeline] sh
+ find . -name '*.php' -exec php -l '{}' ';'
No syntax errors detected in ./database.php
PHP Parse error:  syntax error, unexpected token "if" in ./index.php on line 5
Errors parsing ./index.php
[Pipeline] echo
Build completed in 0.765 seconds
```

Gambar 9. Log kesalahan sintaks pada tahap build

```
Running automated tests...
[Pipeline] sh
+ /var/lib/jenkins/.composer/vendor/bin/phpunit tests/
PHPUnit 11.5.2 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.3.14

.EF                                                                 3 / 3 (100%)

Time: 00:00.059, Memory: 4.00 MB

There was 1 error:

1) DatabaseTest::testQueryExecution
mysqli_sql_exception: Table 'to_do_list.users' doesn't exist

/var/lib/jenkins/workspace/Web To Do List/tests/DatabaseTest.php:12

--

There was 1 failure:

1) IndexTest::testRenderPage
Failed asserting that '\n
<!DOCTYPE html>\n
<html lang="en">\n
<head>\n
    <meta charset="UTF-8">\n
    <meta name="viewport" content="width=device-width, initial-scale=1.0">\n
    <title>To-Do-List</title>\n
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">\n
    <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">\n
    <style>\n
        :root {\n
            --soft-pink: #F3E8E6;\n
            --light-pink: #F9F4F3;\n
            --soft-green: #A8C1A1;\n
            --dark-gray: #8D99AE;\n
            --accent-blue: #7E9DAA;\n
            --accent-orange: #D9A78E;\n
        }\n
\n
        body {\n
            background-color: var(--light-pink);\n
            font-family: 'Roboto', sans-serif;\n
        }\n
\n
        .navbar {\n
            background-color: var(--soft-pink);\n
```

Gambar 10. Log kesalahan logika pada tahap test

Selain error sintaks, kesalahan juga dapat terjadi pada tahap test, terutama jika terdapat masalah logika atau koneksi ke database. Gambar berikut menampilkan log error yang terjadi, seperti tabel database yang tidak ditemukan atau validasi data yang gagal. Informasi ini membantu pengembang untuk mengidentifikasi masalah dengan lebih cepat dan akurat.

4.2. Evaluasi Keberhasilan Implementasi CI/CD

Berikut adalah tabel perbandingan waktu eksekusi pipeline:

Tabel 3. Tabel Perbandingan Waktu Eksekusi

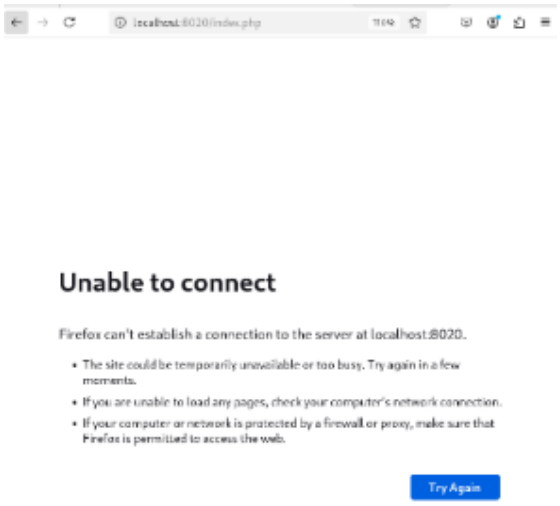
Tahap	Waktu Rata-rata	Variasi Waktu	Tingkat Keberhasilan
Build	3 detik	± 2 detik	100%
Test	5 detik	± 3 detik	95%
Deploy	3 detik	± 2 detik	100%

Berikut merupakan penjelasan dari Tabel 3:

a. Tahap build memiliki waktu rata-rata **3 detik** dengan variasi waktu ± 5 detik dan tingkat keberhasilan 100%, menunjukkan bahwa proses build berjalan stabil dan efektif. Jenkins berhasil mengunduh kode sumber dari repository GitHub dan membangun aplikasi tanpa kesalahan. Proses ini mengotomatiskan pekerjaan manual dan memastikan efisiensi dalam waktu build.
b. Tahap test memiliki waktu rata-rata **5 detik** dengan variasi waktu ± 3 detik dan tingkat keberhasilan 95%, menunjukkan bahwa proses pengujian berjalan dengan baik, tetapi masih ada beberapa kesalahan kecil yang perlu diperbaiki. Pengujian dilakukan untuk memeriksa interaksi antara backend PHP dan database MySQL, termasuk operasi CRUD pada tabel tasks. Proses pengujian otomatis ini sangat efektif untuk mendeteksi bug lebih awal, yang sebelumnya mungkin terlewatkan dalam pengujian manual.
