PERANCANGAN SISTEM OTOMATISASI *CONTINUOUS DEPLOYMENT* BERBASIS *MONOLITHIC REPOSITORY*

Fikri Anggawie[1], Dana Sulistyo Kusumo[2], Donny Richasdy[3]
[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
fikrianggawie@students.telkomuniversity.ac.id[1], danakusumo@telkomuniversity.ac.id[2] ,donnir@telkomuniversity.ac.id[3]

Abstrak

Otomatisasi adalah hal yang sudah seharusnya diterapkan sebuah perusahaan yang mengerjakan proyek besar dengan tim yang terdiri dari dua orang atau lebih dalam pengembangan aplikasi. Tanpa adanya otomatisasi dalam melakukan perubahan *code* meskipun berskala kecil akan memakan banyak waktu dan biaya. Otomatisasi menggunakan metode *continuous deployment* dan aplikasi *jenkins, docker* dan menerapkan prinsip *monolithic repository* akan membantu dalam penerapan otomatisasi. Penelitian ini berfokus pada pengembangan dan integrasi menggunakan metode *continuous deployment* dan *monolithic repository,* dalam proses pengerjaan menggunakan *tools jenkins* dan *docker*. Penerapan *metode continuous deployment* dan *monolithic repository* menggunakan *jenkins* mempermudah melakukan integrasi, ini karena *jenkins* bersifat *open source* dan memiliki banyak opsi *plug-in. Docker* berperan sebagai *container* untuk meletakan semua kebutuhan integrasi aplikasi dan *deployment*. Keberadaan sistem otomatisasi adalah bentuk penghematan sumber daya waktu dan penggunaan yang lebih mudah. Proses integrasi dan penerapan menggunakan *jenkins* dan *docker* dapat menghemat banyak waktu dan jika ada perubahan *code* tidak memerlukan integrasi ulang serta dapat diubah dengan mudah. Aktivitas penelitian dilakukan dengan *deploy* proses otomatisasi, pemantauan proses otomatisasi melalui *prepare, build,* dan *deployment,* dan selanjutnya proses pengujian yang merupakan proses akhir deployment. Hasil pengujian menunjukan *framework* dengan *ansible* lebih baik dalam waktu proses *deployment* dibandingkan *monolithic* pada skenario pengujian yang sama

Kata kunci: c*ontinuous deployment, monolithic repository*

Abstract

Automation is something that should be implemented by a company working on a large project consisting of two or more people developing their application. Without automation, making code changes even though it is small will take a lot of time and money, of course. Automation using the continuous deployment method and the Jenkins, Docker application and applying the monolithic repository principle will help a lot in the application of automation. This research focuses on development and integration using the continuous deployment method and monolithic repository as well as using Jenkins and Docker tools in the process. The application of the continuous deployment method and monolithic repository uses Jenkins to facilitate integration because Jenkins is open source and has many plug-in options. docker itself acts as a container to place all application integration and deployment needs. The existence of an automation system is a form of saving resources such as time and easier use. So integration and deployment using Jenkins and docker we can save a lot of time and also if there are changes to the code we don't need re-integration and can be changed easily.

Keywords: continuous deployment, monolithic repository

I. PENDAHULUAN

Aplikasi harus secara cepat melakukan peningkatan pengembangan produk, hal ini karena adanya kebutuhan pembaruan berdasarkan permintaan pasar dan untuk kepentingan bisnis. Keberadaan otomatisasi membantu *deployment* sebagaimana disampaikan oleh Mysari, *deployment* tanpa otomatisasi akan memakan banyak waktu untuk melakukan perubahan pada *code* yang akan diubah dan melakukan deploy kembali *code* tersebut [1]. Pengembangan produk dilakukan dengan menggunakan *continuous deployment* (CD) yang berperan sebagai metode untuk memastikan bahwa perangkat lunak dapat dirilis kapan saja dengan mengotomatiskan *deployment* dan rilis alur kerja [2]. Tim perangkat lunak perusahaan besar seperti mozilla, google, microsoft telah mengadopsi CD sebagai bentuk peningkatan *development* untuk mempercepat pembangunan dan meningkatkan kualitas perangkat lunak [2].

