**IMPLEMENTASI CI/CD MENGGUNAKAN GITHUB ACTIONS UNTUK PENGEMBANGAN WEB LANDING PAGE UMKM XYZ**

***IMPLEMENTATION OF CI/CD USING GITHUB ACTIONS FOR THE DEVELOPMENT OF A WEB LANDING PAGE FOR UMKM XYZ***

**Firmansyah Putra Apriliyanto1, Yani Prihati2, Cristeddy Asa Bakti3**

1,2,3Universitas AKI

[223200036@student.unaki.ac.id](mailto:223200036@student.unaki.ac.id)

***ABSTRACT***

*The effective application of information technology is crucial for SMEs to enhance their competitiveness in the digital era. SME XYZ faces challenges in the efficient and reliable development and management of their web landing page. This study aims to implement the Continuous Integration/Continuous Deployment (CI/CD) method using GitHub Actions in the development of SME XYZ's web landing page. The method involves setting up a CI/CD pipeline that automatically integrates, tests, and deploys code changes. The findings of the study indicate that the implementation of GitHub Actions successfully improves the efficiency of the development process, reduces testing and deployment time, and minimizes errors in the code. The conclusion of this study is that the application of CI/CD with GitHub Actions significantly contributes to the improvement of the quality and speed of web development for SMEs, allowing the development team to focus more on enhancing features and services without being burdened by manual integration and deployment issues.*

***Keywords****: CI/CD, Github Actions, UMKM, Server, Website*

**ABSTRAK**

Penerapan teknologi informasi yang efektif sangat penting bagi UMKM untuk meningkatkan daya saing di era digital. UMKM XYZ menghadapi tantangan dalam pengembangan dan pengelolaan web landing page yang efisien dan reliabel. Penelitian ini bertujuan untuk menerapkan metode Continuous Integration/Continuous Deployment (CI/CD) menggunakan GitHub Actions dalam pengembangan web landing page UMKM XYZ. Metode yang digunakan melibatkan pengaturan pipeline CI/CD yang otomatis mengintegrasikan, menguji, dan menerapkan perubahan kode. Temuan penelitian menunjukkan bahwa implementasi GitHub Actions berhasil meningkatkan efisiensi proses pengembangan, mengurangi waktu pengujian dan deploy, serta meminimalkan kesalahan dalam kode. Kesimpulan dari penelitian ini adalah bahwa penerapan CI/CD dengan GitHub Actions memberikan kontribusi signifikan terhadap peningkatan kualitas dan kecepatan pengembangan web UMKM, memungkinkan tim pengembang untuk lebih fokus pada peningkatan fitur dan layanan tanpa terbebani oleh masalah integrasi dan deploy manual.

**Kata Kunci**: CI/CD, Github Actions, UMKM, Server, Website.

**PENDAHULUAN**

Perkembangan teknologi baru memberikan dampak yang signifikan pada berbagai bidang kehidupan manusia. Salah satunya adalah kemajuan di bidang penyebaran informasi. Proses penyebaran tersebut bisa melalui media massa seperti radio, televisi, videotron dan media yang saat ini menjadi *trendsetter* adalah internet.(Dwi Kurniawan et al., 2020).

*Internet (Interconnected Networking)* merupakan kumpulan komputer yang saling terhubung membentuk sebuah jaringan yang terkoneksi ke seluruh dunia dan saling bertukar data dimana data tersebut nantinya akan ditampilkan dalam bentuk informasi yang bisa dilihat dari berbagai *platform* seperti *website* (Nurbaiti & Alfarisyi, 2023).

*Website* pada dasarnya adalah perangkat lunak yang berisi dokumen seperti gambar, suara, animasi, teks. *Website* terbagi menjadi dua macam yaitu *website* statis dan *website* dinamis. *Website* statis merupakan *website* yang mempunyai tampilan tetap dan setiap perubahan konten harus dilakukan secara manual oleh pengembang *website*, sedangkan *website* dinamis merupakan *website* yang kontennya dapat berubah secara otomatis berdasarkan interaksi pengguna. *Website* secara umum dibangun menggunakan *markup language (HTML)* dan bahasa pemograman seperti *Javascript, CSS* dan bahasa pemograman lainnya menyesuaikan kebutuhan pengguna (Kesuma Astuti & Sri Agustina, 2022).

Dalam pengembangan perangkat lunak khususnya *website*, terdapat banyak teknologi pendukung yang digunakan untuk memaksimalkan hasil akhir proyek ataupun alur kerja, salah satu yang populer adalah otomatisasi. Proses otomatisasi ini berkaitan erat dengan *CI/CD* (*Continuous Integration/Continuous Deployment*) dimana nantinya *developer* menulis kode proyek untuk kemudian di *pus­h* ke sistem terpusat yang mengelola *source code* seperti Git. Kegiatan push kode ini akan memicu rangkaian kegiatan *CI/CD* meliputi build app, test unit dan deployment ke *server* produksi. Seluruh proses *CI/CD* berjalan secara otomatis sehingga meminimalkan campur tangan *developer*  (Toba et al., 2022).

Sebelumnya, pengembangan perangkat lunak cenderung menggunakan model waterfall. Model ini memiliki jalur yang panjang, berurutan dan saling bergantung satu sama lain, mulai dari konsepsi hingga siap digunakan (Muhamad Adillah Fatih et al., 2024). Namun, munculnya praktik Agile pada awal 2000-an menciptakan keinginan untuk mendapatkan *feedback* yang cepat dari pengguna dan perilisan perangkat lunak yang lebih konsisten. Karena alasan ini muncul istilah CI (Continuous Integration) yang merupakan sebuah praktik yang berfokus pada penggabungan kode ke sistem terpusat, pengujian otomatis dan umpan balik yang cepat dan bebas dari kesalahan. Sedangkan CD (Continuous Deployment) berfokus pada perilisan aplikasi ke lingkungan *production* secara otomatis*,* sehingga pengguna bisa mendapatkan fitur terbaru dan perbaikan *bug* lebih cepat. Dalam kerangka kerja *Agile* dan *DevOps*, *CI/CD* muncul sebagai praktik penting yang membantu organisasi dalam memberikan produk aplikasi secara cepat dan konsisten kepada pengguna mereka (Thatikonda, 2023).

