

SKRIPSI

IMPLEMENTASI CI/CD PADA LAYANAN MIKRO TUKUTU MENGUNAKAN TELEGRAM BOT DAN NODEJS

Diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik
Informatika



Disusun Oleh:

Nama : Ahmad Basir
NIM : A11.2015.09101
Program Studi : Teknik Informatika-S1

**FAKULTAS ILMU KOMPUTER
UNIVERSITAS DIAN NUSWANTORO
SEMARANG
2019**

PERSETUJUAN SKRIPSI

Nama : Ahmad Basir
NIM : A11.2015.09101
Program Studi : Teknik Informatika
Fakultas : Ilmu Komputer
Judul Tugas Akhir : Implementasi CI/CD Pada Layanan Mikro Tukutu
Menggunakan Telegram Bot Dan Nodejs

Tugas Akhir ini telah diperiksa dan disetujui,

Semarang, 5 November 2019

Menyetujui
Pembimbing

Mengetahui
Dekan Fakultas Ilmu Komputer

Dr. Pulung Nurtantio A., S.T, M.Kom

NPP. 0686.11.2006.331

Dr. Drs. Abdul Syukur, MM

NPP.0686.11.1992.017

PENGESAHAN DEWAN PENGUJI

Nama : Ahmad Basir
NIM : A11.2015.09101
Program Studi : Teknik Informatika
Fakultas : Ilmu Komputer
Judul Tugas Akhir : Implementasi CI/CD Pada Layanan Mikro Tukutu
Menggunakan Telegram Bot Dan NodeJS

Tugas akhir ini telah diujikan dan dipertahankan dihadapan Dewan Penguji pada Sidang tugas akhir tanggal 5 November 2019. Menurut pandangan kami, tugas akhir ini memadai dari segi kualitas maupun kuantitas untuk tujuan penganugrahan gelar Sarjana Komputer (S.Kom.)

Semarang, 5 November 2019

Dewan Penguji:

ERNA ZUNI ASTUTI M.Kom

Anggota 1

JUNTA ZENIARJA, M.Kom

Anggota 2

PURWANTO, S.Si, M.Kom, Ph.D

Ketua Penguji

PERNYATAAN KEASLIAN SKRIPSI

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan di bawah ini, saya:

Nama : Ahmad Basir

NIM : A11.2015.09101

Menyatakan bahwa karya ilmiah saya yang berjudul:

IMPLEMENTASI CI/CD PADA LAYANAN MIKRO TUKUTU MENGUNAKAN TELEGRAM BOT DAN NODEJS

merupakan karya asli saya (kecuali cuplikan dan ringkasan yang masing-masing telah saya jelaskan sumbernya dan perangkat pendukung seperti web cam dll). Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, yang disertai dengan bukti-bukti yang cukup, maka saya bersedia untuk dibatalkan gelar saya beserta hak dan kewajiban yang melekat pada gelar tersebut. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada tanggal : 5 November 2019

Yang menyatakan

(Ahmad Basir)

PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai mahasiswa Universitas Dian Nuswantoro, yang bertanda tangan di bawah ini, saya:

Nama : Ahmad Basir

NIM : A11.2015.09101

demikian mengembangkan Ilmu Pengetahuan, menyetujui untuk memberikan kepada Universitas Dian Nuswantoro Hak Bebas Royalti Non-Eksklusif (*Non-exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul:

IMPLEMENTASI CI/CD PADA LAYANAN MIKRO TUKUTU MENGUNAKAN TELEGRAM BOT DAN NODEJS

beserta perangkat yang diperlukan (bila ada). Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Dian Nuswantoro berhak untuk menyimpan, mengcopy ulang (memperbanyak), menggunakan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya dan menampilkan/mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta ijin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Dian Nuswantoro, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini. Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Semarang

Pada tanggal : 5 November 2019

Yang menyatakan,

(Ahmad Basir)

UCAPAN TERIM KASIH

Dengan memanjatkan puji syukur kehadirat Allah SWT. Tuhan Yang Maha Pengasih yang telah melimpahkan segala rahmat, hidayah dan inayah-Nya kepada penulis sehingga tugas akhir berjudul “IMPLEMENTASI CI/CD PADA LAYANAN MIKRO TUKUTU MENGGUNAKAN TELEGRAM BOT DAN NODEJS” dapat penulis selesaikan sesuai dengan rencana karena dukungan dari berbagai pihak yang tidak ternilai besarnya. Oleh karena itu penulis menyampaikan terimakasih kepada :

1. Prof. Dr. Ir. Edi Noersasongko, M.Kom, selaku Rektor Universitas Dian Nuswantoro Semarang.
2. Dr. Drs. Abdul Syukur, MM, selaku dekan Fakultas Ilmu Komputer Universitas Dian Nuswantoro.
3. Dr. Moeljono, S.Si, M.Kom , selaku Ka. Progdi Teknik Informatika dan dosen pembimbing yang telah memberikan bimbingan kepada penulis dalam menyelesaikan tugas akhir ini.
4. Dosen-dosen pengampu di Fakultas Ilmu Komputer Teknik Informatika Universitas Dian Nuswantoro Semarang yang telah memberikan ilmu dan pengalamannya masing – masing, sehingga penulis dapat mengimplementasikan ilmu yang telah disampaikan.
5. Bapak dan Ibu yang tidak pernah lelah untuk memberikan doa serta dorongan kepada penulis untuk terus maju dan berusaha.
6. Tim Tukutu developer yang selalu memberikan semangat dan dukungan moril untuk penulis.
7. Sahabat – sahabat dan teman – teman penulis yang telah membantu dan memberikan semangat kepada penulis.

Akhirnya, Penulis mengucapkan rasa terima kasih kepada semua pihak dan apabila ada yang tidak disebutkan Penulis mohon maaf, dengan besar harapan semoga tugas akhir yang ditulis oleh Penulis ini dapat bermanfaat khususnya bagi Penulis sendiri dan umumnya bagi pembaca.

Semarang, 5 November 2019

(Ahmad Basir)

ABSTRAK

Aplikasi titip jual beli sepatu online Tukutu termasuk perangkat lunak berbasis mobile apps yang terintegrasi dengan RESTFull API sebagai tempat untuk mengirim atau menerima data atau record. Pada implementasi penelitian ini menggunakan Continuous Integration yang berfungsi sebagai penerapan pengembangan perangkat lunak yang dapat melakukan kompilasi dan pengujian secara otomatis. Dipadukan dengan Telegram BOT yang menggunakan sebuah akun khusus tanpa nomor, command pada akun tersebut berfungsi sebagai antarmuka dari sistem yang berjalan yang berkomunikasi melalui Telegram Bot API. Dengan perpaduan keduanya menghasilkan Continuous Delivery atau pengiriman berkelanjutan yang fungsinya untuk memberikan command dari sebuah sistem kemudian akan dilanjutkan dengan command perubahan sistem tersebut keserver production atau secara bertahap.

Kata kunci : *CI/CD, Docker, Telegram Bot, Git, NodeJS*

DAFTAR ISI

PERSETUJUAN SKRIPSI.....	ii
PENGESAHAN DEWAN PENGUJI.....	iii
PERNYATAAN KEASLIAN SKRIPSI.....	iv
PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS.....	v
UCAPAN TERIM KASIH.....	vi
ABSTRAK.....	viii
DAFTAR ISI.....	ix
DAFTAR TABEL.....	xii
DAFTAR GAMBAR.....	xiii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	4
1.3 Batasan Masalah.....	4
1.4 Tujuan Penelitian.....	5
1.5 Manfaat Penelitian.....	5
1.5.1 Bagi Akademik.....	5
1.5.2 Bagi Tukutu.....	5
1.5.3 Bagi Penulis.....	6
1.5.4 Bagi Pembaca.....	6
BAB II LANDASAN TEORI.....	7
2.1 Tinjauan Studi.....	7
2.2 Tinjauan Pustaka.....	11
2.2.1 Continuous Integration And Delivery (CI/CD).....	11
2.2.2 SSH.....	12
2.2.3 NodeJS.....	13
2.2.4 BASH.....	14

2.2.5 Sistem Operasi Linux.....	15
2.2.6 Docker dan docker-compose.....	16
2.2.7 Git.....	18
2.2.8 Telegram Bot.....	19
2.3 Deskripsi Tempat KKI.....	23
2.3.1 Logo dan Makna Tempat KKI.....	24
2.3.2 Struktur Organisasi Tempat KKI.....	25
2.3.3 Visi dan Misi Tempat KKI.....	25
BAB III METODE PENELITIAN.....	26
3.1 Instrumen Penelitian.....	26
3.2 Prosedur Pengambilan Data.....	26
3.2.1 Observasi.....	26
3.2.2 Studi Literatur.....	27
3.2.3 Forum Diskusi dan Tutorial.....	27
3.3 Teknik Analisis Data.....	27
3.4 Metode.....	28
3.4.1 Rapid Application Development (RAD).....	28
3.4.2 Perancangan Sistem Bot.....	30
3.4.3 Pembuatan Akun Telegram Bot.....	31
3.4.4 Pendefinisian Telegram Bot Token.....	34
BAB IV IMPLEMENTASI DAN PEMBAHASAN.....	36
4.1 Gambaran Umum Sistem.....	36
4.1.1 Tukutu.....	36
4.1.2 Telegram Bot Tukutu.....	37
4.2 Work Flow.....	38
4.2.1 Identifikasi Aktor.....	39
4.2.2 Use Case Diagram.....	40
4.2.2.1 Use Case.....	40
4.2.2.2 Use Case Naratif.....	41
4.2.3 Activity Diagram.....	43

4.2.3.1 Activity Diagram Pembaruan Service.....	44
4.2.3.2 Activity Diagram Pengecekan Log Service.....	45
4.2.3.3 Activity Diagram Remote Command Line.....	46
4.2.3.4 Activity Diagram Health Check.....	47
4.3 Implementasi Kode Program.....	48
4.3.1 Struktur Project Telegram Bot Tukutu.....	48
4.3.1.1 .circleci.....	48
4.3.1.2 Deployer.....	49
4.3.1.3 Module.....	54
4.3.1.4 Notification.....	56
4.3.1.5 Elfi.js.....	57
4.4 Pengujian.....	61
4.4.1 White Box Testing.....	62
4.4.1.1 Pseudocode.....	62
4.4.2 Pembuatan Flowgraph.....	64
4.4.2.1 Perhitungan <i>Cyclomatic Complexity</i>	64
4.4.2.2 Penentuan jalur independen.....	65
4.4.2.3 Hasil <i>Test Case</i>	66
4.4.3 Hasil Pengujian Pembaruan Service Tukutu.....	67
4.4.4 Hasil Proses Pada Latar Belakang Service.....	68
BAB V KESIMPULAN DAN SARAN.....	69
5.1 Kesimpulan.....	69
5.2 Saran.....	69
DAFTAR PUSTAKA.....	71

DAFTAR TABEL

Tabel 2.1: State of The Art.....	8
Tabel 2.2: Telegram Bot Parameter.....	21
Tabel 4.1: Identifikasi Aktor.....	39

DAFTAR GAMBAR

Gambar 1.1: CI/CD Work Flow.....	2
Gambar 2.1: Sampel NodeJS server.....	13
Gambar 2.2: Logo PT Dian Nuswantoro Teknologi dan Informasi.....	24
Gambar 3.1: Rapid Application Development Flow.....	28
Gambar 3.2: Proses Pembuatan Bot.....	32
Gambar 3.3: Menu List Bot.....	33
Gambar 3.4: Menu Detail Bot.....	34
Gambar 3.5: Environment Bot.....	34
Gambar 3.6: Pendefinisian Bot Token.....	35
Gambar 4.1: Aplikasi Tukutu.....	36
Gambar 4.2: Tentang Telegram Bot.....	37
Gambar 4.3: Diagram Use Case.....	40
Gambar 4.4: Flow Diagram Update Service.....	44
Gambar 4.5: Flow Diagram Pengecekan Log Service.....	45
Gambar 4.6: Flow Diagram Remote Command Line.....	46
Gambar 4.7: Flow Diagram Health Check.....	47
Gambar 4.8: Struktur Project.....	48
Gambar 4.9: Shell Script Lib.sh I.....	49
Gambar 4.10: Shell Script Lib.sh II.....	50
Gambar 4.11: Shell Script Updater I.....	51
Gambar 4.12: Shell Script Updater II.....	52
Gambar 4.13: Variable.sh.....	53
Gambar 4.14: Logika Authtorize Module.....	54
Gambar 4.15: Logika Git.js Module.....	55
Gambar 4.16: Logika Splitter.js Module.....	56
Gambar 4.17: Shell Script Restart Bot Service.....	57
Gambar 4.18: Logika Execute Elfi.js.....	58
Gambar 4.19: Logika Retrusted Command.....	59

Gambar 4.20: Logika Update Service.....	59
Gambar 4.21: Logika Pengecekan Log Service.....	60
Gambar 4.22: Logika Unauthorize User Check.....	61
Gambar 4.23. Flowgraph Parsing Algorithm.....	64
Gambar 4.24: Pengujian Pada Telegram Bot.....	67
Gambar 4.25: Log Report Pada Pembaruan Service.....	68

BAB I

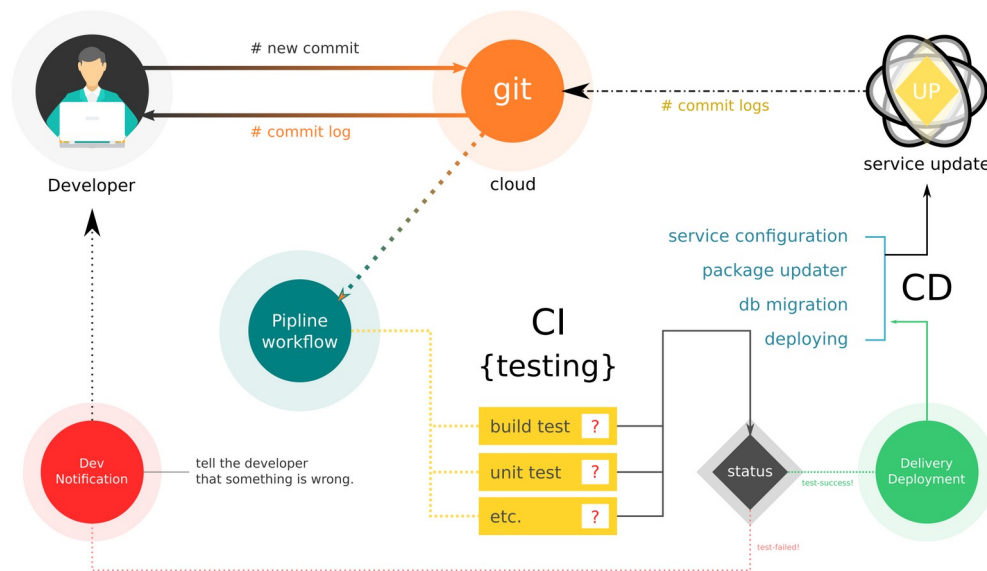
PENDAHULUAN

1.1 Latar Belakang

Saat ini banyak sekali teknologi – teknologi yang di kembangkan oleh developer atau pengembang software, ada berbagai macam tipe dan juga karakteristik seperti perangkat lunak dalam bidang kesehatan, pertanian, sosial media maupun berbelanja dengan memanfaatkan internet. Salah satu dari beberapa tipe tersebut yang saat ini masih menjadi sepuluh besar yaitu dari bidang perbelanjaan atau biasa di sebut dengan belanja online e-commerce. Perkembangan teknologi informasi dapat membantu cara hidup kita menjadi lebih modern, lebih mudah dan lebih praktis dalam kehidupan sehari-hari (Ilyas, 2018).

Aplikasi titip jual beli sepatu online Tukutu yang mempunyai ciri khas tersendiri yaitu dengan memberikan jaminan keaslian dari sepatu yang dijual. Dalam aplikasi Tukutu ada beberapa spesifikasi dari sepatu bekas, langka sampai sepatu baru dari beberapa brand ternama maupun brand local. Hal ini tentu memudahkan komunitas pecinta sepatu original dalam memilah atau mencari tempat dimana harus membeli sepatu yang mereka inginkan. Aplikasi Tukutu saat ini sudah memasuki perangkat lunak berjalan mobile apps yang terintegrasi dengan RESTFull API sebagai tempat untuk mengirim atau menerima data atau record.

