ORKESTRASI CONTINUOUS INTEGRATION / CONTINUOUS DELIVERY (CI/CD) DAN AUTOMATED TESTING PADA DEVOPS MARKETPLACE TOKODISTRIBUTOR

Anggita Aprilia
Octanty Mulianingtyas

STIMIK AMIK Bandung
STIMIK AMIK Bandung

Anggitaaprilia2001@gmail.com
Octanty@stmik-amikbandung.ac.id

Abstrak

Pada fase *build*, *test*, dan deploy yang terjadi pada siklus pengembangan perangkat lunak seringkali memakan banyak waktu dan menyebabkan rilis tidak sesuai dengan jadwal. *Continuous Integration* (CI) dan *Continuous Delivery* (CD) diharapkan dapat memberi solusi masalah tersebut. dimana sistem ini akan otomatis diintegrasikan dengan beberapa layanan ke dalam aplikasi dalam pipeline CI/CD yang dijalankan secara otomatis jika ada *triggered event* yang memicu seperti pada *event push* pada repository. Penelitian ini merupakan praktik penerapan *DevOps* pada SDLC marketplace Tokodistributor yang merupakan perangkat lunak berbasis mobile web, dan sebelumnya sudah mengadopsi SDLC *Agile Scrum*. Dari penelitian ini dihasilkan bahwa *DevOps* dapat diimplementasikan pada pengembangan aplikasi tokodistributor dengan baik di mana alur kerja dapat terstandarisasi dengan baik dari bentuk alur kerja (pipeline) maupun penyetaraan environment. Penggabungan kode terjadi dengan mudah, build harian lancar dan pemeriksaan kelayakan kode terjadi setiap kali ada *commit* dan *push* dari pengembang aplikasi. Proses deployment menggunakan CI/CD mampu mempersingkat proses di angka 1-10 menit dan meningkatkan kinerja dengan memberikan hasil yang lebih teliti dengan penemuan bug pada pengujian adalah 85% sedangkan pada hasil pengujian UAT didapati sekitar 84,99%. DevOps telah memudahkan proses pengembangan perangkat lunak menggunakan *tools* yang digunakan

Kata kunci: *Continuous Integration, Continuous Delivery*, Devops, Orkestrasi CI/CD

Abstract

The build, test, and deploy phases that occur in the software development cycle often take a lot of time and cause releases not to be on schedule. Continuous Integration (CI) and Continuous Delivery (CD) are expected to provide a solution to this problem. where this system will be automatically integrated with several services into the application in a CI/CD pipeline which is executed automatically if there is a triggered event such as a push event in the repository. This research is a practice of implementing DevOps on the Tokodistributor marketplace SDLC, which is mobile web-based software, and previously adopted the Agile Scrum SDLC. From this research, it emerged that DevOps can be implemented in the development of distributor store applications well, where the workflow can be standardized both in terms of workflow (pipeline) and equalization of the environment. Code merging happens easily, daily builds are smooth and code feasibility checks occur every time there is a commit and push from the application developer. The deployment process using CI/CD is able to shorten the process by 1-10 minutes and improve performance by providing more thorough results with bug discovery in testing being 85% while in UAT testing results it is found to be around 84.99%. DevOps has made the software development process easier using the tools used

Keywords : *Continuous Integration, Continuous Delivery, Devops, CI/CD Orchestration*

PENDAHULUAN

Latar Belakang

Kemajuan teknologi dan komunikasi kini sudah semakin berkembang baik dan cepat, membuat berbagai bidang instusi bisnis mulai menggunakan teknologi, yang dapat di akses oleh user menggunakan aplikasi web maupun aplikasi berbasis android. Penelitian membuktikan, bahwa sekitar 86% orang dari berbagai bidang pekerjaan, setiap harinya menggunakan teknologi saat bekerja. Oleh karena itu kini para pengembang aplikasi berlomba - lomba dalam menyediakan alat-alat canggih yang dapat membantu mempermudah serta mempercepat semua pekerjaan manusia. Tak dapat dipungkiri bahwa dengan adanya sebuah aplikasi yang dapat memudahkan banyak pekerjaan dan memberikan keuntungan. Membuat bisnis dapat memberikan nilai tambah secara cepat melalui fitur-fitur aplikasi yang mereka tawarkan, dan pada akhirnya pembisnis memiliki tujuan untuk mempercepat proses pembuatan aplikasi tanpa mengorbankan kualitas serta fitur dari aplikasi tersebut. Perubahan yang terjadi sangat cepat dan drastis dalam sebuah bisnis, seringkali menuntut pengembang aplikasi untuk bekerja lebih cepat, dan sebagaian besar memberikan hasil yang jauh dari harapan. Hal ini di sebabkan karena pembangunan perangkat lunak masih berorientasi secara konvesional, sebagaimana penelitian dari *2ndWatch* menyatakan bahwa lebih dari 1000 profesional TIK, level manajer hingga direktur, sekitar 78% mengakui perusahaannya masih memisahkan antara tim infrastruktur atau operasional dengan tim pengembangan aplikasi. Dimana depatement IT masih berkerja sendiri - sendiri dan harus menunggu gilirannya masing-masing dalam mengerjakan suatu proyek.