Proses *deployment* yang menerapkan *monolithic repository* akan lebih mempermudah dalam banyak aspek. Beberapa aspek tersebut adalah menyederhanakan *management* ketergantungan dari *repository* lain dengan cara melakukan penyimpanan secara terpusat, selain menyederhanakan ketergantungan antar *repository* akan sulit apabila merilis versi baru tanpa menyebabkan kerusakan pada bagian lain karena semua penelponnya harus diperbarui pada saat yang bersamaan [1]. *Jenkins* dalam otomatisasi berperan sebagai pendukung proses integrasi dalam membangun *auto deployment* dan *docker* berperan sebagai *container* yang akan berisikan semua kebutuhan otomatisasi seperti *jenkins* dan perangkat lunak itu sendiri.

Dalam penelitian sebelumnya, proses otomatisasi dengan menggunakan metode *continuous deployment* dan *tools jenkins ansible* menerapkan otomatisasi waktu penyebaran dalam aplikasi tunggal, berkurang secara signifikan yang akan meningkatkan pemeliharaan aplikasi, selain itu otomatisasi berguna ketika aplikasi melakukan perubahan *code* pada *repository*, akan mengurangi waktu dalam peninjauan *code* dan dapat melakukan perubahan *code* secara berulang untuk mengubah aplikasi [1].

Pada penelitian lain disampaikan tentang keuntungan dan kerugian dalam menerapkan *monolithic repository (monorepo)*. Penelitian ini melakukan survei kepada *developers* yang telah melakukan penerapan *monorepo* atau *multi-repo* [3]. Hasil dari survei dapat disimpulkan mayoritas *developers* lebih menyukai *monorepo* dikarenakan kemudahan dan tidak memiliki ketergantungan pada *code-base* lain, akan tetapi pada penerapannya *monorepo* tidak disarankan dalam penggunaan berskala besar karena begitu banyak *code-base* yang akan disimpan dalam satu tempat [3]. Beberapa *developers* lain menyarankan pula apabila penerapan *monorepo* disandingkan dengan manajemen yang baik maka *code* dalam jumlah besar bukan menjadi masalah yang berarti [3].

Sistem otomatisasi yang menerapkan metode *continuous deployment* dan penerapan *monolithic repository* akan banyak memotong waktu dalam melakukan *deploy code* pada *server*. Kebutuhan otomatisasi akan menjadi penting karena banyak keuntungan yang akan di dapatkan selain memangkas waktu dalam proses *deployment*.

Berdasarkan latar belakang tersebut, maka dalam penelitian yang akan dilakukan adalah penerapan proses *deployment* secara otomatis dengan mengadaptasi penerapan *monolithic repository*. Penerapan ini menawarkan konfigurasi manajemen yang lebih baik dari cara yang digunakan selama ini yaitu proses *deployment* dengan tidak diotomatisasi, cara tersebut akan banyak memakan waktu lama dan menimbulkan potensi kesalahan pada proses *deployment* dari sisi *developers* dan sistem seperti muncul *bug* atau gagal *deployment* karena ketidakcocokan pada sistem.

Proses *deployment* sendiri mencakup bagaimana sebuah *code* akan di *deploy* ke *production* dengan cara otomatis, sedangkan *monolithic repository* berperan dalam manajemen *repository*, *code* aplikasi akan tersimpan dalam satu tempat atau satu *repository*, hal tersebut akan mempermudah pengerjaan karena *code* tersimpan dalam satu tempat dan *developer* pun bisa melihat sejauh apa pengerjaan tim lain, sejauh apa pengerjaannya dan bisa menyesuaikan tahapan pengerjaan sehingga bisa selesai pada saat yang sama.

Proses pengerjaan sistem otomatisasi mencakup proses *deployment* dengan menggunakan *tools docker* sebagai *container*, manajemen aplikasi menggunakan *jenkins* untuk melakukan proses *deployment*, *virtual privat server* (VPS) sebagai penyimpan *os* dan *github* sebagai penyimpan *code*. Proses selanjutnya dalam pengerjaan sistem *monolithic repository* akan dilakukan dengan pembuatan *code* sendiri dan tidak melakukan *tools* apapun.

