KEMENTERIAN DIGITAL
JABATAN DIGITAL NEGARA

DevOps Sektor Awam

PANDUAN

PELAKSANAAN DEVOPS DALAM PEMBANGUNAN SISTEM APLIKASI SEKTOR AWAM

Kementerian Digital
Jabatan Digital Negara

Panduan Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam

KANDUNGAN

SENARAI JADUAL

SENARAI RAJAH

PRAKATA
KETUA PENGARAH
JABATAN DIGITAL NEGARA

Assalamualaikum Warahmatullahi Wabarakatuh, Salam Sejahtera, Salam Malaysia MADANI.

Dengan nama Allah yang Maha Pemurah lagi Maha Mengasihani.

Alhamdulillah, syukur ke hadrat Allah yang Maha Esa kerana dengan limpah kurnia dan izin-Nya jua, maka dokumen Panduan Pelaksanaan DevOps bagi Pembangunan Sistem Aplikasi Sektor Awam ini telah berjaya diterbitkan. Pembangunan Panduan Pelaksanaan DevOps Sektor Awam ini adalah amat signifikan dan bertepatan dengan masa, seiring dengan penjenamaan semula Jabatan Digital Negara (JDN) dalam memacu pendigitalan perkhidmatan kerajaan ke arah penyampaian perkhidmatan kerajaan berpaksikan rakyat dan berpacukan data melalui pendekatan *Whole of Government* (WoG).

Saya ingin mengucapkan tahniah dan syabas kepada ahli pasukan projek yang telah bersungguh-sungguh memerah keringat dalam menjayakan penerbitan dokumen panduan ini. Penghasilan dokumen ini amat penting bagi memastikan pembangunan aplikasi kerajaan dapat dilaksanakan dengan lebih efisien melalui pendekatan baharu yang lebih holistik dan secara tidak langsung memberi kelebihan kepada kerajaan untuk memberikan yang terbaik kepada rakyat, lebih-lebih lagi dalam situasi global yang semakin mencabar. Situasi ini memerlukan penjawat awam bertindak dengan lebih responsif dan inovatif dalam menyampaikan perkhidmatan kepada pelanggan.

Sebagai usaha merealisasikan inisiatif pendigitalan dalam penyampaian perkhidmatan, kerajaan telah melancarkan Rangka Tindakan Ekonomi Digital Malaysia bagi

mentransformasikan Malaysia menjadi negara berpendapatan tinggi menjelang 2025. Selain itu, Pelan Strategik Pendigitalan Sektor Awam (PSPSA) 2021-2025 juga telah menggariskan hala tuju strategik pendigitalan sektor awam, selaras dengan aspirasi Wawasan Kemakmuran Bersama 2030 dan Rancangan Malaysia ke-12 (RMKe-12). Hasrat ini tiada lain, selain agar segala inisiatif pendigitalan yang dilaksanakan dapat dimanfaatkan oleh rakyat sepenuhnya, seterusnya mentransformasikan Malaysia kepada negara mampan melalui ekonomi digital selaras dengan aspirasi Malaysia MADANI.

Dalam mengorak langkah ke hadapan, beberapa inisiatif pendigitalan baharu perlu diberi penekanan terutama dalam pembangunan sistem aplikasi yang dilihat perlu berubah kepada lebih berfokuskan pengguna selain mengadaptasi konsep-konsep baharu yang lebih fleksibel seperti *Agile* dan *DevOps.* Pelaksanaan DevOps dapat membantu agensi dalam memperkasakan pembangunan sistem yang merupakan tonggak dalam merealisasikan pendigitalan penyampaian perkhidmatan kerajaan.

Justeru, Dokumen Panduan Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam telah dihasilkan sebagai rujukan dan panduan kepada semua agensi kerajaan dan penjawat awam khasnya, dalam pelaksanaan DevOps bagi pembangunan sistem aplikasi sektor awam dengan pantas dan efisien. Saya yakin dan percaya, melalui kaedah pembangunan baharu yang digariskan dalam dokumen ini, akan berupaya memacu prestasi dan produktiviti penjawat awam ke arah penghasilan sistem aplikasi yang lebih mantap dan efisien bagi memenuhi ekspektasi rakyat.

Sekian, terima kasih.

Ts. Dr. Fazidah binti Abu Bakar
Ketua Pengarah
Jabatan Digital Negara
Kementerian Digital

AKRONIM

DASA	*DevOps Agile Skills Association*
IaC	*Infrastructure as Code*
ICT	*Information and Communication Technology*
PPP	Pelan Pengurusan Projek
PPS	Pelan Pembangunan Sistem
BRS	*Business Requirement Specification*
UAT	*User Acceptance Test*
APM	*Application Performance Monitoring*
RUM	*Real User Monitoring*
SPBM	Sistem Pengurusan Bilik Mesyuarat
API	*Application Programming Interface*
IDE	*Integrated Development Environment*

TAKRIFAN

Agile	Pendekatan pembangunan sistem yang menekankan kaedah iteratif melibatkan perancangan dan pelaksanaan berterusan serta berulang-ulang secara kolaboratif di antara pemilik produk dengan pasukan pembangun bagi membolehkan perubahan dapat dilaksanakan dengan lebih cepat dan berkesan.
Compliance as Code	Pengautomasian pelaksanaan, pengesahan, pemulihan, pemantauan dan pelaporan piawaian pematuhan yang perlu dipatuhi oleh agensi dalam pembangunan sistem aplikasi.
Container	Platform untuk menempatkan kod, fail konfigurasi, kod binari dan *libraries* aplikasi ke dalam satu pakej yang dipanggil imej *container*. Imej ini berfungsi sebagai sistem operasi maya yang ringkas dengan penggunaan sumber sistem yang minimum.
***Definition of Done* (DoD)**	Kriteria dan syarat yang telah dipersetujui oleh pasukan dan pemilik produk sebelum projek atau cerita pengguna, boleh dianggap selesai.
Epic	Keperluan-keperluan pengguna yang mempunyai objektif yang sama tetapi tidak boleh dilaksanakan dalam masa yang singkat. *Epic* hanya boleh dihasilkan melalui beberapa *iteration. Epic* adalah kerja yang boleh dibahagikan kepada tugasan terperinci (iaitu user story) berdasarkan keperluan/permintaan pengguna.
Infrastructure as a Code (IaC)	Pengautomasian penyediaan infrastruktur melalui konfigurasi skrip atau kod yang boleh diubah dengan pantas bagi mempercepatkan penyediaan infrastruktur dalam pembangunan sistem aplikasi.
Iteration	Kitaran pembangunan projek biasanya mengambil masa sekitar dua hingga empat minggu dan akan berulang apabila pasukan membangunkan ciri produk yang boleh digunakan dan menghantarnya ke persekitaran pengeluaran. Sesuatu projek terdiri daripada satu siri lelaran yang dimulakan dengan mesyuarat perancangan dan mesyuarat retrospektif pada penghujungnya *sprint*. *Sprint* adalah contoh *iteration* pembangunan berdasarkan rangka kerja *Scrum*.
Kriteria keluar	Status aktiviti serta tahap pencapaian atau metrik yang menjadi syarat untuk menamatkan sesuatu peringkat. Kriteria keluar yang ditetapkan perlu di bincang dan dipersetujui bersama oleh ahli pasukan.
Monorepo	Kod sumber bagi satu atau beberapa projek dikumpulkan ke dalam satu repositori tunggal atau dikenali sebagai repositori monolitik. Kesemua kod sumber, pengujian dan konfigurasi berada dalam satu

	repositori yang sama supaya pasukan pembangun dapat berkongsi dan mengintegrasikan kod sumber di antara projek yang berbeza.
Pelan Pengurusan Projek (PPP)	Dokumen yang menjadi rujukan utama dalam mengurus dan mengawal projek sistem aplikasi ICT di sektor awam.
Produk	Perisian atau sistem aplikasi yang dibangunkan, diuji dan disampaikan kepada pengguna secara peningkatan (incremental) dan pengulangan (iterative). Produk merangkumi kod sumber, konfigurasi dan artifak berkaitan yang diperlukan untuk pembinaan dan penyampaian ke persekitaran pembangunan, *staging* atau produksi.
Regression test	Pengujian bagi memastikan fungsi-fungsi sistem aplikasi sedia ada tidak terganggu selepas sistem menjalani pembetulan atau pindaan. Aktiviti ini menguji fungsi sedia ada serta fungsi baharu pada satu sesi pengujian.
Retrospektif	Mesyuarat pasukan pada akhir *iteration* di mana ahli pasukan berkumpul untuk membincangkan kaedah kerja pasukan. Retrospektif perlu berlaku semasa projek supaya penambahbaikan yang terhasil boleh digunakan dan relevan dengan kerja yang akan datang pada projek yang sama. Aspek retrospektif seperti masa, ahli pasukan dan format mesyuarat harus konsisten.
Scalability	Keupayaan untuk mengendalikan peningkatan dan penambahan beban kerja dan membolehkan peralatan komputer dan program sistem aplikasi berkembang dari semasa ke semasa.
Semakan analisis statik	Analisis yang digunakan untuk mengenal pasti masalah keselamatan semasa peringkat pengekodan. Semakan analisis statik semasa proses integrasi mengenal pasti masalah keselamatan sebelum proses pelepasan dilaksanakan. Analisis statik yang kerap akan memastikan kod tiada kelemahan keselamatan yang serius dalam setiap peringkat.
Telemetry	Log data yang terhasil daripada sumber yang sukar dicapai dan dihantar ke sistem yang berbeza untuk pemantauan dan analisis. *Telemetry* membolehkan masalah semasa dikenal pasti dengan pantas. Pasukan DevOps boleh mengenal pasti metrik corak pengguna melalui *telemetry* dan mencipta amaran jika anomali berlaku.
***Tools* DevOps**	*Tools* yang digunakan bagi menggalakkan prinsip berterusan DevOps. Setiap peringkat DevOps mempunyai *tools* dengan fungsi yang tersendiri serta terbahagi mengikut kesesuaian dalam kematangan pelaksanaan DevOps. *Tools* DevOps yang dipilih

bergantung kepada model dan infrastruktur dalam pembangunan sistem aplikasi.

User Story Keperluan pengguna yang telah diperhalusi dan boleh dilaksanakan dalam jangka masa satu *iteration* iaitu antara dua hingga empat minggu. *User story* membantu dalam penyediaan skop bagi sistem aplikasi yang akan dibangunkan berdasarkan keperluan pengguna.

RINGKASAN EKSEKUTIF

Panduan Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam ini diterbitkan untuk memperincikan kaedah pelaksanaan DevOps dengan mengambil kira amalan terbaik sebagai rujukan agensi sektor awam. Panduan ini merupakan pelengkap dan perlu dibaca bersama-sama Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam yang telah diterbitkan pada tahun 2022.

Berdasarkan model rangka kerja tersebut, terdapat enam komponen yang telah diperkenalkan iaitu Penerapan Prinsip dan Budaya, Pengukuhan Tadbir Urus, Pengadaptasian Metodologi, Pengukuran Kematangan, Pemerkasaan Teknologi, serta Pematuhan Keselamatan. Panduan Pelaksanaan DevOps Sektor Awam ini akan menumpu dan memperincikan kepada komponen Pemerkasaan Teknologi serta kaedah penggunaan *tools* bagi setiap peringkat dalam kitar hayat DevOps.

Pelaksanaan DevOps membentuk kitaran dan digunakan untuk mengenal pasti peringkat dalam kitaran hayat pembangunan sistem aplikasi. Terdapat lapan peringkat dalam kitar hayat DevOps yang merangkumi peringkat perancangan, pengekodan, pembangunan, pengujian, pelepasan, penempatan, pengoperasian dan pemantauan. Aktiviti utama telah dikenal pasti dan dikelaskan pada setiap peringkat dalam kitar hayat DevOps.

Pada peringkat perancangan, aktiviti utama seperti perancangan produk, pengurusan komunikasi, pengurusan *tools* dan proses CI/CD serta perancangan pengujian akan diterangkan. Peringkat pengekodan melibatkan tiga aktiviti utama iaitu pengurusan kod sumber, konfigurasi *pipeline* CI/CD dan semakan kualiti kod. Pada peringkat pembangunan, terdapat tiga aktiviti utama iaitu *compile code, package code* dan pembinaan kod dan imej *container*. Peringkat pengujian menjelaskan dua pengujian utama iaitu pengujian keperluan fungsian dan pengujian bukan fungsian. Pada peringkat pelepasan dan penempatan, masing-masing mempunyai satu aktiviti utama iaitu pelepasan dan penempatan sistem aplikasi. Peringkat pengoperasian melibatkan tiga aktiviti utama iaitu penyandaran pangkalan data (database backup), pengemasan *container registry* (housekeeping) dan pemulihan sistem aplikasi (system recovery).

Pada peringkat terakhir dalam kitar hayat DevOps, aktiviti seperti pemantauan sistem, pemantau prestasi sistem aplikasi, perancangan kapasiti dan pemantauan maklum balas pengguna akan diterangkan.

Pelaksanaan DevOps dalam pembangunan sistem aplikasi sektor awam adalah menggunakan metodologi *agile* dengan mengadaptasikan amalan dan proses *agile scrum.* Metodologi *agile scrum* dalam pembangunan sistem aplikasi di sektor awam berupaya mempertingkatkan kolaborasi dan komunikasi antara pasukan. Penggunaan *tools* juga menyokong pelaksanaan DevOps bagi mempercepatkan proses penyampaian produk dan mengurangkan tempoh masa dalam penghasilan sesebuah produk. Secara keseluruhan, panduan ini boleh dijadikan rujukan bagi membantu agensi sektor awam dalam pelaksanaan DevOps ke arah transformasi sistem penyampaian perkhidmatan awam yang terbaik.

BAB 1 : PENDAHULUAN

Inisiatif pelaksanaan DevOps dalam pembangunan sistem aplikasi di sektor awam diperkenalkan sebagai satu pendekatan baharu bagi mempertingkatkan kualiti produk sistem aplikasi dan dalam masa yang sama menambah baik amalan pembangunan dengan memperkenalkan metodologi *Agile* serta penggunaan *tools* automasi. Panduan ini disediakan sebagai rujukan komprehensif kepada semua agensi sektor awam dalam mempraktikkan pendekatan DevOps di agensi masing-masing. Bagi mempermudahkan pemahaman, kajian kes disediakan sebagai contoh pelaksanaan yang digunakan merangkumi semua peringkat DevOps.

Proses penghasilan dokumen ini melibatkan *Subject Matter Expert* (SME) dari Bahagian Perundingan ICT, Jabatan Digital Negara dan pegawai teknikal Bahagian Pembangunan Aplikasi, Jabatan Digital Negara untuk memastikan kandungan dokumen adalah lengkap, komprehensif dan bersesuaian dengan konteks pelaksanaan DevOps di agensi sektor awam.

Panduan ini terdiri daripada lima bab iaitu:

Bab 1 menjelaskan tujuan penghasilan dokumen panduan.

Bab 2 menerangkan pengenalan kepada rangka kerja DevOps sektor awam dan pengenalan kajian kes.

Bab 3 menjelaskan pengenalan dan senarai *tools* DevOps yang digunakan oleh agensi awam.

Bab 4 menjelaskan berkenaan kitar hayat DevOps yang merangkumi peringkat perancangan, pengekodan, pembangunan, pengujian, pelepasan, penempatan, pengoperasian dan pemantauan.

Bab 5 merumuskan kandungan dokumen panduan ini dan jangkaan pengguna terhadap dokumen ini.

Dokumen ini perlu dibaca bersama dengan dokumen berkaitan yang telah diterbitkan oleh Jabatan Digital Negara, khususnya:

a. Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam.
b. Panduan Pengurusan Projek ICT Sektor Awam (PPrISA).
c. Panduan Kejuruteraan Sistem Aplikasi Sektor Awam (KRISA).
d. *Independent Verification & Validation (IV&V) Handbook*.
e. *Software Quality Management Using DevOps Approach*.

1.1. SKOP DOKUMEN

Skop penggunaan dokumen panduan ini adalah dikhaskan untuk pembangunan sistem aplikasi secara dalaman yang merangkumi aktiviti pembangunan baharu atau penambahbaikan sistem aplikasi sedia ada. Penggunaan dokumen panduan ini juga terhad kepada penggunaan *tools* yang terdapat di Jabatan Digital Negara dan garis panduan sedia ada yang berkaitan pembangunan sistem aplikasi yang masih berkuat kuasa.

Kumpulan sasaran utama dokumen panduan ini adalah:

a. Pegawai teknikal di agensi sektor awam yang berperanan sebagai ahli pasukan DevOps.
b. Bahagian atau Unit di agensi sektor awam yang berperanan sebagai pemilik produk.
c. Mana-mana pihak berkepentingan yang terlibat dalam pelaksanaan DevOps.

1.2. OBJEKTIF DOKUMEN

Objektif penyediaan dokumen ini adalah seperti berikut:

a. Mengaplikasikan metodologi agile scrum bagi meningkatkan keupayaan dan kualiti penyampaian perkhidmatan sektor awam melalui pelaksanaan DevOps.

b. Memperkenalkan teknologi dan platform khusus bagi pelaksanaan DevOps yang boleh digunakan oleh agensi sektor awam.

c. Memberi pemahaman dan panduan mengenai proses pembangunan sistem aplikasi menggunakan tools yang ditawarkan di Jabatan Digital Negara.

BAB 2 : RANGKA KERJA DEVOPS SEKTOR AWAM

Terma DevOps merujuk kepada gabungan singkatan *development* dan *operations*, iaitu dua aktiviti yang pada kebiasaannya dilaksanakan secara berasingan oleh dua pihak yang berbeza. Pendekatan DevOps menggabungkan dua aktiviti ini supaya kedua-dua pihak dapat bekerja dalam satu pasukan dan dibantu dengan automasi bagi membolehkan proses pembangunan dan penyampaian sistem aplikasi dilaksanakan dengan lebih efisien.

Rajah 2-1: Rangka Kerja Pelaksanaan DevOps

Rajah 2-1 memaparkan model Rangka Kerja Pelaksanaan DevOps Sektor Awam. Terdapat enam komponen yang diperkenalkan dalam model rangka kerja ini iaitu Penerapan Prinsip dan Budaya, Pengukuhan Tadbir Urus, Pengadaptasian Metodologi, Pengukuran Kematangan, Pemerkasaan Teknologi serta Pematuhan

Keselamatan. Manakala lima pemboleh daya yang menyokong kejayaan pelaksanaan komponen DevOps adalah Dasar, Pengurusan Perubahan, Kompetensi, Kawalan dan Automasi.

Rangka Kerja DevOps Sektor Awam menggariskan tiga prinsip asas iaitu kolaboratif, komunikatif dan penambahbaikan berterusan. Pendekatan DevOps menggalakkan kolaborasi dan komunikasi aktif antara pasukan untuk mencapai satu matlamat yang sama iaitu membangunkan produk yang berkualiti dan berdaya tahan. Manakala penambahbaikan berterusan merujuk kepada amalan integrasi, penempatan dan penyampaian berterusan di sepanjang kitar hayat pelaksanaan DevOps. Selain daripada itu, pendekatan DevOps ini juga turut menekankan kepada aspek automasi dan kawalan di mana peringkat yang terlibat dalam pembangunan produk akan dilaksanakan secara automasi yang seterusnya dapat mempercepatkan tempoh pembangunan produk. Aspek kawalan juga dapat diperkukuhkan dan pemantauan dapat diperluaskan bagi mempertingkatkan kecekapan kualiti pembangunan sistem aplikasi.

2.1. PELAKSANAAN DEVOPS SEKTOR AWAM

Pelaksanaan DevOps di agensi sektor awam meliputi perkara-perkara seperti berikut:

a. Penerbitan dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam sebagai asas pengetahuan bagi meningkatkan tahap penyampaian perkhidmatan digital di agensi.

b. Penyediaan *tools* bagi memudahkan pelaksanaan DevOps sektor awam.

c. Aktiviti *coaching* dan latihan bagi memperkukuhkan pengetahuan dalam pelaksanaan DevOps di sektor awam.

2.2. Pengenalan Kajian Kes

Satu kajian kes pembangunan sistem aplikasi akan digunakan sebagai contoh pelaksanaan DevOps dalam dokumen ini iaitu Sistem Pengurusan Bilik Mesyuarat (SPBM). Semua peringkat dalam pelaksanaan pendekatan DevOps termasuk penggunaan metodologi *agile* akan diterangkan secara praktikal melalui kajian kes ini.

Sistem Tempahan Bilik Mesyuarat merupakan sistem aplikasi sedia ada yang dimiliki oleh Bahagian Khidmat Pengurusan bagi menyokong pentadbiran bilik mesyuarat. Sistem tersebut mempunyai kekurangan dari segi fungsi dan kemudahan dalam menyokong cara kerja baru. SPBM adalah merupakan inisiatif bagi menambah baik Sistem Tempahan Bilik Mesyuarat sedia ada bagi menangani isu-isu semasa dalam pentadbiran dan pengurusan bilik mesyuarat dengan teratur dan efisien. Pembangunan produk ini akan dilaksanakan sepenuhnya secara dalaman oleh pegawai IT agensi.

Terdapat dua dokumen asas yang boleh digunakan sebagai prasyarat sebelum pembangunan sistem aplikasi seperti yang diperincikan dalam Buku KRISA. Dokumen yang relevan dan sesuai digunakan sebelum pelaksanaan *agile* dan DevOps dalam pembangunan sistem aplikasi adalah seperti:

a. Pelan Pembangunan Sistem (PPS).
b. Spesifikasi Keperluan Bisnes (BRS).

BAB 3 : PENGENALAN *TOOLS* DEVOPS SEKTOR AWAM

3.1. PENGENALAN *TOOLS* DEVOPS

Rajah 3-1: Komponen 5 Pemerkasaan Teknologi

Merujuk kepada komponen kelima daripada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam iaitu Pemerkasaan Teknologi, infrastruktur DevOps diperlukan bagi mewujudkan persekitaran yang menyokong pembangunan dan penyampaian produk. Justeru itu, pemilihan *tools* yang sesuai akan meningkatkan kejayaan dalam pelaksanaan DevOps. Pendekatan DevOps menekankan automasi proses pembangunan sistem aplikasi seperti pembangunan, pengujian, pelepasan dan lain-lain untuk menghasilkan produk lebih pantas dan berkualiti serta mengurangkan risiko kegagalan pada sistem aplikasi.

Bab ini menerangkan *tools* yang disediakan bagi pelaksanaan DevOps dalam pembangunan sistem aplikasi sektor awam. Penggunaan *tools* pada peringkat kitar hayat DevOps akan dijelaskan dalam Bab 4.