Continuous Integration and Delivery Concept



Gambar 1.1: CI/CD Work Flow

Continuous Integration – Integrasi berkelanjutan merupakan penerapan pengembangan perangkat lunak yang dapat melakukan kompilasi dan pengujian secara otomatis (Hilton, 2016), seperti misalnya pengembang sedang melakukan perubahan pada sebuah sistem, dalam satu waktu perubahan tersebut yang telah dilakukan versioning dapat secara otomatis melakukan builing system dan pengujian secara otomatis. Continuous Delivery – Pengiriman berkelanjutan adalah kemampuan untuk mendapatkan semua jenis perubahan dari sebuah sistem kemudian akan dilanjutkan dengan pemasangan perubahan sistem tersebut ke server production atau secara bertahap (Arachchi, 2018). Telegram BOT - sebuah akun khusus tanpa nomor yang dapat menghandle sebuah Command dari pengguna dan memberikan jawaban sesuai fungsionalitas Command. Akun

tersebut berfungsi sebagai antarmuka dari sistem yang berjalan yang berkomunikasi melalui Telegram Bot API (Nufusula, 2018).

SSH atau biasa disebut secure shell, adalah salah satu perangkat lunak yang menyediakan login access secara aman. Konsep dari ssh seperti remote untuk mengakses sebuah perangkat dari jauh dengan protocol yang rumit. SSH menggunakan kriptografi yang unik serta kuat untuk memberikan otentikasi dan kerahasiaan (Provos, 2001), ssh di gunakan oleh system administrator (SysAdmin) untuk mengakses server yang mereka kelola.

System Administrator (SysAdmin) memiliki peran dalam konfigurasi, penanganan masalah dan pertahanan sistem komputer rumit yang terdiri dari beberapa komponen, misal : system management basis data, server servis, server aplikasi, load balancer, dan server-server lain yang di distribusikan di beberapa jaringan platform system operasi (Barrett, 2004). Dalam sebuah kasus pengembang dari tahap pengembangan sampai tahap produksi, sisi operasional server melakukan deployment dengan cara masuk melalui SSH-SERVER kemudian baru melakukan konfigurasi pada sistem tersebut dengan memasukan beberapa baris perintah bash script. Begitupun apabila pengembang melakukan perubahan setiap saat dan sisi operasional harus melakukan perulangan yang sama sehingga langkah tersebut menjadi berulang-ulang dan tidak efisien.

Dari masalah yang telah diuraikan maka penulis melakukan penelitian yang berjudul “IMPLEMENTASI CI/CD PADA LAYANAN MIKRO TUKUTU MENGGUNAKAN TELEGRAM BOT DAN NODEJS” sebagai pemecahan masalah ci/cd yang berada di servis mikro Tukutu. Kenapa kita harus menggunakan CI/CD dan apa manfaat dari CI/CD itu sendiri terdapat pada efisiensi dalam pengelolaan setiap service yang telah didefinisi atau di konfigurasi, misalkan ada lebih dari 100 servis kita hanya melakukan setup satu kali untuk kebutuhan selanjutnya, CI/CD juga akan menguntungkan dari sisi

developer karena dia bisa langsung tau apa yang salah dan apa yang harus di perbaiki tanpa harus di beritahu oleh SysAdmin.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka di dapatkan rumusa masalah sebagai berikut :

1. Bagaimana merancang dan mengimplementasikan CI/CD berkelanjutan (CI/CD) pada microservice di tukutu

1.3 Batasan Masalah

Dalam penelitian ini terdapat batasan masalah sehingga pembahasan yang dilakukan tidak menyimpang dari tujuan penelitian. Berikut batasa masalah yang di dapat:

1. CI/CD dibuat menggunakan NodeJS.
2. CI/CD diImplementasi pada Telegram BOT.
3. Bot hanya bisa mengeksekusi perintah jika username pada akun telegram terdaftar pada environment.
4. Selain super Admin Bot hanya bisa menerima perintah yang ada pada daftar.
5. Bot terintegrasi dengan git version control.
6. Perintah pada Bot akan dilanjutkan dengan eksekusi sebuah file.
7. Dalam sebuah group, bot bisa mengeksekusi perintah dengan melakukan mention.

8. Beberapa perintah dalam terminal tidak dapat dieksekusi Bot bahkan oleh super admin, untuk menjaga sistem tetap aman.

1.4 Tujuan Penelitian

Adapun tujuan yang dilakukan penelitian ini:

1. Merancang dan implementasi CI/CD menggunakan Telegram BOT pada microservice Tukutu.

1.5 Manfaat Penelitian

Manfaat yang diperoleh dari penelitian ini:

1.5.1 Bagi Akademik

- a) Sebagai indikator sejauh mana pemahaman yang telah di dapat saat menuntut ilmu di Universitas Dian Nuswantoro.
- b) Menambah jenis penelitian pada Universitas Dian Nuswantoro yang dapat digunakan sebagai referensi penelitian lainnya.
- c) Menjalin hubungan kerja sama antara Universitas Dian Nuswantoro dan Tukutu.

1.5.2 Bagi Tukutu

- a) Menjalin hubungan kerja sama antara Tukutu dan Universitas Dian Nuswantoro.
- b) Dengan adanya penelitian ini, Tukutu dapat terbantu pada sistem jual belinya.

1.5.3 Bagi Penulis

- a) Menambah pengetahuan dan pengalaman kerja pada instansi tempat penelitian dilakukan.
- b) Penulis dapat mengembangkan ilmu yang telah di dapat selama berada di Universitas Dian Nuswantoro.

1.5.4 Bagi Pembaca

- a) Menambah wawasan tentang CI/CD.
- b) Dapat digunakan sebagai bahan refrensi penelitian dan pengembangan selanjutnya.

BAB II

LANDASAN TEORI

2.1 Tinjauan Studi

Pada penelitian ini salah satu topik yang berhubungan dengan yang peneliti angkat adalah sebuah buku pada tahun 2017 dengan judul “Xamarin Continuous Integration and Delivery” oleh versluis, G. yang membahas tentang kenapa kita perlu sebuah Automation Pipeline atau dengan kata lain mengotomatisasikan serangkaian proses menjadi lebih efisien, dari buku tersebut dikatakan bahwa pengulangan adalah suatu yang tidak begitu bagus bagi manusia, namun hal ini berbeda dengan komputer yang perannya sangat cocok untuk melakukan hal yang sama berulang-ulang (Versluis, 2017), pengujiannya pada chapter 6 Integration Tests into Your Build menarik sebuah kesimpulan pada kondisi tertentu Automation Pipeline akan lebih bagus digunakan ketika kita melakukan banyak proses sekaligus (multi proccess).

Penelitian lain dengan judul “A Spoonful of DevOps help the GI go Down” pada tahun 2018, yang di temukan oleh Alexandra Elbakyan. Penelitian tersebut membahas tentang Genetic Improvement (GI) yang berfokus pada peningkatan otomatis dari pengembangan perangkat sebuah lunak. Genetic Improvement (GI) bertujuan untuk meningkatkan pengembangan perangkat lunak yang ada melalui pencarian transformasi dan evaluasi secara otomatis, otomatisasi pengembangan dan pengoperasian perangkat lunak secara end-to-end untuk evolusi itu sendiri. Hasil dari penelitian tersebut menyampaikan bahwa otomatisasi dalam skala proses menghilangkan hambatan dalam proses development sampai ke produksi sebuah perangkat lunak (Nicolas, 2018).

Penelitian berikutnya dengan judul “Continuous Deployment of Software Intensive Products and Services: A Systematic Mapping Study” tahun 2015. Memebuat penelitian tentang bagaimana cara membuat suatu struktur pipeline development dengan tahapan yang praktis dan tool yang sudah dikembangkan digunakan kembali pada penelitian utama, didasarkan bagaimana cara membuat sebuah produk lebih cepat di kirim ke customer atau pengguna lebih cepat dari sebelumnya, karena hal ini mempengaruhi proses bisnis (Rodríguez, 2015).

Tabel 2.1: State of The Art

No	Peneliti	Judul	Tool	Hasil
1.	Ferry Satria, Tira Sundara, Hertog Nugroho, Malayusfi (2019)	Pemantau Ruangan MenggunakanRa spberry Pi Terintegrasi Dengan BotTelegram Dan Halaman Web	Raspberry Pi	Semakin besar nilai delta threshold yang digunakan maka semakin semakin sedikit juga motion yang dideteksi.
2.	Rob van der Valk, Patrizio Pelliccione, Patricia Lago, Rogardt Heldal, Eric Knauss, Jacob Juul (2018)	Transparency and Contracts: Continuous Integration and Delivery in the Automotive Ecosystem	Atlas.ti	Transparansi bukanlah kondisi yang diperlukan untuk CI&D antar organisasi. Namun, transparansi dianggap sebagai positif karena menciptakan efek

				sinergis positif dalam hal efisiensi, kepercayaan, dan saling pengertian, sambil menghindari situasi stres yang tidak perlu.
3.	Eric Knauss ¹ , Aminah Yussuf, Kelly Blincoe, Daniela Damian, Alessia Knauss (2016)	Continuous clarification and emergent requirements flows in open-commercial software ecosystems	IBM's Collaborative Lifecycle Management	saluran informasi terbuka mendukung aksi strategis global dan aksi just-in-time lokal. Baik tindakan global maupun lokal diperlukan untuk menjadikan ekosistem perangkat lunak sebagai mitra bisnis yang kompetitif, tetapi untuk memungkinkan keduanya, arus informasi bottomup dan horizontal

				perlu ditangani secara sistematis.
4.	Ricardo Colomo-Palacios, Eduardo Fernandes, Pedro Soto-Acosta, Xabier Larrucea (2018)	A case analysis of enabling continuous software deployment through knowledge management	Meta4	tekanan terhadap adopsi DevOps bersifat internal dan eksternal. Misalnya, teknologi terkini, ketersediaan teknologi, efektivitas dan manfaat yang dilaporkan diidentifikasi sebagai tekanan eksternal utama. Pada saat yang sama, adopsi DevOps terbukti mampu meningkatkan waktu dan kualitas siklus. Namun, responden berpendapat bahwa meskipun efek di tingkat perusahaan terbatas, arsitektur

				layanan mikro dapat memperluas manfaat DevOps di luar sisi teknis.
--	--	--	--	--

2.2 Tinjauan Pustaka

2.2.1 Continuous Integration And Delivery (CI/CD)

Continuous Integration merupakan sebuah teknik pada pengembangan software dimana pengembang dengan melakukan pembaruan atau perubahan pada sebuah repository jika suatu waktu terdapat perubahan pada kode sumber source code baru, serta mengotomatisasikan build & test dengan hasil berupa sebuah produk jadi. Apabila terjadi sebuah kesalahan, error, atau bug pada sebuah microservice atau aplikasi software dalam build/test maka tim pengembang dapat dengan segera melakukan bug fixing sehingga produk dapat di tingkatkan, dan juga dapat mengurangi waktu untuk melakukan validasi sebuah update.

Continuous Delivery adalah sebuah teknik pada pengembangan software atau aplikasi ketika pengembang melakukan sebuah perubahan atau update source-code, build & testing yang kemudian akan otomatis sistem akan melakukan deploying sumber yang telah di perbarui ke sebuah environment produksi, akan tetapi jika suatu ketika sebuah kode sumber terdapat sebuah error atau bug, maka dengan otomatis sistem akan melakukan roll-back pada perubahan sebelumnya karena Continuous Delivery berkaitan dengan proses bisnis. Apabila Continuous Delivery dapat berjalan dengan tanpa kendala tim pengembang bisa melanjutkan

2.2.2 SSH

Protokol Secure Shell (SSH) adalah protokol untuk login jarak jauh yang aman dan layanan jaringan aman lainnya melalui jaringan yang tidak aman. Protokol SSH terdiri dari tiga komponen utama: Transport Layer Protocol menyediakan otentikasi server, kerahasiaan, dan integritas dengan kerahasiaan ke depan yang sempurna. Protokol Otentikasi Pengguna mengautentikasi klien ke server. Protokol Koneksi melipatgandakan terowongan terenkripsi menjadi beberapa saluran logis. Detail protokol ini dijelaskan dalam dokumen terpisah (Ylonen, 2006).

Metode Secure Shell (SSH) digunakan untuk melakukan remote-login atau masuk pada suatu sistem dari jarak jauh pada jaringan yang tidak aman. Secure Shell (SSH) sendiri terbagi menjadi 3 komponen :

- Protocol Transport Layer [SSH-TRANS]
menyediakan otentikasi server, kerahasiaan, dan integritas. Secara opsional dapat juga memberikan kompresi. Lapisan transport biasanya dijalankan melalui koneksi TCP / IP,
- Protokol Otentikasi Pengguna [SSH-USERAUTH]
mengotentikasi pengguna sisi klien ke server. Ini berjalan di atas protokol layer transport.
- The Connection Protocol [SSH-CONNECT]
melipatgandakan terowongan terenkripsi menjadi beberapa saluran logis. Itu berjalan di atas protokol otentikasi pengguna.

2.2.3 NodeJS

NodeJS merupakan sebuah runtime JavaScript yang didorong peristiwa asinkron, Node dirancang untuk membangun aplikasi jaringan yang dapat diskalakan. Dalam contoh "hello world" berikut, banyak koneksi dapat ditangani secara bersamaan. Pada setiap koneksi callback diaktifkan, tetapi jika tidak ada pekerjaan yang harus dilakukan, Node akan tertidur (sleep mode).

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Gambar 2.1: Sampel NodeJS server

Dengan popularitas Node.js, asynchronous, pemrograman eventdriven telah menyebar luas di aplikasi sisi server. Meskipun secara konsep sederhana, pemrograman berbasis event bisa membosankan dan rawan kesalahan. Semantik kompleks dari event-loop Node.js, ditambah dengan berbagai rasa eksekusi asinkron dalam JavaScript, dengan mudah menyebabkan bug (Sun, 2019). Pada sebuah Makalah yang berjudul "Reasoning about the Node.js event loop using async graphs" memperkenalkan model baru yang disebut Async Graph untuk alasan tentang perilaku runtime aplikasi dan interaksinya dengan loop peristiwa Node.js. Berdasarkan model, Penulis telah mengembangkan AsyncG, alat untuk

secara otomatis membangun dan menganalisis Grafik Async dari aplikasi yang sedang berjalan, dan untuk mengidentifikasi bug yang terkait dengan semua sumber eksekusi asinkron di Node.js. AsyncG kompatibel dengan fitur bahasa ECMAScript terbaru dan dapat (de) diaktifkan saat runtime. Dalam evaluasi kami, kami menunjukkan bagaimana AsyncG dapat digunakan untuk mengidentifikasi bug di aplikasi Node.js di dunia nyata.

Event-Driven dari Node.js dan event-loop secara konseptual sederhana: loop peristiwa mendengarkan kejadian dan memicu pengendali peristiwa misalnya pemanggilan kembali fungsi callback ketika suatu peristiwa terjadi. Perulangan event di Node.js sendiri dijalankan oleh satu threads, tetapi ia dapat mensurvei peristiwa dari threads lainnya misalkan threads asli untuk I/O, dan dengan demikian membuat Node.js mampu menangani ribuan permintaan klien secara bersamaan (asynchronous)

Ada beberapa modul atau library dasar node.js yang dapat mentranslasikan sebuah string menjadi sebuah perintah dasar console, dari library tersebut dapat lagi dikembangkan menjadi modul-modul kecil yang sangat berguna, misalkan untuk membuat perintah baru seperti memasukkan perintah console yang di ambil dari sebuah package binary dari sebuah program kecil. dari sini dapat dilakukan pemrograman secara modular dimana untuk pengembangan jangka panjang akan lebih nyaman dan mudah.