c. Tahap deploy memiliki waktu rata-rata **3 detik** dengan variasi waktu ± 2 detik dan tingkat keberhasilan 100%, menunjukkan bahwa proses deploy berjalan sangat cepat dan efektif. Setelah aplikasi lolos tahap build dan test, proses deploy dilakukan ke server lokal menggunakan PHP built-in server, sehingga aplikasi dapat diakses langsung di lingkungan pengujian tanpa memerlukan intervensi manual.

4.3. Tantangan yang Dihadapi

Konfigurasi awal Jenkins untuk menghubungkan repository GitHub dan menyusun pipeline CI/CD membutuhkan waktu dan pemahaman yang mendalam. Meskipun konfigurasi cukup kompleks, pipeline berjalan otomatis dan lancar setelah selesai disiapkan. Namun, kendala muncul saat Jenkins tidak dapat menjalankan PHP built-in server di latar belakang, meskipun log menunjukkan deployment berhasil. Masalah ini teratasi dengan mengubah PHP built-in server menjadi layanan systemd, sehingga Jenkins cukup merestart layanan tersebut pada tahap deployment untuk memastikan server menjalankan aplikasi versi terbaru secara konsisten.



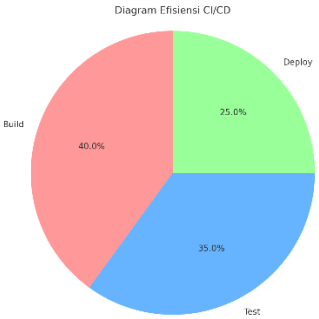
Gambar 11. Aplikasi Tidak Dapat Terhubung ke Port 8020

Pengelolaan sumber daya juga menjadi perhatian, karena Fedora Workstation di VirtualBox hanya memiliki 2 core prosesor dan 4090 MB RAM. Meskipun cukup untuk aplikasi sederhana, aplikasi yang lebih kompleks memerlukan peningkatan alokasi sumber daya guna menjaga kinerja optimal dan memastikan pipeline berjalan tanpa hambatan.

4.4. Pembahasan

Hasil implementasi pipeline CI/CD menggunakan Jenkins pada Fedora Workstation menunjukkan pengaruh signifikan terhadap kecepatan, efisiensi, dan kualitas pengembangan aplikasi berbasis PHP. Setiap tahap pipeline build, test, dan deploy memiliki waktu rata-rata eksekusi yang cepat. Tahap build memerlukan waktu rata-rata 3 detik dengan variasi ±2 detik, mencakup pengunduhan kode dari repository GitHub dan pemeriksaan sintaks menggunakan php -l. Tahap test membutuhkan rata-rata 5 detik dengan variasi ±3 detik, mencakup validasi logika aplikasi dan pengujian koneksi database MySQL, termasuk operasi CRUD. Tahap deploy memiliki waktu rata-rata 3 detik dengan variasi ±2 detik, di mana kode yang telah diuji berhasil dideploy ke server lokal menggunakan PHP built-in server. Tingkat keberhasilan masing-masing tahap pipeline menunjukkan performa yang baik, dengan tahap build dan deploy mencapai 100% keberhasilan, sementara tahap test memiliki tingkat keberhasilan 95% akibat beberapa kesalahan kecil terkait validasi data atau logika aplikasi.