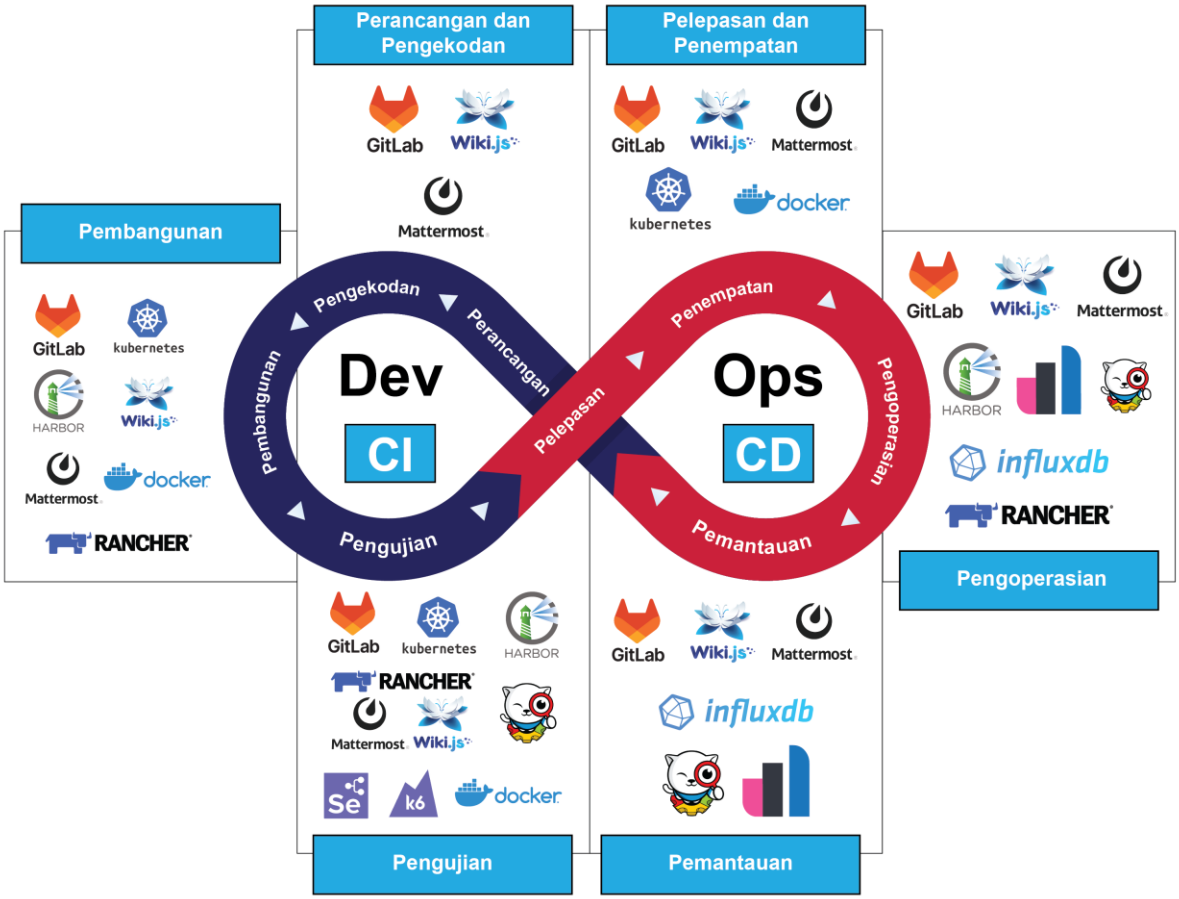
3.2. SENARAI *TOOLS* PELAKSANAAN DEVOPS SEKTOR AWAM

Berikut merupakan senarai dan penggunaan *tools* pada pelaksanaan peringkat DevOps dalam pembangunan sistem aplikasi sektor awam. *Tools* yang disenaraikan pada Jadual 3-1Jadual 3-1 merupakan senarai *tools* yang boleh didapati daripada platform sumber terbuka.

Jadual 3-1: Senarai *Tools* DevOps Sektor Awam.

No.	Penggunaan *Tools*	Peringkat DevOps							
		Perancangan	Pengekodan	Pembangunan	Pengujian	Pelepasan	Penempatan	Pengoperasian	Pemantauan
1.	Git	✓	✓	✓	✓	✓	✓	✓	✓
2.	GitLab	✓	✓	✓	✓	✓	✓	✓	✓
3.	Mattermost	✓	✓	✓	✓	✓	✓	✓	✓
4.	Wiki.js	✓	✓	✓	✓	✓	✓	✓	✓
5.	Docker			✓	✓	✓	✓		
6.	Kubernetes			✓	✓	✓	✓		
7.	Rancher			✓	✓	✓	✓	✓	
8.	Harbor			✓	✓	✓	✓	✓	
9.	Selenium Grid				✓				
10.	K6				✓				
11.	Sitespeed.io				✓			✓	✓
12.	ZAP				✓		✓		
13.	OWASP Dependency-Check				✓		✓		
14.	InfluxDB							✓	✓
15.	Elastic Observability							✓	✓

Rajah 3-2 memaparkan infografik padanan *tools* pada setiap peringkat pada kitar hayat DevOps.



Rajah 3-2: Senarai *Tools* DevOps Sektor Awam.

Rujuk **Lampiran J-1** untuk pautan laman web bagi setiap *tools* yang terdapat di Jabatan Digital Negara.

3.2.1. Git

Git merupakan sistem pengurusan kod sumber yang membantu pasukan pembangun menyelaraskan aktiviti penyimpanan, pengesanan dan pengurusan versi fail kod daripada sumber yang berbeza. Setiap perubahan yang disimpan di repositori Git mempunyai rekod seperti tarikh, masa, nama pengguna dan bahagian kod sumber yang berubah.

Git membolehkan pasukan pembangun melaksanakan kawalan versi pada kod sumber dan melakukan *rollback* ke versi sebelumnya sekiranya perlu. Pasukan pembangun dapat menyelaras *branch* yang berbeza dalam repositori berpusat supaya setiap pembangun dapat membangunkan modul atau tugasan yang berbeza pada masa yang sama.

Pembangun boleh mengakses repositori dari komputer mereka dengan membuat salinan (clone) dari repositori berpusat. Setelah repositori disalin, pembangun boleh mengemas kini kod sumber dan melaksanakan *commit* serta *push* ke repositori berpusat.

3.2.2. GitLab

GitLab merupakan sebuah platform DevOps yang mempunyai fungsi pengurusan kod sumber berasaskan Git di samping fungsi-fungsi lain dalam semua kitar hayat DevOps. GitLab membolehkan pasukan melaksanakan semua tugas berkaitan DevOps dari peringkat perancangan produk hingga pemantauan. Tatacara penggunaan GitLab pada dokumen ini adalah menggunakan GitLab Enterprise Edition (EE) 15.10.1 dengan pelan premium.

Kelebihan utama menggunakan GitLab membolehkan ahli pasukan DevOps bekerjasama dalam pembangunan produk pada setiap peringkat DevOps.

GitLab menyokong metodologi *agile* dan mempunyai fungsi yang bersesuaian dengan setiap aktiviti yang dilaksanakan dalam pendekatan *agile scrum.* Jadual 3-2 merupakan padanan di antara artifak *agile* dan *features* GitLab yang akan digunakan pada Bab 4.

Jadual 3-2: Padanan Artifak *Agile Scrum* dengan *Features* GitLab

Artifak *Agile*	*Features* GitLab	Penjelasan
User story	*Issues*	Fungsi GitLab *Issues* digunakan untuk merekod keperluan pengguna.
Task	*Task lists*	*Task Lists* adalah pecahan tugas bagi setiap GitLab *Issues* untuk diagihkan kepada ahli pasukan.
Epic	*Epics*	Fungsi GitLab *Epics* digunakan untuk mengumpulkan keseluruhan *user story*.
Story Point	*Weights*	Fungsi GitLab *weights* pada setiap GitLab *Issue* digunakan sebagai *story point* bagi menggambarkan saiz dan kerumitan *user story* tersebut.
Product backlog	*Issues List* dan *Labels*	Fungsi GitLab *Issues List* adalah untuk menyenaraikan keseluruhan *issues*. Senarai *issues* ini boleh dijana secara dinamik untuk menjejaki *backlog* dan disusun mengikut keutamaan sebagai *product backlog*. Fungsi GitLab *Labels* boleh dicipta dan diperuntukkan pada setiap *issues* bagi memudahkan pencarian pada *Issues List* berdasarkan hanya satu label atau pelbagai label. *Priority label* juga boleh digunakan untuk menyusun senarai tersebut.
Sprint	*Iterations/* *Milestones*	Fungsi GitLab *Iterations* atau *Milestones* boleh digunakan sebagai penetapan tempoh masa *sprint.* Contohnya, menetapkan enam *sprint* bagi satu produk dengan tempoh setiap *sprint* ditetapkan selama dua minggu. Setiap *user story* atau GitLab *Issues* akan ditetapkan berada pada *sprint* ke berapa mengikut keutamaan yang telah ditetapkan.

Artifak *Agile*	*Features* GitLab	Penjelasan
Milestones	*Milestones*	Fungsi GitLab *Milestones* juga boleh ditetapkan sebagai tarikh untuk melepaskan produk ke persekitaran produksi. Contohnya produk yang *Viable to Release* ditetapkan selepas *sprint* ketiga. Maka tarikh mula *milestones* adalah tarikh bermula *sprint* 1 manakala tarikh tamat *milestones* adalah tarikh akhir *sprint* 3.
Carta *Burndown*	Burndown *chart*	Fungsi GitLab Burndown *Chart* adalah memaparkan perkembangan pembangunan sistem aplikasi.
Agile board	*Issues Boards*	Fungsi GitLab *Issues Boards* adalah memaparkan keseluruhan *issues* di bawah *project/epics* yang sama. Pergerakan *issue* yang dibincangkan semasa *daily scrum* boleh dilakukan pada fungsi ini supaya dapat mencapai matlamat yang sama sepanjang *sprint*. Sepanjang tempoh *sprint*, *issues* akan bergerak dari satu peringkat ke peringkat yang lain seperti *Ready for Dev, In Dev, In QA, In Review* dan *Done* bergantung kepada proses aliran pada setiap agensi.

Antara komponen penting GitLab yang boleh digunakan dalam pelaksanaan DevOps adalah seperti berikut:

a. GitLab CI/CD

GitLab CI/CD merupakan *tools* yang boleh digunakan untuk amalan penambahbaikan berterusan yang ada dibincangkan pada Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam iaitu integrasi berterusan, pengujian berterusan, penyampaian berterusan, penempatan berterusan, pemantauan berterusan dan keselamatan berterusan. Proses pengujian, pembinaan, pelepasan dan penempatan sistem aplikasi boleh dibuat menggunakan GitLab CI/CD tanpa memerlukan integrasi dari aplikasi pihak ketiga. Namun, amalan penambahbaikan berterusan iaitu pemantauan berterusan akan dilakukan dengan menggunakan *tools* yang berbeza.

b. GitLab *Runner*

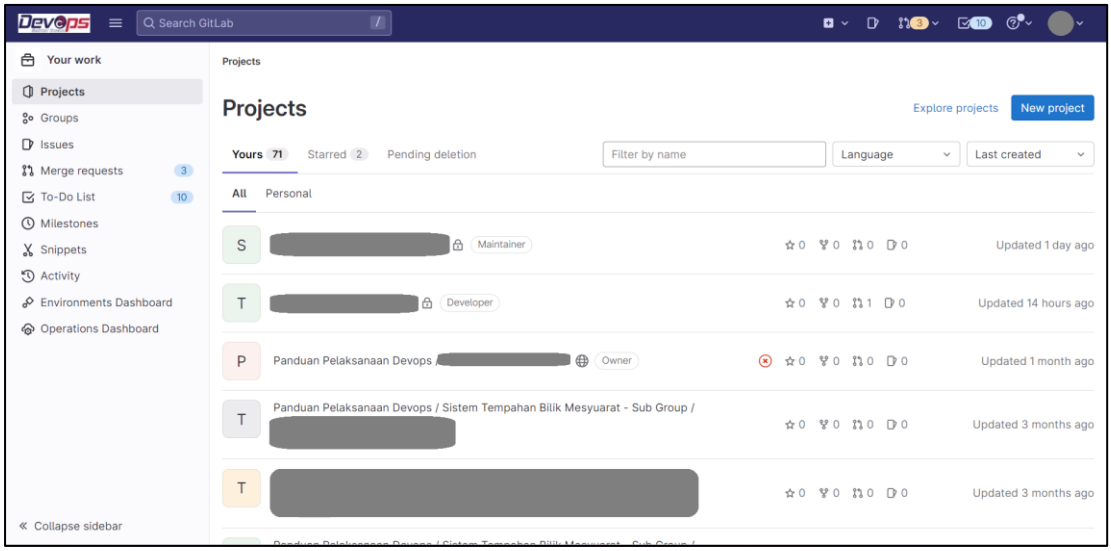
GitLab *runner* merupakan ejen yang dipasang secara berasingan dan mempunyai akses pada persekitaran pembangunan dan pelaksanaan sistem aplikasi untuk melaksanakan *jobs* yang telah dikonfigurasikan pada GitLab CI/CD. Pemasangan GitLab *runner* boleh dibuat secara khusus atau dikongsi oleh beberapa sistem aplikasi. Jadual 3-3 menjelaskan kategori *runner* yang boleh dipasang.

Jadual 3-3: Kategori GitLab *Runner*

Kategori *Runner*	Penjelasan
Specific Runner	Dikhaskan hanya untuk satu projek repositori git sahaja dan tidak dapat digunakan oleh repositori projek Git yang lain. Boleh dijadikan sebagai *shared runner* sekiranya *maintainer* atau pemilik *runner* memutuskan agar *runner* tersebut boleh digunakan oleh repositori yang lain.

Kategori *Runner*	Penjelasan
Shared Runner	Ditujukan untuk seluruh projek repositori Git. Kelemahannya adalah jika *shared runner* sedang digunakan oleh repositori yang lain, maka *pipeline* akan tertangguh dan menunggu repositori lain selesai menggunakan *shared runner*.
Group Runner	Ditujukan untuk beberapa projek repositori Git yang terdaftar di dalam satu kumpulan repositori.

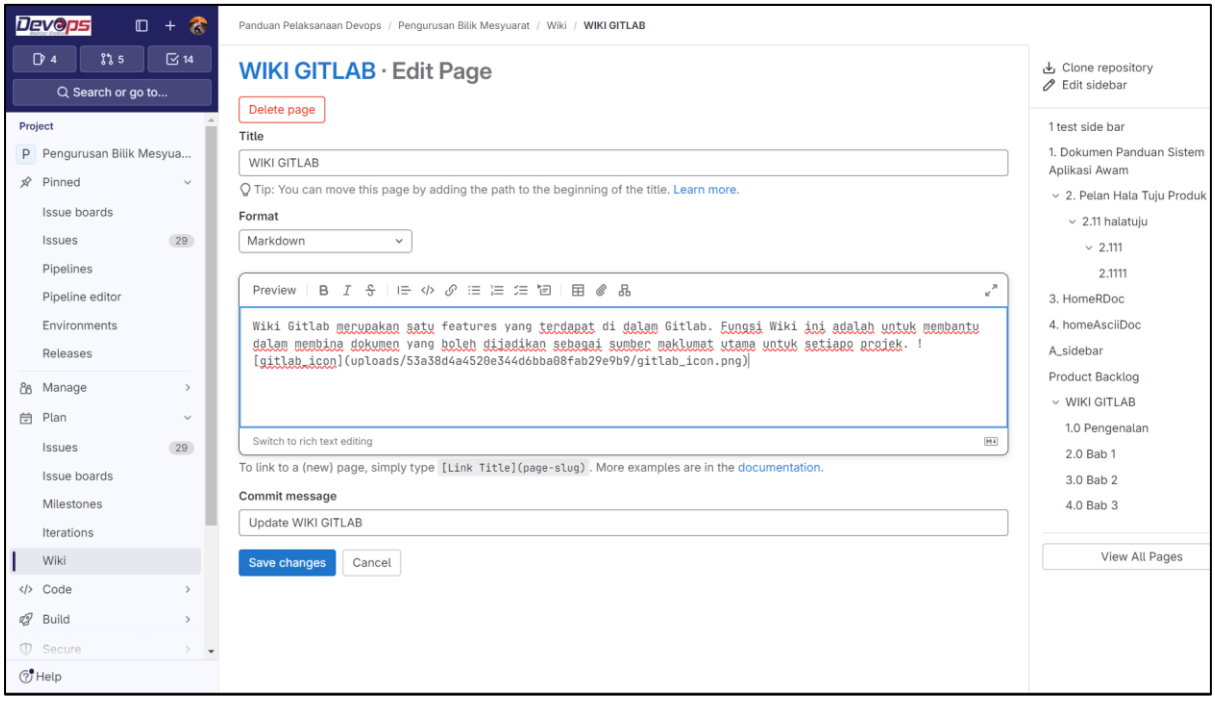
Jadual 3-3 memaparkan contoh paparan *tools* GitLab.



Rajah 3-3: Contoh Paparan GitLab

c. Wiki GitLab

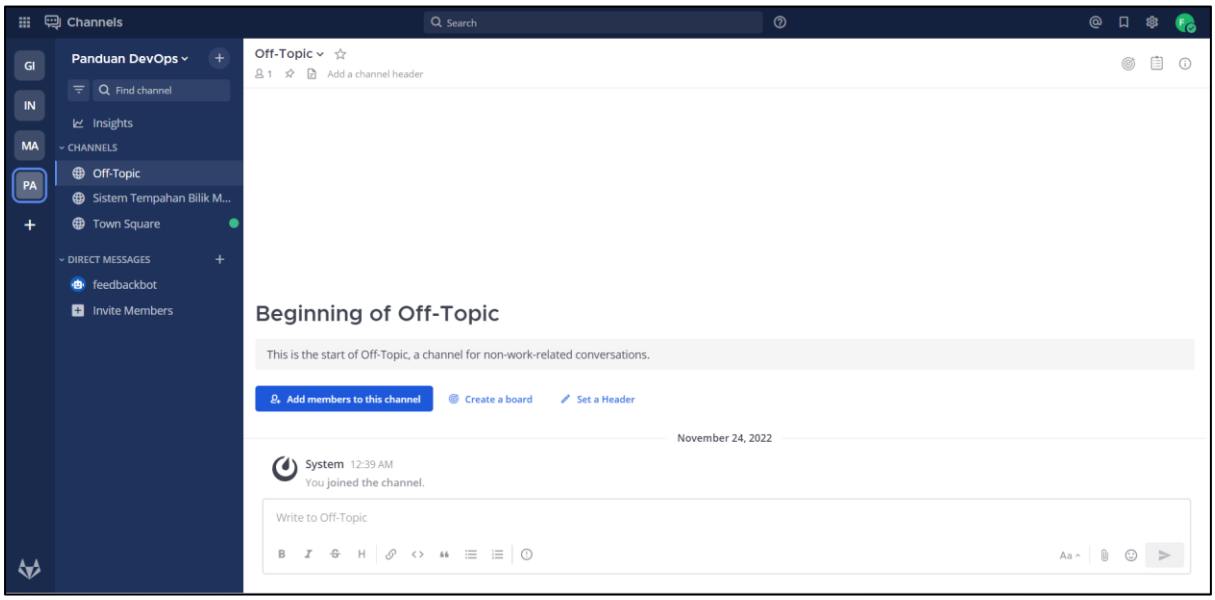
Wiki GitLab merupakan satu *tools* di dalam GitLab yang membolehkan pengguna berkongsi maklumat sesebuah projek atau modul. Perkongsian maklumat boleh dikemaskini secara dalam talian bagi memudahkan pasukan mendapat maklumat secara jelas dan tersusun. Ahli pasukan yang diberi akses boleh mengemaskini setiap maklumat yang dikongsi dalam repositori yang sama.



Rajah 3-4: Paparan Halaman Wiki GitLab

3.2.3. Mattermost

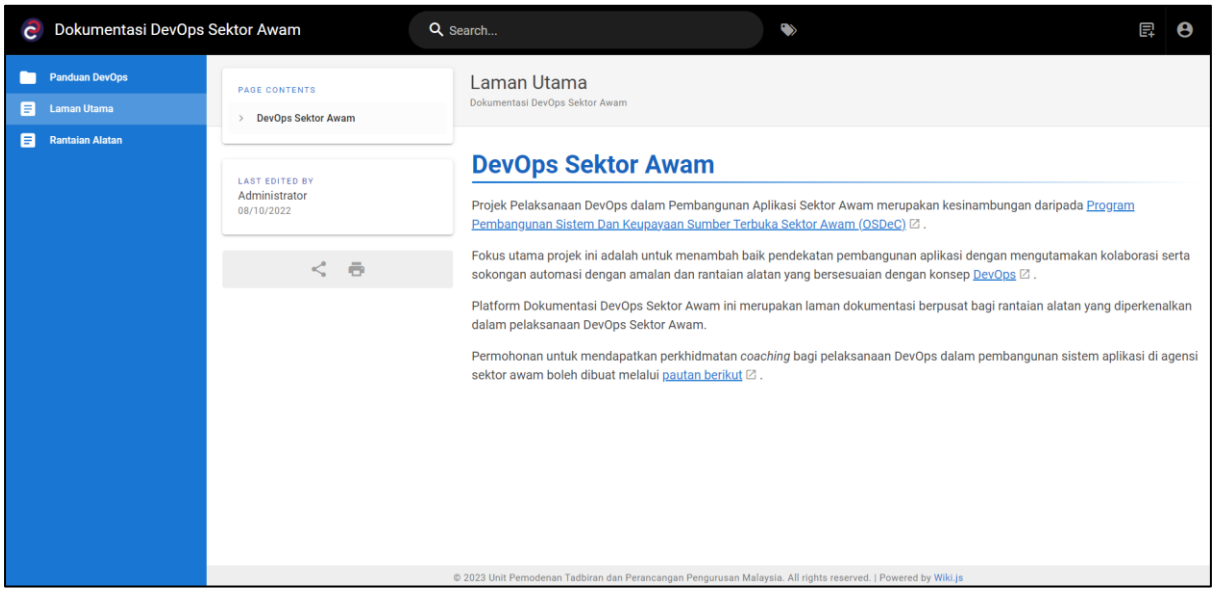
Mattermost merupakan platform yang menyediakan perkhidmatan komunikasi dan kolaborasi antara ahli pasukan. Integrasi Mattermost dengan GitLab boleh dilaksanakan bagi tujuan mewujudkan *Issues* dan memulakan CI/CD *jobs*. Tatacara penggunaan Mattemost *ChatOps* pada dokumen ini adalah menggunakan Mattermost Team Edition versi 7.5.1. Rajah 3-5 memaparkan contoh paparan *tools* Mattermost *ChatOps.*



Rajah 3-5: Contoh Paparan Mattermost *ChatOps*

3.2.4. Wiki.js

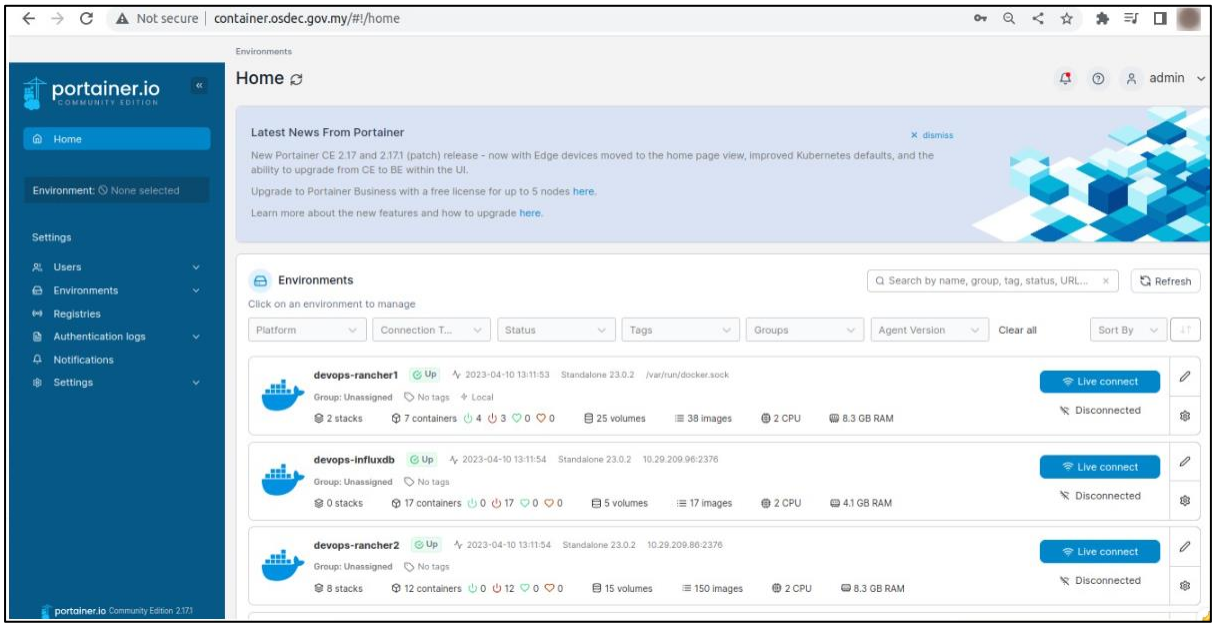
Wiki.js merupakan perisian kolaboratif yang membolehkan pengguna mencipta dan mengemas kini dokumentasi secara dalam talian. Kandungan dokumentasi ditulis menggunakan format Markdown atau fungsi *visual editor.* Wiki.js boleh juga digunakan sebagai repositori dokumentasi sepanjang pelaksanaan pembangunan sistem aplikasi. Tatacara penggunaan Wiki.js pada dokumen ini adalah menggunakan Wiki.js versi 2.5.295. Rajah 3-6 memaparkan contoh paparan *tools* Wiki.js.