2.2.4 BASH

BASH adalah file teks biasa yang berisi serangkaian perintah. Perintah-perintah ini adalah campuran dari perintah yang biasanya kita ketik ourselves pada baris perintah (seperti ls atau cp misalnya) dan perintah yang bisa kita ketik pada baris perintah tetapi umumnya tidak (Anda akan menemukan ini pada beberapa halaman berikutnya). BASH berjalan pada Unix shell, interpreter baris perintah

atau shell yang menyediakan antarmuka pengguna baris perintah untuk sistem operasi mirip Unix. Shell adalah bahasa perintah interaktif dan bahasa scripting, dan digunakan oleh sistem operasi untuk mengontrol eksekusi sistem menggunakan skrip shell. Pengguna biasanya berinteraksi dengan shell Unix menggunakan emulator terminal; Namun, operasi langsung melalui koneksi perangkat keras serial atau Secure Shell adalah umum untuk sistem server. Semua shell Unix menyediakan wildcarding nama file, perpipaan, di sini dokumen substitusi perintah, variabel, dan struktur kontrol untuk pengujian kondisi dan iterasi.

Dalam suatu kasus, perintah command line biasanya di eksekusi pada sebuah console atau terminal dari sebuah sistem operasi itu sendiri, dalam hal lain terdapat deretan atau runtutan perintah command line yang di satukan menjadi sebuah file yang bisa disebut sebagai wizard-script atau bash-script. Bash scripting biasanya digunakan sebagai file automasi dari sebuah perintah yang sering diulang untuk menghemat waktu pada penulisan command line.

2.2.5 Sistem Operasi Linux

Linux adalah keluarga sistem operasi perangkat lunak bebas dan sumber terbuka yang berbasis pada kernel Linux, kernel sistem operasi yang pertama kali dirilis pada 17 September 1991 oleh Linus Torvalds. Linux biasanya dikemas dalam distribusi Linux (atau singkatnya distro). Distribusi mencakup kernel Linux dan perangkat lunak serta perpustakaan sistem pendukung, banyak di antaranya disediakan oleh Proyek GNU. Banyak distribusi Linux menggunakan kata "Linux" dalam namanya, tetapi Free Software Foundation menggunakan nama GNU / Linux untuk menekankan pentingnya perangkat lunak GNU.

Linux pada awalnya dikembangkan untuk komputer pribadi berdasarkan arsitektur Intel x86, tetapi sejak itu porting ke lebih banyak platform daripada

sistem operasi lain. Linux adalah sistem operasi terkemuka di server dan sistem besel besar lainnya seperti komputer mainframe, dan satu-satunya OS yang digunakan pada superkomputer TOP 500 (sejak November 2017, setelah secara bertahap menghilangkan semua pesaing). Ini digunakan oleh sekitar 2,3 persen komputer desktop. Chromebook, yang menjalankan Chrome OS berbasis kernel Linux.

Linux juga berjalan pada sistem tertanam, yaitu perangkat yang sistem operasinya biasanya dibangun di dalam firmware dan sangat disesuaikan dengan sistem. Ini termasuk router, kontrol otomatisasi, televisi, perekam video digital, konsol gim video, dan jam tangan pintar. Banyak ponsel pintar dan komputer tablet menjalankan Android dan turunan Linux lainnya. Karena dominasi Android pada smartphone, Linux memiliki basis instalasi terbesar dari semua sistem operasi serba guna.

Linux adalah salah satu contoh paling menonjol dari kolaborasi perangkat lunak bebas dan sumber terbuka. Kode sumber dapat digunakan, dimodifikasi, dan didistribusikan — secara komersial atau non-komersial — oleh siapa pun di bawah ketentuan lisensi masing-masing, seperti Lisensi Publik Umum GNU.

2.2.6 Docker dan docker-compose

Docker adalah program komputer yang melakukan virtualisasi tingkat sistem operasi. Ini pertama kali dirilis pada 2013 dan dikembangkan oleh Docker, Inc. Docker digunakan untuk menjalankan paket perangkat lunak yang disebut container. container diisolasi dari satu sama lain dan menggabungkan aplikasi mereka sendiri, alat, perpustakaan dan file konfigurasi; mereka dapat berkomunikasi satu sama lain melalui saluran yang jelas. Semua kontainer dijalankan oleh kernel sistem operasi tunggal dan karenanya lebih ringan dari mesin virtual. container dibuat dari gambar yang menentukan konten tepatnya.

Gambar sering dibuat dengan menggabungkan dan memodifikasi gambar standar yang diunduh dari repositori publik.

Docker dikembangkan terutama untuk Linux, di mana ia menggunakan fitur isolasi sumber daya dari kernel Linux seperti cgroups dan ruang nama kernel, dan sistem file yang dapat digunakan oleh serikat pekerja seperti OverlayFS dan lainnya untuk memungkinkan kontainer independen berjalan dalam satu contoh Linux, menghindari overhead memulai dan memelihara mesin virtual (VM). Dukungan kernel Linux untuk ruang nama sebagian besar, mengisolasi pandangan aplikasi dari lingkungan operasi, termasuk pohon proses, jaringan, ID pengguna dan sistem file yang dipasang, sementara cgroup kernel menyediakan pembatasan sumber daya untuk memori dan CPU. Sejak versi 0.9, Docker menyertakan libcontainer library sebagai caranya sendiri untuk secara langsung menggunakan fasilitas virtualisasi yang disediakan oleh kernel Linux, selain menggunakan antarmuka virtualisasi abstrak melalui libvirt, LXC dan systemd-nspawn.

Membangun di atas fasilitas yang disediakan oleh kernel Linux (terutama cgroups dan namespaces), container Docker, tidak seperti mesin virtual, tidak memerlukan atau menyertakan sistem operasi yang terpisah. Sebagai gantinya, ia bergantung pada fungsionalitas kernel dan menggunakan isolasi sumber daya untuk CPU dan memori, dan memisahkan ruang nama untuk mengisolasi pandangan aplikasi dari sistem operasi. Docker mengakses fitur-fitur virtualisasi kernel Linux baik secara langsung menggunakan perpustakaan libcontainer, yang tersedia pada Docker 0.9, atau secara tidak langsung melalui libvirt, LXC (Linux Containers) atau systemd-nspawn.

Sistem container membuat pengembangan menjadi lebih fleksibel, sistem yang terisolasi serta pemisahan setiap environment atau modul tidak mempengaruhi service-service lain yang telah berjalan, serta keamanan pada lokalisasi service dan scaling yang mudah.

Compose (docker-compose) merupakan alat untuk mendefinisikan dan menjalankan aplikasi yang kompleks dengan Docker. Dengan Compose, Anda mendefinisikan aplikasi multi-container dalam satu file, lalu memutar aplikasi Anda dalam satu perintah tunggal yang melakukan semua yang perlu dilakukan untuk menjalankannya. Dalam hal ini lebih praktis dalam menjalankan sebuah service karena dalam satu file dapat meringkas banyak service container.

2.2.7 Git

Git adalah sistem versioning-control yang terdistribusi dari sumber terbuka (opensource) dan gratis, git dirancang untuk melakukan penanganan segala sesuatu mulai dari proyek kecil hingga besar dengan cepat dan efisien. Ada beberapa contoh versioning-control lain selain git seperti CVS, SVN, Mercurial dan Fossil, akan tetapi Git lebih populer atau menduduki peringkat paling atas dari versioning-control yang lain. Git berfungsi sebagai pondasi untuk banyak layanan seperti Github, Gitlab, atau Bitbucket, yang telah menyediakan server repository cloud. Namun beberapa layanan Git dapat kita replikasi menjadi self-host service namun ada beberapa fitur yang pasti berbeda.

Git melakukan proses snapshot pada suatu proyek yang kita kerjakan, dan menyimpannya dalam versi yang unik. Jika ada suatu perubahan yang itu berdampak pada suatu proyek maka kita dapat melakukan rollback ke perubahan yang sebelumnya, dan melakukan memperbaiki kesalahan yang telah di buat. Jika pengembang berkolaborasi, maka ketika pengembang lain mengirim perubahan, pengembang utama dapat menggabungkan perubahan itu ke dalam cabang kerja, dan kemudian pengembang lain dapat mengambil versi proyek yang digabungkan dan terus bekerja dari versi yang baru. Git memungkinkan pengembang untuk mengelola banyak potensi varian karya tunggal, mempertahankan history semua perubahan, dan bahkan memungkinkan untuk versi paralel.

2.2.8 Telegram Bot

Telegram Bot adalah akun telegram yang dioperasikan oleh perangkat lunak, Telegram Bot pada umumnya digunakan untuk wadah sebuah AI. Telegram Bot dapat diintegrasikan dengan layanan microservice atau bahkan meneruskan perintah menjadi Internet Of Things (IoT). Telegram Bot muncul pada 24 Juni 2015 dan sampai sekarang masih dalam proses pengembangan dari pihak telegram. Bot Telegram adalah aplikasi yang di-host di server yang menggunakan API bot Telegram untuk terhubung ke klien Telegram Messenger. Bot Telegram terhubung ke pengguna yang menggunakan pesan teks dan panggilan balik inline yang dienkapsulasi sebagai json. Pengguna dapat mengunggah foto / suara / video ke bot atau mengunduhnya dari bot ke komputer / ponsel mereka. Aplikasi bot yang berjalan di server bisa apa saja, dari aplikasi percakapan sederhana, mesin pencari yang kuat, perpustakaan multimedia yang besar, mesin pemecahan masalah hingga apa pun yang dapat Anda bayangkan. Keuntungan besar dari Telegram bot adalah bahwa mereka tidak memiliki persyaratan pemasangan dan berjalan tanpa terlihat pada semua platform komputer tempat Telegram Messenger dijalankan (Windows, Mac, Linux, Android, iOS, dan semua Browser Web). Dibawah ini adalah beberapa fitur dari Bot Telegram

- Authorizing Bot

Setiap bot diberikan token autentikasi unik ketika dibuat. Tokennya terlihat seperti 123456: ABC-DEF1234ghIkl-zyx57W2v1u123ew11, tetapi pihak telegram hanya akan menggunakan <token> dalam dokumen ini. Pengembang dapat mempelajari tentang mendapatkan token dan membuat token baru dalam situs <https://core.telegram.org/bots/api>

- Pembuatan request

Semua request ke API Bot Telegram harus menggunakan protocol HTTPS dan perlu disajikan dalam endpoint ini:

https://api.telegram.org/bot<token>/METHOD_NAME.

Seperti pada contoh di bawah ini:

<https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe>

Respons berisi objek JSON, yang selalu memiliki bidang Boolean 'ok' dan dapat memiliki bidang String opsional 'deskripsi' dengan deskripsi hasil yang dapat dibaca manusia. Jika 'ok' sama dengan benar, permintaan berhasil dan hasil query dapat ditemukan di bidang 'hasil'. Jika permintaan tidak berhasil, 'ok' sama dengan false dan kesalahannya dijelaskan dalam 'uraian'. Bidang Integer 'error_code' juga dikembalikan, tetapi isinya dapat berubah di masa mendatang. Beberapa kesalahan mungkin juga memiliki bidang opsional 'parameter' dari tipe ResponseParameters, yang dapat membantu menangani kesalahan secara otomatis.

Jika pengembang menggunakan webhook, pengembang dapat melakukan permintaan ke Bot API saat mengirim jawaban ke webhook. Gunakan tipe konten aplikasi / json atau application / x-www-form-urlencoded atau multipart / form-data untuk parameter yang lewat. Tentukan metode yang akan dipanggil dalam parameter metode permintaan. Tidak mungkin mengetahui bahwa permintaan seperti itu berhasil atau mendapatkan hasilnya.

- Melakukan Update

Ada dua cara yang saling eksklusif untuk menerima pembaruan untuk bot - metode getUpdates di satu sisi dan Webhooks di sisi lain. Pembaruan yang masuk disimpan di server sampai bot menerimanya dengan cara apa pun, tetapi pembaruan itu tidak akan disimpan lebih dari 24 jam. Terlepas dari

opsi mana yang pengembang pilih, pengembang akan menerima objek Pembaruan berseri JSON sebagai hasilnya. Objek ini mewakili pembaruan yang masuk. Paling banyak salah satu parameter opsional dapat hadir dalam setiap pembaruan yang diberikan.

Tabel 2.2: Telegram Bot Parameter

Field	Tipe	Deskripsi
update_id	Integer	Pengidentifikasi unik pembaruan. Pengidentifikasi pembaruan dimulai dari angka positif tertentu dan meningkat secara berurutan. ID ini menjadi sangat berguna jika Anda menggunakan Webhooks, karena memungkinkan pengembang untuk mengabaikan pembaruan berulang atau untuk mengembalikan urutan pembaruan yang benar, jika mereka keluar dari urutan. Jika tidak ada pembaruan baru selama setidaknya satu minggu, maka pengidentifikasi pembaruan berikutnya akan dipilih secara acak alih-alih secara berurutan.
message	Message	Pilihan. Pesan masuk baru apa pun - teks, foto, stiker, dll.
edited_message	Message	Pilihan. Versi baru dari pesan yang diketahui oleh bot dan telah diedit.

channel_post	Message	Pilihan. Versi baru dari pesan yang diketahui bot dan telah diedit. Opsional. Posting saluran masuk baru apa pun - teks, foto, stiker, dll.
edited_channel_post	Message	Pilihan. Versi baru dari posting saluran yang diketahui oleh bot dan telah diedit.
inline_query	InlineQuery	Pilihan. Permintaan inline masuk baru
chosen_inline_result	ChosenInlineResult	Pilihan. Hasil kueri sebaris yang dipilih oleh pengguna dan dikirim ke mitra obrolan mereka. Silakan lihat dokumentasi kami tentang pengumpulan umpan balik untuk detail tentang cara mengaktifkan pembaruan ini untuk bot Anda.
callback_query	CallbackQuery	Pilihan. Permintaan panggilan balik masuk baru.
shipping_query	ShippingQuery	Pilihan. Permintaan pengiriman masuk baru. Hanya untuk faktur dengan harga fleksibel.
pre_checkout_query	PreCheckoutQuery	Pilihan. Permintaan pra-checkout yang masuk baru. Berisi informasi lengkap tentang checkout.

2.3 Deskripsi Tempat KKI

PT. Dian Nuswantoro Teknologi dan Informasi merupakan perusahaan di bidang teknologi komputerisasi yang berdiri sejak tahun 2006. PT Dian Nuswantoro Teknologi dan Informasi ini lebih dikenal dengan nama PT Dinustek berfokus pada pelayanan tentang informasi dan teknologi (IT) , Layanan jaringan internet (Internet Service Provider) dan layanan pembuatan aplikasi (Software Development).

Informasi tentang PT Dinustek :

Alamat : Jl. Arjuna No 36 ,Kota Semarang, Jawa Tengah, Indonesia.
Telepon : (024) 3568492
Email : marketing@dinustek.com
Website : www.dinustek.com

2.3.1 Logo dan Makna Tempat KKI



Gambar 2.2: Logo PT Dian Nuswantoro Teknologi dan Informasi

Logo PT Dinustek terbagi menjadi 3 (tiga) bagian yaitu :

1. Simbol Api

Simbol api berwarna oranye melambangkan solusi untuk klien yang bermaksud memberikan penyelesaian masalah terbaik dari sisi teknologi dan informasi bagi bisnis klien.

2. Warna Biru

Warna biru melambangkan solusi teknologi informasi yang kami tawarkan dapat dipercaya kehandalan bagi klien.

3. Bentuk Huruf

Huruf-huruf yang tersusun berbentuk small caps melambangkan keramah-tamahan perusahaan kepada klien, dan nuansa kekeluargaan di perusahaan. Bentuk huruf yang dinamis namun kuat menggambarkan tim perusahaan yang kreatif dan solid.

2.3.2 Struktur Organisasi Tempat KKI

Struktur organisasi PT Dinustek:

1. Direktur Utama

Mohamad Sidiq, S.Si., M. Kom

2. Direktur

Dr. Pulung Nurtantio Andono, ST, M. Kom

3. Manager of Software Department

Abu Salam, M. Kom

4. Kepala Divisi Stratup Software

Moh. Tofa Nurzaki, S. Kom

2.3.3 Visi dan Misi Tempat KKI

Visi : Menjadi perusahaan teknologi informasi dan komunikasi kelas dunia.

Misi : Memberikan solusi keseluruhan teknologi informasi dan komunikasi untuk klien.