PT Distributor Unggul Indonesia merupakan perusahaan yang bergerak di bidang *marketplace* yang menggunakan konsep B2B (Business to Business) dengan platfrom digital bernama Tokodistributor yang menggabungkan bisnis moderen dan tradisional dengan tujuan untuk memperpendek dan menyederhanakan jalur distribusi. Dimana *marketplace* ini menghubungkan pembeli, *reseller* dan *suplier* menjadi platform atau media yang tepat dan terpercaya untuk melakukan transaksi online baik sebagai pembeli, *reseller* dan *supplier*. Tokodistributor sendiri menyediakan berbagai kebutuhan sehari-hari dengan ratusan kategori mulai dari makanan, *fashion*, *lifestyle* dan masih banyak lagi. Berdasarkan observasi awal peneliti mengamati jalanya proses pembangunan perangkat lunak melalui wawancara yang telah dilakukan dengan tim pengembang pada proyek aplikasi *website reseller* tokodistributor. Proses pembangunan, pengembangan dan pemeliharaan aplikasi menggunakan siklus hidup pengembangan sistem *Agile* dengan kerangka kerja *Scrum*. Kerangka kerja Scrum terdapat Sprint dan tim yang akan bekerja di dalamnya. Setelah Sprint selesai, Demo selesai dan Definition of Done (DoD) terpenuhi sesuai dengan user story, lantas setiap tim akan mengintegrasikan kode mereka menggunakan alat manajemen konfigurasi untuk disimpan dalam sebuah repository, kode – kode terintegrasi yang telah di buat akan secara di build akan di kirim ke suatu lingkungan stage atau test untuk melakukan uji kelayakan pada software baik kode maupun server. Apabila hasil pengujian otomatis tersebut telah berhasil, maka perangakat lunak tersebut telah layak untuk di produksi namun jika hasil pengujian otomatis itu tidak berhasil maka kode – kode yang sudah di buat akan di kembalikan untuk di perbaiki dan di lakukan perubahan baik berupa pembaruan fitur, perbaikan bug, hingga perubahan konfigurasi. Perbaikan masalah ini yang akan memakan banyak waktu dan menyebabkan waktu rilis tidak sesuai jadwal. Masalah pada fase build dan deploy seperti kode yang tidak berfungsi dengan baik saat di implementasikan di lingkungan produksi sering terjadi pada lingkungan kerja Agile. Membuat divisi tester memiliki waktu terbatas untuk melakukan tes pada aplikasi, yang pengujianya hanya melihat UI user interface. Hal ini juga yang terjadi dalam pengembangan aplikasi.

Oleh karena itu peneliti ingin melakukan *penerapan Continues Intergrations /Continues Delevery* (CI/CD) dan *Automated Testing* Pada *Devops* aplikasi *mobile web resseler* tokodistributor untuk merubah cara kerja metode lama dimana salah satu departemen masih menggunakan metode kerja manual, yang nantinya tiap department akan memiliki tools yang digunakan dan bekerja sama dalam mengembangkan perangkat lunak sehingga, jika terdapat suatu perubahan situasi, perusahaan bisa lebih cepat merespon dan membentuk strategi dalam mengatasi perubahan yang ada. *Devops* sendiri dapat mengatur alur *souce code* dalam hal pengembangan dan operasional melalui *code, plan, deploy*, dengan berbagai *tools* yang dapat membantu mempermudah dan mengoptimalkan pengembangan aplikasi dan diharapkan *Devops* dapat menjadi solusi hal tersebut.

Identifikasi Masalah

Berdasarkan uraian latar belakang masalah, peneliti menyimpulkan permasalahan sebagai berikut:

1) Dalam Pengembangan perangkat lunak, koding hanyalah tahapan awal, setelah koding, masih banyak hal yang biasanya di lakukan oleh tim pengembang perangkat lunak sebelum aplikasi yang di buat sampai ke *production server*, seperti menjalankan unit tes, dan melakukan pengecekan menggunakan *otomations testing*.
2) Fase *Build* dan *deploy* aplikasi memakan banyak waktu, jika kode tidak berhasil melewati uji kelayakan, akan terjadi *erorr* sehingga menyebabkan rilis tidak sesuai jadwal.

3) Proses *Development* membutuhkan waktu yang lama jika ada salah satu divisi departemen IT masih melakukan budaya kerja manual.
4) Hal – hal menjadi begitu komplek jika pengujian aplikasi masih di lakukan secara manual terus menerus, resikonya adalah sering terjadi kesalahan dan tidak konsiten,
5) Tester memiliki waktu sedikit untuk mengecek aplikasi

Rumusan Masalah

Berdasarkan uraian identifikasi masalah, penelitian menyimpulkan beberapa batasan masalah antara lain

1. Bagaimana proses *development* aplikasi agar tidak memakan banyak waktu, dan membuat *tester* memiliki waktu melakukan testing aplikasi?
2. Bagaimana metode *Devops* digunakan untuk pengembangan apliskasi perangkat lunak?
3. Bagaimana *Devops* di terapkan dan bagaimana cara kerjanya untuk menyelesaikan masalah?
4. Bagaimana melakukan pengujian kualitas aplikasi menggunakan *software tools automated* untuk mengantikan aktivitas tester dalam pengujian aplikasi manual dan dampak untuk kode yang di uji.