Rajah 3-6: Contoh Paparan Wiki.js

3.2.5. Docker

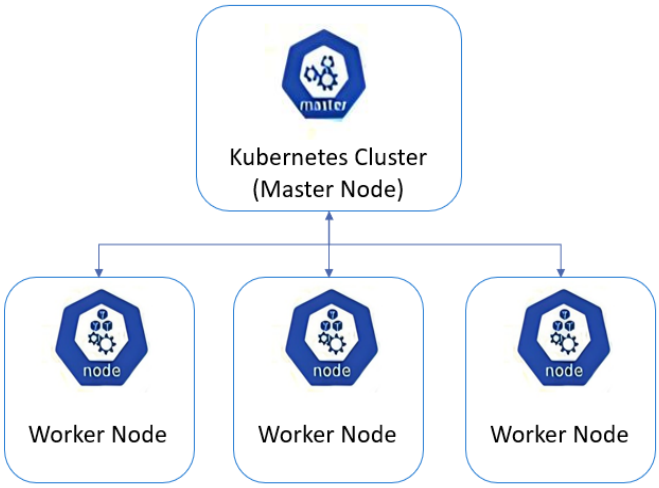
Docker adalah platform yang membolehkan sistem aplikasi dipakejkan kepada satu format standard *Open Container Initiative* (OCI) dipanggil imej *container* yang mengandungi *library*, *runtime* dan kod sumber sistem aplikasi. Platform ini membolehkan pasukan membina, menguji dan menempatkan sistem aplikasi dengan efektif melalui teknologi *container*. Imej *container* boleh disediakan secara automasi dengan bantuan *pipeline* CI/CD khusus kepada persekitaran penempatan tertentu dan disimpan dalam repositori imej *container* berpusat untuk dikongsikan bersama pasukan DevOps. Rajah 3-7 memaparkan contoh paparan *tools* Docker.



Rajah 3-7: Contoh Paparan Docker

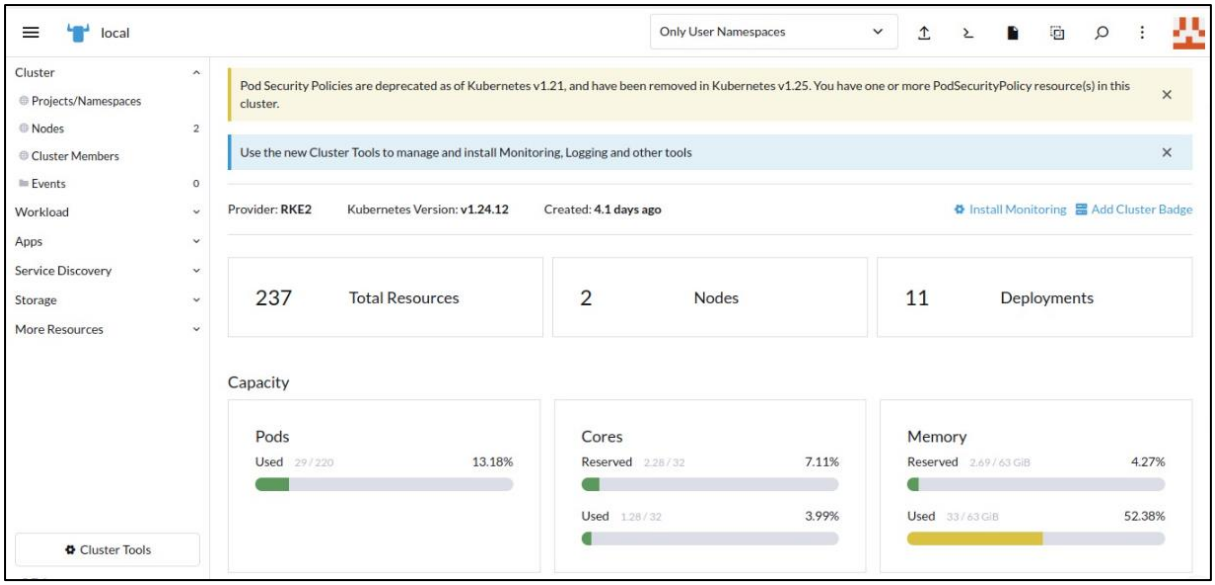
3.2.6. Kubernetes

Kubernetes adalah platform orkestrasi *container* yang mengautomasikan proses penempatan, pengurusan dan penskalaan sistem aplikasi. Kubernetes dilengkapi dengan fungsi *high availability* (HA) kepada persekitaran *container* dan menyokong ciri-ciri *self healing* serta *auto scalling*.



Rajah 3-8: Komponen Kluster Kubernetes

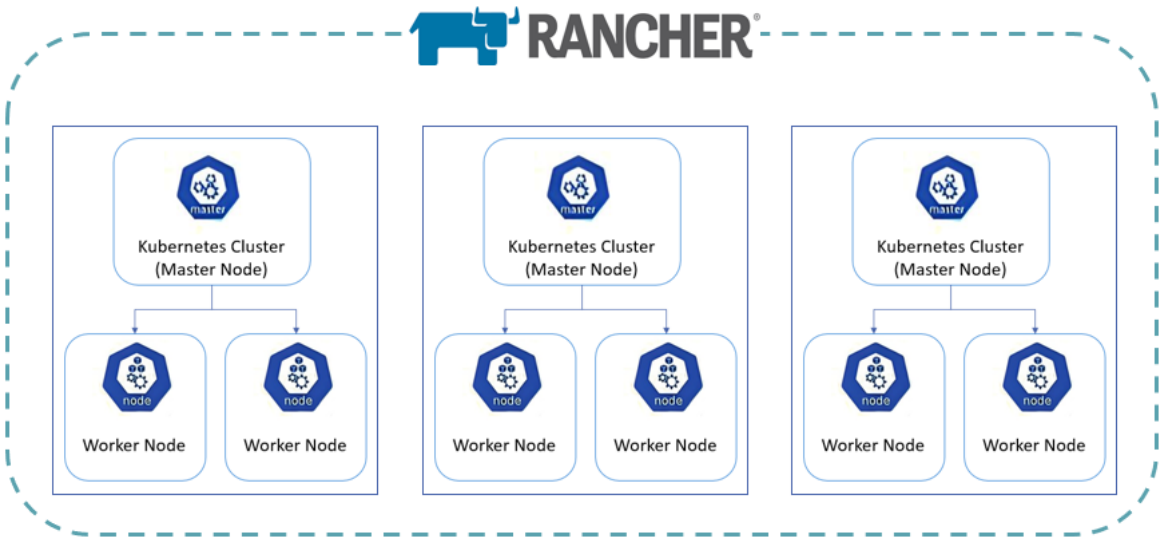
Rajah 3-8 menunjukkan Kluster Kubernetes yang berfungsi sebagai *master node* berupaya menguruskan beberapa *worker node* yang terdiri daripada pelayan *virtual* ataupun fizikal. *Worker node* berfungsi untuk menempatkan dan menjalankan aplikasi *container*. Rajah 3-9 memaparkan contoh paparan *tools* Kubernetes.



Rajah 3-9: Contoh Paparan Kubernetes

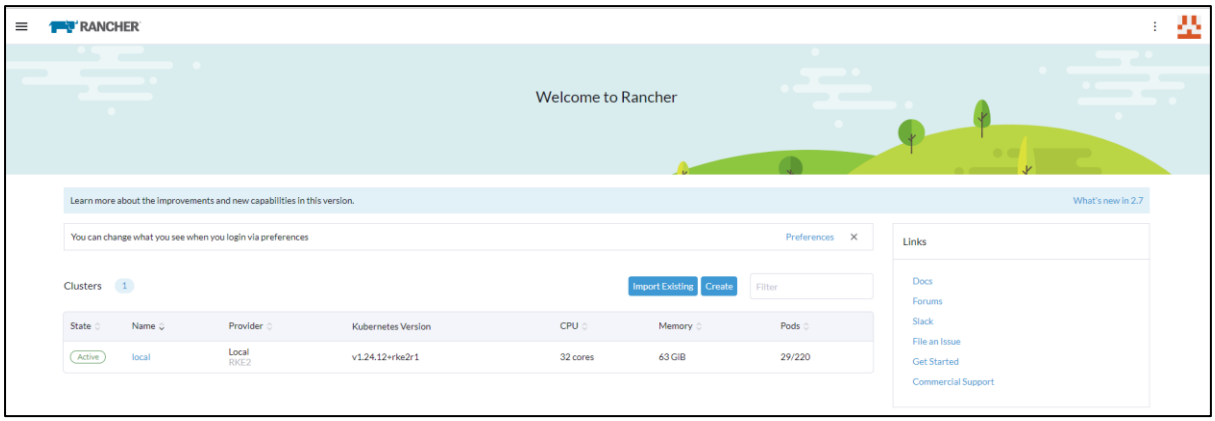
3.2.7. Rancher

Rancher adalah platform untuk menguruskan kluster Kubernetes melalui antara muka web.



Rajah 3-10: Arkitektur Platform Rancher

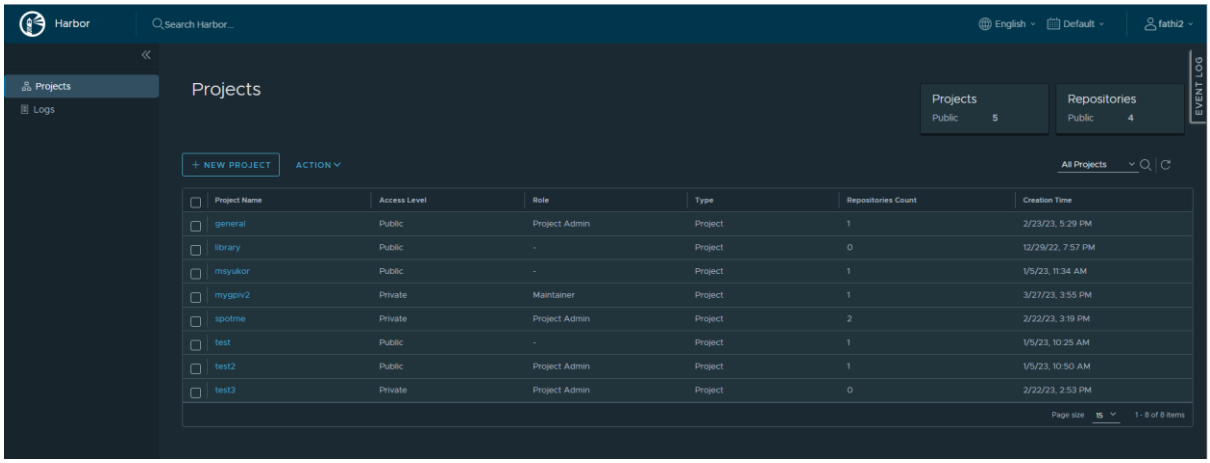
Rajah 3-10 menunjukkan arkitektur platform Rancher yang berfungsi sebagai antara muka pengguna dalam memantau dan menguruskan satu atau lebih kluster Kubernetes. Rajah 3-11 memaparkan contoh paparan *tools* Rancher.



Rajah 3-11: Contoh Paparan Rancher

3.2.8. Harbor

Harbor merupakan repositori *container* yang menyimpan dan melindungi artifak dengan dasar dan kawalan capaian berasaskan peranan, memastikan imej diimbas serta bebas dari isu berkaitan keselamatan. Harbor membolehkan pengurusan imej *container* secara konsisten dan selamat merentasi platform seperti Kubernetes dan Docker. Rajah 3-12 memaparkan contoh paparan *tools* Harbor.



Rajah 3-12: Contoh Paparan Harbor

3.2.9. Selenium Grid

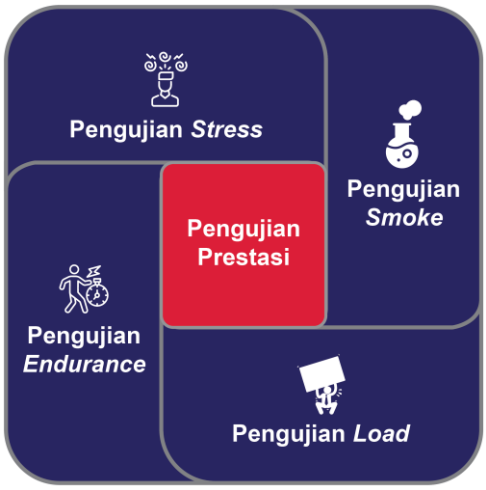
Selenium Grid adalah *tools* untuk melaksanakan pengujian fungsian aplikasi web secara automasi di pelbagai pelayar dan *remote machine* secara serentak dalam persekitaran berbeza. Jadual 3-4 menyenaraikan sistem pengoperasian, bahasa pengaturcaraan dan pelayar web yang disokong oleh Selenium Grid untuk pengujian aplikasi web.

Jadual 3-4: Sistem pengoperasian, bahasa pengaturcaraan dan pelayar web yang disokong oleh Selenium Grid.

Sistem Pengoperasian	Bahasa Pengaturcaraan	Pelayar Web
i. Windows. ii. Linux. iii. MacOS.	i. Python. ii. C #. iii. Ruby. iv. Java. v. JavaScript.	i. Google Chrome. ii. Microsoft Edge. iii. Mozilla Firefox. iv. Microsoft Internet Explorer. v. Apple Safari.

3.2.10. K6

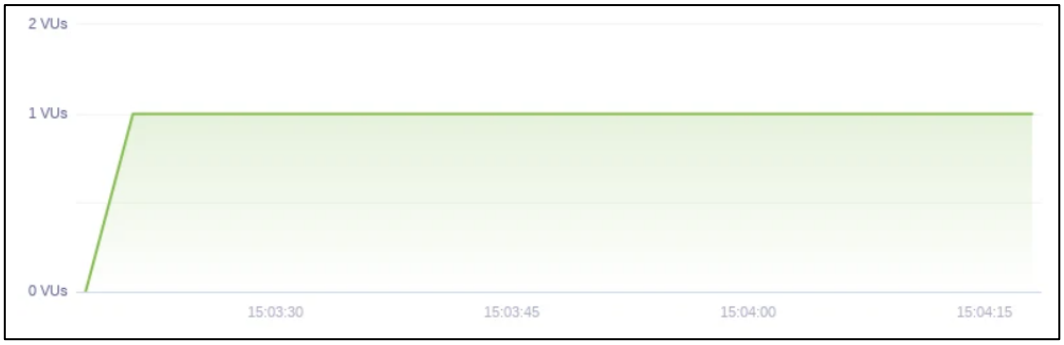
Grafana K6 merupakan *tools* pengujian prestasi *load testing* untuk sistem berasaskan web dan web API. K6 dapat melaksanakan pengujian terhadap *reliability* dan prestasi sistem aplikasi seterusnya mengesan masalah berkaitan prestasi lebih awal. K6 boleh menjalankan pelbagai jenis pengujian prestasi bergantung kepada skop pengujian yang ingin dilaksanakan. Rajah 3-13 menunjukkan jenis pengujian prestasi yang boleh dilaksanakan oleh K6.



Rajah 3-13: Jenis pengujian yang dilaksanakan oleh K6

Berikut adalah penerangan untuk setiap aktiviti pengujian.

a. **Pengujian *Smoke*** mengesahkan bahawa sistem boleh mengendalikan beban minimum, tanpa sebarang masalah. Rajah 3-14 memaparkan contoh graf hasil daripada pengujian *smoke*.



Rajah 3-14: Contoh Paparan Graf bagi Pengujian *Smoke*

b. Pengujian *Load* menilai prestasi sistem dari segi akses pengguna secara serentak atau berdasarkan permintaan sesaat (requests per second). Rajah 3-15 memaparkan contoh graf hasil daripada pengujian *load*.

Rajah 3-15: Contoh Paparan Graf bagi Pengujian *Load*

c. **Pengujian *Stress*** menilai had dan kestabilan sistem dalam keadaan yang maksimum berdasarkan had yang ditetapkan. Rajah 3-16 memaparkan contoh graf hasil daripada pengujian *stress*.

Rajah 3-16: Contoh Paparan Graf bagi Pengujian *Stress*

d. **Pengujian *Endurance*** menilai dan mengesan aktiviti berkaitan prestasi sistem dalam persekitaran produksi secara berterusan. Rajah 3-17 memaparkan graf hasil daripada pengujian *endurance*.

Rajah 3-17: Contoh Paparan Graf bagi Pengujian *Endurance*

3.2.12. Sitespeed.io

Sitespeed.io merupakan *tools* yang membantu dalam memantau, menganalisis dan mengoptimumkan prestasi laman web. Sitespeed.io mempunyai tiga keupayaan utama berikut:

a. Menjalankan pengujian laman web menggunakan metrik penyemak imbas *Navigation Timing API, User Timings and Visual Metrics* (FirstVisualChange, SpeedIndex & LastVisualChange).

b. Mensimulasikan ketersambungan pengguna sebenar dan mengumpulkan metrik yang penting seperti *Speed Index* dan *First Visual Render*.

c. Menganalisis cara laman web dibina dan mencadangkan penambahbaikan yang bersesuaian bagi meningkat prestasi laman web.

3.2.13. Zed Attack Proxy (ZAP)

Zed Attack Proxy atau lebih dikenali sebagai ZAP merupakan aplikasi pengimbas keselamatan yang dibangunkan oleh *Open Web Application Security Project* (OWASP). Aplikasi ini digunakan untuk menguji keselamatan aplikasi web melalui pengujian penembusan serta mengenal pasti kerentanan keselamatan seperti Cross-Site Scripting (XSS) dan SQL Injection. ZAP boleh diintegrasikan pada *pipeline* CI/CD untuk mengautomasikan pengujian keselamatan sebagai sebahagian daripada proses pembangunan yang bertujuan meningkatkan keselamatan aplikasi web.

3.2.14. OWASP Dependency-Check

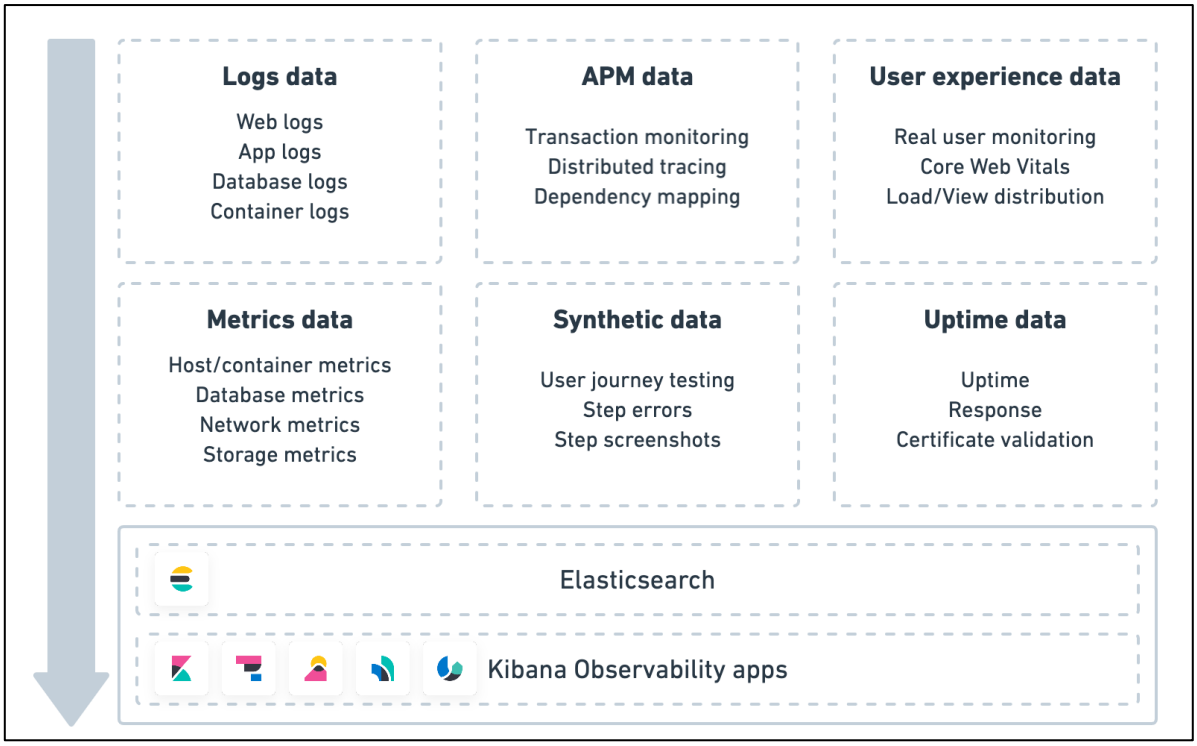
OWASP Dependency-Check merupakan *tool Software Composition Analysis* (SCA) yang berupaya mengesan kelemahan sistem aplikasi yang terdedah. Dependency-Check mengenal pasti kerentanan keselamatan dalam *library* dan kebergantungan pihak ketiga yang digunakan dalam sistem aplikasi. *Tools* ini berguna bagi pembangun dan penguji keselamatan untuk memastikan bahawa mereka tidak menggunakan komponen yang terdedah kepada kerentanan yang diketahui.

3.2.15. InfluxDB

InfluxDB merupakan pangkalan data siri masa yang dibangunkan oleh InfluxData. InfluxDB dioptimumkan untuk penyimpanan dan pengambilan data siri masa yang pantas dan ketersediaan tinggi seperti pemantauan operasi, metrik aplikasi, data sensor IoT dan analisis masa nyata.

3.2.16. Elastic Observability

Elastic Observability menyediakan satu platform yang mengumpulkan data log, metrik infrastruktur, data *uptime*, data jejak (traces) transaksi sistem aplikasi, data pengalaman pengguna dan data sintetik. Penggunaan Elastic Observability bertujuan mempertingkatkan kebolehtafsiran terhadap data yang dikumpul sebelum dipaparkan secara visual. Rajah 3-18 menunjukkan jenis data dan metrik yang diterima oleh Elastic Observability.



Rajah 3-18: Komponen yang terdapat dalam Elastic Observability.

Sumber: Elastic Observability

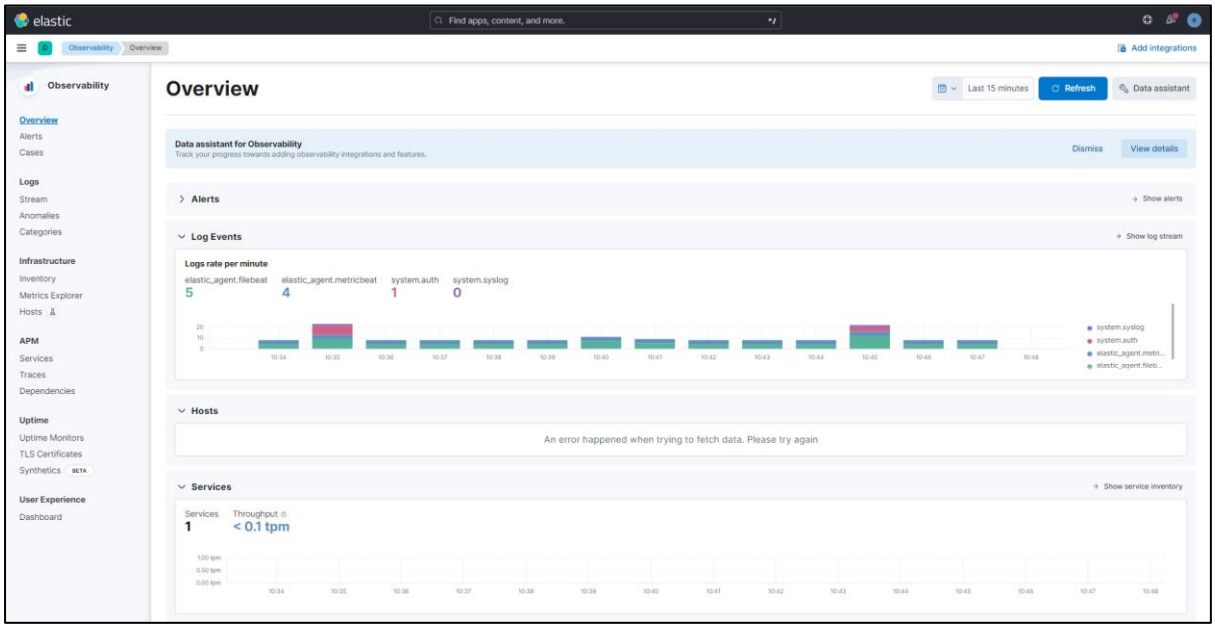
Penerangan berikut menjelaskan aliran data yang dihantar dari sumber ke Elastic Observability.

a. Data dari sumber seperti penggunaan CPU, penggunaan memori serta data aktiviti pengguna akan dihantar terus ke Elasticsearch.

b. Data akan diproses, dianalisis dan diubah kepada format yang tersusun.

c. Data yang telah diproses akan dipaparkan secara visual.