Salah satu *tools* yang mendukung penerapan *CI/CD* adalah Github Actions. Github Actions memungkinkan *developer* untuk membuat dan mengelola alur kerja secara otomatis, menjalankan pengujian, build dan deployment setiap kali terdeteksi ada perubahan pada kode sumber (Wessel et al., 2023).

Penerapan *CI/CD* menggunakan Github Actions juga sejalan dengan prinsip Agile dan Devops yang mendorong kolaborasi tim, respon yang cepat terhadap perubahan, dan pengiriman perangkat lunak yang cepat. Hal ini akan berdampak pada waktu respon terhadap kebutuhan pasar dan pengguna, dan akan menjaga daya saing suatu organisasi/perusahaan di tengah persaingan bisnis yang kian ketat (Dwi Praba & Santoso, 2023).

UMKM XYZ merupakan salah satu UMKM yang berupaya meningkatkan kehadirannya secara digital melalui pengembangan landing page yang modern dan responsif. UMKM XYZ memerlukan *website* yang selalu up-to-date dengan perubahan yang cepat, oleh karena itu penerapan *CI/CD* menjadi solusi efektif untuk mengatasi kebutuhan tersebut. Dengan menerapkan *CI/CD* setiap perubahan pada kode web bisa diuji dan di-deploy secara otomatis.

Penelitian ini bertujuan untuk memberikan panduan praktis mengenai implementasi *CI/CD* menggunakan Github Actions yang akan diterapkan pada *website* landing page UMKM XYZ. Hasil penelitian ini diharapkan dapat memberikan kontribusi signifikan dalam membantu para *developer* untuk mengadopsi teknologi *CI/CD* sehingga dapat meningkatkan kinerja perangkat lunak dan memastikannya dalam kondisi yang optimal.

**METODE**

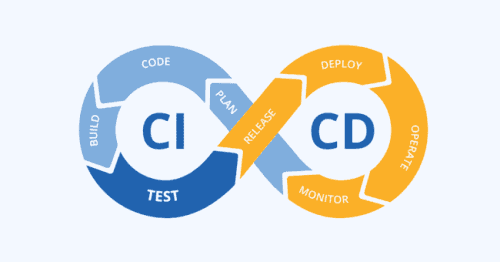
Dalam penelitian ini akan dibahas beberapa tahapan penting yang mendukung kegiatan penelitian seperti yang tertera pada gambar 1.

A screenshot of a computer

Description automatically generated

**Gambar 1. Metodologi kerja**

**Konsep Dasar CI/CD**



**Gambar 2. CI/CD Workflows**

**(Sinha, 2020)**

*CI (Continuous Integration)* merupakan praktik pengembangan perangkat lunak dimana *developer* harus menggabungkan kode ke repositori terpusat setiap ada perubahan kode. Selanjutnya kode akan diverifikasi dan dilakukan pengujian kode secara otomatis sehingga *developer* bisa mendeteksi masalah atau kesalahan kode secepat mungkin (Toba et al., 2022).

*CD (Continuous Deployment)* adalah praktik pengembangan perangkat lunak dimana setelah kode berhasil di-*build* dan lolos *testing*, aplikasi akan di-*deploy* dan dirilis ke lingkungan produksi sehingga bisa sampai ke pengguna akhir (Guna Noviantama & Purno Wahyu, 2021) .

**Software & Tools**

Terdapat beberapa aplikasi *tools* yang bisa dikombinasikan untuk mencapai tujuan mengotomatiskan proses pengembangan software di antaranya:

1. Version Control

Adalah *tools* untuk mencatat dan melacak riwayat perubahan, memfasilitasi rollback, mendukung branching dan merging pada kode yang sedang dikerjakan. Contoh tools nya adalah *Git, SubVersion, Mercurial.*

1. Cloud Version Control Provider

Adalah penyedia layanan *version control* di lingkungan *cloud* sebagai repositori terpusat untuk menyimpan *source code* proyek. Contohnya adalah *Github, BitBucket, Gitlab, Azure Repos.*

1. Cloud Platform

Adalah layanan berbasis internet yang menyediakan berbagai sumber daya komputasi, seperti penyimpanan data, *server*, database, jaringan, perangkat lunak, dan alat pengembangan yang dapat diakses dan dikelola secara online. *Cloud* platform memungkinkan pengguna untuk menyewa sumber daya ini sesuai kebutuhan tanpa perlu investasi besar dalam infrastruktur fisik. Contoh *cloud* platform yang terkenal diantaranya *Alibaba Cloud Service, AWS, Microsoft Azure, Google Cloud Platform.*

1. CI/CD Tools

Adalah alat yang digunakan untuk mendukung praktik CI (Continuous Integration) dan CD (Continuous Deployment) dalam pengembangan perangkat lunak. Contohnya adalah *Github Actions, Jenkins, Gitlab CI/CD, CircleCI, Travis CI.*

1. Runtime

Adalah lingkungan yang menyediakan layanan untuk menjalankan program yang ditulis dalam bahasa pemograman tertentu. Contohnya adalah *Nodejs* sebagai *runtime Javascript, JVM (Java Virtual Machine)* sebagai runtime program *Java, .NET* sebagai runtime program *C#*.