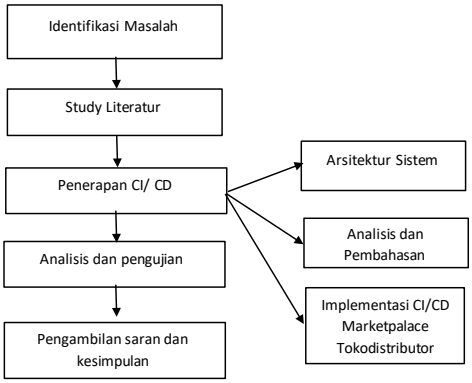
Tujuan Penelitian

Dari uraian rumusan masalah, peneliti menyimpulkan tujuan dari penelitian sebagai berikut

1. Mempermudah tim kerja *Developer* dan *operations,*
2. *Developer* dan *Quality assurance* dapat saling memberikan informasi terkait software yang sedang di kembangkan
3. Mengurangi *erorr* dan mempercepat proses saat pengecekan kode uji kelayakan CI/CD
4. Meningkatkan ketelitian pencarian bug pada *software* aplikasi yang akan di lakukan testing oleh *tester*.
5. Mengurangi kelelahan pada tim IT Departemen

METODE

menjelaskan langkah-langkah yang dilakukan dalam pembuatan sistem, yaitu kontek penerapan CI CD, perangkat pengolahan CI CD. Berikut ini merupakan diagram alir tahap pengerjaan penelitian ini



Gambar 1. Diagram Alir Tahap Pengerjaan

Identifikasi Masalah

Identifikasi masalah adalah tahap krusial untuk merumuskan fokus penelitian, pada masalah yang timbul dalam pembangunan dan pengembangan perangkat lunak sehingga di gunakan sistem Continuous Integration (CI) dan Continuous Delivery (CD) Devops. Pernyataan di atas menjadi penuntun yang digunakan dalam riset ini sebagai penjabaran dari rumusan masalah yang telah dituliskan dalam bagian terdahulu. Berpijak pada pernyataan itu pula, diusulkan sebuah tools yang di gunakan untuk sebuah sistem CI/CD dengan aspek pengelolaan proyek sebagaimana yang digunakan.

Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan sebagai penunjang dan pendukung penulisan skripsi. Teori penunjang dan pendukung skripsi ini meliputi:

a. Devops
b. Jira *Software*
c. Slack
d. Aplikasi Web
e. *Source code*
f. Gitlab
g. *Webhook*
h. Kubernetes
i. Jenkins
j.Docker
k. Keel
l. Katalon studio
m. ECR
n. EKS

Penerapan Sistem CI / CD

Penerapan sistem CI/CD pada skripsi ini digunakan untuk pengembangan, pengelolaan perangkat, pengujian yang terintegrasi, yaitu pengatur orkestrasi yang melibatkan berbagai kelompok tools dan menggabungkannya untuk mencapai tujuan otomatisasi proses pengembangan perangkat lunak, berikut tahap penerapan sistem CI / CD.

1. Arsitektur sistem

Integrasi CI/CD menjelaskan secara detail fitur setiap tools yang digunakan dan bagaimana data pemrosesan, dari setiap tools tersebut dapat diambil. Pada aplikasi yang digunakan dan akan dilakukan uji coba. Namun tahap yang pertama kali dilakukan sebelum melakukan hal tersebut adalah, eksplorasi CI/ CD yang menjelaskan analisis fitur dan instalasi berbagai tools CI/CD, tahap pertama akan dilakukan survei dan analisis fitur dan kemampuan berbagai tools CI/CD yang tersedia. Setelah itu akan dipilih tools yang memiliki fitur yang cukup lengkap dan dapat dicoba terlebih dahulu (*free trial*) atau memiliki biaya lisensi yang terjangkau. Salah satu kriteria penting pemilihan, selain biaya juga harus memiliki arsitektur yang mudah dan efektif untuk membagun proyek aplikasi dengan metodologi devops sehingga dapat dilakukan analisis lanjutan. Selanjutnya setelah itu akan dilakukan analisis pembahasan dari setiap pengguna dan penjabaran orkestrasi integrasi dari setiap tool untuk dipelajari dan di lakukan implementasi pada aplikasi yang akan di buat.

2. Analisis dan Pembahasan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan dari sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan *(requirements)* sistem dan siapa saja yang terlibat di dalamnya. Metode analisis yang digunakan adalah (SDLC) Sistem *Development Life Cicyle* dengan menggunakan metode *Devops* dengan menggunakan Bahasa pemodelan UML (*Unified Modeling Laguage*). Analisis kebutuhan dilakukan dengan mengidentifikasi semua kebutuhan (*requirements*) sistem kemudian akan dimodelkan dalam *Activity diagram* dan *Use case diagram.*

3. Implementasi CI/ CD

Melakukan Implementasi CI/CD terhadap aplikasi yang akan di lakukan uji coba dengan tools yang sudah di pelajari dan di lakukan analisis, kemudian akan diperoleh hasil penggunaan CI/CD berupa umpan balik melalui testing yang memberikan gambaran terhadap potensi dan kualitas proses pengembangan perangkat lunak dengan CI/CD yang dilakukan oleh tim pengembang, khususnya dalam konteks proyek marketplace tokodistributor.