Tujuan dalam penerapan ini adalah mengimplementasikan metode *continuous deployment* untuk menjadi sebuah sistem otomatisasi, membuat sistem otomatisasi yang menggunakan *docker* dan *Jenkins*, menggabungkan metode *continuous deployment* dengan *monolithic repository*

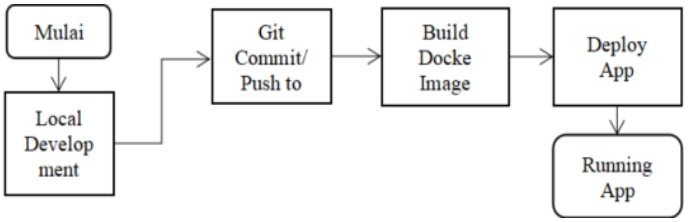
II. KAJIAN TEORI

A. Otomatisasi

Penerapan otomatisasi bertujuan mempermudah pekerjaan dalam sebuah *project* untuk melakukan *deployment* suatu aplikasi dan menghindari proses-proses seperti munculnya *bug* dalam *deploying* [2]. Otomatisasi perlu dilakukan karena begitu banyak kemudahan yang akan didapatkan serta tidak perlu melakukan integrasi berulang ulang dalam setiap pembuatan aplikasi baru.

B. Continuous Deployment

Continuous deployment merupakan jantung *devops* dan membentuk bagian terpenting pada bagian optimasi pengiriman *software* secara menyeluruh. Sebagian besar perusahaan teknologi informasi dan komunikasi mengalami sisi operasi kontributor yang signifikan tetapi mengalami keterlambatan dalam pengiriman *software* [4]. *Continuous deployment* merupakan proses rilis perangkat lunak menggunakan pengujian otomatis guna memvalidasi apakah perubahan *code* sudah sesuai dan stabil untuk penerapan dalam lingkungan produksi. Berikut adalah gambaran tahapan dalam proses *continuous deployment.*



GAMBAR 1
PROSES *CONTINUOUS DEPLOYMENT*

C. Docker

Docker adalah perangkat lunak yang memungkinkan untuk membuat, menguji, dan menerapkan aplikasi dengan cepat [6]. *Docker* dapat mengemas perangkat lunak ke dalam unit standar yang disebut *container*. *Container* sendiri berisikan semua kebutuhan perangkat lunak agar dapat berfungsi. Termasuk pustaka, alat sistem, *code*, dan waktu proses. *Docker* menerapkan dan melaksanakan aplikasi ke *environment* apapun dan memastikan bahwa *code* akan berjalan [5].

D. Jenkins

Jenkins adalah *tools* berbasis *open-source* yang ditulis menggunakan bahasa pemrograman *java* [6]. *Jenkins* sendiri berdiri sejak tahun 2011 yang menawarkan pangsa pasar dominan, *jenkins* digunakan tim dari semua ukuran dalam

sebuah proyek [6]. *Jenkins* menjadi populer dalam kalangan pengembang dikarenakan jenkins mudah digunakan serta sederhana [6], intuitif, dan menarik secara visual. *Jenkins* juga bersifat fleksibel dan mudah beradaptasi untuk tujuan para pengembang. Kemudahan lain yang dimiliki *jenkins* adalah memiliki ratusan *plugin open source* dan *plugin* yang tersedia mencakup semua hal termasuk sistem kontrol versi, alat *build*, metrik kualitas *code*, *build* notifikasi, integrasi dengan sistem external, kustomisasi *ui* serta banyak *plugin* lain yang mudah dalam penginstalannya [6].

E. Virtual Private Server

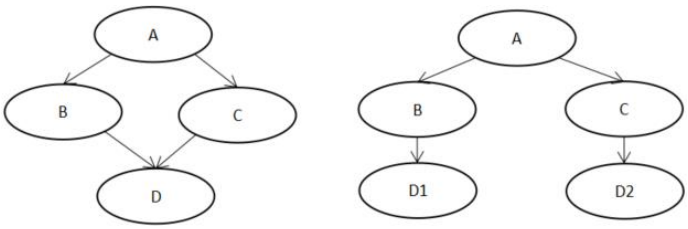
Virtual private server (VPS) adalah *server* yang menggunakan teknologi virtualisasi *hardware server*. *VPS* disebut *virtual* dikarenakan pembagian setiap bagian dilakukan dengan sebuah *software* sehingga dalam satu *server* bisa terdapat beberapa *VPS*.