BAB III METODE PENELITIAN

3.1 Instrumen Penelitian

Penyusunan suatu proyek penelitian, sesuai dengan data-data permasalahan pokok yang dihadapi. Data dapat dikatakan baik apabila data tersebut dapat mewakili objek yang sedang diteliti dan untuk mendapatkan data yang baik diperlukan metode atau serangkaian cara yang sesuai dengan kebutuhan penelitian. Dalam penyusunan sistem penyesuaian infrastruktur dengan data-data dari objek penelitian yang dipakai. Data dikatakan baik apabila data tersebut dapat mewakili object dari yang sedang diteliti dan untuk mendapatkan data yang baik diperlukan metode atau serangkaian cara sesuai dengan kebutuhan peneliti.

3.2 Prosedur Pengambilan Data

Untuk mendapatkan data yang benar, akurat dan relevan serta sesuai dengan sumber data dan tujuan penyusunan Laporan Kuliah Kerja Industri ini, maka peneliti dalam pengumpulan data menggunakan beberapa teknik antara lain adalah sebagai berikut :

3.2.1 Observasi

Teknik pengumpulan data dengan cara melakukan pengamatan yang terlibat langsung pada infrastruktur server Tukutu, dalam sistem pengembangan Bot Telegram serta pemilahan bahasa pemrograman yang tepat dan efisien dalam Tim Pengembangan Bot Telegram Tukutu yang berada dibawah pengawasan PT Dinustek.

Observasi dilakukan untuk melihat potensi Bot untuk penyesuaian penggunaan pada sistem Continuous Integration dan Delivery, Dengan mengumpulkan apa saja yang berhubungan dengan server, service, bot serta environment yang tersedia.

3.2.2 Studi Literatur

Melakukan berbagai pencarian terhadap teori-teori dan literatur yang dapat digunakan sebagai dasar melakukan penelitian. Serta mempelajari penelitian yang pernah dilakukan sebelumnya sehingga dapat lebih memahami teori yang ada.

3.2.3 Forum Diskusi dan Tutorial

Teknik pengumpulan data dengan cara membuat sebuah forum diskusi, yang berisikan para pengembang sistem. Peneliti membuka diskusi pada group chat tim pengembang aplikasi Tukutu, guna mengetahui masalah yang sebenarnya dialami oleh tim pengembang. Secara rutin melakukan pertemuan dengan pengembang aplikasi Tukutu untuk mendiskusikan bagaimana cara membuat Aplikasi Tukutu bisa melakukan rolling update dengan memanfaatkan Telegram Bot sebagai integrasi sistem dan metode deployment.

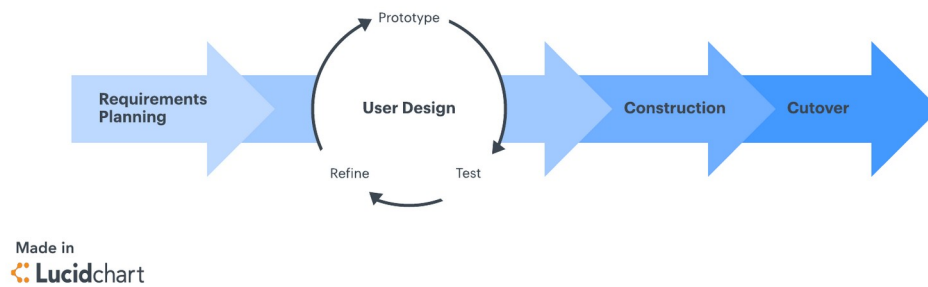
3.3 Teknik Analisis Data

Data yang diperoleh dari literatur, forum diskusi dan Observasi akan diolah sesuai dengan bagian – bagian dari sistem Bot. Dari hasil analisa yang didapatkan akan dilakukan analisa masalah dengan pengembangan fitur yang cocok sebagai solusi dari masalah tersebut, sehingga diharapkan fitur baru yang dibuat akan mampu menyelesaikan masalah yang ada dan berfungsi dengan baik, serta untuk melakukan perbaikan jika terdapat suatu bug/error.

3.4 Metode

Dalam proses mengembangkan Bot Telegram Tukutu ini tim pengembang menggunakan metode RAD (Rapid Application Development) dengan mempersiapkan skenario test case sebelum membuat fungsional code. Berikut ini merupakan tahapan - tahapan yang akan dilakukan dalam penerapan metode RAD:

Rapid Application Development (RAD)



Gambar 3.1: Rapid Application Development Flow

3.4.1 Rapid Application Development (RAD)

Berdasarkan model RAD diatas dapat diuraikan pembahasan masing - masing tahapan sebagai berikut:

- Requirement Planing

Pada tahap ini pengembang melakukan observasi untuk perancangan pada kebutuhan sistem perangkat lunak atau Bot dan untuk menggambarkan area-area dimana definisi lebih jauh untuk iterasi selanjutnya.

- User Design

Pada tahap User Design pengembang harus dapat menentukan rancangan secara detail dari spesifikasi, fitur, serta alur program dengan jelas, sehingga pada tahap selanjutnya dapat langsung dilakukan implementasi program. Terdapat beberapa sub-modul pada poin ini, yaitu:

- Prototyping

Pengembang menciptakan beberapa versi pada program untuk dilakukan pengujian program mana yang akan lebih baik untuk digunakan dan dikembangkan kedepannya.

- Testing

Setiap prototype yang sudah dibuat akan dilakukan pengetesan secara manual oleh pengembang dan juga beberapa user.

- Refine

Prototype yang telah dilakukan pengetesan akan di saring akan di kembangkan menjadi prototype baru, serta untuk prototype yang lulus dalam proses penyaringan akan dilakukan pengembangan lebih lanjut.

- Construction

Pada tahapan ini pengembang akan membangun sistem dengan para pengguna secara intensif dengan melakukan testing dan merancang aspek - aspek bisnis dan non teknis perusahaan. Apabila sudah ditemukan aspek aspek yang telah disetujui maka sistem akan segera di bangun dan disaring, sistem - sistem baru atau bagian dari sistem di ujicoba lalu kemudian diperkenalkan kepada pengguna.

- Cutover

Ini adalah tahap implementasi di mana produk jadi diluncurkan. Ini mencakup konversi data, pengujian, dan penggantian ke sistem baru, serta pelatihan pengguna. Semua perubahan akhir dilakukan saat pengembang dan pengguna terus mencari bug di sistem.

3.4.2 Perancangan Sistem Bot

Perancangan Sistem merupakan tahapan awal yang harus dilakukan pada proses RAD ketika mengerjakan sebuah fitur, spesifikasi, deskripsi dan juga detail kebutuhan serta hasil yang diinginkan harus jelas dalam implementasi. Pada Bot Telegram Tukutu akan terdapat beberapa fitur/perintah Bot utama yaitu:

- **converting text-to-script** : bot mampu melakukan konversi dari pesan yang dikirim oleh user kemudian mentranslasi kan pesan tersebut menjadi sebuah command-line pada server.
- **filtering user** : bot dapat melakukan pengecekan pada user yang mengirim sebuah pesan, jika user tidak terdaftar pada environment Bot Telegram Tukutu maka bot akan menolak akses dan akan mengirim log pada super User yang telah di definisikan CHAT_ID nya pada environment.
- **self update** : pembaruan Sistem Bot Telegram lewat bot itu sendiri
- **healt check** : pengecekan bot apakah dia posisi shutdown/running, jika bot terjadi sebuah kesalahan/bug/error akan ada pengecekan setiap 1 jam yang akan melakukan restart Bot Telegram Secara otomatis, apabila bot tidak terjadi apa - apa maka log akan tertulis dengan keterangan proses Bot Telegram masih berjalan
- **log service** : Bot Telegram akan mengecek log dengan memanfaatkan git, dan akan mengirim history commit 10 baris terakhir dari repository yang ada di remote server.

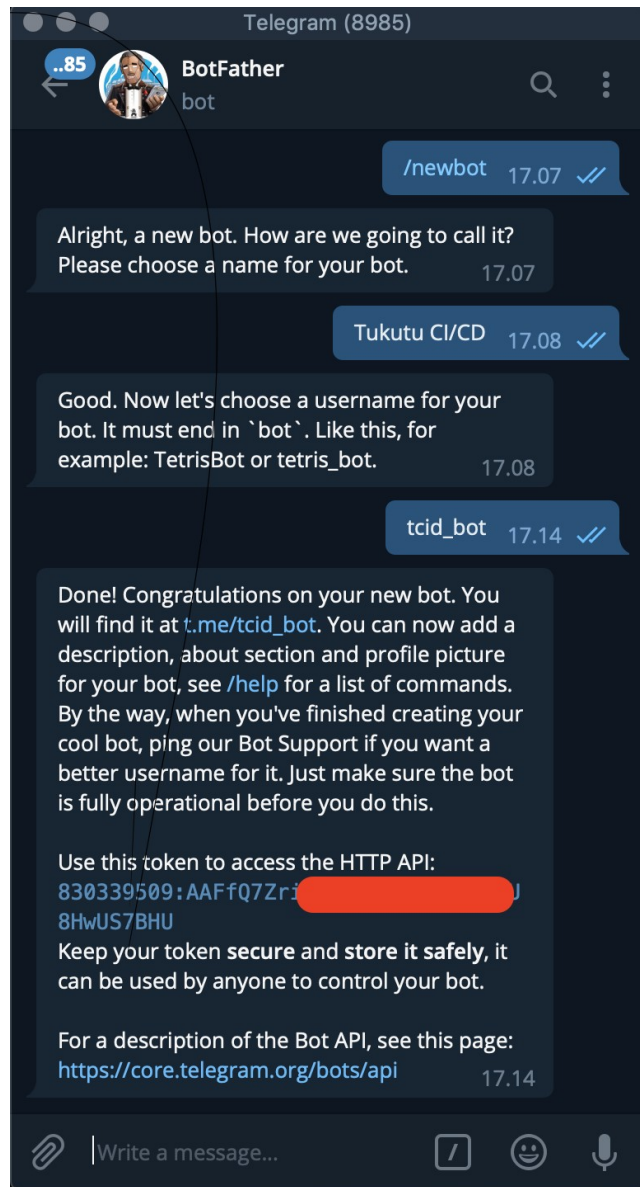
- update service : yang meliputi alur melakukan pembaruan pada service yang ada pada server produksi/development dengan meliputi sebuah script bash sebagai otomasi.

Berdasarkan data fitur-fitur tersebut akan dibuat suatu skenario pengujian dengan ketentuan bahwa setiap sub-fitur akan dibuatkan setidaknya dua kondisi pengujian yaitu kondisi pengujian positif dan negatif. Dua kondisi ini diperlukan agar dapat dipastikan bahwa fungsi yang dibuat dapat menangani ketika seluruh parameter yang diperlukan benar dan dapat mengantisipasi jika parameter yang dikirim ada yang salah.

3.4.3 Pembuatan Akun Telegram Bot

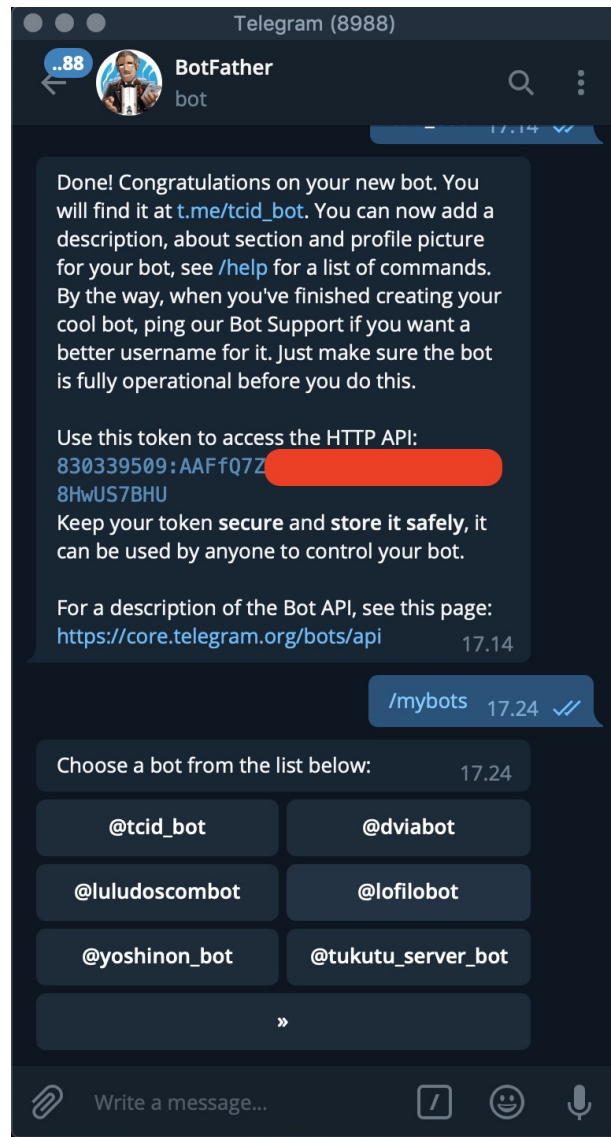
Telegram Bot merupakan sebuah fitur yang ada pada Aplikasi Telegram yang dapat kita manfaatkan secara gratis, pembuatan Bot Telegram dapat dilakukan pada Aplikasi Telegram. Untuk membangun sistem untuk Bot yang terintegrasi dengan Telegram kita membutuhkan sebuah TELEGRAM BOT TOKEN, yang kita bisa dapatkan dengan membuat bot secara gratis pada Telegram BotFather (@BotFather).

Untuk mendapatkan TELEGRAM BOT TOKEN Langkah pertama yaitu membuat bot dengan perintah /newbot pada BotFather, kemudian BotFather akan menanyakan nama yang akan ditampilkan pada Bot, sebagai sample peneliti mengisinya dengan nama TUKUTU CI/CD, jika sudah BotFather akan menanyakan lagi username untuk Bot tersebut, karena ini username maka apabila terjadi pengambilan username yang sudah di gunakan BotFather akan memberitahu jika username tersebut sudah di gunakan dan juga ketentuan username pada bot sekarang harus memiliki sebuah kata yang mengandung kata bot. Sebagai Contoh terdapat pada gambar dibawah



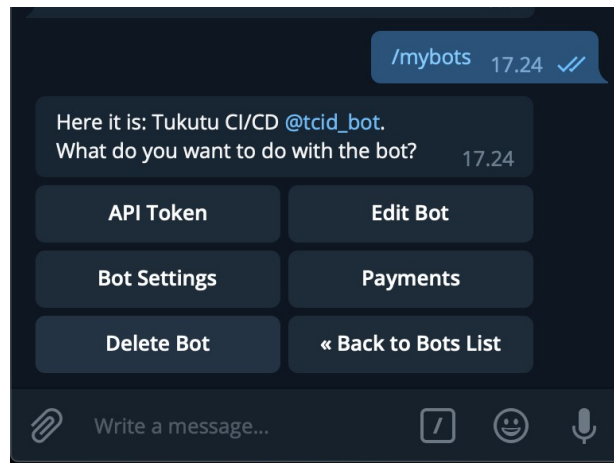
Gambar 3.2: Proses Pembuatan Bot

Dari gambar diatas pengembang sudah membuat Telegram Bot dan mendapatkan sebuah TELEGRAM BOT TOKEN untuk mengakses Telegram Bot API. BotFather menyediakan fitur untuk manajemen Bot, pengembang bisa menggunakan perintah `/mybots` untuk mengetahui Bot apa saja yang sudah dibuat.



Gambar 3.3: Menu List Bot

Untuk mengatur Bot yang telah dibuat, pengembang dapat menekan username Bot yang tersedia.