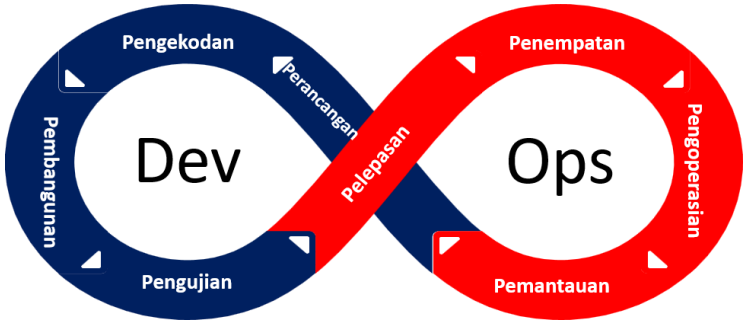
Rajah 3-19 memaparkan contoh paparan *tools* Elastic Observability.



Rajah 3-19: Contoh Paparan Elastik Observability

BAB 4 : KITAR HAYAT DEVOPS

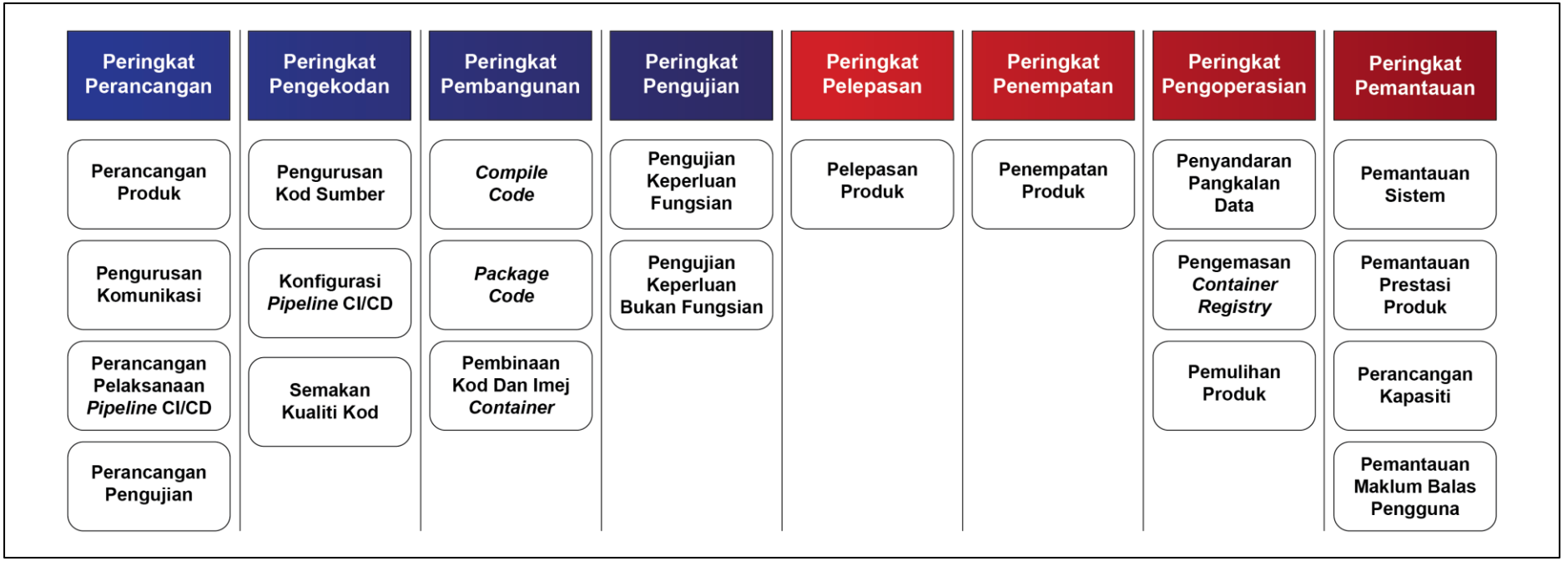
DevOps merupakan penerapan prinsip dan budaya yang membolehkan komunikasi dan kerjasama di antara pihak-pihak berkepentingan menjadi lebih baik terutamanya dalam menentukan hala tuju, membangunkan dan mengendalikan sistem aplikasi serta melaksanakan penambahbaikan berterusan dalam semua aspek kitaran hayat.



Rajah 4-1: Kitar Hayat Pelaksanaan DevOps

Seperti yang diterangkan dalam Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam, kitar hayat DevOps adalah seperti berikut:

a. Peringkat Perancangan.
b. Peringkat Pengekodan.
c. Peringkat Pembangunan.
d. Peringkat Pengujian.
e. Peringkat Pelepasan.
f. Peringkat Penempatan.
g. Peringkat Pengoperasian.
h. Peringkat Pemantauan.



Rajah 4-2: Aktiviti di Peringkat DevOps

Rajah 4-2 memaparkan senarai aktiviti utama yang terdapat pada setiap peringkat DevOps. Penerangan setiap aktiviti tersebut akan diterangkan pada setiap peringkat berikut.

4.1. PERINGKAT PERANCANGAN

Peringkat perancangan merupakan peringkat penting dalam proses pembangunan produk yang berfungsi dan memenuhi keperluan pengguna. Penambahbaikan produk yang dirancang pada peringkat ini dipengaruhi oleh pengumpulan keperluan dan maklum balas pengguna dari pihak berkepentingan. Keutamaan peringkat ini adalah untuk memastikan bahawa matlamat dan hala tuju produk yang sedang dibangunkan difahami dengan jelas oleh setiap ahli pasukan DevOps. Penetapan matlamat yang jelas membolehkan pasukan DevOps merancang dan membangunkan produk yang dapat memenuhi kehendak pengguna dengan tepat. Penerangan berkaitan peringkat perancangan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.2.

Aktiviti *agile* seperti *sprint zero, sprint planning meeting* pertama dan *sprint planning meeting* kedua akan dilaksanakan serta artifak yang terhasil daripada aktiviti tersebut akan menjadi input bagi aktiviti pada peringkat ini. Artifak *product vision, epic, user story,* DoD dan perancangan kapasiti akan didokumenkan dalam bentuk wiki manakala *product backlog* dan *sprint backlog* akan diterjemahkan ke dalam *tools* GitLab**.**

Templat pelaksanaan aktiviti *agile* boleh dirujuk pada **Lampiran A.**

Peringkat perancangan terdiri daripada empat aktiviti utama iaitu:

a. Perancangan Produk,

b. Pengurusan Komunikasi,

c. Pengurusan *Tools* dan Proses *pipeline* CI/CD, serta

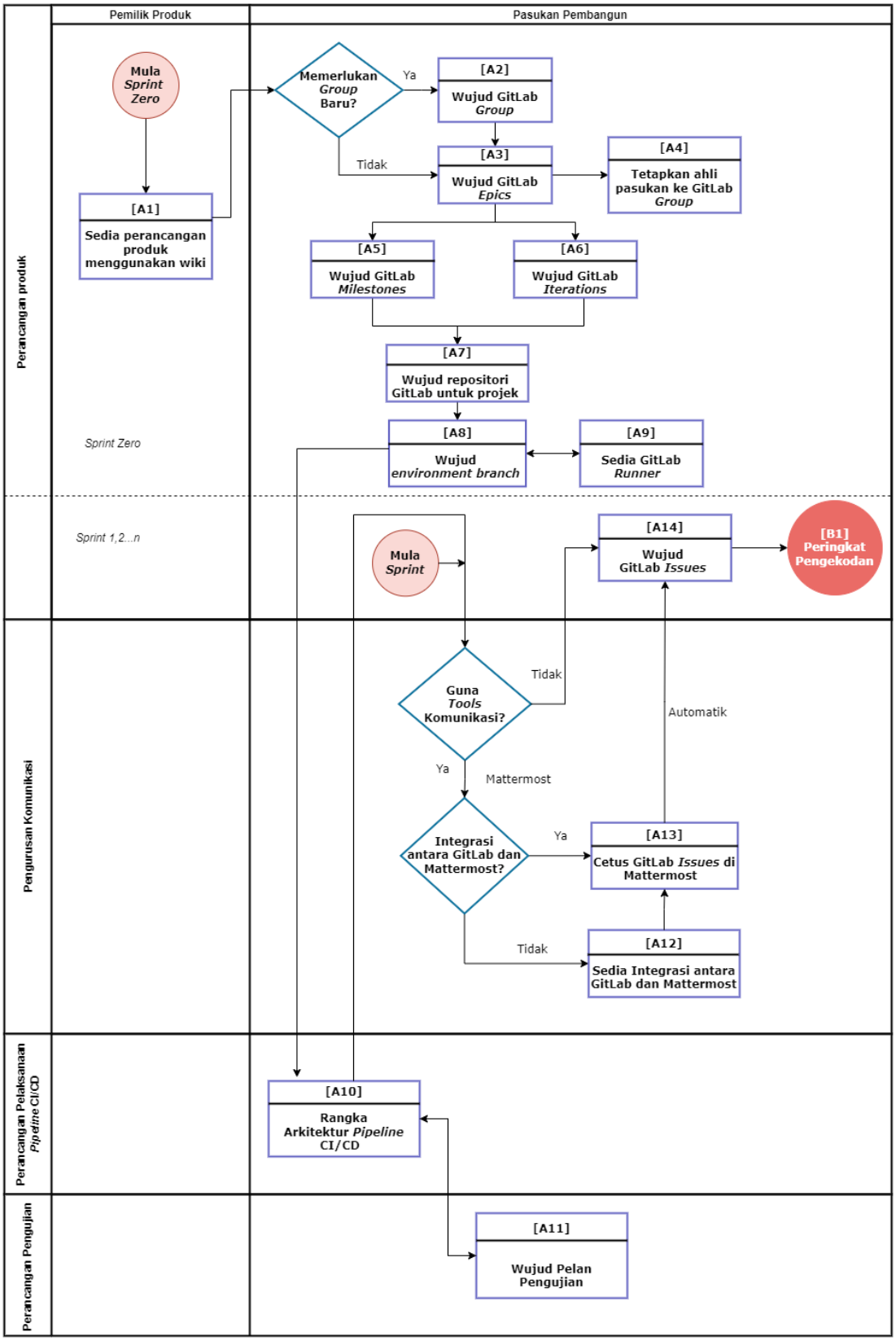
d. Perancangan Pengujian.

4.1.1. Prasyarat

Peringkat perancangan hanya bermula setelah mendapat kelulusan Mesyuarat Jawatankuasa Pemandu ICT (JPICT) serta penyediaan PPS serta BRS telah selesai.

4.1.2. Aliran Proses Kerja Peringkat Perancangan

Peringkat perancangan melibatkan aktiviti-aktiviti seperti memperhalusi keperluan pengguna dalam skop produk dan merancang pengurusan bagi penggunaan *tools* serta *pipeline* CI/CD. Penyediaan *tools* boleh dibuat pada *sprint zero* setelah pelan hala tuju produk ditetapkan. Rajah 4-3 memaparkan aliran proses kerja yang terlibat dalam peringkat perancangan.



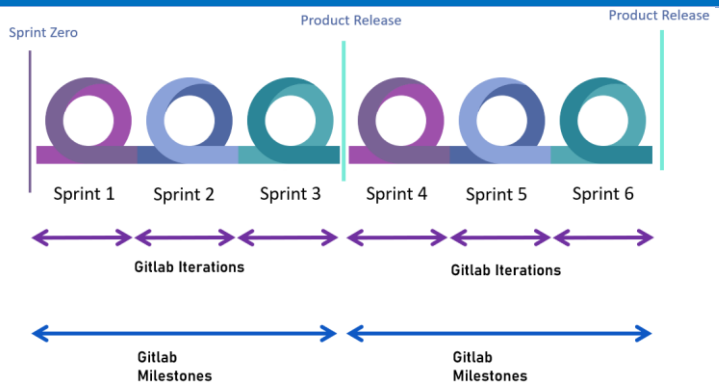
Rajah 4-3: Aliran Kerja Peringkat Perancangan

Jadual 4-1 menerangkan proses yang berlaku dalam peringkat perancangan berdasarkan Rajah 4-3.

Jadual 4-1: Penerangan Aktiviti dalam Peringkat Perancangan

Aktiviti	Penerangan
Sedia perancangan produk [A1]	Pelan perancangan produk yang telah diterjemahkan kepada templat *product vision* dan *user story*. Artifak ini akan didokumenkan ke dalam aplikasi wiki seperti Wiki.js atau wiki di dalam GitLab.
Wujud GitLab *Group* [A2]	GitLab *Group* membolehkan ahli pasukan mengakses repositori projek dengan memberi kebenaran pada peringkat kumpulan. GitLab *Group* digunakan untuk: i. Mengurus satu atau lebih projek yang berkaitan pada masa yang sama. ii. Mengurus kebenaran akses kepada projek yang berkaitan kepada ahli kumpulan. iii. Melihat semua isu dan *merge request* untuk projek dalam kumpulan. iv. Melihat *dashboard* yang menunjukkan aktiviti dan kemajuan kumpulan. v. Menggunakan *group* untuk berkomunikasi dengan semua ahli kumpulan sekali gus.
Wujud GitLab *Epics* [A3]	GitLab *Epics* membenarkan *user story* atau GitLab *Issues* dikumpulkan dan berkongsi tema yang sama merentasi projek dan *milestones*. Ini membolehkan portfolio projek diuruskan dengan lebih cekap. GitLab *Epics* sesuai digunakan bagi senario berikut: i. Melibatkan perbincangan dan pemantauan terhadap isu sistem atau *feature* pada projek yang berbeza di dalam kumpulan yang sama, ii. Menjejaki tugasan bagi senarai tugasan yang disasarkan mengikut *milestone* yang ditetapkan merentasi projek di dalam kumpulan yang sama, atau iii. Mengumpulkan idea, skop dan ciri-ciri produk secara generik melalui perbincangan di dalam kumpulan yang sama.

Aktiviti	Penerangan
Tetapkan ahli pasukan ke dalam GitLab *Groups* [A4]	Ahli pasukan DevOps dimasukkan ke dalam GitLab *Groups* bagi melihat dan mengakses kepada projek yang berkaitan.
Wujud GitLab *Milestones* [A5]	GitLab *Milestones* digunakan untuk mengurus dan mengesan kemajuan *user story* bagi tempoh masa tertentu dengan menggunakan carta *Burndown*. *User story* adalah dikelaskan mengikut label dan status (unassigned, assigned dan completed). Perkara yang perlu dititikberatkan semasa mewujudkan GitLab *Milestones* adalah seperti berikut: i. *Milestones* boleh diwujudkan pada kedua-dua peringkat *Project* dan *Group*, ii. *Milestones* boleh menetapkan tarikh yang akan digunakan pada *GitLab Epics* (sekiranya *epic* menggunakan tarikh *inherits* daripada *issues* dan *milestones*), iii. *Milestone* boleh menetapkan tempoh masa tertentu (tarikh mula dan tarikh tamat), iv. *GitLab Issues* akan ditambah ke *milestones* dan v. Senarai *Milestone* boleh disusun mengikut nama dan tarikh (sama ada tarikh akhir atau tarikh mula). *Milestones* juga boleh digunakan untuk perancangan pelepasan produk ke persekitaran produksi. Sebagai contoh produk berfungsi yang telah dibangun, dijadualkan untuk pelepasan ke persekitaran produksi selepas *sprint* 3. Maka tarikh mula yang dicadangkan pada *milestones* ini adalah pada hari bermulanya *sprint* 1 manakala tarikh tamat adalah tarikh akhir *sprint* 3. Penetapan tarikh *milestones* boleh diubah bergantung persetujuan bersama sepanjang pembangunan sistem aplikasi tersebut.
Wujud GitLab *Iterations* [A6]	GitLab *Iterations* adalah cara untuk menjejaki isu sepanjang tempoh masa. Ini membolehkan pasukan menjejaki metrik *velocity* dan prestasi. *Iterations* digunakan untuk merancang *agile* atau *agile-like sprint* untuk mengenal pasti tugasan yang perlu diselesaikan dalam tempoh masa yang ditetapkan. Rajah 4-4 menunjukkan gambaran tempoh masa bagi GitLab *Iterations* dan *Milestones*.

Aktiviti	Penerangan
	 **Rajah 4-4: Tempoh Masa bagi GitLab *Iterations* dan *Milestones***
Wujud repositori GitLab untuk projek [A7]	GitLab *Project Repository* digunakan untuk menyimpan dan mengintegrasikan kod sumber daripada setiap ahli pasukan pembangun.
Wujud *environment branch* [A8]	GitLab *environment branch* digunakan sebagai salinan repositori daripada *branch* utama (main). Antara contoh *environment branch* adalah seperti *staging* dan produksi.
Sedia GitLab *Runner* [A9]	GitLab *Runner* digunakan untuk menjalankan *jobs* yang ditetapkan dan menghantar hasilnya kepada GitLab. Contoh *jobs* adalah *compile code.*
Rangka Arkitektur *Pipeline* CI/CD [A10]	*Pipeline* CI/CD boleh dirangka terlebih dahulu sebelum penulisan skrip disediakan.
Wujud Pelan Pengujian [A11]	Pelan pengujian akan diwujudkan semasa *sprint planning meeting* kedua untuk digunakan pada peringkat pengujian.
Sedia Integrasi antara GitLab dan Mattermost [A12]	Projek yang menggunakan Mattermost sebagai saluran komunikasi, memerlukan integrasi di antara Mattermost dan GitLab bagi membenarkan proses komunikasi dua hala.
Cetus GitLab *Issues* dari Mattermost [A13]	Ahli pasukan boleh mencetuskan GitLab *Issues* baharu menggunakan skrip arahan dari Mattermost *ChatOps*.
Wujud GitLab *Issues* [A14]	*Product backlog* dan *sprint backlog* akan diwujudkan di bawah GitLab *Issues*. GitLab *Labels* boleh digunakan untuk membezakan antara *product backlog* dan *sprint backlog.*

4.1.3. Perancangan Produk

Perancangan produk menumpukan kepada pelan hala tuju produk bagi menyelaraskan objektif dan memfokuskan kepada ciri-ciri produk yang akan dibangunkan. Aktiviti perancangan produk juga merangkumi penyediaan *tools* yang akan digunakan sepanjang pembangunan sistem aplikasi. Pelan hala tuju produk perlu direkodkan bagi membantu pemantauan kemajuan sepanjang aktiviti pembangunan. Pelan ini terdiri daripada artifak *product vision, epic* dan *user story.* Persediaan *tools* boleh dibuat setelah *sprint zero* dilaksanakan iaitu setelah pelan hala tuju produk telah ditetapkan.

4.1.3.1. Penggunaan *Tools*

Tools yang akan digunakan bagi aktiviti perancangan produk adalah seperti berikut:

a. Wiki.js atau Wiki GitLab
b. *Features* GitLab yang terlibat adalah seperti berikut:
    i. *Groups*
    ii. *Epics*
    iii. *Milestones*
    iv. *Iterations*
    v. *Roadmap*
    vi. *Projects*
    vii. *Issues*
    viii. *Tasks*
    ix. *Branches*

4.1.3.2. Ringkasan Tatacara Penggunaan *Tools*

a. Sedia Perancangan Produk menggunakan Wiki.js atau Wiki GitLab [A1]

Perancangan yang telah dibincangkan bersama-sama ahli pasukan boleh diterjemahkan ke dalam bentuk wiki bagi tujuan dokumentasi. Antara dokumen yang boleh didokumenkan pada aktiviti ini adalah pelan hala tuju produk, DoD dan perancangan kapasiti.

Tatacara untuk menyediakan pelan hala tuju produk pada Wiki.js boleh dirujuk pada **Lampiran B-1** manakala untuk Wiki GitLab pada **Lampiran B-2**.

b. Wujud GitLab *Groups* [A2]

GitLab *Groups* digunakan untuk mengurus satu atau lebih projek dalam masa yang sama. Di bawah *groups* ini juga *epics* akan dihasilkan.

Tatacara untuk mewujudkan GitLab *Groups* baharu boleh dirujuk pada **Lampiran B-3**.

c. Wujud GitLab *Epics* [A3]

Setelah mengenal pasti *groups* untuk pengurusan produk tersebut, *epics* boleh diwujudkan. *Epics* merupakan produk premium daripada GitLab dan pengguna GitLab perlu mempunyai akses sekurang-kurangnya *reporter role* bagi mengakses *epics*.

Tatacara untuk mewujudkan GitLab *Epics* boleh dirujuk pada **Lampiran B-4**.

d. Tetapkan Ahli Pasukan ke GitLab *Groups* [A4]

Setelah mengenal pasti *group*s untuk pengurusan produk, pemilik produk boleh menambah pengguna baharu ke *groups.*

Tatacara untuk menetapkan ahli pasukan ke GitLab *Groups* boleh dirujuk pada **Lampiran B-5**.

e. Wujud GitLab *Milestones* [A5]

GitLab *Milestones* boleh diwujudkan sama ada di bawah *Groups* atau *Projects*. Paparan Carta *Burndown* pada GitLab *Roadmap* terdapat di bawah GitLab *Milestones* ini.

Tatacara untuk mewujudkan *Milestones* di bawah *Group* boleh dirujuk pada **Lampiran B-6**.

f. Wujud GitLab *Iterations* [A6]

GitLab *Iterations* merupakan tempoh *sprint* yang ditetapkan pada *sprint zero*. Sebagai contoh yang ditetapkan pada kajian kes ini iaitu selama dua minggu.

Tatacara untuk mewujudkan GitLab *Iterations* boleh dirujuk pada **Lampiran B-7**.

g. Wujud Repositori GitLab untuk Projek [A7]

GitLab *Project* digunakan untuk menguruskan projek atau produk dan dijadikan repositori. Pasukan boleh mewujudkan projek untuk menjejak isu, merancang kerja, bekerjasama pada kod dan terus membina, menguji dan menggunakan *pipeline* CI/CD untuk pembangunan produk. Projek boleh disediakan secara terbuka, dalaman atau persendirian, tanpa had bilangan.

Repositori merupakan tempat simpanan kod sumber dan membuat perubahan kod padanya. Setiap perubahan akan dijejaki dengan kawalan versi. Setiap projek di GitLab akan mengandungi repositori tersendiri. Repositori akan turut diwujudkan setelah projek diwujudkan.

Tatacara mewujudkan projek baharu pada GitLab boleh dirujuk pada **Lampiran B-8**.

h. Wujud GitLab Environment *Branch* [A8]

GitLab *environment branch* merujuk kepada *branch* Git yang digunakan untuk menguruskan kod sumber yang khusus untuk persekitaran tertentu sepanjang pembangunan produk. Penggunaan *branch* yang berbeza bagi setiap persekitaran membolehkan pasukan pembangun menguruskan perubahan kod

sumber pada setiap persekitaran secara berasingan dan mengurangkan risiko masalah yang berlaku di persekitaran lain semasa pembangunan produk.

Semasa pembangunan produk, terdapat beberapa persekitaran yang digunakan dan setiap persekitaran memerlukan konfigurasi dan kod sumber yang berbeza seperti persekitaran pembangunan, *staging* dan produksi. Penetapan *environment branch* ini boleh dibuat pada peringkat awal sebelum *sprint* bermula. Jadual 4-2 menerangkan penetapan *environment branch*.