1. Framework

Adalah kumpulan *library* yang menyediakan struktur standar, pola desain dan panduan untuk untuk membangun sebuah aplikasi. Contohnya adalah *ReactJS, Django, Laravel, Spring, Flutter.*

1. Teks editor

Adalah aplikasi untuk menulis dan mengedit teks. Dalam pengembangan perangkat lunak, teks editor digunakan untuk menulis *source code.*

**Kebutuhan Sistem untuk CI/CD**

Pada tahapan ini akan dijabarkan mengenai kebutuhan minimum sistem yang digunakan untuk menjalankan proses *development* aplikasi dan proses *CI/CD* *website* *Landing Page* di *server*. Pemilihan perangkat untuk kebutuhan *CI/CD* perlu memperhatikan berdasarkan beban kerja dari aplikasi yang dibuat dan pertimbangan banyaknya jumlah pengguna di masa mendatang. Oleh karena itu pemilihan spesifikasi perangkat sangat penting untuk diperhatikan.

1. Laptop

**Tabel 1. Spesifikasi minimum**

|  |  |  |
| --- | --- | --- |
| **No** | **Part** | **Tipe** |
| 1  2  4  5  6 | CPU  RAM  Storage  Screen  OS | Ryzen 3 atau setara  8 GB  128 GB SSD  14 Inch  Windows 10 |
|  | | |

1. Aplikasi Server

**Tabel 2. Aplikasi CI/CD**

**pada server**

|  |  |  |
| --- | --- | --- |
| **No** | **Paket** | **Jenis** |
| 1  2  3 | Web server  Process Manager  Runtime | Nginx  PM2  NodeJS |

1. Software

**Tabel 3. Software untuk development aplikasi**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Part** | | **Tipe** |
| 1 | Text Editor | VS Code | |
| 2  3  4  5  6  7 | Browser  Terminal  CI/CD Tools  Version Control  Framework  Runtime | Google Chrome  Gitbash  Github Actions  Git  ReactJS  Nodejs | |

**HASIL DAN PEMBAHASAN**

1. **Eksplorasi Perangkat**

Setelah menganalisa dan membandingkan beberapa fitur *CI/CD* Tools dan platform cloud dari berbagai sumber literatur, diputuskan untuk menggunakan Github Actions sebagai *CI/CD* tools dan server dari Alibaba Cloud sebagai tempat *deployent.* Github Actions dipilih karena mempunyai lisensi gratis yang bisa digunakan untuk proyek penelitian sederhana atau proyek bisnis dengan skala yang kecil hingga besar. Selain itu Github Actions dipilih karena mempunyai integrasi yang sangat baik dengan repositori Github melalui konfigurasi file YAML yang sangat fleksibel. Github Actions juga bisa diintegrasikan dengan berbagai ekosistem cloud contohnya seperti Alibaba, Google Cloud Computing, AWS, Microsoft Azure.

**Tabel 4. Spesifikasi**

**VPS (Virtual Private Server) Alibaba Cloud**

|  |  |  |
| --- | --- | --- |
| **No** | **Part** | **Tipe** |
| 1  2  3  4  5 | vCPU  RAM  Storage  Bandwidth  OS | 1 Core  1 GB  40 GB  1 Mb/s  Centos 7.9 |

Pemilihan *cloud server* untuk *deployment* web *landing page* UMKM XYZ seperti pada tabel 4 dilakukan berdasarkan analisis menyeluruh terhadap kebutuhan spesifik proyek aplikasi. *Cloud server* ini menawarkan *CPU* sebanyak 1 *core,* memori 1 GB, media penyimpanan 40 GB, *bandwidth* 1 Mb/s dan sistem operasi Centos 7.9. Selain itu, keberadaan *data center* yang terdapat di Indonesia memberikan keuntungan dalam hal latensi dan kepatuhan terhadap regulasi data lokal. Layanan dan fitur yang ditawarkan termasuk dukungan tambahan untuk ekosistem pengembangan web dan integrasi dengan *tools* *CI/CD* turut menjadi faktor penting dalam pemilihan ini.

1. **Arsitektur Sistem**

Pada tahapan ini akan dijelaskan bagaimana alur kerja/*workflows* dari penelitian ini. Detail alur dapat dilihat pada gambar 1.

A diagram of a software process

Description automatically generated

**Gambar 3. Alur kerja penelitian CI/CD**

1. Push Code

Pada tahapan ini *developer* menulis kode dan kemudian melakukan push ke repositori Github. tindakan mengirimkan perubahan kode yang telah di-commit dari repositori lokal ke repositori remote (misalnya, repositori di GitHub). Push dilakukan setelah commit untuk memperbarui repositori remote dengan perubahan terbaru yang dibuat di repositori lokal.

1. Github Repository

Setelah kode di-push, kode tersebut disimpan dalam repositori GitHub. Repositori GitHub adalah tempat penyimpanan yang terstruktur untuk semua kode sumber, termasuk riwayat perubahan, cabang (branch).

1. Continuous Integration

Pada tahapan ini, setelah kode tersimpan di repositori. Github akan mendeteksi ada perubahan kode dan mentrigger proses CI (Continuous Integration). Disini build dan testing akan dilakukan secara otomatis oleh *server CI* untuk memastikan bahwa kode baru telah ditambahkan dan tidak terdapat kesalahan kode yang terdeploy ke lingkungan *production*.

1. Continuous Deployment

Pada tahapan ini setelah build dan testing berhasil, kode yang telah diverifikasi akan otomatis terdeploy ke lingkungan *production.* Proses ini memastikan bahwa perubahan yang lulus pengujian segera tersedia untuk pengguna. Di Github Actions, proses CD (Continuous Deployment) dapat diintegrasikan dengan berbagai layanan cloud seperti Alibaba, AWS, Azure, Google yang bisa disesuaikan kebutuhan penelitian.