F. Git

Git adalah sebuah penyimpanan *historical*, setiap perubahan *code* yang telah kita lakukan akan tersimpan dalam *git* sehingga memudahkan kita untuk mengetahui perubahan apa saja yang telah dilakukan oleh para pengembang.

G. Monolithic Repository

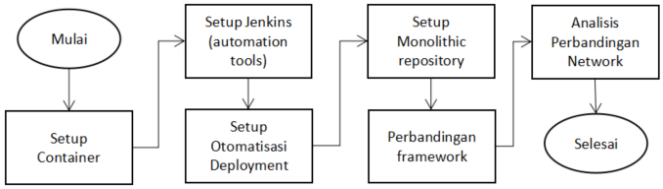
Monolithic repository disebut juga *monorepo* merupakan model *repository* yang ditawarkan oleh google dan telah diterapkan oleh perusahaan besar lain, yaitu sebuah *repo* yang menyimpan semua *code* repositori dalam satu *repository* saja [7]. *Monorepo* memberikan keuntungan apabila ada sebuah *project* atau *taks* memerlukan *project* lain sebagai acuan dasar maka akan lebih mudah karena ketika semua tersimpan dalam satu *repository*, *developers* akan dapat melihat sejauh mana pekerjaan tim lain dan dapat menyesuaikan alur pengerjaannya [7]. Keunggulan lain dari *monorepo* ialah manajemen ketergantungan yang disederhanakan, mampu mengkolaborasi lintas tim, tidak ada batasan kepemilikan *code*, visibilitas *code* dan meminimalisir kerusakan ketika melakukan pembaruan *code* dikarenakan masalah ketergantungan *code-base* yang dapat dilihat seperti perumpamaan di bawah ini antara *monorepo* dan *multi-repo*.



GAMBAR 2
PROSES KETERGANTUNGAN PERUBAHAN *CODE*

III. METODE

Implementasi sistem menggunakan metode *continuous deployment* dan *monorepo* guna menerapkan otomatisasi *deploy software* kepada *production*. Dalam penerapan ini menggunakan beberapa aplikasi sebagai pendukung proses *deployment* yaitu *VPS* yang akan berjalan sebagai server serta tempat penyimpanan semua aplikasi kebutuhan *deployment*. Aplikasi yang akan berjalan dalam *VPS* adalah *docker* sebagai *container* dan *jenkins* sebagai aplikasi yang menyimpan kebutuhan *deployment*. Berikut adalah gambaran bagaimana proses otomatisasi *deploy* berjalan.



GAMBAR 3
PROSES PENERAPAN SISTEM OTOMATISASI

A. Setup Docker

Penginstalan *docker* pada *virtual private server* dilakukan pada komputer lokal melalui *ssh* kepada *virtual private server* yang telah di sewa. Pada pengerjaan kali ini menggunakan *docker* versi 20.10.8 spesifikasi *virtual private server* menggunakan *ubuntu* versi 21.04 dengan 2 *vcpu ram* 2048 MB dan penyimpanan 40 GB.

B. Setup Jenkins

Instalasi *Jenkins* pada *container* dalam *docker* yang telah dibuat dan melakukan beberapa penginstalan sebagai berikut :

1. *Jenkins container setup jenkins* dalam *docker container* serta menentukan *port* yang akan diakses yaitu *port* 8080. Proses selanjutnya adalah mengakses *jenkins* sesuai *ip* yang ada lalu melakukan registrasi dan penginstalan *jenkins* pada *virtual private server*.
2. Membuat *branch pipline* dengan *github*. Membuat *pipline* pada *jenkins* serta menghungkannya kepada *github* sebagai sumber semua *code* tersimpan. *Tools jenkins* yang ada menggunakan *jenkins* versi 2.303.1

C. Setup Otomatisasi Deploy

Pada bagian *setup* ini dilakukan *hard code* pada *github* yang telah dikonfigurasi dengan *jenkins* yang terinstal pada *virtual private server*.