Jadual 4-2: Jadual Penetapan *Environment Branch*

No.	*Environment Branch*	Penjelasan
1.	*Branch* Utama (Main)	i. Merupakan *branch* pembangunan setelah pasukan pembangun selesai mengemas kini kod sumber. ii. Setiap pembangun akan menggabungkan kod sumber yang telah dikemas kini ke *branch* utama bagi tujuan pengujian di persekitaran pembangunan.
2.	*Branch Staging*	i. Merupakan *branch* yang digunakan untuk aktiviti pengujian pada persekitaran *staging.* ii. Pembangun akan menggabungkan kod sumber dari *branch* utama ke *branch staging* untuk menyelaraskan kod sumber.
3.	*Branch* Produksi	i. Merupakan *branch* yang digunakan untuk pelepasan ke persekitaran produksi. ii. Pembangun akan menggabungkan kod sumber dari *branch staging* ke *branch* produksi untuk menyelaraskan kod sumber.

Tatacara untuk menetapkan *environment branch* boleh dirujuk pada **Lampiran B-9**.

i. Sedia GitLab *Runners* [A9]

GitLab *runners* boleh dipasang dan digunakan pada GNU/Linux, macOS, FreeBSD dan Windows. Pengguna boleh memasang *runners* dengan beberapa cara berikut:

a. Pemasangan dalam *container,*

b. Pemasangan dengan memuat turun fail pakej rpm/deb dari repositori GitLab*.*

GitLab *runners* yang menyokong binari bagi arkitektur x86, AMD64, ARM64, ARM, s390x dan ppc64le. *Official Packages* boleh didapati bagi Linux *distributions* seperti CentOS, Debian, Ubuntu, RHEL, Fedora, Mint, Oracle dan Amazon.

Tatacara untuk menyediakan *runners* boleh dirujuk pada **Lampiran B-10**.

j. Wujud GitLab *Issues* [A14]

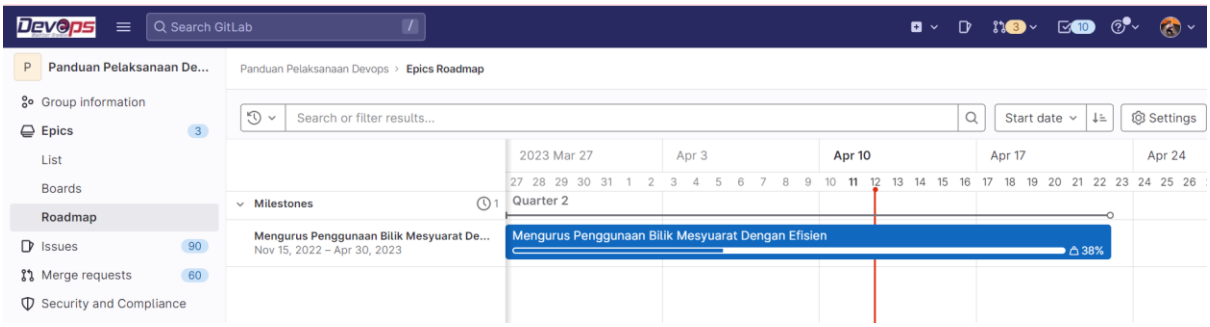
Setiap *product backlog* dan *sprint backlog* akan diwujudkan menggunakan GitLab *Issue* dan dilabelkan.

Tatacara untuk mewujudkan GitLab *Issues* boleh dirujuk pada **Lampiran B-11**.

k. Lihat Paparan GitLab *Roadmap*

Epics dan *milestones* di dalam *group* yang mengandungi tarikh mula dan tarikh siap boleh digambarkan dalam bentuk carta Gantt. Halaman *roadmap* menunjukkan *epics* dan *milestones* di dalam *group* yang sama. Bar *epic* akan memaparkan tajuk dan peratusan kemajuan yang telah selesai. Rajah 4-5 memaparkan contoh paparan GitLab *Roadmap*.

Tatacara untuk melihat paparan GitLab *Roadmap* boleh dirujuk pada **Lampiran B-12**.



Rajah 4-5: Paparan GitLab *Roadmap*

4.1.4. Pengurusan Komunikasi

Pengurusan komunikasi merangkumi komunikasi, perbincangan dan pengumuman di antara ahli pasukan sepanjang pembangunan produk. Penglibatan semua ahli pasukan boleh menyumbang kepada keberkesanan komunikasi untuk menyokong aspek budaya DevOps. Antara contoh aktiviti dalam pengurusan komunikasi termasuklah penggunaan *ChatOps,* pengumuman berkaitan pelepasan versi produk dan juga perkongsian fail.

4.1.4.1. Penggunaan *Tools*

Tools yang akan digunakan bagi aktiviti pengurusan komunikasi adalah Mattermost *ChatOps*.

4.1.4.2. Ringkasan Tatacara Penggunaan *Tools*

Tatacara berikut melibatkan penggunaan Mattermost sebagai saluran komunikasi sepanjang pembangunan produk. Antara aktiviti pengurusan komunikasi adalah seperti berikut:

a. Wujud Mattermost *Team*

Team perlu diwujudkan sebelum mewujudkan *channel* dan integrasi dengan GitLab. **Tatacara untuk mewujudkan *team*** di dalam aplikasi Mattermost boleh dirujuk pada **Lampiran B-13**.

b. Wujud Mattermost *Channel*

Setelah *Team* diwujudkan atau disertai, *channel* perlu diwujudkan sebelum integrasi dengan GitLab. **Tatacara untuk mewujudkan *channel*** di dalam aplikasi Mattermost boleh dirujuk pada **Lampiran B-14**.

c. Sedia Integrasi antara GitLab dan Mattermost [A12]

Integrasi antara GitLab dan Mattermost merujuk kepada penyatuan hubungan antara dua *tools* ini. Kedua-dua platform ini boleh diintegrasikan supaya pasukan DevOps dapat berkolaborasi dan berkomunikasi secara lebih efektif dalam pelaksanaan pembangunan produk. Integrasi ini membantu mempercepatkan proses pembangunan produk dan membantu pasukan DevOps lebih mudah berkomunikasi secara terus dalam satu platform.

Tatacara untuk menyediakan integrasi antara GitLab dan Mattermost boleh dirujuk pada **Lampiran B-15.**

d. Cetus GitLab *Issues* di Mattermost [A13]

Mattermost *ChatOps* mempunyai kebolehan untuk *create new* GitLab *Issues* melalui *command line* pada *channel*. Melalui *ChatOps*, ahli pasukan boleh mencadangkan *product backlog* atau *user story*. Jika cadangan dipersetujui, GitLab *Issues* akan diwujudkan.

Tatacara untuk mencetus GitLab *Issue* daripada Mattermost boleh dirujuk pada **Lampiran B-16**.

4.1.5. Perancangan Pelaksanaan *Pipeline* CI/CD

Senarai *jobs* bagi setiap peringkat yang perlu diautomasikan pada *pipeline* CI/CD perlu dirancang lebih awal bagi persediaan sebelum memasuki ke peringkat pengekodan. **Tatacara untuk perancangan pelaksanaan pipeline CI/CD** boleh dirujuk pada **Lampiran B-17**.

4.1.5.1. *Pipeline* CI/CD

Pipeline merupakan komponen terpenting dalam asas penambahbaikan berterusan.

Pipeline merangkumi dua komponen utama iaitu:

a. *Jobs*

Komponen ini menjelaskan ‘apa yang perlu dilakukan’. Contoh job adalah *compile code*.

b. *Stages*

Komponen ini menjelaskan ‘bila *jobs* perlu dijalankan’. Contohnya stage `test` perlu dilaksanakan selepas stage `build`.

Jobs akan dilaksanakan oleh *runners*. Sekiranya semua *jobs* pada satu-satu *stages* berjaya dilaksanakan, *pipeline* akan melaksanakan *jobs* pada peringkat berikutnya. Jika terdapat *jobs* yang gagal, *pipeline* akan tamat lebih awal. Secara umumnya, *pipeline* akan dilaksanakan secara automatik dan tanpa memerlukan intervensi manual sebaik sahaja konfigurasi *pipeline* ditetapkan.

Pipeline pada kebiasaannya merangkumi empat *stages*, dilaksanakan mengikut aturan berikut:

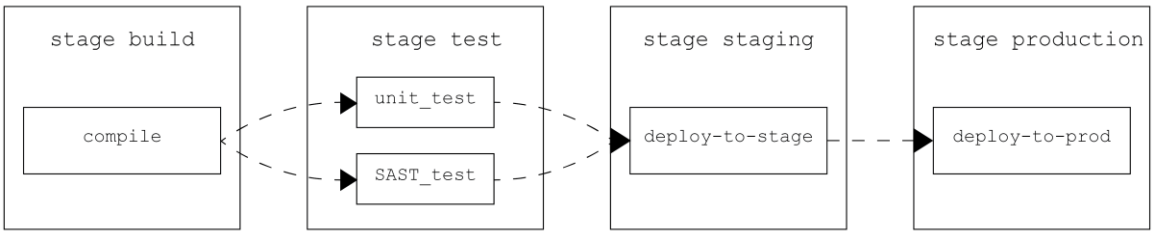
a. ***Stage*** `build` dengan *job* yang dipanggil `compile`.

b. ***Stage*** `test` dengan dua *jobs* yang dipanggil `unit_test` dan `SAST_test`.

c. ***Stage*** `staging` dengan *job* yang dipanggil `deploy-to-stage`.

d. ***Stage*** `production` dengan *job* yang dipanggil `deploy-to-prod`.

Rajah 4-6 memaparkan contoh *pipeline* yang lazim digunakan.



Rajah 4-6: Contoh *Pipeline*

4.1.5.2. Jenis-Jenis *Pipelines*

Pipelines boleh dikonfigurasikan dengan menggunakan beberapa pendekatan. Antara pendekatan yang biasa digunakan adalah seperti berikut:

a. ***Pipeline Basic*** yang menjalankan semua *job* pada setiap peringkat secara serentak diikuti dengan peringkat seterusnya.

b. ***Pipeline Directed Acyclic Graph* (DAG)** merupakan perkaitan antara *jobs* dan boleh dilaksanakan dengan lebih cepat daripada *pipeline basic*.

c. ***Pipeline Parent-child*** dipecahkan dari *pipeline* yang kompleks kepada satu *parent pipeline* yang akan *trigger* pelbagai *child sub-pipelines* dimana semuanya dibawah satu projek. *Pipeline* jenis ini biasanya digunakan pada monorepo.

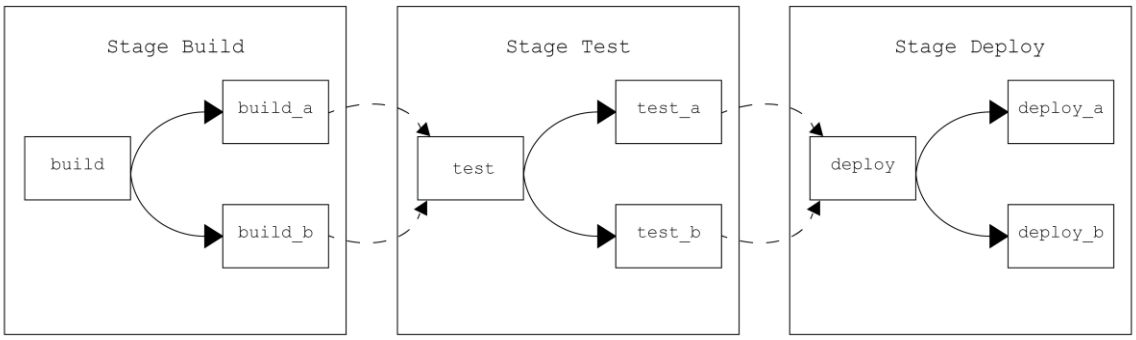
d. ***Pipeline Multi-project*** menggabungkan dua atau lebih projek yang berlainan di dalam satu *pipeline* yang sama.

4.1.5.3. Arkitektur *Pipelines*

Pipeline merupakan struktur asas bagi pembentukan CI/CD dalam pelaksanaan DevOps. *Pipeline* boleh disusun dengan pendekatan yang berbeza bergantung kepada kelebihan dan keperluan masing-masing. Pendekatan ini boleh dicampur dan dipadankan jika diperlukan. Berikut adalah antara contoh arkitektur *pipeline*:

a. Pipeline Basic

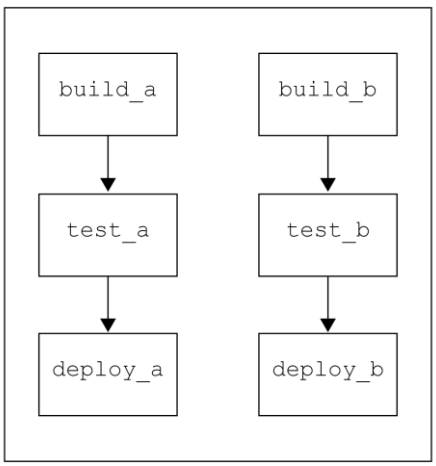
Sesuai bagi projek yang kecil apabila semua konfigurasi berada pada lokasi yang sama. Rujuk Rajah 4-7 bagi gambaran arkitektur *pipeline basic.*



Rajah 4-7: Arkitektur *Pipeline Basic*

b. Pipeline DAG

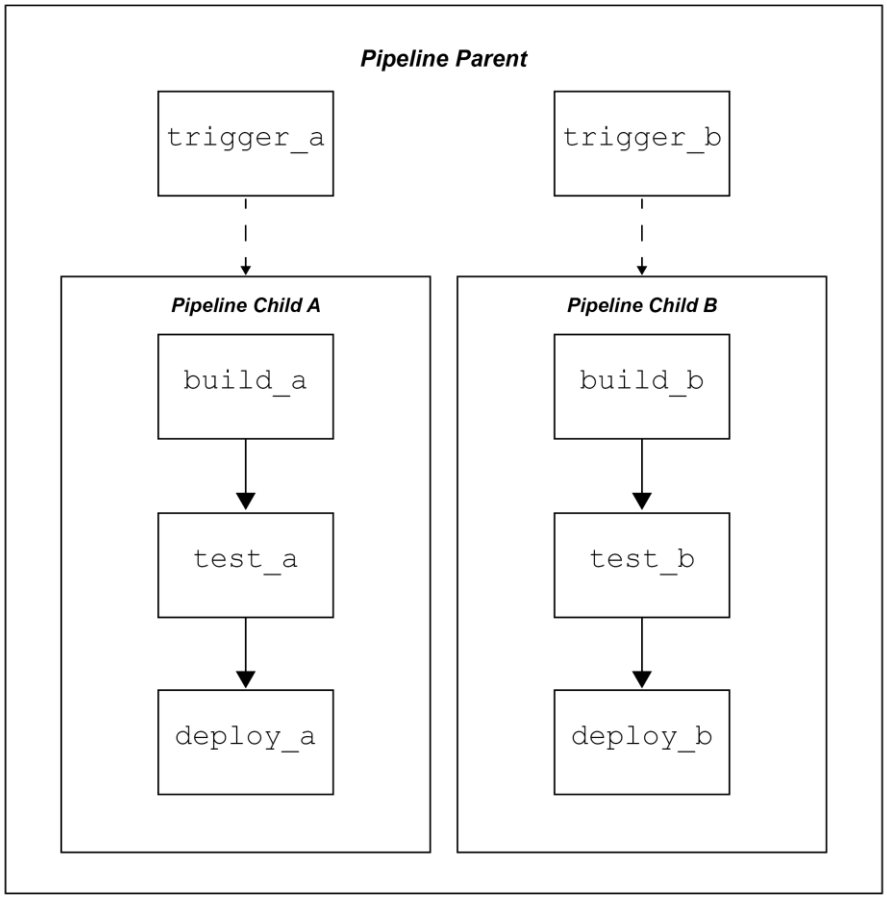
Sesuai bagi projek yang besar dan lebih kompleks yang memerlukan pelaksanaan yang lebih cekap. Rujuk Rajah 4-8 bagi gambaran arkitektur pipeline DAG.



Rajah 4-8: Arkitektur *Pipeline* DAG

c. Pipeline Parent-child

Sesuai bagi monorepo dan projek yang mempunyai komponen yang tersendiri. Rujuk Rajah 4-9 bagi arkitektur *pipeline parent-child*.



Rajah 4-9: Arkitektur *Pipeline Parent-child*

d. Pipeline Multi-Project

Sesuai untuk produk yang lebih besar yang memerlukan kebergantungan antara projek seperti arkitektur *microservices*.

Berdasarkan arkitektur *pipeline* yang telah dibincangkan, *pipeline* yang disyor untuk digunakan adalah arkitektur *basic* dan DAG, tertakluk kepada saiz dan kompleksiti sistem aplikasi yang dibangunkan.

4.1.5.4. Fail `.gitlab-ci.yml`

Selain daripada kod sumber sistem aplikasi, fail `.gitlab-ci.yml` yang merupakan komponen utama di dalam pembentukan *pipeline* CI/CD turut disimpan di dalam repositori Git bagi sesebuah projek. Lokasi fail ini ditempatkan di *root* repositori dan mengandungi tetapan konfigurasi CI/CD bagi projek tersebut.

Berikut merupakan perkara yang perlu dirancang sebelum mewujudkan fail `.gitlab-ci.yml` :

a. Skrip yang diperlukan untuk dilaksanakan,

b. Fail konfigurasi lain atau templat yang hendak dimasukkan,

c. *Dependencies* dan *caches* yang diperlukan,

d. *Command* yang hendak dilaksanakan mengikut urutan atau secara serentak,

e. Persekitaran untuk menempatkan produk tersebut (pembangunan/*staging*/produksi), dan

f. Ketetapan sama ada skrip ini dilakukan secara automatik atau manual.

4.1.5.5. Penggunaan *Tools*

Tools yang diperlukan untuk membuat konfigurasi *pipeline* CI/CD adalah dengan menggunakan *feature* sedia ada di dalam GitLab yang dinamakan sebagai GitLab CI/CD. Kaedah membuat konfigurasi *pipeline* melalui penulisan skrip akan diterangkan di dalam para 4.2.4.

4.1.5.6. Ringkasan Tatacara Penggunaan *Tools*

Penjelasan penggunaan *pipeline* CI/CD boleh dirujuk pada para 4.2.4. **Tatacara untuk mewujud, mengakses dan mengemas kini fail `.gitlab-ci.yml`** boleh dirujuk pada **Lampiran B-16**.

4.1.6. Perancangan Keselamatan

Pasukan perlu melaksanakan perancangan keselamatan untuk mengenal pasti risiko keselamatan seawal mungkin dalam proses pembangunan dan sebagai input semasa fasa reka bentuk sistem aplikasi yang selamat. Aktiviti berikut dilaksanakan semasa perancangan keselamatan.

a. Penilaian Risiko (Risk Assessment).

Penerangan berkaitan penilaian risiko boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.6.2.1.

b. Pemodelan Ancaman (Threat Modeling).

Penerangan berkaitan permodelan ancaman boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.6.2.2.

4.1.7. Perancangan Pengujian

Untuk menjalankan aktiviti pengujian, pasukan perlu memastikan ketersediaan perkara berikut.

a. Dokumentasi:

i. Keperluan fungsian telah dipersetujui dan disemak di antara pasukan pembangun dan pemilik produk.

ii. Pelan Pengujian telah dibangun dan disemak oleh pemilik produk.

iii. Kes Pengujian telah dibangun dan disemak di antara pasukan pembangun dan pemilik produk.

Penerangan berkaitan penyediaan dokumentasi perancangan pengujian boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.2.4.

b. Persekitaran dan *tools*:

i. Konfigurasi GitLab *Runner* supaya menyokong arahan kerja berkaitan aktiviti pengujian.

ii. *Tools* pengujian yang menyokong bahasa pengaturcaraan yang digunakan.

Penerangan berkaitan tatacara penggunaan *tools* untuk konfigurasi GitLab boleh dirujuk pada dokumen panduan pelaksanaan DevOps dalam pembangunan sistem aplikasi sektor awam di **Lampiran E: Peringkat Pengujian**.

4.1.8. Serahan/Output

Serahan/output bagi peringkat ini adalah:

a. Pelan hala tuju produk

b. Artifak *product backlog*

c. Artifak *sprint backlog*

d. Pelan pengujian

e. Kes pengujian

f. Persekitaran GitLab yang telah dikonfigurasi seperti berikut:

   i. Groups

   ii. Epics

   iii. Projects

   iv. Milestones

   v. Iterations

   vi. *Runners*

   vii. *Issues*

g. Persekitaran komunikasi Mattermost *ChatOps*.

4.2. PERINGKAT PENGEKODAN

Peringkat pengekodan merupakan peringkat pembangunan kod aplikasi berdasarkan *user story* yang telah ditetapkan pada *sprint* semasa. Pengemaskinian kod sumber turut berlaku pada peringkat ini sekiranya berlaku perubahan. Aktiviti *daily scrum meeting* akan diadakan pada peringkat ini bagi tujuan pemantauan dan pelaporan tahap kemajuan pembangunan produk yang dibangunkan. Pembangun akan mendaftar masuk dan mengintegrasikan kod sumber ke repositori sistem pengurusan kod sumber berpusat (repositori berpusat). Penerangan berkaitan peringkat pengekodan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.3.

Setelah pasukan pembangun menyelesaikan *sprint task*, mereka akan *commit* dan *push* kod sumber ke repositori berpusat. Pembangun akan menghantar perubahan kod sumber dengan *merge request* iaitu permintaan untuk menggabungkan kod baharu mereka ke repositori berpusat. Penilai iaitu pembangun yang mempunyai kepakaran dalam semakan kod sumber akan menyemak perubahan yang telah pasukan pembangun laksanakan. Setelah analisis dibuat dan tidak menemui sebarang masalah, mereka akan meluluskan *merge request*. Semakan secara manual ini sepatutnya lebih pantas, tetapi berkesan bagi mengenal pasti masalah lebih awal.

Terdapat 3 aktiviti utama pada peringkat ini:

a. Pengurusan Kod Sumber,

b. Konfigurasi *Pipeline* CI/CD dan

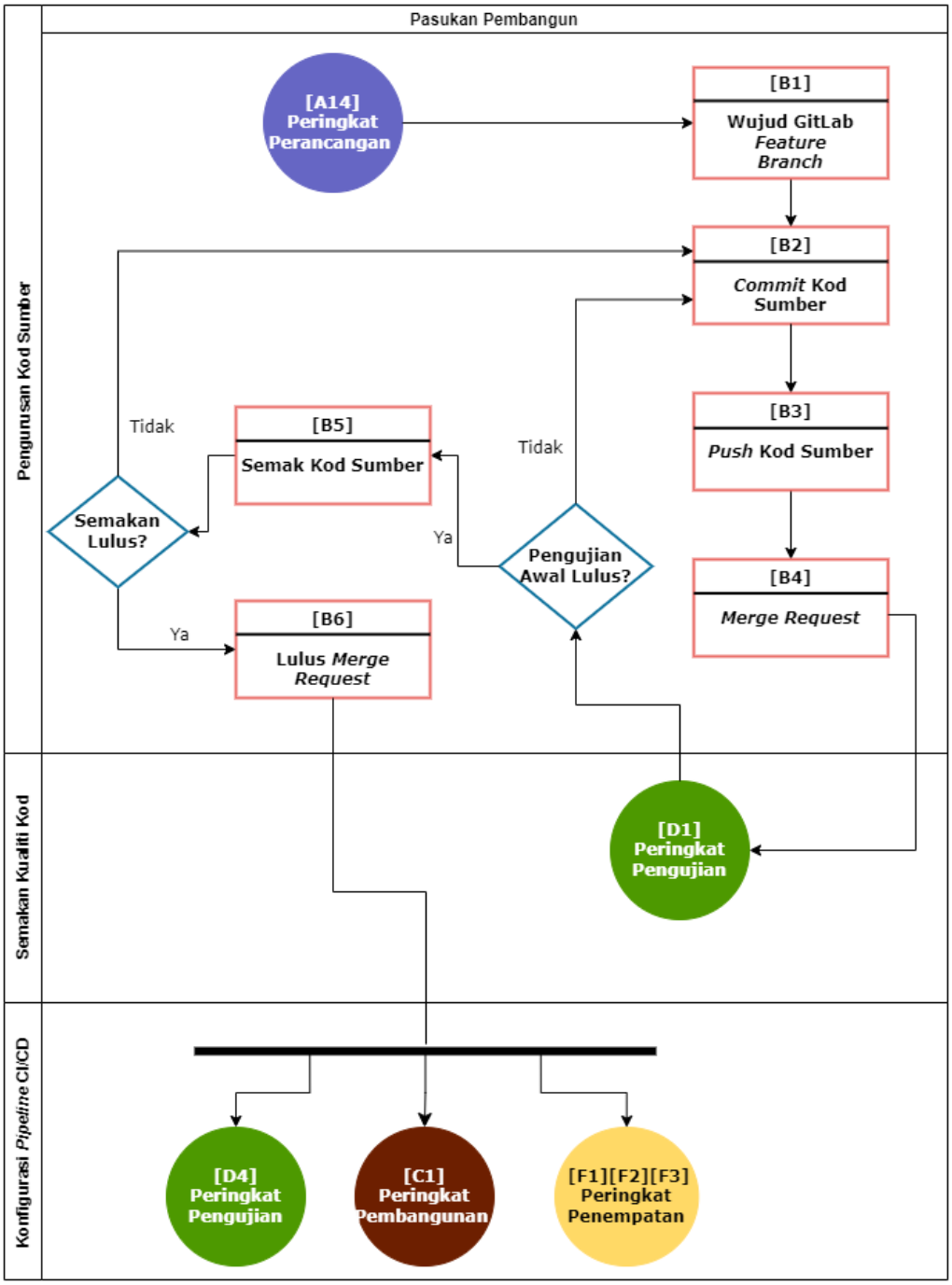
c. Semakan Kualiti Kod.