1. **Setup Server**

Pada tahap ini penelitian dimulai dengan menyiapkan *server* yang akan dijadikan tempat deploy *website* *landing page*. *Server* ini menggunakan sistem operasi Centos 7.9 yang nantinya akan dipasang berbagai aplikasi seperti web *server* Nginx, PM2 dan Nodejs sebagai runtime Javascript. Tujuan pemasangan aplikasi tersebut agar *website* dapat berjalan di *server* dan bisa diakses oleh pengguna.

Penggunaan Nginx sebagai web *server* dilakukan karena aplikasi tersebut juga mendukung reverse proxy yang nantinya dapat menerima dan meneruskan permintaan ke port suatu aplikasi yang sedang berjalan di *server*, dalam hal ini adalah proyek landing page. Penggunaan PM2 dilakukan karena process manager tersebut dikhususkan untuk aplikasi yang dibuat dengan runtime Nodejs. PM2 mendukung fitur restart otomatis yang dapat memulai ulang aplikasi apalagi terdapat crash atau kegagalan, sehingga memastikan ketersediaan yang tinggi. Sedangkan penggunaan Nodejs dilakukan karena *website* yang dibuat pada penelitian ini menggunakan framework Reactjs, dimana Reactjs adalah salah satu *framework Javascript* yang penggunaannya diharuskan menggunakan *runtime* agar kode bisa dieksekusi pada komputer lokal.

1. **Repositori Github & Github Actions**

Pada tahap ini akan diuraikan tentang repositori Github yang menjadi fitur krusial dalam proses *CI/CD* yang akan dipakai dalam penelitian.

A screenshot of a computer

Description automatically generated

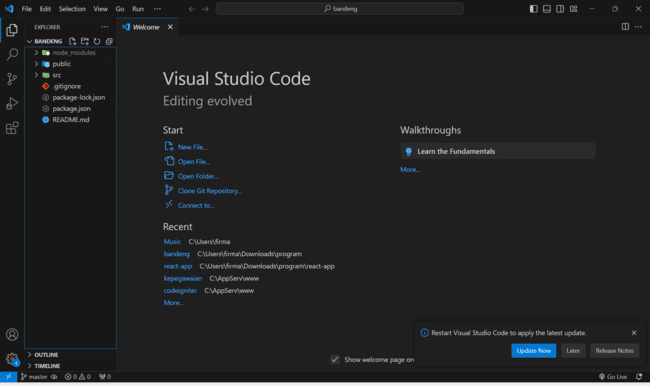
**Gambar 4. Membuat repositori Github**

Pada gambar 4 menunjukan akan dibuat sebuah repositori baru bernama bandeng, repositori ini nantinya yang akan menjadi wadah atau tempat menyimpan kode sumber proyek *website* landing page.

1. **Pembuatan Website**

Pada tahap ini akan dibuat *website* landing page yang nantinya akan digunakan UMKM XYZ dengan menggunakan framework ReactJS dan runtime Nodejs di lingkungan lokal.

Dengan bantuan terminal GitBash, pembuatan proyek Reactjs dimulai dengan perintah **“npx create-react-app bandeng”**. Perintah tersebut akan membuat aplikasi React baru dengan folder bernama “bandeng” seperti yang bisa dilihat pada gambar 5.



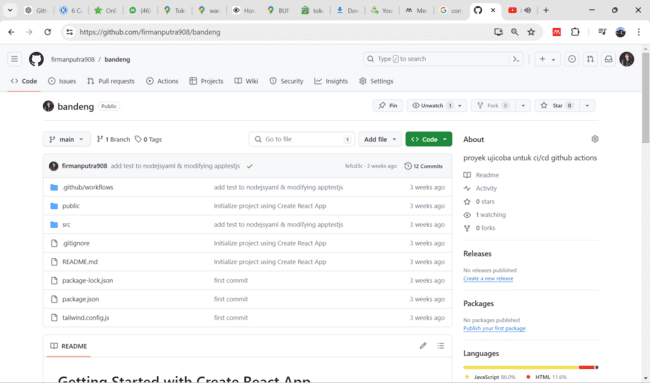
**Gambar 5. Proyek Reactjs**

Gambar 5 diatas merupakan folder dan file yang otomatis dibuat dari perintah sebelumnya. Pada folder inilah berbagai struktur *website* seperti gambar, kode akan disusun oleh *developer* menjadi sebuah *website* *landing page* yang nantinya akan di push ke repositori Github.

1. **Push Code**

Setelah *developer* selesai membuat atau melakukan perubahan pada sumber kode di lingkungan lokal. *Developer* akan mengunggah kode tersebut ke repositori Github. Terdapat beberapa tahapan yang perlu dilakukan sebelum melakukan *push* kode.

1. **git add .** ,perintah tersebut digunakan untuk menambahkan perubahan terbaru ke area staging.
2. **git commit -m “pesan perubahan”** , perintah tersebut digunakan untuk merekam snapshot perubahan dari area staging ke repositori lokal dengan pesan tertentu.
3. **git push -u origin main** , perintah tersebut digunakan untuk mengirimkan commit dari repositori lokal ke *branch* “main” yang ada di repositori Github.