D. Setup Monorepo

Setup repository dilakukan bersamaan dengan pembuatan sistem otomatisasi dengan menambahkan beberapa code. Penambahan *code* tersebut adalah fitur pengecekan apabila terdapat aktivitas *commit* pada *github* maka sistem akan melakukan *build* dan *folder* yang berubah saja. (Penjelasan *flow app*)

E. Pengukuran Tingkat Keberhasilan

Pengukuran tingkat keberhasilan dalam penerapan ini adalah dengan membandingkan hasil dari proses *deployment.*

IV. HASIL DAN PEMBAHASAN

Sebelum memasuki tahapan hasil dari pengujian, berikut adalah tahapan bagaimana proses sistem otomatisasi berjalan, diawali dengan mendeteksi adanya perubahan *code* yang ada pada *repository* dan mencakup 3 tahapan, yaitu:

1. *Prepare*

Pada tahapan ini sistem bekerja dengan mendeteksi adanya *commit* pada penyimpanan yang ada dan melakukan *get change list* melihat bagian penyimpanan yang berubah serta melakukan pengecekan apakah semua kondisi telah memenuhi syarat untuk dilanjutkan kepada proses *build*.

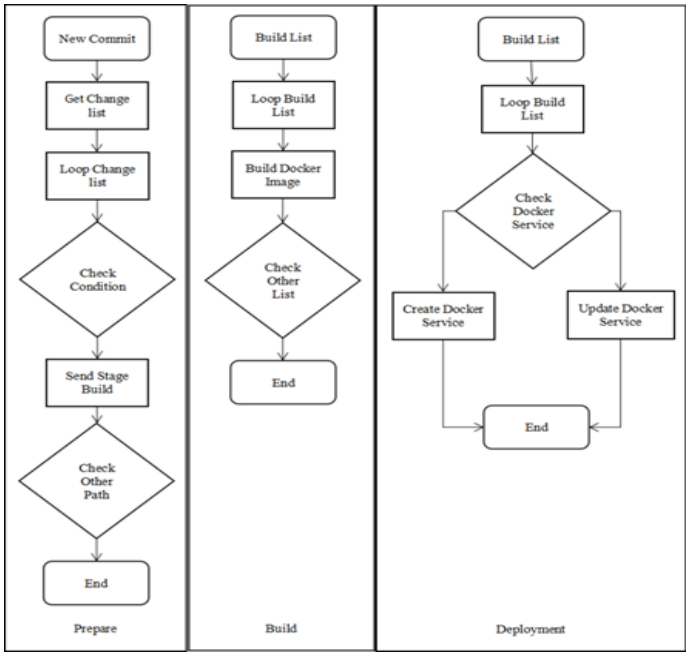
2. *Build*

Dalam proses *build*, melakukan *build list* yang telah didapatkan dari proses *prepare*, setelah itu membuat *docker image* dan dalam tahapan terakhir melakukan pengecekan kembali apakah ada *list* yang tidak terbaca ataupun memenuhi syarat untuk lanjut ke tahapan selanjutnya.

3. *Deployment*

Aktivitas pada proses ini adalah melakukan pengecekan apakah terdapat *docker service* dalam peyimpanan yang diubah, selanjutnya apabila telah ada *docker service* maka dilanjutkan ke dalam tahapan *update docker service* dengan perubahan yang baru, namun apabila belum memiliki *docker service* maka akan melakukan pembuatan *docker service* dan melakukan *deployment* aplikasi.

Proses *prepare*, *build* dan *deployment* ditunjukan dengan gambar 4,



GAMBAR 4
PROSES SISTEM OTOMATISASI BERJALAN

Proses pengujian selanjutnya adalah membandingkan hasil dari kedua *framework* yang ada dengan hasil *stages* (proses akhir *deployment*).

a. *Framework* Pertama

Sistem otomatisasi yang menjadi pembanding pada bahasan penelitian ini dibangun dengan menggunakan metode *continuous deployment* dan *integration* serta menggunakan *tools jenkins ansible*. *Jenkins* berperan sebagai *tools* otomatisasi dan *ansible* digunakan sebagai penyebaran aplikasi yang berisikan alamat *ip* dari sistem yang meng-*host* aplikasi [1]. Kemudian mendeklarasikan *variabel* seperti nama pengguna sistem tempat meng-*host* kemudian kata sandi dan menentukan nomor *port* bila dibutuhkan [1].