4.2.1. Prasyarat

Input bagi peringkat pengekodan melibatkan senarai perkara pada para 4.1.8.

4.2.2. Aliran Proses Kerja Peringkat Pengekodan

Peringkat ini memberi tumpuan kepada pengurusan kod sumber meliputi kawalan versi dan penjejakan perubahan kod sumber. Semakan kualiti kod akan dilaksanakan pada *pipeline* CI/CD. Rajah 4-10 memaparkan aliran proses kerja yang terlibat dalam peringkat pengekodan.



Rajah 4-10: Aliran Kerja Peringkat Pengekodan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 4-10.

a. Wujud GitLab *Feature Branch* [B1]

Branch digunakan dalam sistem pengurusan kod sumber untuk mengurus perubahan yang dibuat pada kod sumber. *Branch* juga memudahkan pembangun mengenal pasti perbezaan kod sumber untuk membuat pembetulan ralat dan menggabungkan kod sumber terkini selepas pengujian dilaksanakan. *Feature branch* merujuk kepada *branch* yang diwujudkan khusus untuk membangunkan ciri baru atau membuat perubahan pada produk.

Setiap pembangun perlu *check out* kod sumber terkini dari sistem pengurusan kod sumber pada komputer masing-masing. *Check out* merupakan proses mendapatkan kod sumber daripada sistem pengurusan kod sumber. Pembangun disarankan untuk mewujudkan satu *feature branch* berasingan. Ini bertujuan mengasingkan perubahan yang dilakukan daripada kod sumber utama.

b. *Commit* Kod Sumber [B2]

Pembangun melaksanakan *commit* ke atas perubahan yang telah dilakukan pada kod sumber. *Commit* adalah proses menyimpan perubahan kod sumber ke dalam *feature branch*. Aktiviti *commit* kod sumber ini boleh dilakukan beberapa kali apabila pembangun membuat perubahan pada kod sumber sebelum bersedia untuk *push* ke repositori berpusat.

c. *Push* Kod Sumber [B3]

Setelah pasukan pembangun selesai mengemas kini kod sumber, pembangun akan *push feature branch* yang mengandungi perubahan kod terkini ke repositori berpusat. *Push* adalah proses menghantar perubahan kod sumber setelah *commit* dilaksanakan.

d. Merge Request [B4]

Merge requests merujuk kepada proses penggabungan *branch* dari satu *branch* ke *branch* yang lain. Setelah pembangun *push* kod ke repositori berpusat, pembangun boleh membuat *merge request* untuk menggabungkan kod sumber ke *branch* utama. GitLab akan memulakan proses *pipeline* CI/CD secara automatik untuk menguji kod sumber dan memastikan bahawa perubahan yang dibuat tidak merosakkan fungsi produk. *Pipeline* CI/CD akan melalui siri pengujian awal bagi tujuan semakan kualiti kod.

e. Semak Kod Sumber [B5]

Setelah pengujian awal selesai, *merge request* akan dinilai dan disemak oleh pembangun lain atau pembangun yang mempunyai kepakaran dalam semakan kod sumber. Jika perubahan kod sumber tidak diluluskan, pembangun perlu membuat penambahbaikan pada kod sumber tersebut sehingga menepati kualiti semakan kod sumber.

f. Lulus Merge Request [B6]

Setelah perubahan kod sumber diperiksa dan dipersetujui, penilai oleh meluluskan *merge request* tersebut dengan mengklik butang *Merge* pada *merge request*. GitLab akan melakukan proses penggabungan kod sumber, mengesahkan *merge request* dan menghapuskan *feature branch* jika perlu. *Pipeline* CI/CD seterusnya akan melalui peringkat seterusnya bergantung kepada konfigurasi yang telah ditetapkan.

4.2.3. Pengurusan Kod Sumber

Pengurusan kod sumber melibatkan aktiviti pewujudan *feature branch* bagi setiap ahli pasukan pembangun berdasarkan *product backlog* atau *sprint backlog* yang telah ditetapkan bergantung kepada kesesuaian dan penggunaan pasukan pembangun. Selain daripada itu, aktiviti lain dalam pengurusan kod sumber ini adalah *merge request* daripada *feature branch* ke repositori berpusat bagi tujuan pengujian.

4.2.3.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pengurusan kod sumber adalah GitLab. Manakala *features* GitLab yang terlibat adalah seperti berikut:

a. *Project*

b. *Repository*

c. *Merge Request*

4.2.3.2. Ringkasan Tatacara Penggunaan *Tools*

a. Wujud GitLab *Feature Branch* [B1]

Pasukan pembangun akan menempatkan kod yang telah dikemas kini pada *feature branch* sebelum digabungkan ke *branch* utama. **Tatacara untuk mewujudkan *feature branch*** boleh dirujuk pada.

b. *Commit* Kod Sumber [B2] Lampiran C-1

Pembangun perlu *commit* perubahan kod sumber yang telah dibangunkan atau dikemas kini. *Integrated Development Environment* (IDE) yang bersesuaian seperti Visual Studio Code, Sublime, Eclipse, NetBean boleh digunakan sebagai platform pembangunan kod sumber. **Tatacara untuk mengemas kini kod** dengan menggunakan GitLab boleh dirujuk pada **Lampiran C-2**.

c. *Push* Kod Sumber [B3]

Setelah pengemaskinian kod selesai, pasukan pembangun boleh melaksanakan *commit* dan *push* perubahan kod tersebut daripada *feature branch* ke *branch* utama. **Tatacara untuk *commit* dan *push* kod sumber** boleh dirujuk pada **Lampiran C-3**.

d. Merge Request [B4]

Pasukan pembangun membuat *merge request* untuk menggabungkan kod yang telah diubahsuai ke *branch* utama. **Tatacara untuk *merge request*** boleh dirujuk pada **Lampiran C-4**.

e. Semak Kod Sumber [B5]

Semakan kod sumber (*Code review)* merupakan kaedah manual daripada penilai yang mempunyai kepakaran dalam menilai kualiti kod. GitLab menyediakan *features* atau ujian bagi membantu penilai untuk menilai perubahan kod sumber yang dibuat oleh pembangun. **Tatacara untuk melaksanakan *code review*** boleh dirujuk pada **Lampiran C-5**.

f. Lulus Merge Request [B6]

Setelah penilai menilai perubahan kod yang dibuat oleh pembangun, penilai tersebut boleh meluluskan permohonan tersebut. Sekiranya permohonan tidak diluluskan, penilai boleh memberikan komen pada ruangan komen yang disediakan. **Tatacara untuk meluluskan *merge request*** boleh dirujuk pada **Lampiran C-6.**

4.2.4. Konfigurasi *Pipeline* CI/CD

Pipeline CI/CD akan dikonfigurasikan pada peringkat ini berdasarkan kesesuaian sistem aplikasi. Konfigurasi *pipeline* CI/CD adalah dibuat berdasarkan perancangan pada para 4.1.5.

4.2.4.1. Penggunaan *Tools*

Tools yang akan digunakan semasa konfigurasi *pipeline* CI/CD adalah GitLab CI/CD.

4.2.4.2. Ringkasan Tatacara Penggunaan *Tools*

a. Konfigurasi *Pipeline* CI/CD Untuk Sistem Aplikasi Web

Rujuk Lampiran C-7 bagi contoh skrip `.gitlab-ci.yml` untuk *pipeline* CI/CD berdasarkan arkitektur yang diterangkan pada para 4.1.5.3.

b. Konfigurasi *Pipeline* CI/CD Untuk Sistem Aplikasi Mudah Alih

Rujuk Lampiran C-8 bagi contoh skrip `.gitlab-ci.yml` untuk *pipeline* CI/CD bagi sistem aplikasi mudah alih.

4.2.5. Semakan Kualiti Kod

Semakan kualiti kod bertujuan memastikan kod sumber dibina berasaskan standard pembangunan sistem aplikasi yang telah ditetapkan berdasarkan Jaminan Kualiti Perisian(SQA). Penerangan berkenaan SQA dan atribut kualiti perisian boleh didapati dalam buku panduan KRISA. Bagi memastikan SQA terjamin di bawah pelaksanaan DevOps, proses semakan awal pada *pipeline* CI/CD perlu dilaksanakan. Antara pengujian awal yang boleh dibuat adalah seperti berikut:

a. Pengujian Unit

Aktiviti pengujian bagi menilai unit terkecil (lowest level) pada sistem aplikasi seperti kod sumber, fungsi dan sebagainya.

b. Pengujian SAST

Aktiviti pengujian ini bagi menyemak *vulnerabilities* yang terdapat pada kod sumber tersebut.

c. Kualiti kod

Aktiviti pengujian atau analisis ini dibuat pada *pipeline* CI/CD bagi memastikan perubahan kod tersebut dapat meningkatkan prestasi sistem aplikasi tersebut.

Penerangan lebih lanjut pengujian awal ini akan diterangkan pada peringkat pengujian.

4.2.5.1. Penggunaan *Tools*

Skrip pengujian awal untuk aktiviti kawalan kualiti kod boleh dilakukan dengan menggunakan *pipeline* CI/CD sebelum memasuki ke peringkat pembangunan.

4.2.5.2. Ringkasan Tatacara Penggunaan *Tools*

Pengujian awal pada *pipeline* CI/CD adalah proses semakan awal secara automatik dengan menggunakan templat sedia ada. Tatacara penyediaan skrip pengujian bagi kawalan kualiti kod boleh dirujuk pada Lampiran C-8.

4.2.6. Serahan/Output

Peringkat pengekodan merupakan peringkat pihak pembangun membangun dan mengemas kini kod sumber. Kebiasaannya pihak pembangun akan membangunkan produk pada terminal masing-masing dan setelah selesai, pembangun akan *commit* dan *push* kod sumber tersebut ke *branch* utama. Siri pengujian awal seperti pengujian unit, pengujian SAS dan pengujian kod kualiti perlu dibuat sebelum pergi ke peringkat pembangunan.

4.3. PERINGKAT PEMBANGUNAN

Peringkat pembangunan merupakan proses pengubahsuaian fail dan aset lain dibawah tanggungjawab pasukan pembangun menjadi produk dalam bentuk akhir atau boleh digunakan. Pembinaan ini termasuk menyusun (compiling) fail kod sumber. Pada peringkat ini, setelah *merge request* diluluskan, kod sumber yang dibangunkan akan dibina. Peringkat pembangunan ini dikonfigurasikan dalam *pipeline* CI/CD. Penerangan berkaitan peringkat pembangunan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.4.

Peringkat Pembangunan terdapat tiga aktiviti utama iaitu:

a. *Compile Code*,

b. *Package Code* dan

c. Pembinaan Kod dan Imej *Container*.

4.3.1. Prasyarat

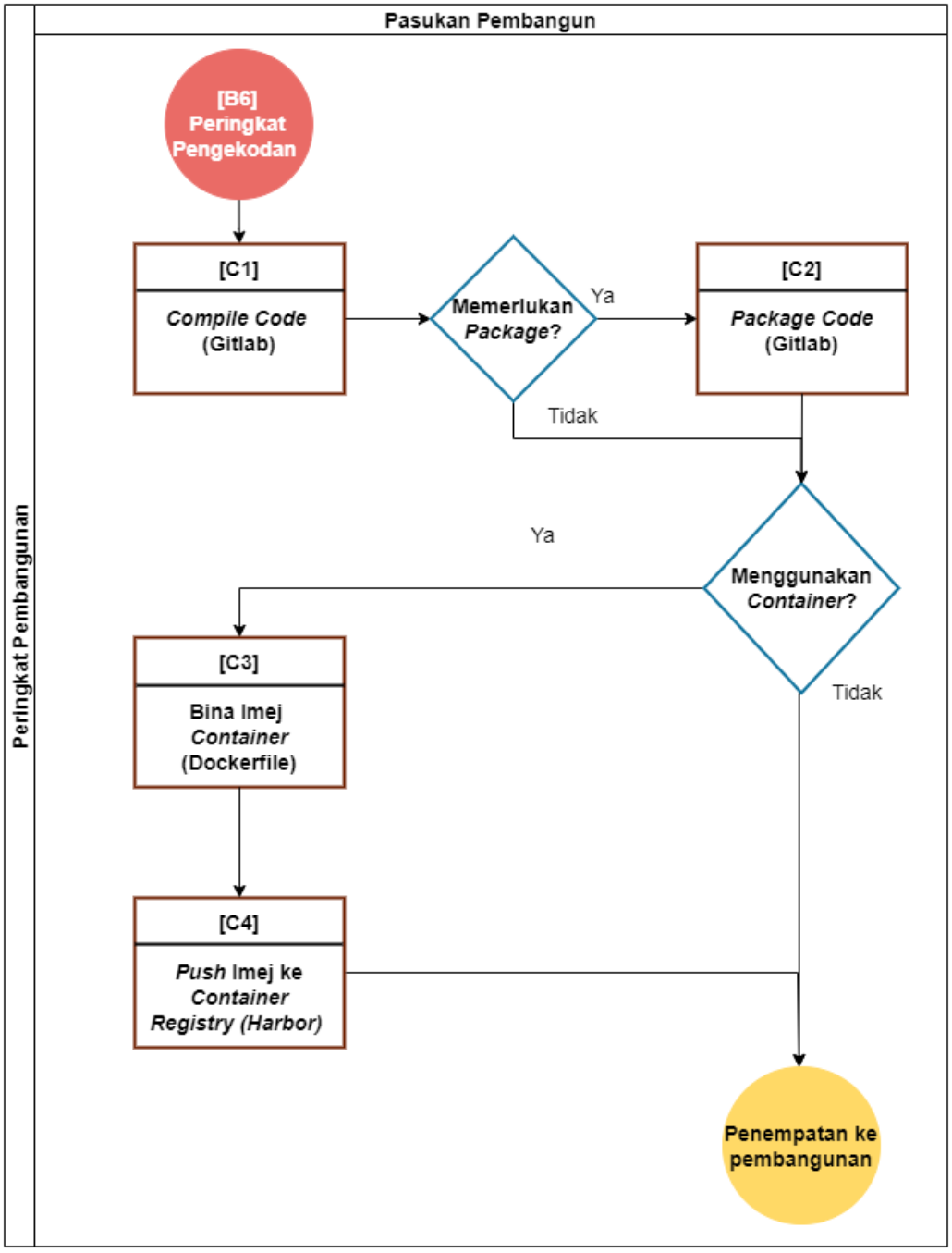
Untuk menjalankan aktiviti CI/CD berkaitan Docker *Container*, pasukan perlu:

a. Menkonfigurasi GitLab *runner* supaya menyokong arahan kerja berkaitan untuk menjalankan Docker *Container*.

b. Menetapkan pilihan *container* dengan menkonfigurasikan imej dalam fail `.gitlab-ci.yml`.

4.3.2. Aliran Proses Kerja Peringkat Pembangunan.

Proses pembangunan kod sumber akan bermula apabila penilai telah menjalankan verifikasi dan menggabungkan kod sumber ke repositori berpusat. GitLab CI akan memulakan proses membina kod berdasarkan konfigurasi dari `.gitlab-ci.yml` . Rajah 4-11 memaparkan aliran proses kerja yang terlibat dalam peringkat pembangunan.



Rajah 4-11: Aliran Kerja Peringkat Pembangunan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 4-11:

a. Aktiviti *compile code* akan dilaksanakan setelah *merge request* diluluskan.

b. Bagi kod sumber yang memerlukan fail pakej, aktiviti *package code* akan dilaksanakan.

c. Sekiranya produk menggunakan *container,* imej *container* akan dibina berdasarkan *dockerfile* yang telah ditetapkan.

d. Imej *container* yang terbina akan dihantar (push) ke *container registry*.

e. Produk yang selesai dibina akan ditempatkan pada persekitaran yang bersesuaian.

4.3.3. *Compile Code* [C1]

Compile code merupakan proses penyusunan dan pengubahsuaian kod sumber daripada *human-readable code* kepada *machine-executable code*. Proses ini melibatkan penggunaan *code compiler* bagi menganalisis dan menterjemahkan kod sumber kepada bentuk fail *executable* serta mengandungi *libraries* dan fail yang berkaitan. Fail ini boleh dipasang pada mesin atau komputer untuk menjalankan program tersebut.

4.3.3.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti *compile code* adalah GitLab CI/CD.

4.3.3.2. Ringkasan Tatacara Penggunaan *Tools – Compile Code*

Code compiler yang digunakan bergantung kepada bahasa pengaturcaraan kod sumber.

Lampiran D-1 menerangkan contoh skrip *code build* yang digunakan.

4.3.4. *Package Code* [C2]

Package code merujuk kepada satu atau lebih fail kod yang telah disusun dan dikumpulkan bersama bagi membentuk satu pakej fail (distributable format) yang sedia untuk diimport atau digunakan semula ke aplikasi pelayan di peringkat penempatan. *Package code* memudahkan pengurusan kod sumber kerana membolehkan pembangun memisahkan kod sumber menjadi modul yang berbeza dan memfokuskan pada setiap modul pada masa yang berbeza.

4.3.4.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti *package code* adalah GitLab CI/CD dan *Package Registry*.

4.3.4.2. Ringkasan Tatacara Penggunaan *Tools – Package Code*

Package code yang dibina akan disimpan di *Package Registry*. GitLab *Package Registry* adalah koleksi *package* yang membantu dalam pencarian, konfigurasi dan pemasangan pakej. Dengan menggunakan GitLab *Package Registry*, pasukan boleh menggunakan sebagai *private* atau *public registry* untuk mengurus pelbagai pakej yang disokong. Rujuk Lampiran D-2 bagi contoh penggunaan *Package Registry*.

4.3.5. Pembinaan Kod dan Imej *Container*

Pembinaan kod sumber dan imej *container* produk melibatkan proses pembangunan atau pembinaan produk daripada kod sumber kepada sistem yang boleh *execute* dan diuji. Pada peringkat ini, aktiviti pengintegrasian berterusan (CI) seperti menyusun kod sumber, memuat turun *dependencies*, membina imej *container* dan menghasilkan pakej perisian yang sedia untuk digunakan. Penggunaan *pipeline* CI/CD akan dikonfigurasi pada peringkat ini berdasarkan kesesuaian produk.

4.3.5.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pembinaan kod dan imej *container* adalah seperti berikut:

a. *Pipeline* CI/CD GitLab.

b. Docker.

c. Harbor.

4.3.5.2. Ringkasan Tatacara Penggunaan *Tools*

a. Bina Imej *Container* [C3]

GitLab CI/CD boleh digunakan dengan Docker untuk membina imej Docker. Sebelum menggunakan Docker *command* pada CI/CD *jobs*, GitLab *runner* perlu dikonfigurasikan untuk menyokong docker *command*. Rujuk Lampiran D-3 bagi contoh skrip untuk membina imej Docker.

b. *Push* Imej ke *Container Registry* [C4]

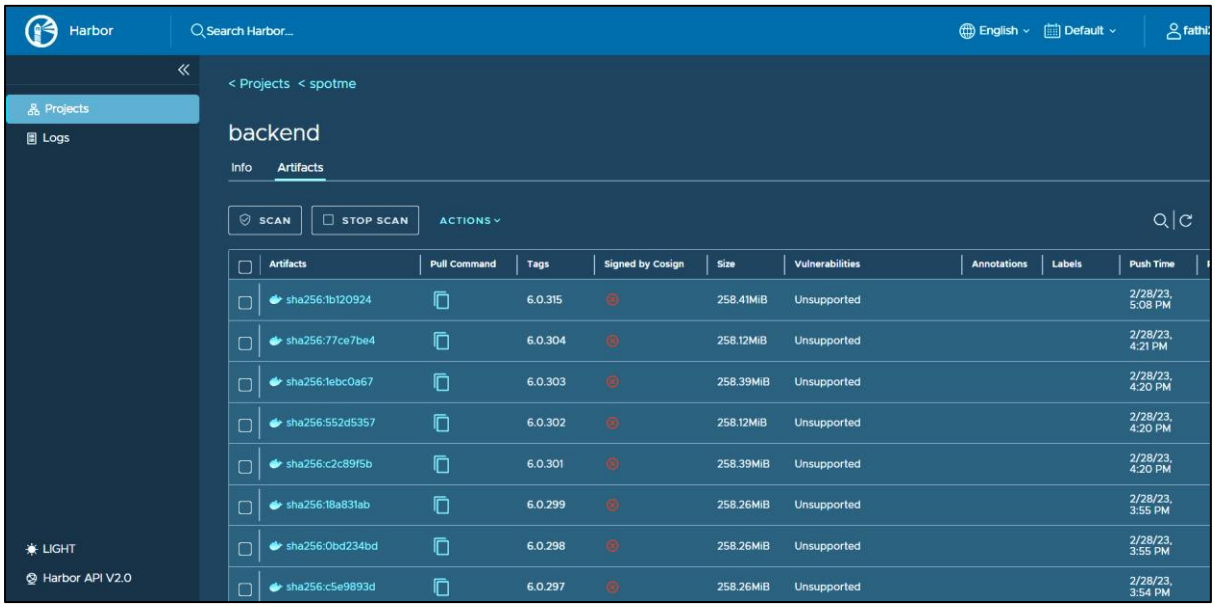
Setelah imej Docker dibina, imej tersebut perlu dihantar ke *container registry*. Rujuk Lampiran D-4 bagi contoh skrip untuk *push* imej Docker ke *container registry*.

4.3.6. Serahan/Output

Peringkat pembangunan merupakan peringkat membina sistem aplikasi daripada kod sumber kepada *executable* sistem yang boleh digunakan pada peringkat pengujian. Antara contoh perkara yang boleh dijadikan serahan pada peringkat ini adalah seperti berikut:

a. *Binary package* seperti fail JAR/WAR, atau

b. Imej *container* seperti yang dipaparkan pada Rajah 4-12.



Rajah 4-12: Paparan Kod Imej pada Harbor

4.4. PERINGKAT PENGUJIAN

Pengujian merupakan aktiviti verifikasi yang dilakukan terhadap komponen atau sistem aplikasi untuk memastikan sistem dibangunkan berdasarkan kepada spesifikasi keperluan dan reka bentuk sistem. Jenis-jenis pengujian yang dijalankan adalah pengujian keperluan fungsian, pengujian keperluan bukan fungsian serta verifikasi terhadap ralat yang telah dibaiki. Penerangan berkaitan peringkat pengujian boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.5.