Pipeline CI/CD memberikan efisiensi signifikan dibandingkan metode manual, di mana seluruh proses pipeline dapat diselesaikan dalam waktu kurang dari 15 detik. Dengan otomatisasi ini, pengembang dapat mengalokasikan lebih banyak waktu untuk pengembangan fitur baru daripada tugas rutin seperti pengujian manual dan deployment. Pipeline juga memberikan umpan balik langsung melalui log, yang mencatat detail kesalahan pada setiap tahap. Jika terjadi kesalahan sintaks pada tahap build, pipeline akan berhenti, dan log Jenkins memberikan informasi rinci tentang jenis dan lokasi kesalahan. Demikian pula, jika terjadi kesalahan logika atau koneksi database pada tahap test, log mencatat detail masalah seperti tabel yang tidak ditemukan atau validasi data yang gagal.



Gambar 12. Diagram Perbandingan Efisiensi Build, Test, dan Deploy

a. Build (40%): Tahap ini memiliki kontribusi terbesar terhadap efisiensi pipeline CI/CD. Proses build mencakup pengunduhan kode terbaru dari repository GitHub dan pemeriksaan sintaks menggunakan PHP lint (php -l). Langkah ini sangat penting untuk memastikan bahwa kode bebas dari kesalahan sintaks sebelum melanjutkan ke tahap berikutnya.
b. Test (35%): Tahap pengujian otomatis berkontribusi sebesar 35% terhadap efisiensi keseluruhan. Proses ini memverifikasi kualitas kode dengan menjalankan pengujian pada aplikasi, seperti memeriksa konektivitas database MySQL dan validasi operasi CRUD. Deteksi bug lebih awal pada tahap ini memungkinkan pengembang untuk menghindari masalah besar di lingkungan produksi.
c. Deploy (25%): Tahap deploy memberikan kontribusi sebesar 25%. Pada tahap ini, kode yang telah diuji disebarkan ke server menggunakan PHP built-in server yang berjalan sebagai service. Proses deploy yang cepat memastikan aplikasi dapat langsung diakses di lingkungan pengujian atau produksi tanpa memerlukan intervensi manual.

Kendala teknis yang muncul, seperti ketidakmampuan Jenkins menjalankan PHP built-in server di latar belakang, diselesaikan dengan mengubah server menjadi layanan systemd yang dapat dikelola oleh sistem operasi. Dengan solusi ini, pipeline hanya perlu merestart layanan PHP pada tahap deployment, memastikan aplikasi selalu berjalan dengan versi terbaru tanpa perlu intervensi manual.

Namun, pengelolaan sumber daya menjadi tantangan tambahan, karena Fedora Workstation di VirtualBox hanya dialokasikan 2 core prosesor dan 4090 MB RAM. Meskipun konfigurasi ini cukup untuk aplikasi sederhana, pada aplikasi yang lebih kompleks alokasi sumber daya perlu ditingkatkan untuk menjaga kinerja optimal. Secara keseluruhan, implementasi pipeline CI/CD menggunakan Jenkins memberikan hasil yang signifikan dalam meningkatkan efisiensi, kualitas, dan stabilitas pengembangan perangkat lunak, menjadikannya solusi yang efektif untuk pengembangan aplikasi modern.

5. KESIMPULAN DAN SARAN

Kesimpulan penelitian ini menunjukkan bahwa implementasi pipeline CI/CD menggunakan Jenkins pada aplikasi web berbasis PHP di lingkungan Fedora Workstation berhasil meningkatkan efisiensi dan kualitas pengembangan perangkat lunak. Otomatisasi tahapan build, test, dan deploy mengurangi kesalahan manual, mempercepat waktu pengerjaan, serta memastikan aplikasi selalu diperbarui ke versi terbaru tanpa intervensi manual. Pipeline Jenkins mencapai tingkat keberhasilan 100% pada tahap build dan deploy, serta 95% pada tahap test, meskipun terdapat tantangan pada