Gambar 3.4: Menu Detail Bot

3.4.4 Pendefinisian Telegram Bot Token

Untuk dapat menggunakan service Telegram Bot API diperlukan sebuah Telegram Bot Token, setiap Bot memiliki Token yang unik, token didefinisikan pada sebuah file .env atau file yang berisi nilai environment - environment yang bersifat dynamic atau berubah - ubah, seperti pada gambar dibawah.

```

1 TBOT_TOKEN=
2 ADMIN=
3 CHAT_ID=
4

```

Gambar 3.5: Environment Bot


```
/**
 * @param {required} node_modules
 */
const TelegramBot = require('telegram-bot-nova')
const util = require('util')
const {
  exec
} = require('child_process');

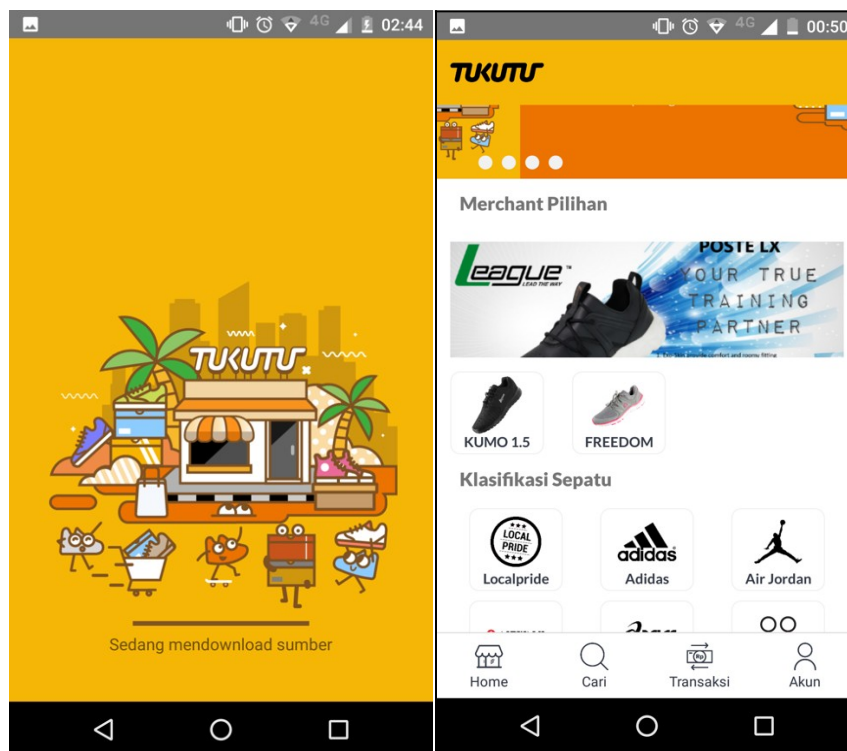
/**
 * @define {bot}
 */
var bot = new TelegramBot(env.token)
const bash = '/bin/bash'
```

Gambar 3.6: Pendefinisian Bot Token

BAB IV IMPLEMENTASI DAN PEMBAHASAN

4.1 Gambaran Umum Sistem

4.1.1 Tukutu

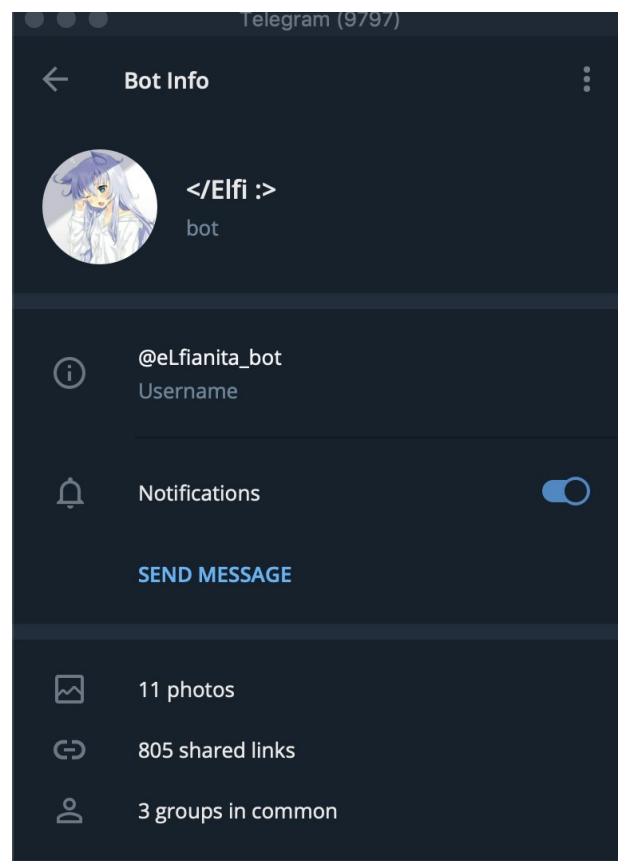


Gambar 4.1: Aplikasi Tukutu

PT Dian Nuswantoro Teknologi adalah perusahaan pengembangan perangkat lunak berbasis web maupun mobile, Aplikasi Tukutu dikembangkan di PT Dinustek. Tukutu merupakan aplikasi yang digunakan untuk jual beli sepatu secara online dengan target merket yaitu pecinta sepatu, yang ingin membeli atau mungkin menjual sepatu koleksi mereka ke masyarakat umum.

Tujuan pengembangan Aplikasi Tukutu lebih untuk membantu para pemilik merk sepatu lokal untuk dapat memasarkan produk mereka sehingga diharapkan dapat bersaing dengan merk internasional yang sudah lebih dahulu dikenal oleh masyarakat luas. Platform Mobile dipilih karena sebagian besar target market place pengguna merupakan pengguna Aplikasi smartphone.

4.1.2 Telegram Bot Tukutu



Gambar 4.2: Tentang Telegram Bot

Telegram Bot Tukutu adalah Bot Telegram yang dikembangkan di PT Dinustek yang digunakan untuk melakukan remote terhadap server tukutu, dan juga sebagai alat untuk melakukan integrasi kelanjutan dan pemasangan. selain itu

bot dapat mentranslasikan sebuah text menjadi command line pada server apa bila terjadi hal yang tidak di inginkan dalam keadaan darurat Bot dapat menjadi jalan pertama dalam penanganan server. Fokus pada Bot ini sendiri adalah untuk menjalankan Update pada Microservice yang terintegrasi dengan Git.

Telegram Bot Tukutu dikembangkan dengan teknologi NodeJS yang di kombinasi dengan Shell Script, pesan yang dikirim dari Telegram Bot Tukutu di translasikan menjadi sebuah command line, pada Telegram Bot Tukutu terdapat 2 jenis perintah, yaitu pertama perintah langsung (terdefinisi) merupakan perintah yang sudah diatur menjadi perintah default dan menjadi perintah sekali jalan, kemudian yang kedua adalah perintah kostum (custom) merupakan perintah yang diciptakan dari kreasi System Administrator itu sendiri. Kedua perintah tersebut berawal dari sebuah pesan string yang dikirim dari Telegram Bot Tukutu kemudian di pecah dari perintah utama dan perintah parameter, kemudian text atau String yang sudah tertrlansasi menjadi sebuah perintah Command Line akan di jalankan dalam interpreter Shell yang akan mengeluarkan Stdout dan Stderr, Stdout merupakan keluaran dari perintah yang dijalankan dan Stderr merupakan keluaran berupa pesan error apabila terjadi sebuah error dari perintah yang dijalankan.

4.2 Work Flow

Penggambaran Work Flow Telegram Bot Tukutu seperti pada gambar no. [no_gambar], Bot berjalan pada latar belakang yang melakukan long polling dengan menunggu request dari pengguna Bot. Request dari pengguna yang dilewatkan sebuah API (Application Programming Interface) dari Telegram Bot berupa sebuah pesan text atau String. Bot membaca text kemudian dilakukan sebuah Parsing Text pada proses belakang, memecah text menjadi beberapa

bagian kemudian melakukan eksekusi sebuah perintah dari pesan text yang di kirim oleh pengguna.

4.2.1 Identifikasi Aktor

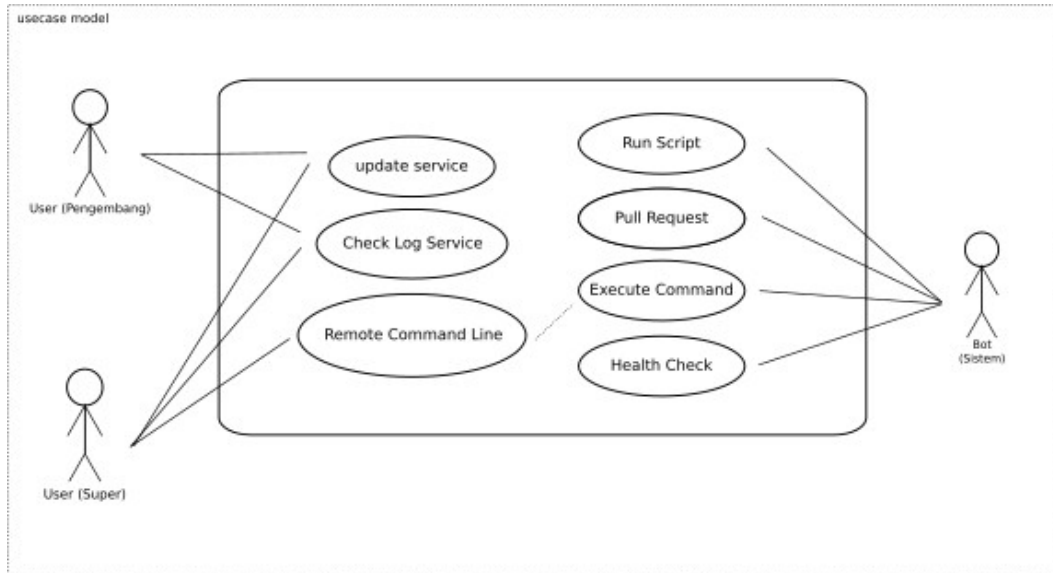
Aktor pada Telegram Bot Tukutu terdiri atas tiga aktor yaitu User(pengembang), User(Super), dan Bot(Sistem). Berikut ini merupakan penjelasan dari tiap aktor yang terlibat.

Tabel 4.1: Identifikasi Aktor

Aktor	Deskripsi
User (Pengembang)	merupakan aktor yang dapat melakukan pengecekan pada microservice yang berjalan pada server tukutu dan dapat melakukan update pada tiap - tiap service yang sedang berjalan
Bot (Sistem)	merupakan aktor yang berperan dalam sistem, dari melakukan pengecekan, pembaruan, serta sistem kerja
User (Super)	merupakan user yang memiliki kendali penuh terhadap Bot itu sendiri, User (Super) dapat melakukan pembaruan terhadap service, melakukan pengecekan, memasukan perintah langsung ke dalam server dengan sebuah pesan yang dapat diterjemahkan menjadi sebuah perintah

4.2.2 Use Case Diagram

4.2.2.1 Use Case



Gambar 4.3: Diagram Use Case

Dari use case diagram diatas, maka dapat dijelaskan terdapat beberapa aktor dimana aktor tersebut memiliki peran masing - masing. Pada aktor User (Pengembang) dapat melakukan pengecekan pada microservice yang sedang berjalan dan melakukan pembaruan pada service. Lalu pada aktor User (Super) merupakan aktor yang memiliki kendali penuh terhadap Bot, User (Super) akan mendapatkan notifikasi secara pribadi dari Bot jika terdapat pengguna yang tidak terdaftar mencoba menggunakan Bot, User (Super) juga dapat melakukan perintah seperti pada User (Pengembang) dan sebagai tambahan User (Super) dapat melakukan perintah langsung pada Bot yang perintah tersebut akan di eksekusi pada server. Sedangkan Bot (Sistem) memiliki peran dalam proses latar belakang dari input User (Pengembang) dan juga User (Super), seperti proses dalam melakukan pembaruan microservice dan pengechekan service yang sedang

berjalan, mentranslasikan perintah dari User (Super) menjadi sebuah command line yang dapat di eksekusi ke server.

4.2.2.2 Use Case Naratif

Use case naratif digunakan sebagai dokumentasi untuk menjelaskan langkah-langkah yang terjadi pada setiap interaksi dari use case yang telah dibuat. Berikut adalah use case naratif:

Nama Use Case	Aktor	Tujuan	Deskripsi
Update Service	User (Pengembang)	Pembaruan Service	User (Pengembang) dapat melakukan pembaruan sistem pada sebuah service yang berjalan diserver Tukutu
Check Log Service	User (Pengembang)	Pengecekan commit Log	User (Pengembang) dapat melakukan pengecekan commit log yang terdapat pada repository Git
Update Service	User (Super)	Pembaruan Service	User (Super) dapat melakukan pembaruan sistem pada sebuah service yang berjalan diserver Tukutu
Check Log Service	User (Super)	Pengecekan commit Log	User (Super) dapat melakukan

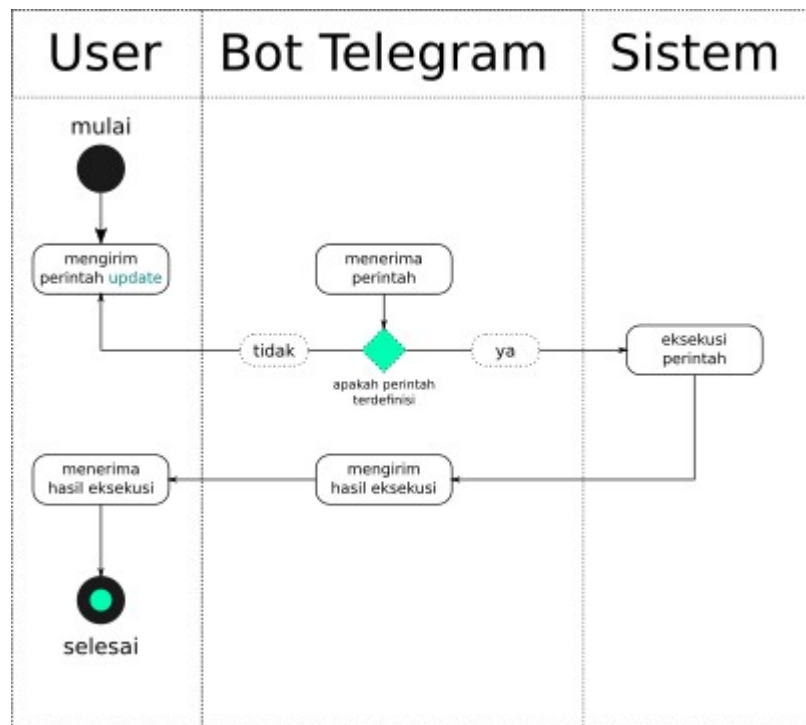
			pengecekan commit log yang terdapat pada repository Git
Remote Command Line	User (Super)	Eksekusi Perintah ke Server	User (Super) dapat melakukan remote command pada server menggunakan Bot, Bot akan mentranslasikan pesan dari User (Super) ke dalam Shell Script
Script Runner	Bot (Sistem)	Eksekusi Shell Script	Bot dapat mengeksekusi Shell Script yang telah dikembangkan oleh Administrator
Pulling Repo	Bot (Sistem)	Pembaruan Source Code	Bot dapat melakukan pembaruan Source Code pada service
Message Translator	Bot (Sistem)	Translasi Pesan ke Shell Command	Bot (Sistem) dapat menerjemahkan pesan menjadi sebuah perintah shell pada server dengan syarat User menambahkan symbol "/" pada pesan yang dikirim di awal kalimat. dan juga perintah

			tersebut harus tersedia pada server.
Healt Check	Bot (Sistem)	Pengechekan Bot	Bot (Sistem) dapat melakukan pengechekan terhadap dirinya sendiri, apabila terjadi sebuah kegagalan sistem maka Bot (Sistem) akan otomatis mengulang kembali proses latar belakang pada dirinya sendiri.

4.2.3 Activity Diagram

Pada Tahapan ini akan dijelaskan Alur Activity Diagram pada Telegram Bot Tukutu dari proses Pengguna sampai pada Proses Latar Belakang Bot.