4.4.1. Prasyarat

Untuk melaksanakan pengujian, pasukan perlu melaksanakan aktiviti berikut:

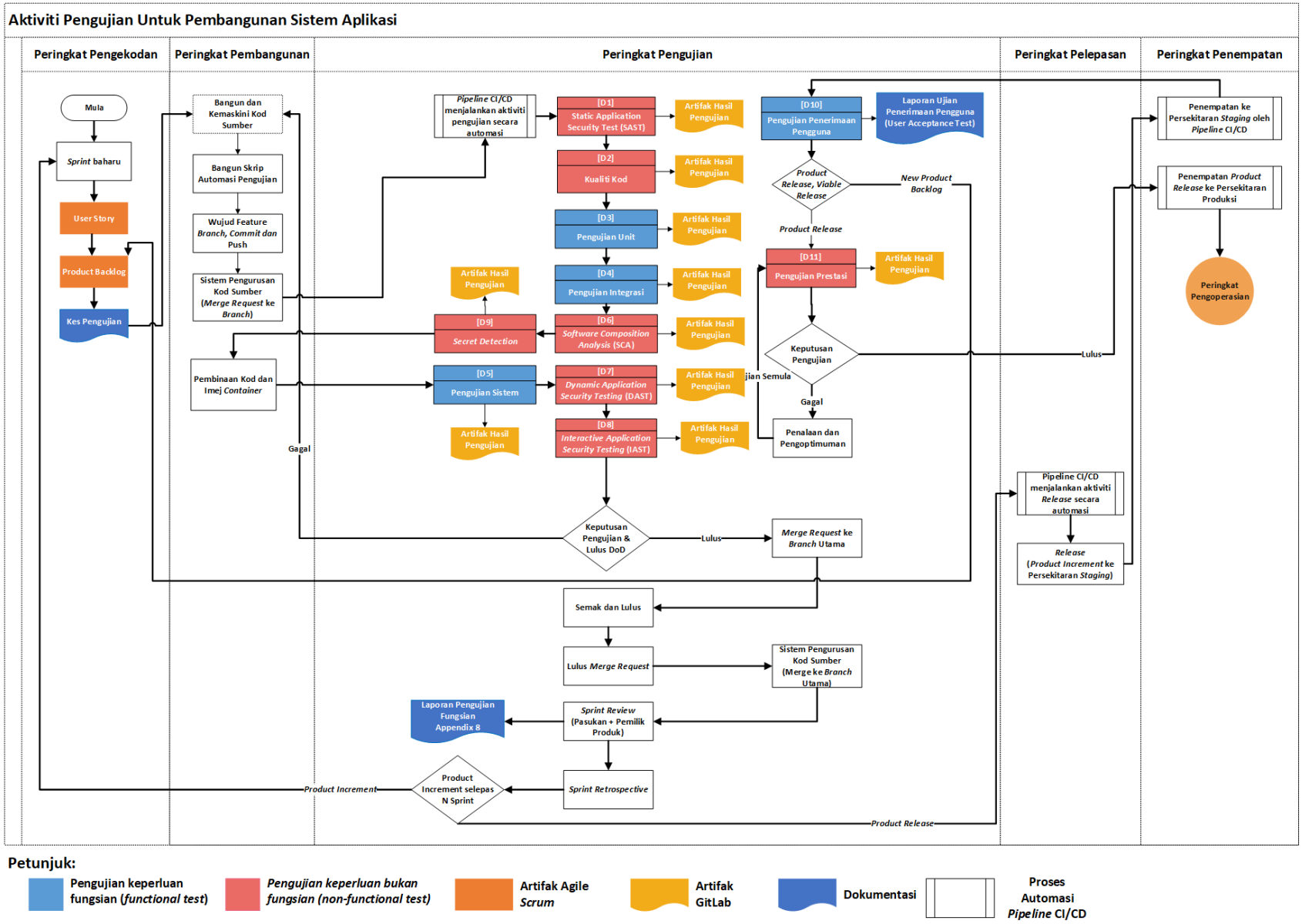
a. Mengkonfigurasi GitLab *Runner* untuk menyokong arahan kerja berkaitan proses pengujian.

b. Menyediakan akses kepada repositori kod sumber oleh *pipeline* CICD.

c. Menyediakan skrip pengujian berdasarkan bahasa pengaturcaraan yang digunakan.

4.4.2. Aliran Proses Kerja Peringkat Pengujian

Aliran proses aktiviti pengujian berdasarkan pendekatan pembangunan *Agile Scrum* pada Rajah 4-13 berikut telah diadaptasikan kepada proses pembangunan sistem aplikasi di sektor awam.



Rajah 4-13: Aktiviti Pengujian *Agile Scrum* untuk Pembangunan Sistem Aplikasi

Keterangan setiap proses dalam aliran aktiviti pengujian *Agile Scrum* untuk pembangunan sistem aplikasi adalah seperti yang berikut:

a. Repositori kod sumber akan menerima *merge request* dari pembangun seterusnya menjalankan *pipeline* CI/CD.

b. *Pipeline* CI/CD secara automatik akan menjalankan aktiviti pengujian ke atas kod sumber seperti yang telah dikonfigurasi dalam fail `.gitlab-ci.yml`.

c. Aktiviti pengujian SAST, kualiti kod, pengujian unit, pengujian integrasi, pengujian SCA, pengujian *secret detection*, pengujian sistem, pengujian DAST dan pengujian IAST akan dijalankan serta diuji secara automasi berdasarkan konfigurasi *pipeline* CI/CD.

d. Setelah aktiviti pengujian selesai, hasil pengujian dalam bentuk artifak GitLab akan dijana secara automasi.

e. Jika pengujian gagal, pembangun akan dimaklumkan untuk mengemas kini kod sumber.

f. Jika pengujian berjaya, *merge request* akan diluluskan oleh wakil pasukan untuk menyatukan kod sumber kepada *branch* utama.

g. Berdasarkan *Release Planning,* wakil pasukan akan memutuskan untuk melakukan *Product Release* atau meneruskan ke *Sprint* seterusnya.

h. Pengujian penerimaan pengguna akan dilaksanakan di mana pengguna akan mengakses sistem aplikasi yang telah di pasang pada persekitaran *staging* untuk melaksanakan pengujian.

i. Setelah pengujian penerimaan pengguna lulus, pengujian prestasi akan dilaksanakan. Penalaan dan pengoptimuman akan dilakukan untuk memastikan sistem aplikasi mencapai hasil yang ditetapkan.

j. Sistem aplikasi akan dipasang pada persekitaran produksi setelah lulus pengujian prestasi.

4.4.3. Pengujian Keperluan Fungsian (Functional Test)

4.4.3.1. Pengujian Unit [D3]

Jadual 4-3 menerangkan tatacara pelaksanaan pengujian unit.

Jadual 4-3: Tatacara Pelaksanaan Pengujian Unit

Perkara	Keterangan
Entry Criteria	i. Keperluan fungsian telah dipersetujui dan disemak di antara pasukan pembangun dan pemilik produk. ii. Kod sumber telah dibangunkan dan sedia untuk diuji. iii. Skrip automasi pengujian unit telah dibangunkan. iv. Persekitaran pengujian telah tersedia
Exit Criteria	i. Skrip automasi pengujian integrasi telah dijalankan dan diuji oleh *pipeline* CI/CD seperti yang telah ditetapkan. ii. Ralat telah berjaya diperbaiki dan diselesaikan iii. Pengujian unit telah memenuhi kriteria penerimaan berdasarkan *acceptance criteria* dalam *Definition of Done*. iv. Keputusan kes pengujian unit tersedia dalam bentuk artifak GitLab.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian akan dijalankan di persekitaran pembangunan dan persekitaran repositori kod sumber. ii. *Tools* pengujian yang digunakan: a. IDE digunakan sebagai platform pengekodan dan pembangunan sistem aplikasi. b. *Framework* pengujian unit diintegrasikan bersama IDE. Digunakan untuk membina skrip automasi pengujian unit.
Pelaksanaan Pengujian	i. Kod sumber dibangunkan atas platform IDE ii. Kes ujian unit dibangunkan berdasarkan *User Story* dan *Product Backlog*, iii. Skrip pengujian unit dibangunkan dengan bantuan *framework* pengujian unit.

	iv. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. v. *Pipeline* CI/CD akan menjalan pengujian unit secara automasi.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – Pengujian Unit

Tatacara pengujian unit boleh dirujuk pada **Lampiran E-1**.

4.4.3.2. Pengujian Integrasi [D4]

Jadual 4-4 menerangkan tatacara pelaksanaan pengujian integrasi.

Jadual 4-4: Tatacara Pelaksanaan Pengujian Integrasi

Perkara	Keterangan
Entry Criteria	i. Keperluan fungsian telah dipersetujui dan disemak di antara pasukan pembangun dan pemilik produk. ii. Pengujian unit telah dilaksanakan dan lulus. iii. Ralat yang dilaporkan telah diperbaiki dan diselesaikan. iv. Skrip automasi pengujian integrasi telah dibangunkan.
Exit Criteria	i. Skrip automasi pengujian integrasi telah dijalankan dan diuji oleh *pipeline* CI/CD seperti yang telah ditetapkan. ii. Ralat telah berjaya diperbaiki dan diselesaikan. iii. Pengujian integrasi telah memenuhi kriteria penerimaan berdasarkan *acceptance criteria* dalam *Defination of Done*. iv. Keputusan kes pengujian integrasi tersedia dalam bentuk artifak GitLab.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian akan dijalankan di persekitaran repositori kod sumber. ii. *Tools* pengujian yang digunakan: a. IDE digunakan sebagai platform pengekodan dan pembangunan sistem aplikasi. b. *Framework* pengujian unit (PHPUnit) dan diintegrasikan bersama IDE. Digunakan untuk menbina skrip kod pengujian integrasi.

Perkara	Keterangan
Pelaksanaan Pengujian	i. Skrip pengujian integrasi dibangunkan dengan bantuan *Framework* pengujian, PHPUnit. ii. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. iii. *Pipeline* CI/CD akan menjalan pengujian integrasi secara automasi. iv. Laporan hasil pengujian unit akan dijana oleh GitLab.
Pengujian Semula (Regression Testing)	i. Mengenal pasti skrip kes pengujian yang terlibat pada *sprint* sebelum untuk digabung dan diuji semula pada *sprint* semasa. ii. *Pipeline* CICD secara automasi akan menjalankan pengujian semula berdasarkan skrip pengujian yang telah digabungkan. iii. Mengenal pasti ralat baharu yang mungkin timbul apabila pasukan pembangunan membetulkan ralat yang telah sedia ada. iv. Ketika pengujian semula, penguji akan memastikan perkara berikut: a. Penambahan fungsi baharu pada *sprint* semasa tidak menimbulkan isu baharu. b. Isu sedia ada diselesaikan. v. Tiada isu baharu yang timbul kesan dari pembaikan isu sedia ada.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – Pengujian Integrasi

Pengujian integrasi dilaksanakan menggunakan dua kaedah berikut.

a. Pengujian integrasi di antara komponen-komponen menggunakan skrip pengujian integrasi

b. Pengujian integrasi *API* menggunakan Newman

Tatacara pengujian integrasi boleh dirujuk pada **Lampiran E-2**.

4.4.3.3. Pengujian Sistem [D5]

Jadual 4-5 menerangkan tatacara pelaksanaan pengujian sistem.

Jadual 4-5: Tatacara Pelaksanaan Pengujian Sistem

Perkara	Keterangan
Entry Criteria	i. Pengujian integrasi telah dilaksanakan dan lulus. ii. Ralat yang dilaporkan dalam pengujian integrasi telah diselesaikan. iii. Kes pengujian seperti di Lampiran B-18: Templat Kes Pengujian telah disediakan dan disemak di antara pasukan pembangun dan pemilik produk. iv. Skrip automasi pengujian sistem telah dibangunkan. v. Konfigurasi skrip pengujian sistem telah dipasang dalam *pipeline* CI/CD. vi. Imej Docker telah tersedia pada *Container Registry*.
Exit Criteria	i. Skrip automasi pengujian sistem telah dijalankan dan diuji oleh *pipeline* CI/CD seperti yang telah ditetapkan. ii. Ralat telah berjaya diperbaiki dan diselesaikan. iii. Pengujian sistem telah memenuhi kriteria penerimaan berdasarkan *acceptance criteria* dalam *Definition of Done*. iv. Keputusan kes pengujian unit tersedia dalam bentuk artifak GitLab.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian dijalankan di persekitaran *staging*. ii. *Tools* pengujian yang digunakan: a. Selenium *Framework*: digunakan menjana dan mencipta skrip kod pengujian berdasarkan bahasa pengaturcaraan yang digunakan. b. Selenium *Runner*: digunakan untuk untuk mengautomasikan ujian aplikasi web pada pipeline CICD secara *headless browser*.
Pelaksanaan Pengujian	i. *Pipeline* CICD secara automasi akan menjalankan skrip Selenium untuk melaksanakan pengujian fungsian. ii. Selenium *Runner* akan mengakses imej docker yang telah di *deploy* pada Docker *Container*. iii. Pengujian fungsian secara antara muka pengguna akan dijalankan secara automasi oleh Selenium *Runner* berdasarkan skrip kes ujian secara *headless browser*.

Perkara	Keterangan
	iv. Hasil pengujian akan dipaparkan pada log *pipeline* CICD.
Pengujian Semula (Regression Testing)	i. Mengenal pasti skrip kes pengujian yang terlibat pada *sprint* sebelum untuk digabung dan diuji semula pada *sprint* semasa. ii. *Pipeline* CICD secara automasi akan menjalankan pengujian semula berdasarkan skrip pengujian yang telah digabungkan. iii. Mengenal pasti ralat baharu yang mungkin timbul apabila pasukan pembangunan membetulkan ralat yang telah sedia ada. iv. Ketika pengujian semula, penguji akan memastikan perkara berikut: a. Penambahan fungsi baharu pada *sprint* semasa tidak menimbulkan isu baharu. b. Isu sedia ada diselesaikan. c. Tiada isu baharu yang timbul kesan dari pembaikan isu sedia ada.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – Pengujian Sistem

Selenium *Runner* yang dipasang pada *pipeline* CID/CD akan mengakses aplikasi yang telah dipasang pada Docker *Container*. Selenium *Runner* akan melaksanakan pengujian secara antara muka pengguna dengan memasukkan input data secara automasi, melaksanakan kes ujian berdasarkan skrip dan mengeluarkan output pengujian.

Tatacara pengujian sistem boleh dirujuk pada **Lampiran E-3**.

4.4.3.4. Pengujian Penerimaan Pengguna (User Acceptance Test (UAT)) [D6]

Pengujian Penerimaan Pengguna dilaksanakan untuk mendapatkan kelulusan penerimaan pengguna mengenai kebolehfungsian dan kecekapan sistem aplikasi yang dibangunkan dalam mengendalikan proses bisnes. Kes pengujian dibangunkan berdasarkan Lampiran E-4.

Jadual 4-6 menerangkan tatacara pelaksanaan pengujian penerimaan pengguna.

Jadual 4-6: Tatacara Pelaksanaan Pengujian Penerimaan Pengguna

Perkara	Keterangan
Entry Criteria	i. Pengujian sistem telah dilaksanakan dan lulus. ii. Ralat yang dilaporkan dalam sebelumnya telah diselesaikan. iii. Laporan Ujian Penerimaan Pengguna telah disediakan dan disemak di antara pasukan pembangun dan pemilik produk. iv. Persekitaran *staging* telah disediakan. v. Sistem aplikasi telah dipasang pada persekitaran *staging*.
Exit Criteria	i. Kes pengujian penerimaan pengguna telah dilaksanakan 100%. ii. Tiada ralat dengan tahap kritikal tinggi. iii. Pengesahan keputusan pengujian akan didokumenkan dalam Laporan Ujian Penerimaan Pengguna.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian akan dijalankan di persekitaran *staging*. ii. *Tools* pengujian tidak digunakan kerana ujian dijalankan secara manual.
Pelaksanaan Pengujian	i. Melaksanakan pengujian pengesahan dan penerimaan oleh pemilik produk. ii. Menganalisis dan menyemak hasil pengujian. iii. Melaksanakan penambahbaikan oleh pembangun. iv. Melaksanakan pengujian semula.

Perkara	Keterangan
	v. Keputusan pengujian didokumenkan dalam Laporan Ujian Penerimaan Pengguna.
Laporan	Laporan Ujian Penerimaan Pengguna

a. Ringkasan Tatacara Penggunaan *Tools* – Pengujian Penerimaan Pengguna (UAT)

Pengujian penerimaan pengguna dilaksanakan secara manual. Pengguna akan mengakses sistem aplikasi melalui browser dan melaksanakan pengujian fungsian berdasarkan kes ujian yang telah ditetapkan. Tatacara penggunaan tools dibawah adalah untuk mengeksport senarai issues ke format fail csv untuk penyediaan kes ujian. Senarai issues yang juga dikenali sebagai product backlog akan dipindahkan ke dalam templat kes ujian seperti **Lampiran E-4**.

4.4.4. Pengujian Keperluan Bukan Fungsian (Non-Functional Test)

4.4.4.1. *Static Application Security Test* (SAST) [D1]

Jadual 4-7 menerangkan tatacara pelaksanaan pengujian SAST.

Jadual 4-7: Tatacara Pelaksanaan Pengujian SAST

Perkara	Keterangan
Entry Criteria	i. Kod sumber telah dibangunkan dan sedia untuk diuji. ii. Pemasangan dan konfigurasi *tools* pengujian SAST pada pipeline CI/CD yang menyokong bahasa pengaturcaraan kod sumber. iii. Persekitaran pengujian telah tersedia.
Exit Criteria	Keputusan analisis keatas kod sumber bebas dari ralat tahap berikut berdasarkan *severity level* dari SAST. i. Tahap *critical*

Perkara	Keterangan
	ii. Tahap *high* iii. Tahap *medium*
Persekitaran dan *Tools* Pengujian	i. Pengujian SAST dijalankan di persekitaran repositori kod sumber. ii. *Tools* pengujian menggunakan p*ipeline* CI/CD dengan konfigurasi skrip SAST.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. ii. *Pipeline* CI akan mengakses kod sumber dan menjalankan pengujian SAST secara automasi. iii. Menganalisis dan menyemak hasil pengujian. *Tools* pengujian akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber. iv. Melaksanakan penambahbaikan oleh pembangun. v. Melaksanakan pengujian semula.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – *Static Application Security Test* (SAST)

Tatacara pengujian unit boleh dirujuk pada **Lampiran E-7**.

4.4.4.2. Kualiti Kod [D2]

Kualiti kod akan menganalisis kod sumber dari segi kualiti dan kerumitan struktur pengekodan. Jadual 4-8 menerangkan tatacara pelaksanaan kawalan kualiti kod.

Jadual 4-8: Tatacara Pelaksanaan Kawalan Kualiti Kod

Perkara	Keterangan
Entry Criteria	i. Kod sumber telah dibangunkan dan sedia untuk diuji. ii. Pemasangan dan konfigurasi *tools* pengujian kualiti kod pada *pipeline* CI/CD yang menyokong bahasa pengaturcaraan kod sumber. iii. Persekitaran pengujian telah tersedia.
Exit Criteria	Kod sumber bebas daripada ralat tahap berikut berdasarkan rujukan tahap kritikal dari dokumentasi GitLab. i. Tahap 1 – *blocker.* Ralat yang memberi kesan yang besar kepada fungsi utama, data atau operasi pengguna yang tiada penyelesaian. ii. Tahap 2 – *critical.* Ralat tetapi dengan penyelesaian yang rumit dan sukar dilaksanakan. iii. Tahap 3 – *major.* Ralat tetapi mempunyai penyelesaian dan boleh dilaksanakan.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian akan dijalankan di persekitaran pembangunan dan persekitaran repositori kod sumber. ii. *Tools* pengujian menggunakan Code Climate yang dipasang pada *pipeline* CI/CD.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. ii. *Pipeline* CI akan menjalankan pengujian kualiti kod secara automasi. iii. Penggunaan *tools* untuk menganalisis kod sumber. *Tools* pengujian juga akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber. iv. Pembangun melaksanakan penambahbaikan dan pengujian semula.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – Kawalan Kualiti Kod

Tatacara pengujian unit boleh dirujuk pada **Lampiran E-6**.

4.4.4.3. *Software Composition Analysis* (SCA) [D6]

Jadual 4-9 menerangkan tatacara pelaksanaan pengujian SCA.

Jadual 4-9: Tatacara Pelaksanaan Pengujian SCA

Perkara	Keterangan
Entry Criteria	i. Kod sumber telah dibangunkan dan sedia untuk diuji. ii. Pemasangan dan konfigurasi *tools* pengujian SCA pada pipeline CI/CD yang menyokong bahasa pengaturcaraan kod sumber. iii. Persekitaran pengujian telah tersedia. iv. Semua komponen perisian yang digunakan dalam projek harus dikenal pasti seperti *libraries*, *framework*, dan *dependencies* pihak ketiga. Masukkan tempalte checklist v. Penetapan garis panduan mengenai komponen yang diterima dan tidak diterima seperti lesen, kelemahan keselamatan, dan keperluan pematuhan keselamatan lain.
Exit Criteria	Keputusan analisis keatas kod sumber bebas dari ralat tahap berikut berdasarkan *severity level* dari SCA. i. Tahap *critical* ii. Tahap *high* iii. Tahap *medium*
Persekitaran dan *Tools* Pengujian	i. Pengujian SCA dijalankan di persekitaran repositori kod sumber. ii. *Tools* pengujian menggunakan p*ipeline* CI/CD dengan konfigurasi skrip SCA.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. ii. *Pipeline* CI akan menjalankan pengujian SCA secara automasi. iii. Menganalisis dan menyemak hasil pengujian. *Tools* pengujian akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber. iv. Melaksanakan penambahbaikan oleh pembangun.

Perkara	Keterangan
	v. Melaksanakan pengujian semula.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools* – *Software Composition Analysis* (SCA)

Pengujian SCA dilaksanakan secara automasi iaitu menggunakan *pipeline* CI/CD. Bagi tujuan kajian kes, persekitaran telah dikonfigurasi menggunakan imej *docker* OWASP Dependency-Check.

Tatacara konfigurasi pengujian SCA boleh dirujuk pada **Lampiran E-8**.

4.4.4.4. *Dynamic Application Security Testing* (DAST) [D7]

Jadual 4-10 menerangkan tatacara pelaksanaan pengujian DAST

Jadual 4-10: Tatacara Pelaksanaan Pengujian DAST

Perkara	Keterangan
Entry Criteria	i. Pemasangan dan konfigurasi *tools* pengujian DAST pada pipeline CI/CD yang menyokong bahasa pengaturcaraan kod sumber seperti OWASP ZAP, Burp Suite, dan Acunetix. ii. Imej *Docker* telah tersedia pada *Container Registry*.
Exit Criteria	Keputusan analisis keatas kod sumber bebas dari ralat tahap berikut berdasarkan *severity level* dari DAST. i. Tahap *critical* ii. Tahap *high* iii. Tahap *medium*

Perkara	Keterangan
Persekitaran dan *Tools* Pengujian	i. Pengujian DAST dijalankan di persekitaran repositori kod sumber. ii. *Tools* pengujian menggunakan p*ipeline* CI/CD dengan konfigurasi skrip DAST.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. ii. *Pipeline* CICD akan menjalankan pengujian DAST secara automasi dengan mengakses imej *docker* sistem aplikasi yang telah tersedia pada Docker *Container*. iii. Pengujian bukan fungsian dilaksanakan dengan mengimbas dan mensimulasikan vektor serangan seperti *cross-site scripting* (XSS), *SQL injection* (SQLi), and *cross-site request forgery* (CSRF) terhadap sistem aplikasi dan API. iv. Hasil pengujian akan dipaparkan pada log *pipeline* CICD.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools – Dynamic Application Security Testing* (DAST)

Pengujian DAST dilaksanakan secara automasi iaitu menggunakan *pipeline* CI/CD. Bagi tujuan kajian kes, persekitaran telah dikonfigurasi menggunakan imej *docker* OWASP ZAP (Zed Attack Proxy).