Pengujian dan Analisis

Pengujian sistem pengembangan perangkat lunak pada skripsi ini dilakukan agar dapat menunjukkan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang telah dianalisis sebelumnya. Strategi pengujian sistem yang akan digunakan yaitu pengujian *Time based metric*, Pengujian *Quality based metric*, dan pengujian UAT (*User Acceptance Test*) pada tahap akhir. Metode pengujian yang di lakukan pada *Quality based metric* yaitu dengan memperhatikan kualitas dari sistem yang di lakukan. Selanjutnya *Time based metric* Proses dan waktu yang di butuhkan oleh proses sistem dalam melakukan keseluruhan tahapan proses, dan yang terakhir Pengujian UAT dilakukan untuk melihat tingkat penerimaan pengguna terhadap sistem dengan variabel penilaian antara lain, yaitu penerimaan pengguna berdasarkan kemudahan penggunaan sistem (*perceived ease of use*) dan penerimaan pengguna berdasarkan pencapaian kegunaan (*perceived usefulness*). Analisis dilakukan dari hasil pengujian sistem sehingga dapat diambil kesimpulan dari pengembangan sistem yang telah dilakukan.

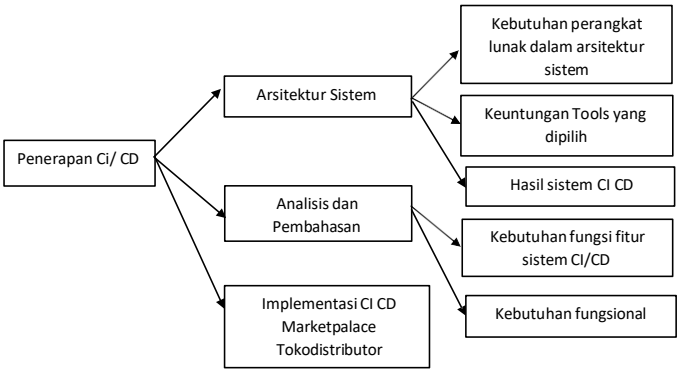
Pengambilan Kesimpulan dan Saran

Pengambilan kesimpulan dilakukan setelah semua tahapan perancangan sistem pengembangan perangkat lunak, implementasi sistem pengembangan perangkat lunak, dan pengujiansistem pengembangan perangkat lunak telah diselesaikan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem pengembangan aplikasi yang dibangun. Tahap terakhir dari penulisan adalah saran yang dimaksudkan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta sebagai acuan untuk pengembangan perangkat lunak selanjutnya.

HASIL DAN PEMBAHASAN

Penerapan CI / CD

Pada proses perancangan aplikasi meliputi 3 (tiga) tahapan, yaitu tahap pertama adalah eksplorasi perangkat lunak, tahap kedua adalah perancangan interaksi tools, dan yang terakhir tahap adalah evaluasi potensi pemanfaatan CI/CD, Pohon perancangan sistem *Devops* dapat dilihat dalam pohon Tahap perancangan umum sistem menjelaskan mengenai gambar proses kerja sistem secara umum.



Gambar 2. Penerapan CI/CD

Arsitektur Sistem

Berikut adalah arsitektur Sistem CI/CD:

1. Kebutuhan perangkat lunak dalam arsitektur sistem

Mengeksplorasi perangkat lunak Continuous Integration (CI) dan Continuous Deployment/Delivery (CD) adalah langkah penting untuk memahami fungsionalitas, antarmuka, dan kemampuan yang ditawarkan oleh berbagai perangkat tools. Sistem Continuous Integration sendiri (CI) dan Continuous Deployment (CD) umumnya adalah orkestrasi yang melibatkan banyak tools dan menggabungkannya untuk mencapai tujuan otomatisasi proses pengembangan perangkat lunak. [5] Berbagai kelompok tools beserta contoh-contohnya diantaranya adalah:

1. Source code control merupakan tools untuk mengatur penggabungan sebuah kode sumber secara terpusat semua pada programmer yang terlibat. Hal ini berfungsi sebagai backup kode sumber, pengatur jika terjadi bentrokan antar kode yang dikirim programmer yang berbeda, dan pengaturan versi kode secara otomatis. Contoh tools version control: Git
2. Server cloud penyedia layanan version control, merupakan layanan di internet yang menyediakan server atau lingkungan cloud untuk pengaturan kode sumber. Contohnya adalah: Github, Gitlab, dan Bitbucket.
3. Perangkat pembangun kode sumber menjadi kode tujuan sesuai bahasa program dan framework yang digunakan. Contohnya: Java Script, PHP , C++ compiler, Go
4. Perangkat untuk pengujian perangkat lunak, yang dapat dibagi lagi menjadi pengujian sebuah fungsi atomik (unit testing), pengujian penggabungan berbagai komponen perangkat lunak (integration test), dan pengujian antarmuka perangkat lunak (user interface/user acceptance test). Contohnya adalah: JUnit, NUnit, TestNG, Selenium, dan Mockito, Sonarqube
5. Perangkat untuk melakukan deployment/instalasi perangkat lunak. Perangkat untuk sistem CI sendiri cukup banyak, contohnya: TeamCity, Jenkins, Circle CI, Gitlab CI, Team Foundation Server, Travis CI, Go CD, Bamboo, CodeShip, Buddy Build, dan AWS Code Pipeline.