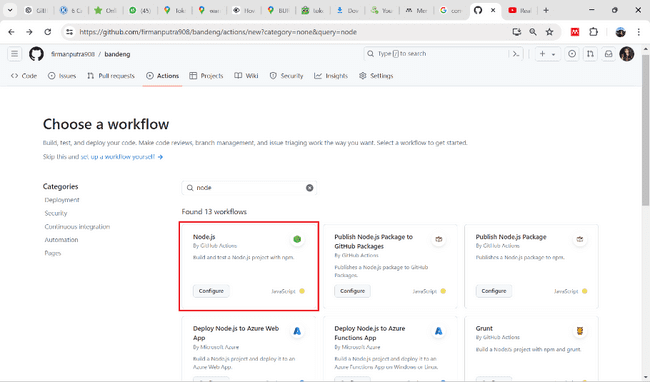


**Gambar 6. Repositori Github**

Pada gambar 6, setelah kode berhasil di push, maka perubahan kode yang telah dilakukan *developer* di lingkungan lokal akan berpindah ke repositori yang ada di Github.

1. **CI/CD Workflows**

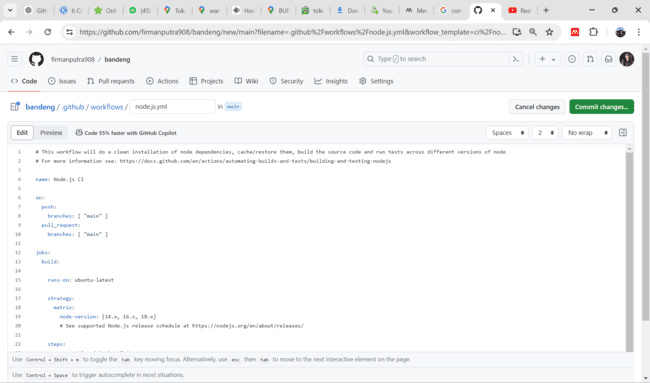
Pada tahapan ini akan ditambahkan sebuah file dengan format YAML yang didapat dari menu Actions yang berisi konfigurasi tentang alur kerja yang akan dijalankan otomatis ketika terdeteksi ada perubahan di repositori Github.



**Gambar 7. Workflows Nodejs**

Github Actions mempunyai integrasi yang kuat dengan berbagai bahasa pemograman, platform cloud dan framework lainnya. Pada gambar 7 terdapat salah satu *workflow* dengan nama

Node.js (kotak merah) yang nantinya akan dijadikan sebagai acuan konfigurasi *CI/CD* dalam pengerjaan penelitian ini.



**Gambar 8. Konfigurasi bawaan node.js.yml**

Pada gambar 8 terdapat file YAML dengan konfigurasi bawaan, nantinya *developer* bisa mengubah konfigurasi sesuai kebutuhan aplikasi. Nantinya file node.js.yml ini harus diletakan di dalam folder .github/workflows.



**Gambar 9. File Node.js.yml**

Penempatan folder .github/workflows akan sangat berpengaruh terhadap terjadinya proses *CI/CD*. Oleh karena itu folder tersebut harus diletakan di *root* folder aplikasi seperti yang tertera pada gambar 9.

1. **Konfigurasi File YML**

Pada tahapan ini akan dilakukan konfigurasi file node.js.yml yang berfungsi untuk mendefinisikan dan mengkonfigurasi alur kerja otomatis yang akan dijalankan Github Actions. Pada penelitian ini dibagi menjadi 4 tahapan yang dilakukan untuk membangun kode YML meliputi pendefinisian nama *workflow, build, test, deploy*, dimana tiap fase akan berjalan secara berurutan. Berikut adalah konfigurasi file node.js.yml yang akan digunakan dalam penelitian :

1. Mendefinisikan nama workflow

Langkah pertama adalah mendefinisikan nama workflow. Penelitian ini menggunakan **Node.js CI** sebagai nama *workflow*. Nantinya nama ini akan muncul di antarmuka Github Action, sehingga memudahkan untuk mengidentifikasi *workflow* yang sedang berjalan. Selanjutnya adalah pendefinisian kapan *workflow* akan dijalankan berdasarkan *event* yang terjadi pada repositori. *Push main* berarti workflow akan dipicu setiap kali ada commit yang di-push ke branch main, sedangkan *pull request* main berarti *workflow* akan dipicu setiap kali ada *pull request* yang menargetkan *branch main*.

1. Build

Pada tahap *build*, *runner* yang digunakan adalah milik bawaan *Github Actions* yaitu *ubuntu-latest*. Penelitian ini menggunakan Node.js versi 16.x sebagai lingkungan berjalannya aplikasi. Selanjutnya dengan perintah **npm ci**, seluruh dependensi yang ada di file package.json akan diunduh dan perintah **npm run build** --**if-present** akan membangun aplikasi jika kode sumber tersedia.

1. Testing

Tahapan testing dilakukan setelah tahap build berhasil dilaksanakan. Proses ini melibatkan pengambilan ulang kode sumber dari repositori. Dependensi proyek diinstal ulang menggunakan npm ci, dan pengujian dijalankan setelah mencetak pesan "Running react test". Pengujian ini dijalankan menggunakan perintah npm test untuk memastikan kode yang dibangun berfungsi dengan benar.

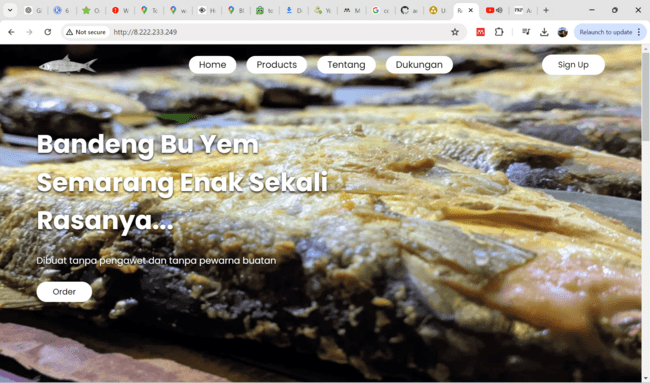
1. Deployment

Tahapan deploy akan dieksekusi setelah tahap test berhasil dilakukan. Tahap deploy melibatkan koneksi ke *server* melalui SSH. Pada tahap ini perintah yang dijalankan meliputi navigasi ke direktori aplikasi di *server* Centos, menarik perubahan terbaru yang ada di repositori Github khususnya pada branch main menggunakan perintah **git pull origin main**, dan merestart aplikasi *website* menggunakan PM2. Langkah-langkah ini memastikan bahwa versi terbaru dari aplikasi akan segera dirilis ke *server* produksi.