Tujuan dalam pembangunan sistem otomatisasi ini untuk mengurangi waktu yang dibutuhkan dalam menghosting aplikasi [1]. Selain waktu yang telah berkurang banyak, *tools* yang digunakan memberi banyak kemudahan karena bersifat *open source* dan memiliki banyak *plugin* [1]. Hasil dari pembuatan sistem otomatisasi ini adalah waktu penyebaran aplikasi tunggal telah berkurang secara signifikan yang akan meningkatkan pemeliharaan aplikasi [1]. Otomatisasi juga akan sangat berguna bagi aplikasi yang sering berubah [1].

Stage View

Average stage times: (Average full run time: ~37s)	Declarative: Checkout SCM	Ansible
	2s	30s
#21 Feb 20 19:52 — 1 commit	1s	30s
#20 Feb 20 19:46 — 1 commit	3s	29s
#19 Feb 20 19:40 — 1 commit	1s	30s
#18 Feb 20 19:14 — 1 commit	4s	35s
#17 Feb 20 19:08 — 1 commit	1s	44s

GAMBAR 5
HASIL STAGES DARI ACUAN YANG ADA

b. *Framework* kedua

Pembangunan sistem otomatisasi ini menggunakan metode *continuous deployment* berbasis *monolithic repository* dengan menggunakan *tools docker* dan *jenkins* yang sama-sama menawarkan berbagai kemudahan seperti acuan pada pembangunan sistem pertama, tetapi menambahkan basis *monolithic repository* sebagai penyimpanan *source code*.

Tujuan penerapan *basis monolithic repository* adalah untuk kemudahan karena semua penyimpanan tersimpan pada satu wadah besar serta tidak perlu melakukan konfigurasi sistem otomatisasi baru apabila akan melakukan pembuatan aplikasi baru yang berbeda dari aplikasi sebelumnya. Berikut adalah gambaran pengukuran waktu *deployment* sistem otomatisasi pada *jenkins*.

Stage View

Average stage times: (Average full run time: ~1min 40s)	Declarative: Checkout SCM	Prepare	Build	Building blog app	Deploy	Deploying app
	3s	2s	825ms	55s	1s	29s
#24 Feb 20 19:56 1 commit	6s	1s	617ms	44s	4s	23s
#23 Feb 20 19:49 1 commit	1s	1s	737ms	43s	762 ms	30s
#22 Feb 20 19:17 1 commit	4s	3s	1s	58s	1s	31s
#21 Feb 20 19:11 1 commit	2s	1s	1s	56s	653ms	32s
#20 Feb 20 19:05 1 commit	1s	1s	793ms	1min 3s	700ms	29s

GAMBAR 6
HASIL DARI *STAGES* PENERAPAN YANG BARU

Skenario pengujian dengan cara melakukan proses *deployment* web, *deploy* web dilakukan sebanyak lima kali dan menggunakan spesifikasi serta isi perubahan web yang sama dengan tujuan mendapatkan hasil yang setara dan optimal. Dalam proses perbandingan menggunakan dua *virtual private server* yang mengatur setiap *framework* yaitu *ansible* dan *monolithic*. Tujuan dari penempatan masing masing *virtual private server* agar mendapatkan hasil yang optimal dan setara dari kedua *framework*.

TABEL 1
PROSES WAKTU *DEPLOYMENT* KEDUA *FRAMEWORK*

No	Ansible	Monolithic
1	45	95,4
2	39	92,6
3	31	98
4	32	76,4
5	31	78,6
Rata-Rata	35,6	88,2

Tabel 1 menunjukan hasil waktu proses *deployment* antara kedua *framework* dan rata-rata keseluruhan proses *deployment* setiap *framework*. Hasil perbandingan antara perhitungan *framework ansible* dan *monolithic repository* pada tabel 7 menunjukan bahwa *framework* dengan *ansible* lebih baik dalam proses *deployment* pada skenario pengujian yang sama dari segi spesifikasi dan isi *deployment*. Hal tersebut dapat terjadi karena *ansible* hanya memiliki satu aplikasi disetiap penyimpanan sedangkan *monolithic* memiliki beberapa aplikasi dalam satu penyimpanan dan dalam proses *deployment* terdapat beberapa proses pengecekan standar *deployment* yang harus dilewati sehingga akan memakan waktu lebih lama.