Orkestrasi sistem CI/CD umumnya mengikuti model pipeline, yaitu output dari sebuah tools akan menjadi input untuk tools berikutnya. Oleh karena itu diperlukan adanya riset khusus untuk mengkombinasikan tools yang cocok berdasarkan metrik pengukuran yang diperlukan.

2. Keuntungan Tools yang dipilih

Penggunaan sistem CI/CD sudah merupakan best practice dalam industri perangkat lunak modern, dan hal ini membuka peluang untuk memaksimalkan pemanfaatannya. Secara umum sebuah sistem CI/CD yang digunakan oleh suatu tim pengembang perangkat lunak akan menghasilkan banyak data dari pemrosesannya, Perangkat (tools) untuk memproses Continuous Integration (CI) dan Continuous Delivery (CD) sangat penting untuk mengotomatisasi dan mengelola alur kerja pengembangan perangkat lunak. Pilihan perangkat CI/CD akan tergantung pada kebutuhan, preferensi, dan infrastruktur yang digunakan oleh tim pengembangan. Beberapa tim mungkin juga memilih untuk membuat solusi CI/CD yang sesuai dengan kebutuhan project menggunakan tools sebagai panduan. Berikut adalah Analisis fitur dan instalasi berbagai tools CI/CD yang diperlukan untuk pengembangan perangkat lunak, dilaksanakan untuk perangkat berikut ini:

a. Jenkins : Jenkins adalah alat otomatisasi open-source yang digunakan untuk memfasilitasi dan mendukung proses Continuous Integration (CI) dan Continuous Deployment (CD) dalam pengembangan perangkat lunak. Namun pada sistem ini Jenkins ditugaskan untuk memantau repositori kode, memicu build setiap kali ada perubahan kode, melakukan integrasi continue, dan melakukan otomatisasi berbagai langkah dalam proses pengembangan. Jenkins juga dapat melakukan pengujian otomatis dan memberikan laporan tentang hasil build, pengujian, dan lintasan perubahan kode.

b. Sonarqube: SonarQube adalah platform analisis kode sumber open-source yang digunakan untuk mengukur dan meningkatkan kualitas perangkat lunak. Tools Jenkins akan menjalankan pengujian otomatis atau code riview Sonarqube dan memproses data yang akan di hasilkan oleh tools sonarqube, berikut data yang Diproses oleh Sonarqube adalah:

1. Kode Sumber: Kode yang diambil dari repositori gitlab yang akan dianalisis.
2. Informasi Proyek: Data proyek yang diperlukan untuk menganalisis dan memahami konteks proyek yang di buat.
3. Hasil Pembangunan: Hasil dari proses pembangunan (build) yang mencakup informasi tentang pengujian.
   Berikut hasil Analisis yang di jalankan oleh Sonarqube adalah:
   a. Metrik Kualitas Kode: indeks kualitas kode dan kepatuhan terhadap standar pemrograman.
   b. Laporan Kualitas Kode: Rangkuman hasil analisis yang mencakup penilaian tentang keamanan, keandalan, kinerja, dan perawatan kode.
4. Rekomendasi Perbaikan: Saran untuk memperbaiki kode berdasarkan standar dan praktik terbaik.

c. Docker: Docker adalah platform perangkat lunak yang digunakan untuk mengotomatisasi penyebaran dan pengelolaan aplikasi dalam lingkungan yang terisolasi yang disebut sebagai kontainer. Docker akan memproses data yang sudah di lakukan pengujian otomatis lalu menghasilkan penyebaran aplikasi, berikut data yang Diproses oleh Docker adalah:

1. Kode Sumber Aplikasi: Kode aplikasi yang akan diintegrasikan, dibangun, dan diimpor ke dalam kontainer Docker.
2. Konfigurasi Dockerfile: Instruksi untuk membangun kontainer Docker yang sesuai dengan kebutuhan aplikasi.
3. Parameter Konfigurasi: Data konfigurasi yang diperlukan untuk mengelola dan mengintegrasikan aplikasi ke dalam kontainer Docker.
4. Hasil Build: Hasil dari proses membangun aplikasi dan kontainer Docker, termasuk laporan kegagalan atau keberhasilan.
   Berikut hasil Hasil Penyebaran Aplikasi adalah:
   a. Kontainer Docker: Jenkins akan membantu membangun, menjalankan, dan memantau kontainer Docker yang memuat aplikasi atau layanan yang ingin diimplementasikan.
   b. Pengelolaan Siklus Hidup Kontainer: Jenkins dapat memastikan bahwa kontainer Docker dimulai, dihentikan, diperbarui, atau dihapus sesuai dengan kebutuhan implementasi dan manajemen aplikasi.
   Jika kode yang telah di uji Sonarqube berada di atas standar dari Sonarqube, maka Jenkins akan memerintahkan untuk menjalankan build image codingan yang dibuat menggunakan Docker

d. ECR : Amazon Elastic Container Registry (ECR) adalah layanan manajemen kontainer yang disediakan oleh Amazon Web Services (AWS). ECR akan memproses data yang berasal dari docker lalu menghasilkan menejemen container, berikut data yang diproses oleh ECR adalah:

1. Kontainer Docker: Kontainer yang akan diunggah ke AWS ECR untuk disimpan dan dikelola.
2. Metadata Kontainer: Informasi tentang kontainer, termasuk tag, deskripsi, dan versi.
3. Izin Akses dan Keamanan: Pengaturan izin akses, kunci API, dan konfigurasi keamanan untuk memastikan kontainer hanya dapat diakses oleh entitas yang sah.