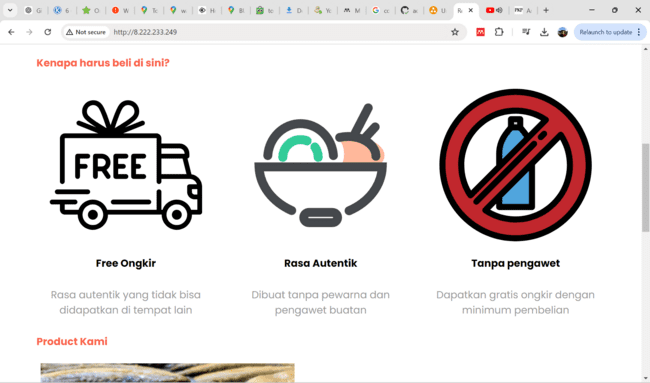
Dengan struktur ini workflow memastikan bahwa setiap perubahan kode yang masuk ke branch main melalui proses build dan pengujian yang ketat sebelum akhirnya di-deploy ke *server* produksi. Proses ini memberikan jaminan kualitas dan stabilitas kode yang di-deploy.

1. **Implementasi**

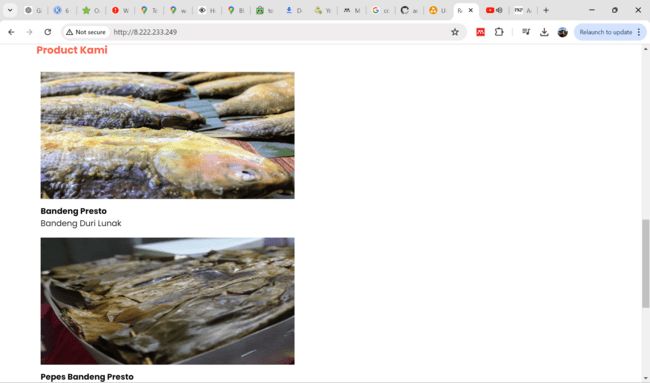
Pada tahapan ini akan dilakukan implementasi *CI/CD* dengan cara melakukan perubahan kode pada sumber kode aplikasi landing page. Seperti yang bisa dilihat pada poin F, *developer* akan *commit* dan *push* sumber kode yang telah diubah ke repositori Github. Setelah itu proses build, testing, dan deploy akan dijalankan secara otomatis oleh Github Actions. Proses tersebut akan terus dilakukan berulang ketika terdapat perubahan kode yang dilakukan oleh *developer.* Dalam penelitian ini perubahan kode dilakukan di lingkungan lokal dengan bantuan teks editor Visual Studio Code.



**Gambar 10. Home page**



**Gambar 11. Feature page**



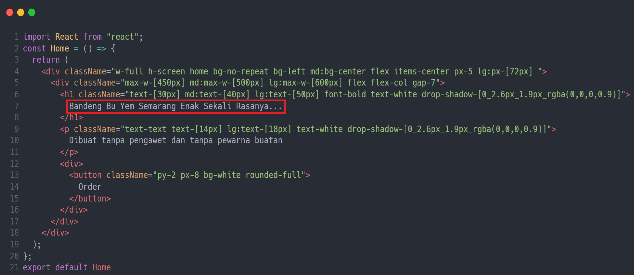
**Gambar 12. Product page**

A screenshot of a computer

Description automatically generated

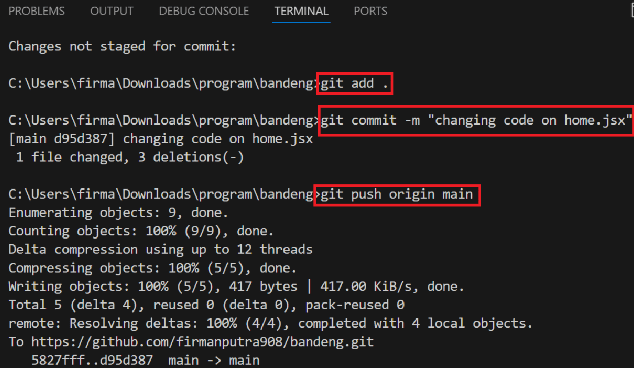
**Gambar 13. Footer**

Gambar 10-13 adalah tampilan landing page yang telah dibuat oleh *developer*, memiliki 4 bagian utama meliputi home page, feature page, product page dan footer.