4.2.3.1 Activity Diagram Pembaruan Service

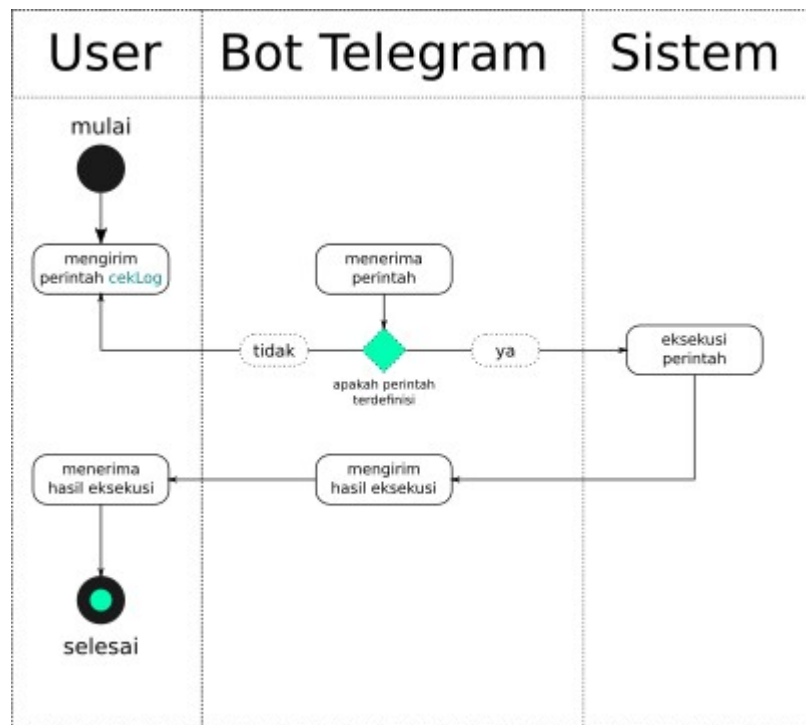


Gambar 4.4: Flow Diagram Update Service

Pada proses Pembaruan Service yaitu User mengirimkan sebuah pesan Text kepada Telegram Bot Tukutu, pengembang hanya memiliki akses pada perintah yang sudah tersedia pada Telegram Bot Tukutu. pengembang hanya mengirimkan perintah seperti `"/update_service"` kemudian bot akan melakukan eksekusi pada sebuah shell script yang sudah di buat, shell script tersebut memiliki bebrapa inti perintah, yaitu :

- pembaruan commit log.
- pembaruan package atau modul jika ditemukan package atau modul baru.
- migrasi table jika terdeteksi migrasi table baru pada database.
- pembersihan konfigurasi cache pada project.
- pemuatan ulang service pada kontainer.

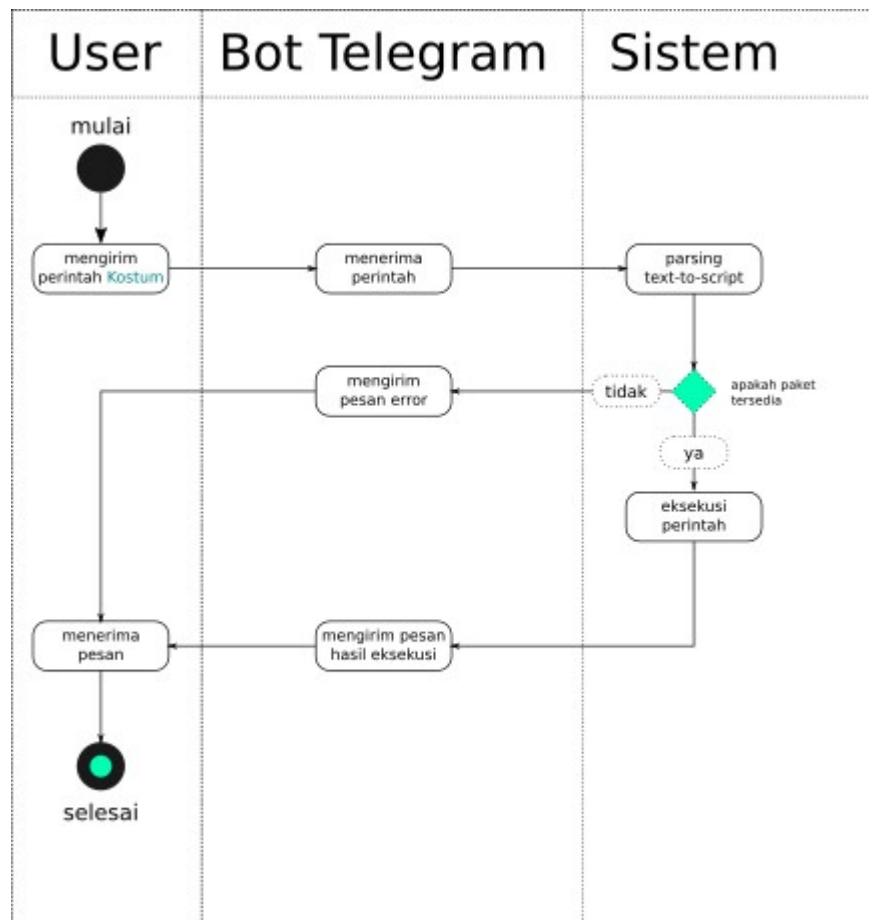
4.2.3.2 Activity Diagram Pengecekan Log Service



Gambar 4.5: Flow Diagram Pengecekan Log Service

Proses pengecekan Log Service diatas User mengirimkan sebuah perintah `/log_service` kepada Telegram Bot Tukutu, proses ini memiliki kesamaan dengan Pembaruan Service, namun pada Proses ini Bot hanya melakukan pengecekan pada commit log repository Git pada service yang ingin di lihat. Bot melakukan eksekusi Shell Script yang sudah ada dan Script memiliki satu inti perintah yaitu pengecekan commit log pada repository.

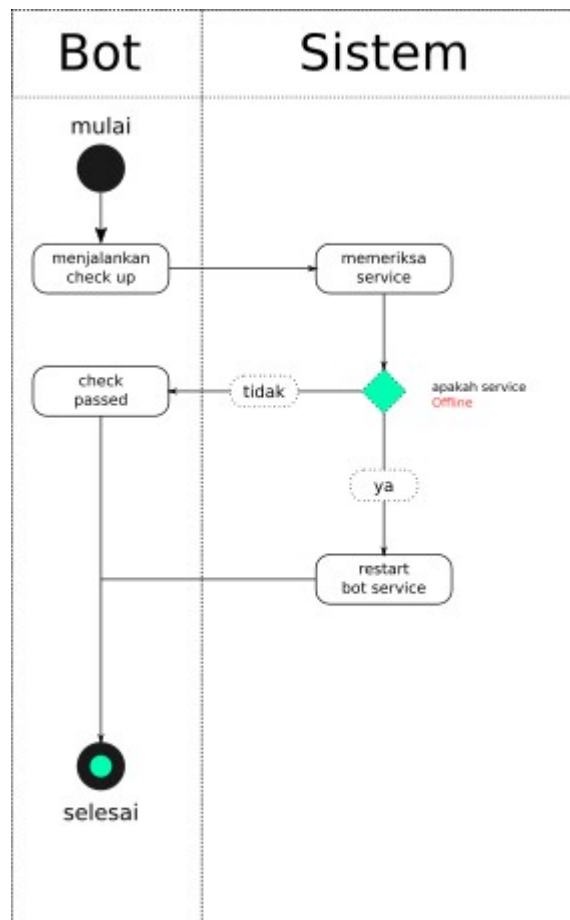
4.2.3.3 Activity Diagram Remote Command Line



Gambar 4.6: Flow Diagram Remote Command Line

Proses remote Command Line merupakan sebuah fitur dari Telegram Bot Tukutu yang tersedia untuk melakukan eksekusi perintah langsung pada server dengan mengirimkan sebuah perintah dengan diawali symbol `"/perintah"` dimana perintah tersebut merupakan binary file atau sebuah package yang sudah tersedia pada server.

4.2.3.4 Activity Diagram Health Check

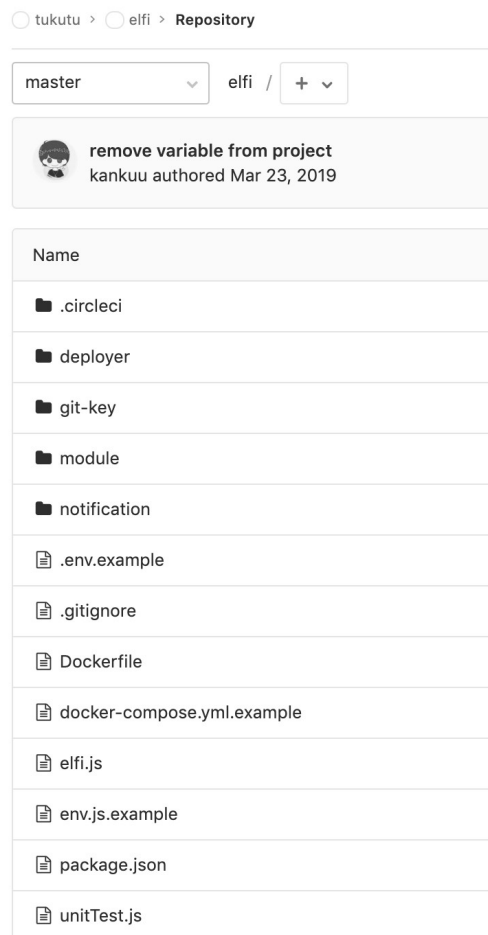


Gambar 4.7: Flow Diagram Health Check

Health Check berjalan pada latar belakang yang dilakukan penjadwalan setiap satu jam sekali, terdapat sebuah Shell Script yang akan dijalankan untuk melakukan pengecekan pada service Telegram Bot Tukutu yang berjalan di latar belakang. script ini hanya memiliki 2 kondisi yaitu online dan offline. jika service Telegram Bot Tukutu terdeteksi pada posisi Online maka tidak ada tindakan yang akan dilakukan, jika posisi service Telegram Bot Tukutu terdeteksi dalam kondisi Offline maka script akan melakukan pengulangan kembali (restarting) service Telegram Bot Tukutu.

4.3 Implementasi Kode Program

4.3.1 Struktur Project Telegram Bot Tukutu



Gambar 4.8: Struktur Project

Gambar diatas merupakan struktur project dari Telegram Bot Tukutu, setiap folder diatas mendefinisikan sebuah paket dan memiliki peran masing - masing, berikut deskripsi peran terpenting:

4.3.1.1 .circleci

.circleci memiliki peran dalam continuous integration dari Telegram Bot Tukutu sendiri.

4.3.1.2 Deployer

terdapat 5 (lima) file pada paket ini yaitu :

- .gitignore
- lib.sh
- sandbox.sh
- updater.sh
- variable.sh.example

```

lib.sh 2.33 KB
1  #!/usr/bin/env bash
2  # this library of bash function representation of deployer
3  # kind of function is to pull repo, update dependency, and
4  # configuration set of project
5  #
6  # maintainable by <abas> akhmadbasir5@gmail.com
7
8  set -e
9
10 # function to update project
11 function update_repo () {
12     # $git_dir=$1 # git directory
13     # $git_work_tree=$2 # git work tree
14     # $remote=$3 # remote repositories
15     # $branch=$4 # branch to to update
16
17     echo "updating repo.."
18     GIT_PULL=$(git --git-dir=$1 --work-tree=$2 pull $3 $4 > /dev/null 2>&1 ; echo $?)
19     if [[ $GIT_PULL -gt 0 ]]; then
20         echo "error pulling resource, errcode : $GIT_PULL"
21         echo "EOP"
22     else
23         echo "source updated!"
24     fi
25 }
26
27 # function to install dependency with composer
28 function composer_install () {
29     # $service_path=$1 # the working dir
30
31     COMPOSER_INSTALL=$(composer --working-dir=$1 install --no-dev > /dev/null 2>&1 ; echo $?)
32     if [[ $COMPOSER_INSTALL -gt 0 ]]; then
33         echo "composing failed, errcode : $COMPOSER_INSTALL"
34         echo "EOP"
35     else
36         echo "dependency updated"
37     fi
38 }

```

Gambar 4.9: Shell Script Lib.sh I

```

39
40 # database migration : update table, using docker instead
41 function artisan_migrate_force () {
42     # $service_name=$1 # name of container service
43
44     docker exec -i $1 php artisan migrate --force
45     if [[ $(docker exec -i $1 php artisan migrate --force > /dev/null 2>&1 ; echo $? ) -gt 0 ]];then
46         echo "something wrong here"
47     else
48         echo "table migrated!"
49     fi
50 }
51
52 # dockerizer config restart queue and config clear
53 function artisan_config_cache () {
54     # $service_name=$1 # name of service or container name
55
56     docker exec -i $1 php artisan config:cache
57 }
58 function artisan_config_clear () {
59     # $service_name=$1 # name of service or container name
60
61     docker exec -i $1 php artisan config:clear
62 }
63 function artisan_queue_restart () {
64     # $service_name=$1 # name of service or container name
65
66     docker exec -i $1 php artisan queue:restart
67 }
68
69 # restarting docker container service
70 function docker_compose_restart () {
71     # $service_name=$1 # name of service or container name
72     # $service_path=$2 # path project
73
74     RESTART_CONTAINER=$(docker-compose -f $2/docker-compose.yml restart > /dev/null 2>&1 ; echo $? )
75     if [[ $RESTART_CONTAINER -gt 0 ]];then
76         echo "something wrong when restarting container $1"
77     else
78         echo "container successfully restarted"
79     fi
80 }

```

Gambar 4.10: Shell Script Lib.sh II

lib.sh berisikan berbagai fungsi yang digunakan sebagai kebutuhan service Telegram Bot Tukutu. lib.sh dapat dikatakan sebagai modul shell Script yang memiliki berbagai fungsi, dimana fungsi - fungsi tersebut yang nanti akan di panggil dalam perintah yang di minta.


```

1  #!/usr/bin/env bash
2  # this library of bash function representation of deployer
3  # kind of function is to pull repo, update dependency, and
4  # configuration set of project
5  #
6  # maintainable by <abas> akhmadbasir5@gmail.com
7
8  set -e
9  source $(find . -name lib.sh)
10 source $(find . -name variable.sh)
11
12 update_option=$1
13 case "$update_option" in
14     "--dashboard" | "-d")
15         # dashboard update statement
16         echo "updating on dashboard service"
17         update_repo $DASHBOARD_SERVICE_PATH/.git $DASHBOARD_SERVICE_PATH $DASHBOARD_REMOTE $DASHBOARD_BRANCH
18         composer_install $DASHBOARD_SERVICE_PATH
19         artisan_config_cache $DASHBOARD_SERVICE_NAME
20         artisan_config_clear $DASHBOARD_SERVICE_NAME
21         docker_compose_restart $DASHBOARD_SERVICE_NAME $DASHBOARD_SERVICE_PATH
22         ;;
23     "--mainservice" | "-ms")
24         # mainservice update statement
25         echo "updating on main service"
26         update_repo $MAIN_SERVICE_PATH/.git $MAIN_SERVICE_PATH $MAIN_REMOTE $MAIN_BRANCH
27         composer_install $MAIN_SERVICE_PATH
28         artisan_migrate_force $MAIN_SERVICE_NAME
29         artisan_config_cache $MAIN_SERVICE_NAME
30         artisan_config_clear $MAIN_SERVICE_NAME
31         artisan_queue_restart $MAIN_SERVICE_NAME
32         docker_compose_restart $MAIN_SERVICE_NAME $MAIN_SERVICE_PATH
33         ;;
34     "--userservice" | "-us")
35         # userservice update statement
36         echo "updating on user service"
37         update_repo $USER_SERVICE_PATH/.git $USER_SERVICE_PATH $USER_REMOTE $USER_BRANCH
38         composer_install $USER_SERVICE_PATH
39         docker_compose_restart $USER_SERVICE_NAME $USER_SERVICE_PATH
40         ;;
41     "--merchantservice" | "-cs")
42         # merchantservice update statement
43         echo "updating on merchant service"
44         update_repo $MERCHANT_SERVICE_PATH/.git $MERCHANT_SERVICE_PATH $MERCHANT_REMOTE $MERCHANT_BRANCH
45         composer_install $MERCHANT_SERVICE_PATH
46         docker_compose_restart $MERCHANT_SERVICE_NAME $MERCHANT_SERVICE_PATH
47         ;;