Berikut hasil Manajemen Kontainer adalah:

a. Penyimpanan Kontainer: Kontainer Docker yang disimpan dan dikelola di AWS ECR untuk digunakan dalam pengembangan dan penyebaran aplikasi.

b. Manajemen Repositori: Repositori kontainer Docker yang dapat diorganisasi, dikelola, dan diakses sesuai dengan kebutuhan.

e. KEEL: Keel adalah alat (tool) open-source yang digunakan untuk otomatisasi proses Continous Deployment (CD) dalam pengembangan perangkat lunak. KEEL akan memproses data lalu menghasilkan penyebaran oleh Keel, berikut data yang di proses KEEl adalah:

1. Manifests Kubernetes: Konfigurasi aplikasi dalam bentuk manifest Kubernetes, seperti file YAML yang mendefinisikan objek seperti Pods, Services, Deployments.
2. Metadata Aplikasi: Informasi tambahan terkait aplikasi seperti versi, deskripsi, dan label yang diperlukan untuk memantau dan mengelola penyebaran.

   Berikut hasil Penyebaran oleh Keel :

   a. Penyebaran Aplikasi: Jika perubahan memenuhi kriteria, Keel akan memulai penyebaran otomatis ke AWS EKS.

   b. Monitoring dan Log: Keel dapat memantau proses penyebaran dan menyediakan log dan informasi terkait status penyebaran.

   c. Notifikasi dan Pembaruan: Keel dapat memberikan pemberitahuan kepada tim pengembangan dan operasi mengenai penyebaran yang berhasil atau jika terjadi kesalahan.

f. EKS: Amazon Elastic Kubernetes Service (Amazon EKS) adalah layanan manajemen Kubernetes yang disediakan oleh Amazon Web Services (AWS). EKS akan memproses data lalu menghasilkan menejemen siklus hidup aplikasi, berikut data yang diproses oleh EKS adalah:

1. Kode Sumber Aplikasi: Kode aplikasi yang akan diintegrasikan, dibangun, dan diimpor ke dalam klaster Kubernetes.
2. Konfigurasi Kubernetes: Konfigurasi Kubernetes yang diperlukan untuk menentukan cara aplikasi akan dijalankan dan dielola.
3. Hasil Build dan Deploy: Hasil dari proses membangun aplikasi dan penyebaran ke klaster Kubernetes, termasuk laporan kegagalan atau keberhasilan.

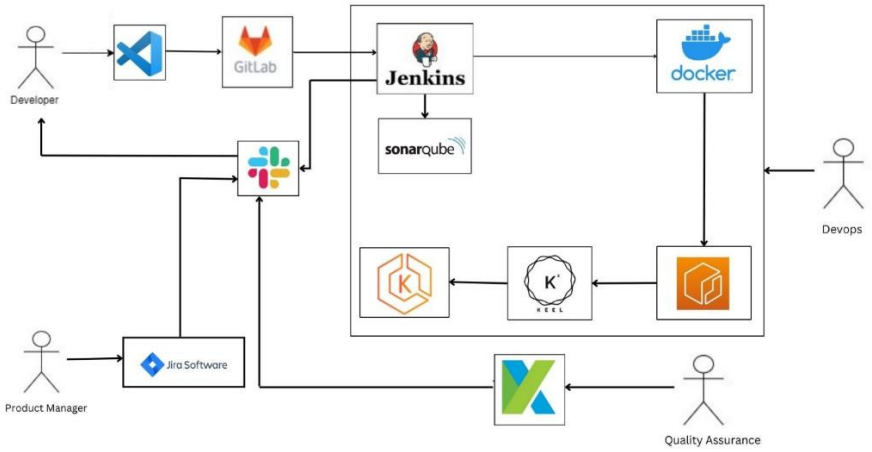
Berikut Hasil Manajemen Siklus Hidup Aplikasi adalah:

a. Pengembangan Aplikasi: Proses pembangunan dan pengujian aplikasi menggunakan Jenkins.

b. Penyebaran Aplikasi: Penyebaran aplikasi ke klaster Kubernetes menggunakan Keel dan EKS.

3. Hasil sistem CI CD

Perancangan interaksi antara alat-alat (tools) dalam konteks Continuous Integration/Continuous Deployment (CI/CD) dan DevOps adalah langkah penting untuk memastikan alur kerja pengembangan perangkat lunak berjalan lancar dan efisien. Berikut perancangan umum sistem digunakan sebagai representasi arsitektur sistem yang dibuat secara umum. Gambar 3.2 dibawah ini menunjukkan integrasi tools sistem.