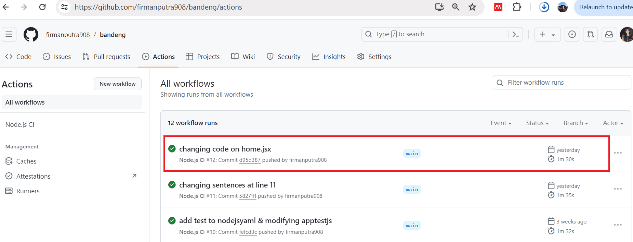
**Gambar 14. Kode sumber home.jsx**

Pada gambar 14 terdapat kode sumber file home.jsx yang membangun tampilan Homepage. Pada gambar tersebut juga terdapat baris dengan kota merah yang terdapat kalimat **“Bandeng Bu Yem Semarang Enak Sekali Rasanya”** . Kalimat ini nantinya akan diubah menjadi **“Nikmati Bandeng Bu Yem Semarang, Sensasi Rasa Tak Terlupakan…**”

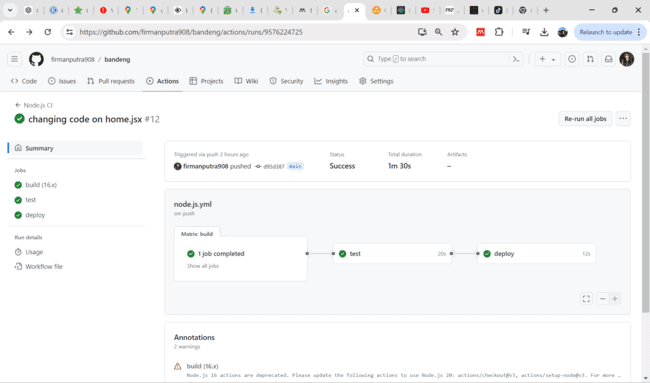


**Gambar 15. Push kode**

Setelah *developer* melakukan perubahan kode, maka langkah selanjutnya adalah menambahkan semua perubahan ke staging area, lalu membuat commit yang berisi perubahan yang telah dilakukan dengan pesan deskriptif tertentu yang menjelaskan perubahan tesebut. Disini terdapat pesan **“changing code on home.jsx”** yang menunjukan bahwa terdapat perubahan kode yang dilakukan pada file home.jsx. Selanjutnya *developer* akan mengirimkan commit tersebut ke branch main pada repoositori di Github. Github Actions akan otomatis melakukan build, testing, dan deploy.



**Gambar 16. Alur kerja Github**



**Gambar 17. Ringkasan alur kerja**

A black screen with a black background

Description automatically generated

**Gambar 18. Build job**

A screenshot of a computer

Description automatically generated

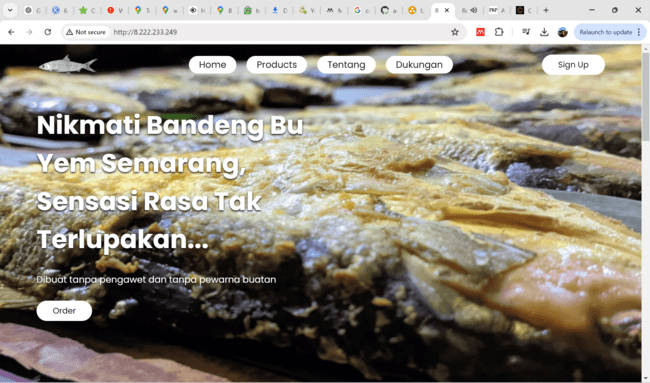
**Gambar 19. Test job**

A black screen with a black background

Description automatically generated

**Gambar 20. Deploy Job**

Pada gambar 18-20 terdapat jobs yang telah didefinisikan pada file node.js.yml. Jobs tersebut diproses secara berurutan dimulai dengan job build untuk memastika bahwa kode dapat di-*compile* dan dibangun dengan benar, hal ini mencakup langkah-langkah seperti mengambil kode dari repositori, menginstall dependensi menggunakan **“npm ci”** dan menjalankan kode *build* dengan **“npm run build”**. Job test dijalankan untuk menjalankan tes unit termasuk tes integrasi, proses pengujian tersebut dijalankan dengan perintah **“npm test.** Job *deploy* dilaksanakan untuk mengirim versi terbaru dari aplikasi landing page ke *server*, proses ini melibatkan langkah seperti remote *server* melalui SSH, menarik perubahan terbaru dari repositori Github dan melakukan restart aplikasi menggunakan PM2.



**Gambar 21. Home page baru**

Pada gambar 21 dapat dilihat bahwa kalimat pada bagian homepage telah berubah. Ini menunjukan bahwa penerapan *CI/CD* menggunakan Github Action berhasil dilakukan. *Developer* hanya perlu melakukan perubahan kode pada aplikasi sesuai kebutuhan dan melakukan push kode ke repositori Github. Selanjutnya Github Actions yang menangani proses *build,testing* dan *deploy* ke *server*.

1. **Pembahasan**

Setelah menerapkan *CI/CD* melalui Github Action pada *website* landing page, selanjutnya akan dijelaskan tentang waktu yang diperlukan aplikasi untuk menyelesaikan satu alur kerja (Time Based Metric) dan kualitas (Quality Metric) yang meliputi frekuensi kegagalan dan waktu rilis aplikasi (Zulhakim & Kurniawan, 2024).

1. Time Based Metrics

Merupakan waktu yang diperlukan aplikasi untuk menyelesaikan satu alur kerja dalam membangun satu aplikasi. Berikut ini dilakukan uji coba alur kerja Github Actions sebanyak 5 kali.

A screenshot of a computer

Description automatically generated

**Gambar 22. Hasil Uji Coba Alur Kerja Github Actions**

**Tabel 5. Hasil uji alur kerja**

**Github Actions**

|  |  |  |
| --- | --- | --- |
| No. Build | Github Actions | |
| Build Time | Status |
| 1  2  3  5  6 | 1m 30s  1m 35s  1m 32s  54s  58s | Sukses  Sukses  Sukses  Sukses  Sukses |

1. Quality Metrics

Pada penelitian ini digunakan 2 metrik yaitu metrik frekuensi kegagalan (failure rate) dan waktu rilis (release time). Penggunaan metrik ini dilakukan untuk memastikan aplikasi bisa berjalan dengan baik dan sesuai dengan kebutuhan pengguna.