validasi data dan logika aplikasi. Penggunaan PHP built-in server untuk deployment lokal efektif, namun kendala kegagalan server di latar belakang berhasil diatasi dengan mengubahnya menjadi layanan systemd. Selain itu, penggunaan Fedora Workstation di VirtualBox dengan alokasi sumber daya terbatas membuktikan keefektifannya untuk aplikasi sederhana, meskipun peningkatan sumber daya diperlukan untuk aplikasi yang lebih kompleks. Penelitian ini menekankan pentingnya otomatisasi dalam pengembangan perangkat lunak dan merekomendasikan eksplorasi lebih lanjut pada aplikasi berskala besar dengan pengelolaan sumber daya yang lebih optimal.

DAFTAR PUSTAKA

[1] A. Alpery and M. A. F. Ridha, "Implementasi CI/CD dalam Pengembangan Aplikasi Web Menggunakan Docker dan Jenkins," ABEC Indonesia, vol. 9, pp. 287-296, 2021.

[2] A. Mulyanto and W. Setiawan, "Penerapan Metode Web Engineering Menggunakan Laravel 5 Dalam Pengembangan Penjualan Toko Online Hijapedia Berbasis Website Di Cikarang Bekasi," Jurnal Informatika SIMANTIK, vol. 5, no. 2, pp. 69-74, 2020.

[3] C. E. Lisangan and I. Dwinanto, "Implementasi sistem keamanan komputer host menggunakan sistem operasi Fedora Linux," Jurnal Maklumatika, vol. 7, no. 2, pp. 109–118, 2021.

[4] D. Enda, S. Supria, I. Yulia, M. F. Amirul, and M. F. Asyrofi, "Penerapan Continuous Integration (CI) Pada Aplikasi Web Profil Karang Taruna (Studi Kasus: Karang Taruna Kabupaten Bengkalis)," in Seminar Nasional Industri dan Teknologi, Bengkalis, Indonesia, pp. 56-63, 2022.

[5] I. P. Hariyadi, R. Azhar, H. Santoso, K. Marzuki, and I. M. Y. Dharma, "Implementasi Continuous Integration/Continuous Deployment Untuk Mengotomatisasi Manajemen Konfigurasi Infrastruktur Jaringan Komputer," TEKNIMEDIA: Teknologi Informasi dan Multimedia, vol. 4, no. 2, pp. 168-181, 2023. http://dx.doi.org/10.46764/teknimedia.v4i2.127

[6] M. S. A. Amrullah and G. I. Marthasari, "Otomatisasi Proses Deployment dengan Metode CI/CD Menggunakan Jenkins dan Docker Pada Web Service i-Lab," Jurnal Repositor, vol. 6, no. 4, pp. 313-322, 2024.

[7] R. A. Nuryadin, T. A. Ramadhani, J. Karaman, and M. Reza, "Analisis perbandingan performa virtualisasi server menggunakan VMware ESXi, Oracle VirtualBox, VMware Workstation 16 dan Proxmox," Methomika: Jurnal Manajemen Informatika & Komputerisasi Akuntansi, vol. 7, no. 2, pp. 175–180, 2023. https://doi.org/10.46880/jmika.Vol7No2.pp175-180

[8] R. Y. Endra, Y. Aprilinda, Y. Y. Dharmawan, and W. Ramadhan, "Analisis Perbandingan Bahasa Pemrograman PHP Laravel dengan PHP Native pada Pengembangan Website," Expert, vol. 11, no. 1, p. 346061, 2022. http://dx.doi.org/10.36448/expert.v11i1.2012

[9] R. Setiabudi, "Analisis Efektivitas CI/CD Dan Manual (Tradisional) Dalam Pengembangan Website Rakyatweb.com," ISMETEK, vol. 17, no. 2, pp. 88-93, 2023.

[10] Y. R. Astino and V. P. Sabandar, "Pengembangan Dan Penerapan Sistem Computer Assisted Test (CAT) Untuk Mengelola Ujian Berbasis Website," Jurnal Informatika dan Rekayasa Perangkat Lunak, vol. 4, no. 3, pp. 253-259, 2023. https://doi.org/10.33365/jatika.v4i3.2603