Gambar 3 integrasi Tools Sistem

Product manager akan membuat tiket atau antrian pekerjaan bagi setiap tim pada Jira yang merupakan alat menejemen proyek, dengan metodelogi Agile menggunakan metode kerja Scrum atau Kaban, dimana setiap tim akan berkaloborasi secara efektif untuk menemukan masalah dan terus bekerja pada solusi yang lebih baik, Selanjutnya jira mengirimkan pesan notifikasi ke channel di dalam Slack, Setelah Tim perdivisi mendapatkan pekerjaannya masing – masing. Developer melakukan push ke dalam repository di Gitlab, proses push tersebut akan ditangkap oleh webhook untuk melakukan trigger otomatisasi di dalam Jenkins Continues integrations. Jenkins akan menjalankan code review Sonarqube jika code tersebut berada di atas standar dari Sonarqube, maka Jenkins akan memerintahkan untuk menjalankan build image codingan yang dibuat menggunakan Docker. Setelah

image telah dibuat Jenkins menyuruh docker untuk tag dan push image tersebut pada AWS ECR (Elastic Container Registry), selanjutnya Jenkins mengirim trigger ke webhook Keel, sehingga Keel sebagai tools Continues Deployment menyiapkan manifest deployment dan di deploy ke AWS EKS (Elastic Kubernetes Service) Ketika seluruh proses berhasil, maka Jenkins akan menjalankan trigger untuk mengirim pesan notifikasi ke channel di dalam Slack. Selanjutnya Quality Assurance akan melakukan testing berdasarkan document tes case yang sudah di buat dan Katalon akan merekam dan mengirimkan notifikasi ke Slack.

4. Pengujian

Pengujian Time based metric

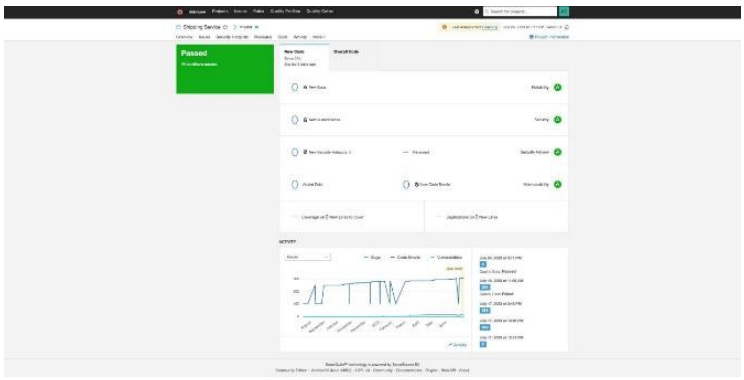
Proses analisis terhadap hasil pengujian Time based metric dilakukan dengan melihat waktu yang dibutuhkan CI &CD dalam melakukan seluruh tahapan pipeline. Berdasarkan hasil pengujian menggunakan proses CI/CD, didapati bahwa rata-rata total build time bersifat stagnan di angka 1 sampai 10 menit Hal ini terjadi tergantung bahasa pemograman yang di gunakan dan pembuatan code pada aplikasi yang di bangun, hasil pengujian ini jauh lebih baik perbandinganya, dibandingkan dengan proses CI/CD tanpa menggunakan tools katalon studio terdapat beberapa faktor yang mempengaruhi lamanya durasi waktu, seperti manual testing, dan terjadinya human error seperti kurang telitinya Quality assurance tim pengembang saat mengecek code ataupun salah dalam mengunggah file pada server.

Gambar 4. Pengujian Time based metric

Pengujian Quality based metric

Proses analisis terhadap hasil pengujian Quality based metric dilakukan dengan melihat mengevaluasi kualitas suatu produk, layanan, atau proses. Berdasarkan pengujian keberhasilan dengan menggunakan proses CI/CD adalah 85%. Hasil ini didapatkan dari 2 kali percobaan yang ada , terdapat 1 percobaan yang berhasil dan 1 percobaan yang gagal karena tidak lolos quality gate dari Sonarqube. Hasil tidak lolos dari quality gate ini akan langsung memberitahu pengembang untuk segera memperbaiki codenya dari bug yang ada, melalui slack. Hasil yang di dapatkan dari 2 kali percobaan yang ada, terdapat 1 yang gagal karena sonarqube mendeteksi bug dan security dan 1 percobaan lainnya mendapatkan hasil sukses deployment passed Berdasarkan observasi ketika menggunakan proses CI/CD, bug yang terdapat pada aplikasi secara langsung dapat ditangkap pada proses code review di dalam pipeline, hal ini merupakan keunggulan dari proses CI/CD dikarenakan bug dapat dideteksi sebelum masuk ke dalam aplikasi yang dirilis, maka hal tersebut akan dapat langsung di benarkan oleh pihak pengembang Developer. Hasil temuan bug dari proses CI/CD berjumlah 1 yang ditemukan pada percobaan pertama. Untuk sisa 1 percobaan sudah tidak ditemukan bug lagi dan proses CI/CD berjalan dengan sukses.