```

Gambar 4.11: Shell Script Updater I

```

48     "--logservice" | "-ls")
49         # logservice update statement
50         echo "updating on log service"
51         update_repo $LOG_SERVICE_PATH/.git $LOG_SERVICE_PATH $LOG_REMOTE $LOG_BRANCH
52         composer_install $LOG_SERVICE_PATH
53         docker_compose_restart $LOG_SERVICE_NAME $LOG_SERVICE_PATH
54     ;;
55     "--coprofile" | "-cp")
56         # company profile update statement
57         echo "updating company profile"
58         update_repo $COPROFILE_SERVICE_PATH/.git $COPROFILE_SERVICE_PATH $COPROFILE_REMOTE $COPROFILE_BRANCH
59         composer_install $COPROFILE_SERVICE_PATH
60         artisan_migrate_force $COPROFILE_SERVICE_NAME
61         artisan_config_cache $COPROFILE_SERVICE_NAME
62         artisan_config_clear $COPROFILE_SERVICE_NAME
63         docker_compose_restart $COPROFILE_SERVICE_NAME $COPROFILE_SERVICE_PATH
64     ;;
65     "--yoshinon" | "-ys")
66         # yoshinon update statement
67         echo "updating yoshinon"
68         update_repo $YOSHINON_SERVICE_PATH/.git $YOSHINON_SERVICE_PATH $YOSHINON_REMOTE $YOSHINON_BRANCH
69     ;;
70 *)
71     echo "nothing option $update_option yet, see the manual below"
72     echo "=="
73     echo "\$ updater [option]"
74     echo "=="
75     echo "option:"
76     echo "  --dashboard      -d      update dashboard service"
77     echo "  --mainservice     -ms     update main service"
78     echo "  --userservice     -us     update user service"
79     echo "  --merchantservice -cs     update mserchant service"
80     echo "  --coprofile       -cp     update company profile service"
81     echo "  --yoshinon        -ys     update yoshinon"
82     ;;
83 esac

```

Gambar 4.12: Shell Script Updater II

updater.sh memiliki peran sebagai main script yang akan mengeksekusi perintah berdasarkan parameter yang diberikan, parameter yang diberikan oleh User menentukan perintah apa yang harus dieksekusi, dan updater sh akan memanggil fungsi - fungsi yang ada pada lib.sh dan juga memanggil variable yang tersedia pada variable.sh.

```

variable.sh.example 523 Bytes
1 DASHBOARD_SERVICE_NAME=
2 DASHBOARD_SERVICE_PATH=
3 DASHBOARD_REMOTE=
4 DASHBOARD_BRANCH=
5
6 COPROFILE_SERVICE_NAME=
7 COPROFILE_SERVICE_PATH=
8 COPROFILE_REMOTE=
9 COPROFILE_BRANCH=
10
11 MAIN_SERVICE_NAME=
12 MAIN_SERVICE_PATH=
13 MAIN_REMOTE=
14 MAIN_BRANCH=
15
16 USER_SERVICE_NAME=
17 USER_SERVICE_PATH=
18 USER_REMOTE=
19 USER_BRANCH=
20
21 MERCHANT_SERVICE_NAME=
22 MERCHANT_SERVICE_PATH=
23 MERCHANT_REMOTE=
24 MERCHANT_BRANCH=
25
26 LOG_SERVICE_NAME=
27 LOG_SERVICE_PATH=
28 LOG_REMOTE=
29 LOG_BRANCH=
30
31 YOSHINON_SERVICE_NAME=
32 YOSHINON_SERVICE_PATH=
33 YOSHINON_REMOTE=
34 YOSHINON_BRANCH=
35

```

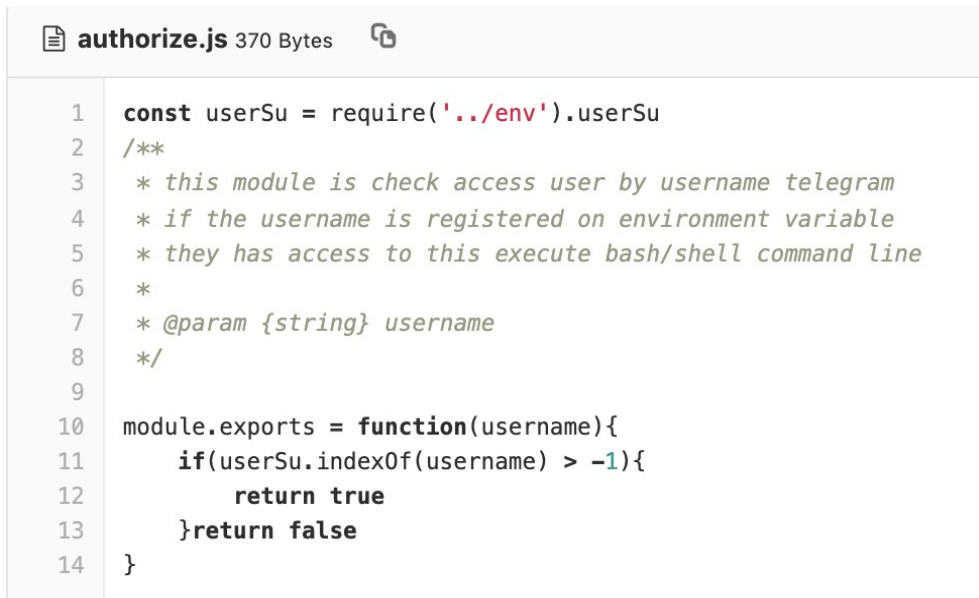
Gambar 4.13: Variable.sh

variable.sh.example merupakan file yang berisikan variable - variable yang disediakan pada tiap service, variable.sh.example merupakan file yang merupakan file yang dinamis harus dipisahkan, variable.sh.example akan di ubah penamaan mennjadi variable.sh dan isi dari tiap - tiap variable harus diisi dengan nilai yang mutlak, karena variable disini berperan penting dalam penentuan jalannya script.

4.3.1.3 Module

paket module berisikan modul - modul yang tertulis dengan bahasa pemrograman javascript native, pemisahan module dilakukan untuk kemudahan dalam penggunaan ulang, ada beberapa module terpenting sebagai berikut :

- authorize.js
- git.js
- log.js
- splitter.js



```

1  const userSu = require('../env').userSu
2  /**
3   * this module is check access user by username telegram
4   * if the username is registered on environment variable
5   * they has access to this execute bash/shell command line
6   *
7   * @param {string} username
8   */
9
10 module.exports = function(username){
11     if(userSu.indexOf(username) > -1){
12         return true
13     }return false
14 }

```

Gambar 4.14: Logika Authtorize Module

module authorize.js berfungsi sebagai pengecekan pengguna Telegram Bot Tukutu, merupakan otentikasi sederhana dengan membandingkan list user yang terdefinisi dengan user yang sedang melakukan chat dengan Bot, jika user terdeteksi tidak dalam list pengguna Telegram Bot Tukutu maka Bot akan tidak melakukan apapun pada pesan user tersebut tapi Bot akan mengirimkan sebuah

pesan notifikasi pada User (Super) bahwa user yang tidak terdaftar tersebut telah melakukan interaksi dengan Bot.



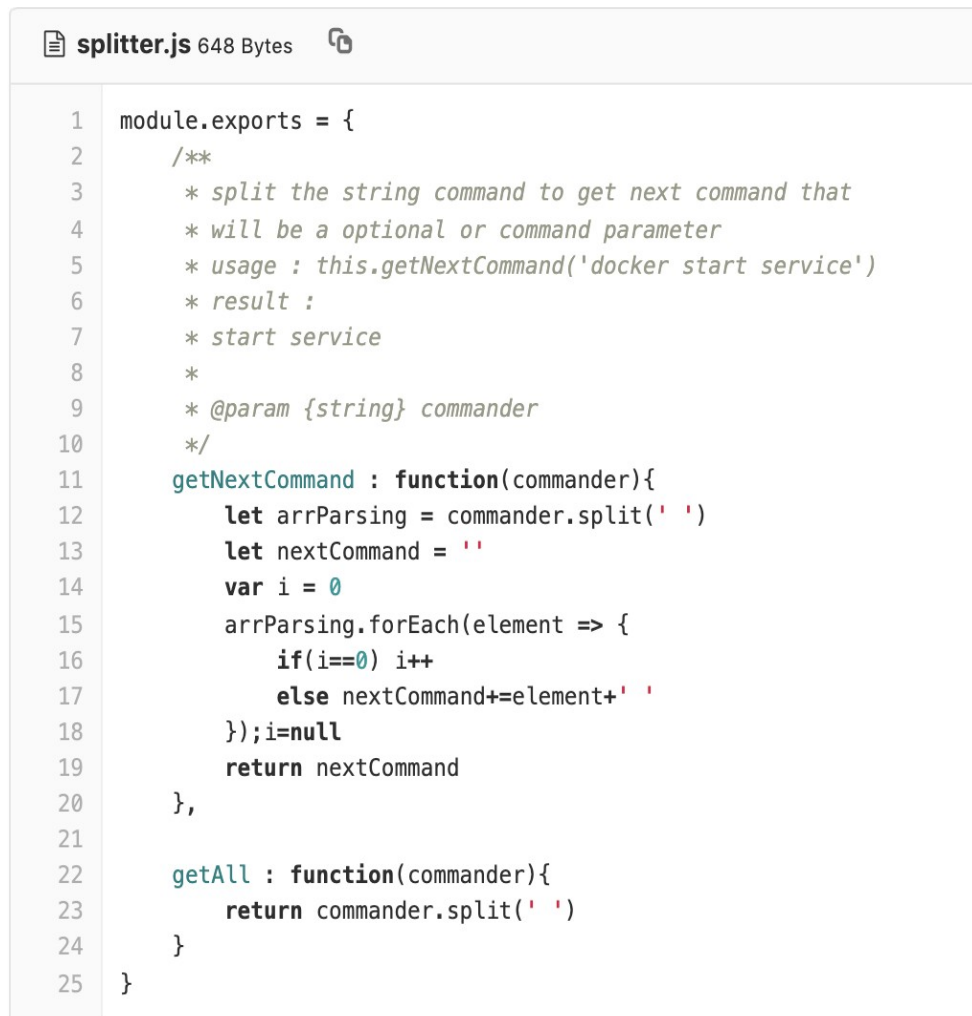
```

1  module.exports = {
2    /**
3     * to show log of project dir with how many line to show up
4     * usage : this.log(1, '/root/project/.git', '/root/project')
5     * result:
6     * 76fcxxx commit_log
7     *
8     * @param {integer} manyLine
9     * @param {string} gitDir
10    * @param {string} workTree
11    */
12    log : function(manyLine, gitDir, workTree){
13      return '--no-pager'+
14        ' --git-dir='+gitDir+
15        ' --work-tree='+workTree+
16        ' log --oneline -n '+manyLine
17    },
18  }
19 }

```

Gambar 4.15: Logika Git.js Module

module git.js berfungsi untuk pengecekan commit log pada git repository local server, untuk memudahkan pada proses pemanggilan git.js menyediakan fungsi dengan sebuah parameter directory yang dituju.



```

1  module.exports = {
2    /**
3     * split the string command to get next command that
4     * will be a optional or command parameter
5     * usage : this.getNextCommand('docker start service')
6     * result :
7     * start service
8     *
9     * @param {string} commander
10   */
11   getNextCommand : function(commander){
12     let arrParsing = commander.split(' ')
13     let nextCommand = ''
14     var i = 0
15     arrParsing.forEach(element => {
16       if(i==0) i++
17       else nextCommand+=element+' '
18     });i=null
19     return nextCommand
20   },
21
22   getAll : function(commander){
23     return commander.split(' ')
24   }
25 }

```

Gambar 4.16: Logika Splitter.js Module

module splitter.js memiliki fungsi sebagai pemecah text atau String, pesan yang dikirim dari pengguna akan dipecah berdasarkan spasi, dan diambil binary package dari kata pertama yang tersedia.

4.3.1.4 Notification

Dalam paket Notification terdapat fungsi untuk melakukan Heath Checking, proses pengecekan pada diri Telegram Bot Tukutu itu sendiri dengan

menggunakan penjadwalan secara berkala yaitu satu jam sekali, berikut merupakan gambaran script health check pada service Telegram Bot Tukutu.



```
1  #!/usr/bin/env bash
2  if [ $(pm2 status | grep stopped | grep elfi > /dev/null 2>&1 ;echo $? ) -eq 0 ];then
3      echo "restarted elfi on $(date)" >> elfi/log/restart.log
4      pm2 restart elfi --watch
5  else
6      echo "healty check passed!"
7  fi
```

Gambar 4.17: Shell Script Restart Bot Service

4.3.1.5 Elfi.js

Selain 4 (empat) Paket diatas, ada main program utama yang menjalankan semua proses tersebut, yaitu elfi.js. Elfi.js merupakan main program yang berjalan di latar belakang menggunakan proses NodeJS, elfi.js berjalan dengan metode long polling, melakukan listen request pada dengan integrasi Telegram Bot API. elfi.js pada Telegram Bot Tukutu memiliki fungsi utama yaitu :

- execute function

```

34  /**
35   * @param {string} first_command
36   * @param {string} following_command
37   */
38  function execute(cmd, fCmd, chat_id) {
39    let script = cmd + ' ' + fCmd
40    log.resMessage(script)
41    exec(script, (error, stdout, stderr) => {
42      if(stderr.includes('/bin/sh') && stderr.includes('not found')){
43        stderr = "i am sorry, i dont know what to do >.<"
44      }
45      if (error) {
46        // log.resExecErr(error)
47        log.resStdout(stdout)
48        log.resStderr(stderr)
49        if(stderr != null){
50          bot.sendText(chat_id, stderr)
51        }
52        bot.sendText(chat_id, stdout)
53      } else if (stdout == '' || stdout == null) {
54        bot.sendText(
55          chat_id,
56          'command execute with stdout `null` result'
57        )
58        log.resMessage('execute with null output')
59      } else {
60        if(stderr != null){
61          bot.sendText(chat_id, stderr)
62        }
63        bot.sendText(chat_id, stdout)
64        log.resStdout(stdout)
65      }
66    })
67  }

```

Gambar 4.18: Logika Execute Elfi.js

execute function merupakan fungsi untuk mengkonversi pesan dari User, pesan diterjemahkan dalam perintah shell script dan kemudian akan dieksekusi pada server.

pada beberapa kasus dalam perintah server, ada perintah - perintah yang harus dihindari, seperti melakukan perintah poweroff, halt, reboot, etc perintah - perintah yang seharusnya tidak boleh di eksekusi oleh Telegram Bot Tukutu. dalam hal tersebut elfi.js akan melakukan pengecekan pada sebuah kondisi apakah pesan yang berupa perintah ke server bukan merupakan perintah yang sensitif, dijelaskan pada kode berikut.

```
restart.sh 234 Bytes
1  #!/usr/bin/env bash
2  if [ $(pm2 status | grep stopped | grep elfi > /dev/null 2>&1 ;echo $? ) -eq 0 ];then
3      echo "restarted elfi on $(date)" >> elfi/log/restart.log
4      pm2 restart elfi --watch
5  else
6      echo "healthy check passed!"
7  fi
```

Gambar 4.19: Logika Retristed Command

- update service

```
case 'main_update':
    execute(bash, update+' --mainservice')
    bot.sendMessage(chat.id,'updating.... #main_service',{replyToMessageId:messageId})
    bot.sendChatAction(chat.id,'typing',(status)=>{
        if(status.Error){
            bot.sendMessage(chat.id,"hmm, cant you check my logs? there must be error there.")
        }else{
            setTimeout(()=>{
                bot.sendMessage(chat.id,'source has been updated, check the last commit',{
                    disable_notification: false,
                    disable_web_page_preview: true,
                    replyToMessageId: messageId
                },()=>{
                    execute('git',git.log(
                        6,env.service.main+'/.git',env.service.main
                    ),chat.id)
                })
            },5000)
        }
    })
    break
```

Gambar 4.20: Logika Update Service

Gambar diatas merupakan salah satu kode yang akan menjalankan perintah pembaruan sistem pada service Tukutu, terjadi beberapa proses pada opsi tersebut, yaitu pembaruan pada latar belakang kemudian setelah pembaruan akan dilakukan pengecekan commit log, dan hasil dari commit log akan di kirimkan kepada user.

- log service

```
case 'glmain':
  bot.sendChatAction(chat.id, 'typing', ()=>{
    bot.sendText(chat.id, "history of commit log", {
      replyToMessageId: messageId
    }, ()=>{
      execute('git', git.log(
        6, env.service.main+'/.git', env.service.main
      ), chat.id)
    })
  })
break
```

Gambar 4.21: Logika Pengecekan Log Service

Gambar diatas merupakan salah satu kode yang akan menjalankan perintah pengecekan commit log pada service Tukutu. Bot akan melakukan pengecekan dari History commit log, di ambil 5 history terakhir yang akan di kirimkan kembali kepada user.