Metrik frekuensi kegagalan (failure rate) memberi informasi tentang tingkat kegagalan aplikasi dalam menjalankan fungsinya. Dengan informasi ini, *developer* bisa mengambil tindakan yang diperlukan ketika terjadi masalah pada aplikasi yang sedang dikerjakan. Pada tabel 5, dilakukan 5 kali uji coba alur kerja dimana dalam uji coba tersebut mempunyai tingkat keberhasilan 100%, ini menunjukan bahwa *Github Actions* mampu menjalankan proses *CI/CD* yang meliputi *build, testing, deployment* dengan baik.

Metrik waktu rilis (release time) dapat membantu *developer* untuk melihat waktu rilis aplikasi setiap terjadi perubahan. Hal ini penting dilakukan agar *developer* bisa mendapat estimasi waktu rilis rata-rata untuk merespons kebutuhan pengguna yang sangat cepat. Berdasarkan data yang tertera pada tabel 5, waktu rata-rata yang dibutuhkan oleh Github Actions untuk menyelesaikan 5 kali uji coba adalah 77,8 detik atau 1 menit 17,8 detik. Dengan waktu terlama adalah 1 menit 35 detik dan waktu tercepat adalah 54 detik

**SIMPULAN**

Dalam penelitian ini telah berhasil dilakukan implementasi *CI/CD* menggunakan Github Actions yang diterapkan pada website landing page UMKM XYZ. Dengan pendekatan *CI/CD* melalui Github Action, proses build,testing dan deploy aplikasi ke *server* menjadi lebih mudah dan hasilnya dapat langsung dilihat oleh pengguna UMKM XYZ.

Penggunaan *Github Actions* sebagai *tools CI/CD* memberi kemudahan dalam konfigurasi dan pemeliharaan aplikasi. Penggunaan repositori Github sebagai tempat penyimpanan sumber kode juga menjadikan konfigurasi lebih terpusat dan sederhana. Selain itu penggunaan file dengan format YML memiliki struktur bahasa yang mudah dipahami sehingga memudahkan *developer* dalam membuat alur kerja *CI/CD*.

Menerapkan (*CI/CD*) pada pengembangan aplikasi membawa berbagai manfaat signifikan. *CI/CD* otomatis mengintegrasikan dan menguji kode secara berkala, mengidentifikasi bug lebih awal dan memastikan bahwa setiap perubahan kode yang di-deploy telah lulus serangkaian pengujian yang ketat. Hal ini mempercepat siklus pengembangan, memungkinkan rilis fitur yang lebih cepat, dan meminimalkan risiko kesalahan dalam produksi. Selain itu, *CI/CD* meningkatkan kolaborasi tim dengan menyediakan feedback cepat dan konsisten, yang secara keseluruhan meningkatkan stabilitas dan k aplikasi kehandalan aplikasi yang sedang dikembangkan.

**DAFTAR PUSTAKA**

Dwi Kurniawan, W., Prijo Budijono, A., & Yunus. (2020). *Pengembangan Web Sebagai Media Informasi dan Promosi Program Studi S1 PENDIDIKAN TEKNIK MESIN JURUSAN TEKNIK MESIN UNESA*.

Dwi Praba, A., & Santoso, T. (2023). PENGEMBANGAN APLIKASI POINT OF SALES MENGGUNAKAN METODE AGILE DENGAN POLA SCRUM. In *JIKA: Vol. ISSN*.

Guna Noviantama, I., & Purno Wahyu, A. W. (2021). MILLENNIA SOLUSI INFORMATIKA. In *Ari Purno Wahyu W Jurnal Ilmiah Teknologi Informasi Terapan* (Vol. 8, Issue 1).

Kesuma Astuti, F., & Sri Agustina, D. (2022). Membangun *Website* MTS Negeri 01 OKU Timur Menggunakan Php dan Mysql. In *JIK* (Vol. 13, Issue 1).

Muhamad Adillah Fatih, Reza Arif Maulana, Rizki Reza Pratama, Muhammad Darwis, & Retno Hendrowati. (2024). PENGEMBANGAN APLIKASI DKM REGISTRATION SYSTEM BERBASIS WEB MENGGUNAKAN METODE WATERFALL. *Jurnal Teknoif Teknik Informatika Institut Teknologi Padang*, *12*(1), 36–46. https://doi.org/10.21063/jtif.2024.v12.1.36-46

Nurbaiti, & Alfarisyi, M. F. (2023). *Sejarah Internet di Indonesia*.

Sinha, D. (2020). *How CI/CD Can Save App Development Time And Create Robust*. https://www.techaheadcorp.com/blog/how-ci-cd-save-app-development-time/

Thatikonda, V. K. (2023). Beyond the Buzz: A Journey Through *CI/CD* Principles and Best Practices. *European Journal of Theoretical and Applied Sciences*, *1*(5), 334–340. https://doi.org/10.59324/ejtas.2023.1(5).24

Toba, H., Gautama, T. K., Narabel, J., Widjaja, A., & Sujadi, S. F. (2022). JEPIN (Jurnal Edukasi dan Penelitian Informatika) Evaluasi Metodologi *CI/CD* untuk Pengembangan Perangkat Lunak dalam Perkuliahan. *Jurnal Edukasi Dan Penelitian Informatika*, *8*(2).

Wessel, M., Vargovich, J., Gerosa, M. A., & Treude, C. (2023). GitHub Actions: The Impact on the Pull Request Process. *Empirical Software Engineering*, *28*(6). https://doi.org/10.1007/s10664-023-10369-w

Zulhakim, Z., & Kurniawan, A. (2024). *Implementasi Continuous Integration Dan Continuous Deployment Pada Pengembangan Aplikasi Website Menggunakan Docker Dan Github Actions*. *16*(1).