Sistem otomatisasi menggunakan metode *continuous deployment* dan *monolithic repository* dengan *tools docker* dan *jenkins* adalah suatu proses otomatisasi yang direkomendaikan untuk digunakan pada saat ini. Selain berbagai kemudahan yang didapatkan, *tools* yang ada mudah digunakan karena bersifat *open source* dan gratis. Sistem otomatisasi dengan metode *continuous deployment* yang mengadaptasi *monolithic repository* memang mendapatkan hasil yang jauh berbeda dengan penelitian pada jurnal acuan sebelumnya dikarenakan *monolithic repository* menawarkan kemudahan dalam bentuk yang lain.

Kemudahan menerapkan metode *monolithic repository* adalah tidak ada ketergantungan pada *code* yang ada, dengan *monolithic repository* apabila kita akan membuat sebuah aplikasi baru maka tidak perlu melakukan konfigurasi ulang terkait *framework* yang ada karena semua *code* pada dasarnya hanya tersimpan pada satu wadah yang sama.

V. KESIMPULAN

Perbandingan antara penerapan sistem menggunakan *monolithic repository* dengan *ansible* memang menunjukkan perbedaan yang jauh dalam waktu *deployment*, *ansible* menunjukkan waktu *deployment* lebih baik, tetapi penerapan *monolithic repository* memiliki kelebihan lain yaitu dengan adanya penyimpanan secara terpusat maka keseluruhan *code* dapat dilihat oleh *developers* dari berbagai divisi untuk menyesuaikan proses pengembangan aplikasi sehingga berjalan secara beriringan sampai semua proses pembuatan aplikasi selesai. Apabila *code* penyimpanan tersimpan di banyak tempat maka *framework ansible* akan lebih memudahkan karena hanya perlu melakukan *ssh* dari satu tempat yang sama untuk mengatur *server* yang lain. Sedangkan *framework monolithic* penggunaannya lebih kepada manajemen data terpusat yang tersusun dengan baik sehingga dengan satu *repository* saja bisa menyimpan begitu banyak aplikasi.

REFERENSI

[1] Mysari, S. and Bejgam, V., 2020, February. *Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible*. In 2020 *International Conference on Emerging Trends in Information Technology and Engineering* (ic-ETITE) (pp. 1-4). IEEE. doi.org/10.1109/ic-etite47903.2020.239

[2] Gallaga, K., 2019, September. *Improving the robustness and efficiency of continuous integration and deployment. In* 2019 IEEE *International Conference on Software Maintenance and Evolution* (ICSME) (pp. 619-623). IEEE. doi.org/10.1109/icsme.2019.00099

[3] Jaspan, C., Jorde, M., Knight, A., Sadowski, C., Smith, E.K., Winter, C., and Murphy-Hill, E., 2018. *Advantages and Disadvantages of a Monolithic Repository*. doi.org/10.1145/3183519.3183550

[4] Virmani, M., 2015, May. *Understanding DevOps & bridging the gap from continuous integration to continuous delivery. In Fifth international conference on innovative computing technology* (*fintech* 2015) (pp. 78-82). IEEE. doi.org/10.1109/intech.2015.7173368

[7] Laanti,Potvin, R. and Levenberg, J., 2016. *Why does Google stores billions of lines of code in a single repository. Communications of the* ACM, 59(7), pp.78-87. doi.org/10.1145/2854146

[5] Ahmed, A. and Pierre, G., 2018, July. *Docker container deployment in fog computing infrastructures*. In 2018 IEEE. doi.org/10.1109/edge.2018.00008

[6] Smart, J.F., 2011. Jenkins: *The Definitive Guide: Continuous Integration for the Masses*. " O'Reilly Media, Inc.".