- Unauthorize User

```

} else {
  let unAuthorizeUsername = 'this user : @' + from.username + '\n' +
    'try to accessing bot with bash/shell command line!\n' +
    'follow this link https://t.me/' + from.username +
    ' to contact this user.\n\n'+
    'command : '+command
  if(command === 'start'){
    bot.sendText(chat.id, 'wasup?')
    bot.sendText(env.chat_id, 'user : @'+from.username+' starting me!')
    log.resExec(from.username, command)
  }else{
    /**
     * if user send command that no have access
     */
    bot.sendText(chat.id, ysnRes.noAccess)
    log.resMessage(unAuthorizeUsername)
    // reporting message to env.chat_id : @kankuu telegram user admin
    bot.sendText(env.chat_id, unAuthorizeUsername)
  }
}
return
`

```

Gambar 4.22: Logika Unauthorize User Check

Jika terdapat user yang mencoba mengakses Telegram Bot Tukutu secara private, maka User (Super) akan mendapatkan notifikasi, dari kode diatas, Bot juga akan menulis Log pada sistem.

4.4 Pengujian

Setelah pengkodean sistem dan sistem (dalam hal ini adalah aplikasi) sudah dapat digunakan. Maka sistem/aplikasi tersebut harus diuji terlebih dahulu. Pada penelitian kali ini penulis menggunakan teknik pengujian :

4.4.1 White Box Testing

White Box merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kasus-kasus uji [19]. Menggunakan metode ujicoba ini dapat menghasilkan kasus-kasus uji coba seperti:

1. Menjamin bahwa seluruh independent paths dalam modul telah dilakukan sedikitnya satu kali.
2. Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah.
3. Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya.
4. Menguji struktur data internal untuk memastikan validitasnya.

4.4.1.1 Pseudocode

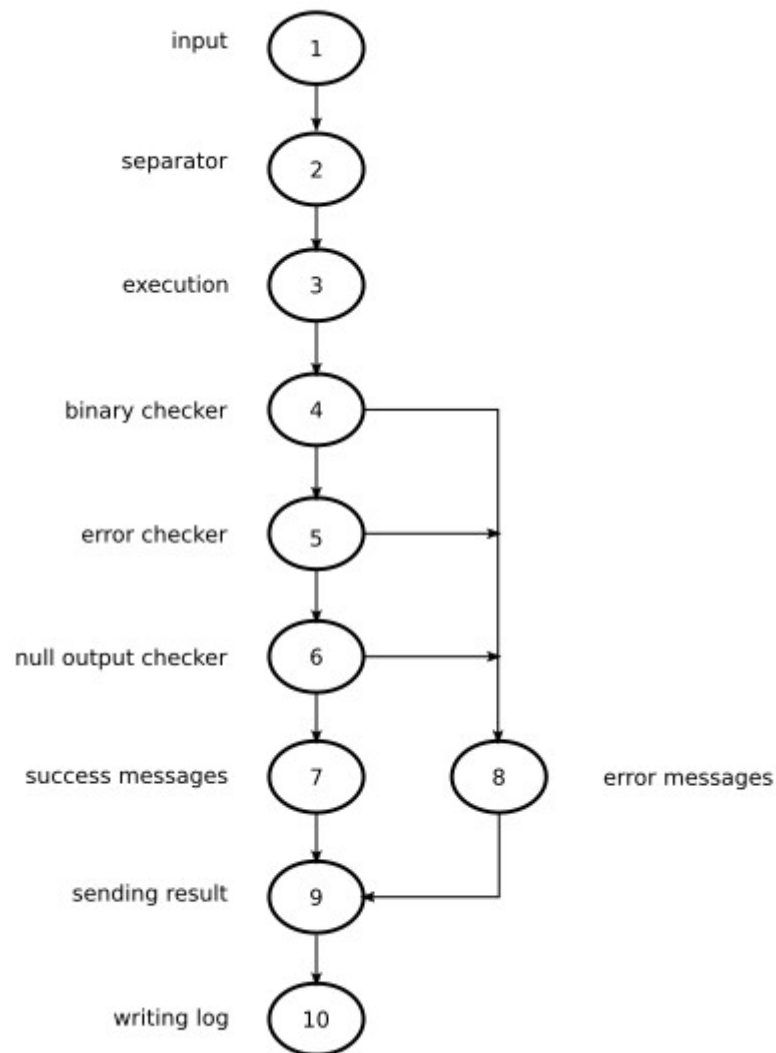
```
function execute (cmd, fCmd, chat_id)
  script <- cmd + ' ' + fCmd
  write log from script
  start exec script begin
    anonim function (error, stdout, stderr)
      if stderr includes /bin/sh, and stderr includes not found then
        stderr <- "i am sorry, i dont know what to do"
      endif

      if error then
        write log from stdout
        write log from stderr
        if stderr != null then
          bot send text notif stderr
        endif
        bot send text stdout

      else if stdout was "", or stdout was null then
        bot send text 'command execute with stdout `null` result'
```

```
write log from 'execute with null output'  
  
else  
  if stderr != null then  
    bot send text stderr  
  endif  
  bot send text stdout  
  write log from stdout  
  
endif  
end anonim function  
end exec script  
end function
```

4.4.2 Pembuatan Flowgraph



Gambar 4.23. Flowgraph Parsing Algorithm

4.4.2.1 Perhitungan Cyclomatic Complexity

Setelah flowchart dalam bentuk flowgraph, maka langkah selanjutnya adalah menghitung Cyclomatic complexity $V(G)$ dengan rumus :

$$V(G) = E - N + 2$$

Dimana:

E = jumlah *edge* pada grafik

N = jumlah *node* pada grafik

Dari flowgraph pada gambar 40 dapat dihitung cyclomatic complexitynya sebagai berikut :

$$V(G) = 12 - 10 + 2$$

$$V(G) = 4$$

Jadi dari perhitungan diatas didapatkan bahwa cyclomatic complexity dari flowgraph gambar 4.23 adalah 4.

4.4.2.2 Penentuan jalur independen

Dari perhitungan cyclomatic complexity tadi akan ditentukan jalur independen. Berdasarkan hasil perhitungan maka akan ada 4 jalur independen :

Jalur 1 = 1-2-3-4-8-9-10

keterangan : pengguna melakukan input berupa text command, kemudian di pisah menjadi 2 bagian perintah, pertama adalah perintah dasar dari binary file yang akan di jalankan. Jika binary yang di panggil tidak ada makan hasil output akan berupa error, dan bot akan mengirim error kepada pengguna dan diakhir dengan penulisan log error.

Jalur 2 = 1-2-3-4-5-8-9-10

keterangan : setelah melewati proses pengecekan binary, proses selanjutnya adalah melakukan pengecekan error execution script. Jika terjadi sebuah error bot akan mengirim error kepada pengguna dan diakhir dengan penulisan log error.

Jalur 3 = 1-2-3-4-5-6-8-9-10

keterangan : setelah melewati proses pengecekan error execution script selanjutnya adalah pengecekan null output, karena tidak semua output dari proses dikeluarkan oleh system. Maka jika terjadi null output bot akan mengirim pesan berupa null output kepada pengguna dan diakhir dengan penulisan log error.

Jalur 4 = 1-2-3-4-5-6-7-9-10

keterangan : jika input yang diberikan oleh pengguna tidak ada permasalahan maka proses execution script akan dianggap sukses oleh system, dan bot akan mengirim pesan log kepada pengguna dan diakhir dengan penulisan log error.

4.4.2.3 Hasil Test Case

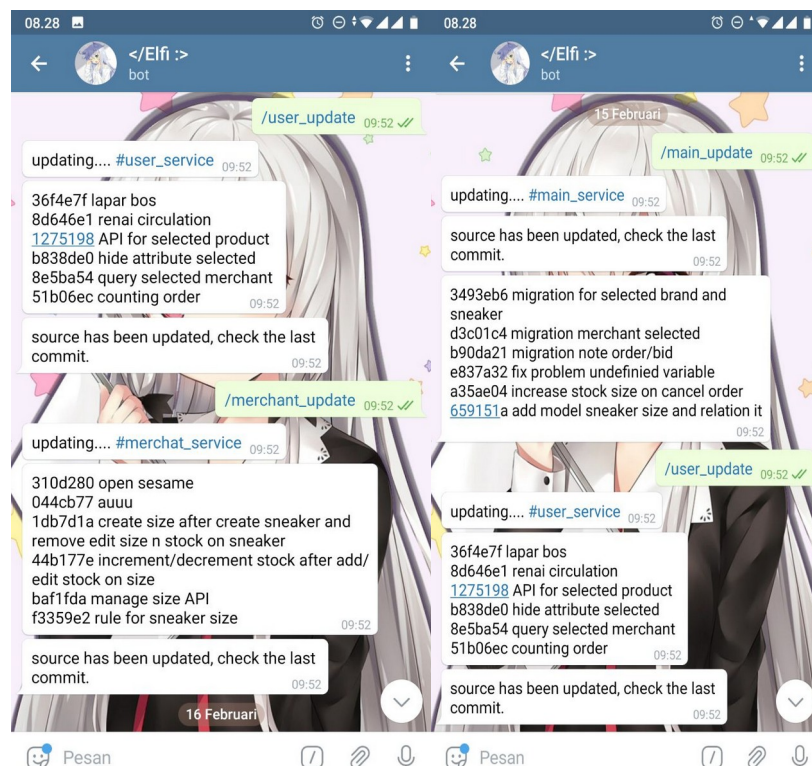
Hasil pengujian test case dapat dilihat pada tabel di bawah ini :

No	Test Case										Ketercapaian	
											Ya	Tidak
1	1	2	3	4	8	9	10				V	

2	1	2	3	4	5	8	9	10		V	
3	1	2	3	4	5	6	8	9	10	V	
4	1	2	3	4	5	6	7	9	10	V	

Tabel 4.2 Table Hasil Test Case White Box

4.4.3 Hasil Pengujian Pembaruan Service Tukutu



Gambar 4.24: Pengujian Pada Telegram Bot

Pengujian dilakukan pada Telegram Bot Elfi, response pertama yaitu Bot sedang melakukan Update dan diikuti dengan response commit log lima baris terakhir dari sumber kode.

4.4.4 Hasil Proses Pada Latar Belakang Service

```

/bin/bash /home/sochen/elixir/deployer/updater.sh --main-service
[log write], chat by :cipowela Fri Feb 15 2019 02:52:11 GMT+0000 (UTC) /main_update
git --no-pager --git-dir=/home/dev/dryer/main_service/.git --work-tree=/home/dev/dryer/main_service log --oneline -n 6
stdout :
3493eb6 migration for selected brand and sneaker
d3c01c4 migration merchant selected
b90da21 migration note order/bid
e837a32 fix problem undefined variable
a35ae04 increase stock size on cancel order
659151a add model sneaker size and relation it

stdout :
updating on main service
updating repo..
source updated!
dependency updated
Migrating: 2019_02_11_041834_add_costFee_and_selcected_merchant
Migrated: 2019_02_11_041834_add_costFee_and_selcected_merchant
Migrating: 2019_02_14_064526_add_table_column_selected_sneaker
Migrated: 2019_02_14_064526_add_table_column_selected_sneaker
Migrating: 2019_02_14_064556_add_table_column_selected_sneaker_brand
Migrated: 2019_02_14_064556_add_table_column_selected_sneaker_brand
table migrated!
Configuration cache cleared!
Configuration cached successfully!
Configuration cache cleared!
Broadcasting queue restart signal.
container successfully restarted

```

Gambar 4.25: Log Report Pada Pembaruan Service

Pada gambar diatas merupakan log hasil eksekusi dari perintah pembaruan service tukutu.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pembahasan yang terdapat pada bab sebelumnya dapat diambil kesimpulan sebagai berikut:

1. Project CI/CD menggunakan Telegram Bot dapat berjalan pada sistem operasi Linux, dengan memanfaatkan kombinasi antara Shell Script dan Bahasa pemrograman NodeJS.
2. Shell Script yang digunakan hanya support untuk sistem operasi linux
3. Project menggunakan bahasa pemrograman NodeJS dengan arsitektur Native Programming, sehingga kesesuaian sistem tergantung pada pengembang Project.

5.2 Saran

Penulis menyadari bahwa terdapat beberapa kekurangan dalam pembuatan aplikasi Tukutu dengan menggunakan bahasa pemrograman Kotlin, oleh sebab itu penulis memberikan beberapa saran:

Terdapat beberapa aspek kekurangan dalam CI/CD menggunakan Telegram Bot yang didasari dengan menggunakan bahasa pemrograman NodeJS, dikarenakan hal tersebut pengembang memberikan beberapa saran untuk pengembangan kedepannya:

1. Sebelum mengembangkan CI/CD menggunakan Telegram Bot, akan lebih baik jika pengembang memahami konsep dasar pada deployment service.
2. Penyempurnaan sistem pendefinisian variable pada Shell Script.

3. Beberapa library yang diambil dari node_modules bisa terdapat Vulnerability atau celah, alangkah baiknya jika pengembang selalu melakukan update pada library node_modules versi terbaru.
4. Buatlah dokumentasi tentang arsitektur pattern yang digunakan di dalam pembuatan project, agar dapat dipahami oleh anggota tim yang baru bergabung.

DAFTAR PUSTAKA

- Ilyas, Y. 2018. *PENGARUH HARGA, KUALITAS LAYANAN DAN TEKNOLOGI E-COMMERCE TERHADAP KEPUTUSAN UNTUK MENGGUNAKAN LAYANAN GO-JEK DI WILAYAH KECAMATAN CIBINONG*. *Economicus*, Vol. 9 No. 1.
- Hilton, M. & dkk. 2016. *Usage, Costs, and Benefits of Continuous Integration in Open-Source Projects*. ASE 16, September 3–7,.
- Arachchi, S.A.I.B.S. & Perera, I. 2018. *Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management*. Moratuwa Engineering Research Conference (MERCon).
- Nufusula, R. & Susanto, A. 2018. *Rancang Bangun Chat Bot Pada Server Pula Menggunakan Telegram Bot API*. *Journal of Information System*.
- Provos, N. & Honeyman, P. 2001. *ScanSSH – Scanning the Internet for SSH Servers*. 2001 LISA XV – December 2-7.
- Barrett, R. & dkk. 2004. *Field Studies of Computer System Administrators: Analysis of System Management Tools and Practices*. *Management of Computing and Information Systems*.
- Versluis, G. 2017. *Xamarin Continuous Integration and Delivery: Team Services, Test Cloud, and HockeyApp*. Apress.
- Nicolas, B. B. & dkk. 2018. *A spoonful of DevOps helps the GI go down*. ACM/IEEE 4th International Genetic Improvement Workshop.

- Rodríguez, P. & dkk. 2015. *Continuous Deployment of Software Intensive Products and Services: A Systematic Mapping Study*. The Journal of Systems & Software.
- Ylonen, T. 2006. *The Secure Shell (SSH) Protocol Architecture*. SSH Communications Security Corp.
- Sun, H. 2019. *Reasoning about the Node.js Event Loop using Async Graphs*. Washington, DC, USA Research Papers.
- Palacios, R. C. & dkk. 2017. *A case analysis of enabling continuous software deployment through knowledge management*. R. International Journal of Information Management.
- Satria, F. & dkk. 2018. *Pemantau Ruangan Menggunakan Raspberry Pi Terintegrasi Dengan Bot Telegram Dan Halaman Web*. Seminar Nasional Teknik Elektro.
- Valk, R. V. D. & dkk. 2018. *Transparency and Contracts: Continuous Integration and Delivery in the Automotive Ecosystem*. ACM/IEEE 40th International Conference on Software Engineering: Software Engineering in Practice.
- Macam-Macam Metode Pengembangan Perangkat Lunak*. 2019. [Online]. Available: <https://fatharaannisaa.wordpress.com/2014/08/22/macam-macam-metode-pengembangan-perangkat-lunak/> [Diakses 10-04-2019]
- Unix Shell*. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Unix_shell [Diakses 10-04-2019]
- Linux*. 2019. [Online]. Available: <https://en.wikipedia.org/wiki/Linux#History> [Diakses 10-04-2019]
- Docker glossary*. 2019. [Online]. Available: <https://docs.docker.com/glossary/?term=Compose> [Diakses 10-04-2019]

Berkenalan dengan DevOps. 2019. [Online]. Available: <https://www.codepolitan.com/berkenalan-dengan-devops-5ab7bbe4947b4> [Diakses 10-04-2019]

What is git ?. 2019. [Online]. Available: <https://opensource.com/resources/what-is-git> [Diakses 10-04-2019]