Sedangkan pada pengujian katalon studio keberhasilan melakukan pengujian adalah 90% terdapat 4 kali percobaan yang berhasil. Hasil ini akan langsung di report pada csv atau pdf namun jika terdapat bug maka katalon studio akan langsung memberitahu pengembang untuk segera memperbaiki codenya dari bug yang ada, melalui slack. Hasil yang di dapatkan dari 4 kali percobaan yang ada, semuanya lolos dari bug passed Berdasarkan observasi ketika menggunakan proses pengujian aplikasi secara langsung dapat ditangkap memperlihatkan code program aplikasi saat melakukan pengujian, hal ini merupakan keunggulan dari pengujian katalon studio karena dapat memberitahu tim pengembang developer code yang terdapat bug sehingga masalah dapat cepat terselesaikan.



Gambar 4. Pengujian Quality Based Metric

Pengujian *UAT*

Proses analisis terhadap hasil pengujian *UAT* dilakukan dengan menggunakan skala *Likert*. Interpretasi skor *Likert* atau persentase dari setiap skor *Likert* ditunjukkan pada Tabel 5.4 Untuk hasil analisis pengujian UAT Perceived Ease of Use dan UAT Perceived Usefullness ditunjukkan pada Tabel 5.5

Tabel 1. Skor Likert

Skor Likert	Interpretasi skor dengan interval =20	Pilihan
1	0% - 19.99&	Sangat tidak memuaskan
2	20% - 39.99%	Tidak Memuaskan
3.	40% -59.99%	Biasa
4.	60% -79.99%	Memuaskan
5.	80% -100%	Sangat Memuaskan

Keterangan:
Interval = 20 didapatkan dari pembagian nilai 100 dengan jumlah skor Likert

Tabel 2. Analisis Hasil Pengujian UAT

No	Pernyataan	STS	TS	N	S	SS	Total Skor	Index
1.	Saya merasa lebih mudah ketika menggunakan sistem ini			1	3	2	25	83,33
2.	Saya merasa sistem ini memberikan respon kinerja yang lebih cepat, ketika saya operasikan		1		2	3	25	83,33
3.	Saya merasa tools pengembangan perangkat lunak yang di sarankan pada sistem ini mudah bagi saya			1	3	2	25	83,33
4.	Saya merasa tools pengembangan perangkat lunak ini efektif dalam menjalankan project aplikasi yang sedang di bangun ataupun di rancang			2	2	2	24	80
5.	Saya suka dengan pilihan tools yang di gunakan dalam sistem ini			1	2	3	26	86,66
6.	Saya merasa interaksi dan informasi pada tools yang di gunakan dalam aplikasi mudah di mengerti				2	4	28	93,33
7.	Saya merasa lebih mudah untuk mempelajari setiap tools yang di sarankan dalam sistem ini			1	2	3	26	86,66
8.	Tools dalam sistem ini sangat berguna bagi pengembangan perangkat lunak			1	2	3	26	86,66
9.	Tools dalam sistem ini memudahkan saya dalam melakukan pekerjaan saya			1	2	3	26	86,66

10.	Tools dalam sistem ini membatu mempercepat pekerjaan saya		1	1		4	25	83,33
11.	Tools dalam sistem ini fiturnya mudah di mengerti dan mudah di gunakan			1	3	2	25	83,33
12.	Tools dalam sistem ini meningkatkan kepercayaan tim dan diri saya sendiri dalam mengerjakan tugas saya			1	3	2	25	83,33

KESIMPULAN

Berdasarkan hasil analisis, perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Dengan implementasi CI/CD alur kerja dapat distandarisasi, baik standarisasi dari bentuk alur kerja (pipeline) maupun penyetaraan environment. Sehingga hal ini dapat menghilangkan adanya dependensi antara tim pengembang dan operasi.
2. Penggabungan kode terjadi dengan mudah, build harian lancar dan pemeriksaan kelayakan kode terjadi setiap kali ada commit dan push dari pengembang aplikasi
3. Proses deployment menggunakan CI/CD mampu mempersingkat proses dan meningkatkan kinerja. Selain itu CI/CD memberikan hasil yang lebih teliti dengan penemuan bug pada pengujian tersebut.

DAFTAR PUSTAKA

M. J. Gulo and P. Simanjutak, "Jurnal Comasie," ANALISIS PENERAPAN METODE DEVOPS PADA APLIKASI RESTORAN BERBASIS ANDROID, vol. VOL. 04, p. 2, 2021.

DevOps, Insight, "i-3 Inovasi Informatika Indonesia," 6 February 2019. [Online]. Available: https://i-3.co.id/siapkah-perusahaan-anda-menerapkan-devops/.

D. Fardian, "glair.ai," 16 February 2022. [Online]. Available: https://glair.ai/blog-posts-id/pentingnya- devops-dalam-menghadapi-transformasi-digital.

Tohirin, S. F. Utam, S. R. Widianto and W. A. Mauludyansah, "JURNAL MULTINETICS," Implementasi DevOps pada Pengembangan Aplikasi e-Skrining Covid-19, vol. VOL.6, p. 2, May 2020.

H. T. T. k. j. n. a. W. S. F. Sujadi, "Evaluasi Metodologi CI/CD untuk Pengembangan Perangkat Lunak dalam Perkuliahan," Jurnal Edukasi dan Penelitian Innformatika (JEPIN), vol. 8, 2022.