Tatacara konfigurasi pengujian DAST boleh dirujuk pada **Lampiran E-9**.

4.4.4.5. *Interactive Application Security Testing* (IAST) [D8]

Jadual 4-11 menerangkan tatacara pelaksanaan pengujian IAST

Jadual 4-11: Tatacara Pelaksanaan Pengujian IAST

Perkara	Keterangan
Entry Criteria	i. Kod sumber telah dibangunkan dan sedia untuk diuji. ii. Pemasangan dan konfigurasi *tools* pengujian IAST pada pipeline CI/CD yang menyokong bahasa pengaturcaraan kod sumber. iii. Persekitaran pengujian telah tersedia.
Exit Criteria	Keputusan analisis keatas kod sumber bebas dari ralat tahap berikut berdasarkan *severity level* dari IAST. i. Tahap *critical* ii. Tahap *high* iii. Tahap *medium*
Persekitaran dan *Tools* Pengujian	i. Pengujian IAST dijalankan di persekitaran repositori kod sumber. ii. *Tools* pengujian menggunakan p*ipeline* CI/CD dengan konfigurasi skrip IAST.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab. ii. *Pipeline* CI akan menjalankan pengujian IAST secara automasi. iii. Menganalisis dan menyemak hasil pengujian. *Tools* pengujian akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber. iv. Melaksanakan penambahbaikan oleh pembangun. v. Melaksanakan pengujian semula.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan Tools – *Interactive Application Security Testing* (IAST)

Pengujian IAST memerlukan *tools* komersial seperti Contrast Security, Vera*code* dan Synopsys Seeker untuk membolehkan pengujian penuh dilaksanakan. Pengujian SAST, DAST, SCA dan pengimbasan imej container boleh dilaksanakan sebagai alternatif kepada pengujian IAST menggunakan *tools* sumber terbuka.

4.4.4.6. *Secret Detection* [D9]

Jadual 4-12 menerangkan tatacara pelaksanaan pengujian *secret detection.*

Jadual 4-12: Tatacara Pelaksanaan Pengujian *Secret Detection*.

Perkara	Keterangan
Entry Criteria	i. Kod sumber telah dibangunkan dan sedia untuk diuji. ii. Penetapan format dan senarai semak peraturan pengesanakan seperti kata laluan, API dan token. iii. Pemasangan dan konfigurasi *tools* pengujian *secret detection* pada *pipeline* CI/CD yang menyokong bahasa pengaturcaraan kod sumber. iv. Persekitaran pengujian telah tersedia.
Exit Criteria	Keputusan analisis keatas kod sumber bebas dari ralat tahap berikut berdasarkan *severity level* dari *secret detection*. i. Tahap *critical* ii. Tahap *high* iii. Tahap *medium*
Persekitaran dan *Tools* Pengujian	iii. Pengujian *secret detection* dijalankan di persekitaran repositori kod sumber. iv. *Tools* pengujian menggunakan p*ipeline* CI/CD dengan konfigurasi skrip *secret detection*.
Pelaksanaan Pengujian	i. Pembangun melaksanakan *commit* dan *push* kod sumber dari IDE ke repositori GitLab.

Perkara	Keterangan
	ii. *Pipeline* CI akan menjalankan pengujian *secret detection* secara automasi dengan mengklon repositori untuk mengakses versi terkini kod sumber. iii. Tugasan akan mengimbas kod sumber untuk mengenal pasti corak yang sepadan dengan format rahsia berdasarkan peraturan pengesanan yang telah ditetapkan iv. Menganalisis dan menyemak hasil pengujian. *Tools* pengujian akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber. v. Melaksanakan penambahbaikan oleh pembangun. vi. Melaksanakan pengujian semula.
Laporan	Hasil pengujian dijana secara automatik oleh *pipeline* CI/CD dan disimpan dalam bentuk artifak.

a. Ringkasan Tatacara Penggunaan *Tools – Secret Detection*

Pengujian secret detection dilaksanakan secara automasi iaitu menggunakan *pipeline* CI/CD.

Tatacara konfigurasi pengujian *secret detection* boleh dirujuk pada **Lampiran E-10**.

4.4.4.7. Pengujian Prestasi [D7]

Jadual 4-13 menerangkan tatacara pelaksanaan pengujian prestasi.

Jadual 4-13: Tatacara Pelaksanaan Pengujian Prestasi

Perkara	Keterangan
Entry Criteria	i. Pengujian Penerimaan Pengguna telah selesai dilaksanakan. ii. Persekitaran pengujian untuk produksi telah disediakan. iii. Menetapkan strategi dan tatacara pengujian. Rujuk Bab 6.6, Langkah 5: Buku Kejuruteraan Sistem Aplikasi Sektor Awam (KRISA). iv. Menetapkan prosedur pengujian telah dipersetujui oleh pasukan pembangun dan pemilik produk.
Exit Criteria	i. Semua transaksi bisnes yang dipersetujui telah direkodkan dan diuji. ii. Purata masa tindak balas adalah sama atau kurang dari yang telah ditetapkan. iii. *Server Utilization* (CPU dan *memory*) adalah tidak melebih tahap yang ditetapkan. iv. Penyediaan laporan pengujian prestasi.
Persekitaran dan *Tools* Pengujian	i. Persekitaran pengujian akan dijalankan di persekitaran produksi. ii. *Tools* pengujian dilaksanakan menggunakan *tools* K6 secara automasi oleh *pipeline* CI/CD atau secara manual dengan bantuan *tools* pengujian prestasi.
Pelaksanaan Pengujian	i. Pasukan menjalankan *pipeline* CI/CD akan menjalankan pengujian prestasi. ii. Menganalisis dan menyemak hasil pengujian berdasarkan kriteria penerimaan pengujian. iii. Mengenal pasti dan melaksanakan penalaan dan pengoptimuman. iv. Melaksanakan pengujian semula.
Laporan	Laporan pengujian prestasi dijana secara automatik oleh *pipeline* CI dalam bentuk artifak GitLab.

a. Ringkasan Tatacara Penggunaan *Tools* – Pengujian Prestasi

Tatacara pengujian unit boleh dirujuk pada Lampiran E-11.

4.4.5. Serahan/Output

Serahan untuk peringkat pengujian adalah seperti berikut:

a. Artifak GitLab hasil Pengujian SAST
b. Artifak GitLab hasil Pengujian Kualiti Kod
c. Artifak GitLab hasil Pengujian Unit
d. Artifak GitLab hasil Pengujian Integrasi
e. Artifak GitLab hasil Pengujian Sistem
f. Artifak GitLab hasil Pengujian Prestasi
g. Laporan Ujian Penerimaan Pengguna

4.5. PERINGKAT PELEPASAN

Peringkat pelepasan adalah proses untuk mengeluarkan versi baru produk kepada persekitaran penempatan. Pelepasan perubahan kod ke persekitaran penempatan boleh dilaksanakan secara automatik dalam *pipeline* CI/CD atau manual. Pelepasan produk ke persekitaran produksi telah ditetapkan pada GitLab *Milestones* berdasarkan hasil persetujuan bersama pemilik produk dan ahli pasukan DevOps. Pada peringkat pelepasan, penetapan versi perlu dilaksanakan untuk digunakan pada peringkat penempatan. Penerangan berkaitan peringkat pelepasan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.6.

Peringkat pelepasan melibatkan tiga persekitaran utama iaitu:

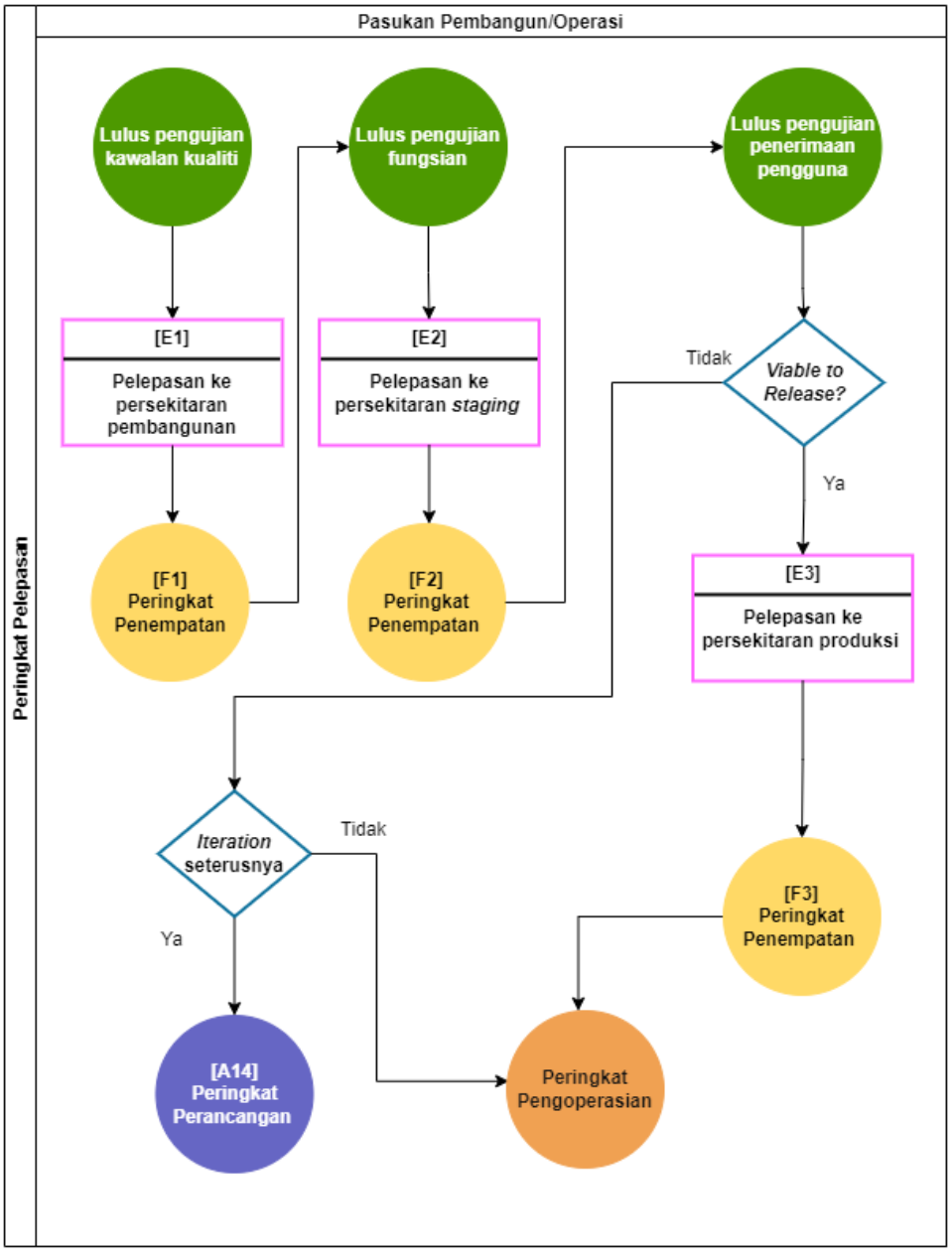
a. Pelepasan ke persekitaran pembangunan,
b. Pelepasan ke persekitaran *staging* dan
c. Pelepasan ke persekitaran produksi.

4.5.1. Prasyarat

Pada peringkat ini, setiap perubahan kod telah selesai melepasi aktiviti pengujian secara automasi atau manual berdasarkan pada peringkat pengujian dan laporan pengujian fungsian (Lampiran E-4) telah disemak dan disahkan.

4.5.2. Aliran Proses Kerja Peringkat Pelepasan

Pelepasan produk bergantung kepada persekitaran yang ingin dilepaskan. Rajah 4-14 memaparkan aliran proses kerja yang terlibat dalam peringkat pelepasan.



Rajah 4-14: Aliran Kerja Peringkat Pelepasan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 4-14:

a. Setelah pengujian kawalan kualiti lulus, produk akan dilepaskan ke persekitaran pembangunan.

b. Produk akan dilepaskan ke persekitaran *staging* apabila telah lulus pengujian fungsian.

c. Hanya produk yang *viable to release* dan lulus pengujian penerimaan pengguna akan dilepaskan ke persekitaran produksi.

4.5.3. Pengimbasan Imej Container

Pengimbasan imej *docker* pada *container registry* dilaksanakan untuk mengenal pasti sebarang isu berkaitan keselamatan sebelum pelepasan sistem aplikasi dikeluarkan. Rujuk Lampiran F-1 bagi contoh skrip untuk pengimbasan imej *docker* pada *container registry*.

4.5.4. Pelepasan Sistem Aplikasi

Pelepasan sistem aplikasi boleh dikeluarkan mengikut versi persekitaran iaitu versi pembangunan, *staging* atau produksi. Versi pembangunan dan *staging* biasanya dilepaskan untuk tujuan pengujian dan mendapatkan maklum balas pengguna sebelum dilancarkan secara rasmi. Versi produksi pula merupakan versi rasmi yang dilepaskan setelah sistem aplikasi telah diuji dengan baik dan dianggap sedia untuk digunakan oleh pengguna. Pelepasan sistem aplikasi boleh dijalankan secara berterusan pada setiap *sprint* atau dalam jadual yang telah ditentukan contohnya selepas *sprint* ketiga.

4.5.4.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pengurusan kod sumber adalah GitLab. Manakala *features* GitLab yang terlibat adalah seperti berikut:

a. *Tags*.

b. *Releases*.

c. *Pipeline* CI/CD.

4.5.4.2. Ringkasan Tatacara Penggunaan *Tools*

a. Pelepasan ke persekitaran pembangunan

Pelepasan sistem aplikasi ke persekitaran pembangunan boleh dilakukan dengan menggunakan *pipeline* CI/CD. Tatacara pelepasan ke persekitaran pembangunan boleh dirujuk pada Lampiran F-1.

b. Pelepasan ke persekitaran *staging*

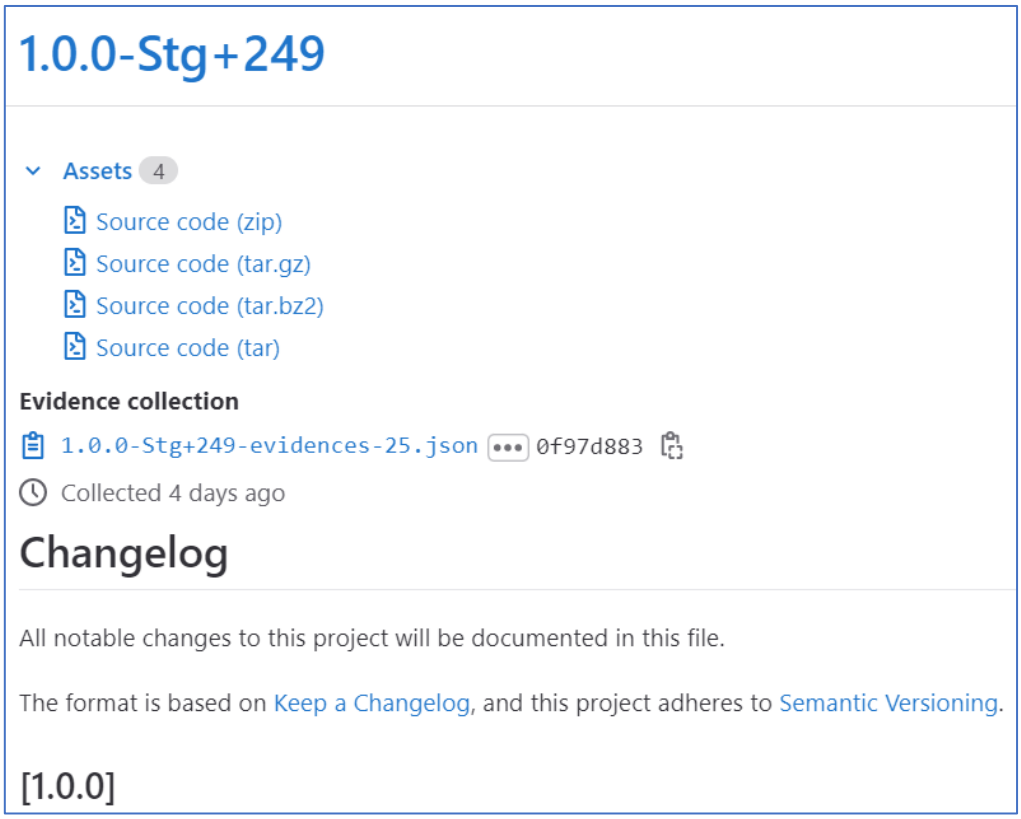
Pelepasan sistem aplikasi ke persekitaran *staging* boleh dilakukan dengan menggunakan *pipeline* CI/CD. Tatacara pelepasan ke persekitaran *staging* boleh dirujuk pada Lampiran F-2.

c. Pelepasan ke persekitaran produksi

Pelepasan sistem aplikasi ke persekitaran produksi boleh dilakukan dengan menggunakan *pipeline* CI/CD. Tatacara pelepasan ke persekitaran produksi boleh dirujuk pada Lampiran F-3.

4.5.5. Serahan/Output

Nota pelepasan boleh dilihat pada artifak GitLab *Releases*. Rajah 4-15 memaparkan artifak daripada GitLab *Releases.*



Rajah 4-15: Paparan Nota Pelepasan pada GitLab *Releases*

4.6. PERINGKAT PENEMPATAN

Penempatan sistem aplikasi secara automasi oleh *pipeline* CI/CD memudahkan pasukan menempatkan versi baharu sistem aplikasi dengan pantas, mengurangkan masa henti dan mengurangkan ralat semasa proses pemasangan imej sistem aplikasi ke persekitaran penempatan. Penerangan berkaitan peringkat penempatan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.7.

4.6.1. Prasyarat

Untuk menempatkan sistem aplikasi, pasukan perlu melaksanakan aktiviti berikut:

a. Mengkonfigurasi GitLab *Runner* untuk menyokong arahan kerja berkaitan proses penempatan.

b. Menyediakan pakej fail sistem aplikasi pada repositori kod sumber.

c. Membina imej Docker dalam *Container Registry*.

d. Menyediakan persekitaran pembangunan, *staging* dan produksi untuk penempatan sistem aplikasi.

e. Menyediakan nota pelepasan daripada artifak GitLab *Releases* untuk rujukan penempatan.

4.6.2. Aliran Proses Kerja Peringkat Penempatan

Peringkat ini memberi tumpuan kepada proses penempatan sistem aplikasi ke persekitaran pembangunan, *staging* dan produksi secara automasi oleh *pipeline* CI/CD. Persekitaran penempatan dilaksanakan menggunakan pelayan seperti berikut.

Persekitaran

a. Pembangunan

b. Staging

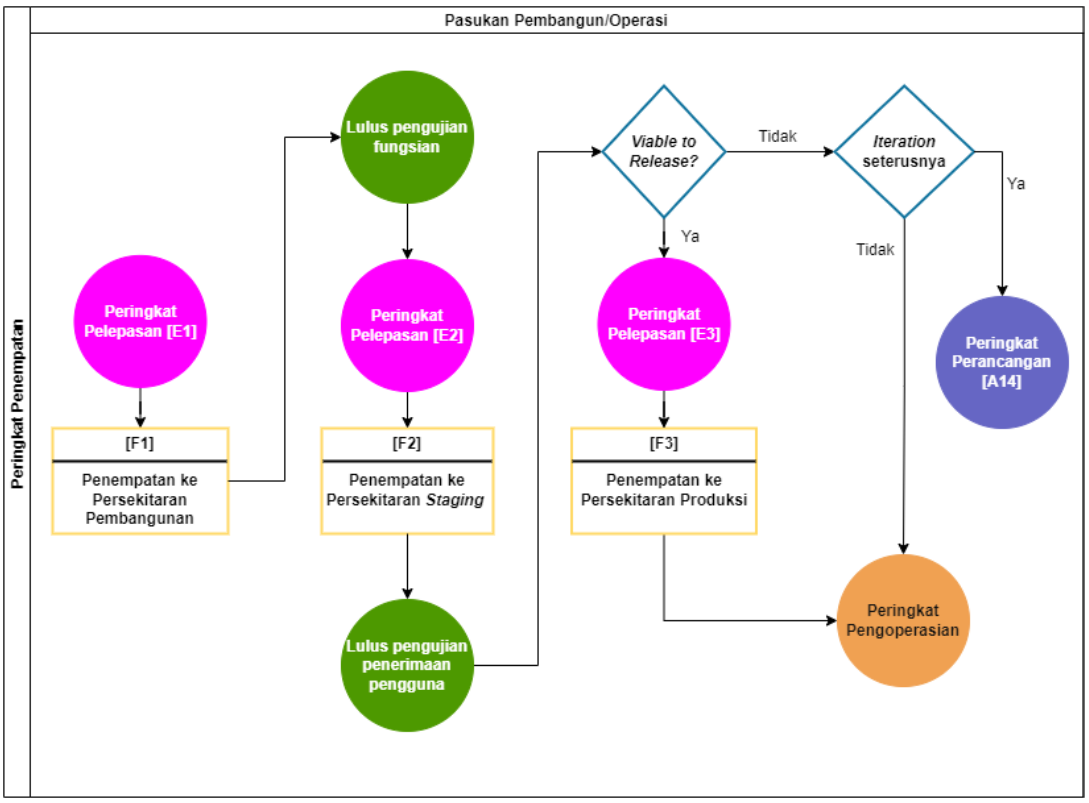
c. Produksi

Pelayan

a. *Virtual Machine* (VM)

b. Docker *Container*

c. Kubernetes



Rajah 4-16: Aliran Kerja Peringkat Penempatan

Berikut adalah penerangan proses penempatan yang berlaku pada *pipeline* CI/CD berdasarkan Rajah 4-16.

a. Penempatan ke Virtual Machine

Pipeline CI/CD akan menempatkan fail pakej sistem aplikasi yang telah dibina ke *virtual machine* secara automasi bagi setiap *merge* ke *branch* repositori pembangunan bagi tujuan pengujian.

b. Penempatan ke Docker *Container*

Pipeline CI/CD akan menempatkan imej Docker yang telah dibina ke Docker *Container* secara automasi bagi setiap *merge* ke *branch* repositori *staging* bagi tujuan pengujian.

c. Penempatan ke Kubernetes

Setelah proses pengujian selesai, kod sumber akan di *merge* ke *branch* repositori produksi dan imej Docker akan dibina. *Pipeline* CI/CD akan menempatkan imej Docker dalam Kubernetes yang berada pada persekitaran produksi.

4.6.3. Pengimbasan Infrastruktur sebagai Kod (IaC)

Pengimbasan Infrastruktur sebagai Kod (IaC) dilaksanakan dengan mengimbas kod yang digunakan untuk menentukan serta menyediakan infrastruktur seperti pelayan, rangkaian, dan storan. Rujuk **Lampiran G-1** untuk contoh konfigurasi skrip pengimbasan Infrastruktur sebagai Kod.

4.6.4. Penempatan Sistem Aplikasi

Pengurusan penempatan sistem aplikasi ke pelbagai persekitaran secara automasi dapat dilaksanakan pada *pipeline* CI/CD dengan konfigurasi yang berasingan untuk setiap persekitaran.

4.6.4.1. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti penempatan sistem aplikasi adalah GitLab. Manakala *features* GitLab yang terlibat adalah seperti berikut:

a. Project
b. *Repository*
c. *Merge Request*
d. *Deployment > Environment.*

4.6.4.2. Ringkasan Tatacara Penggunaan *Tools*

a. Penempatan ke VM

Pasukan perlu memastikan ketersediaan persekitaran VM mengikut bahasa pengaturcaraan sistem aplikasi yang digunakan. Bagi tujuan kajian kes, VM untuk panduan ini telah dikonfigurasi seperti berikut.

i. Sistem Pengoperasian: Linux Ubuntu

ii. Pelayan Web: Nginx

iii. Pangkalan Data: Postgress

iv. Javascript Framework: NodeJS

v. Pengurusan Proses: PM2

Tatacara penempatan sistem aplikasi ke VM boleh dirujuk pada Lampiran G-2.

b. Penempatan menggunakan imej dari Docker *Container*

Pasukan perlu memastikan ketersediaan imej Docker *Container* mengikut bahasa pengaturcaraan sistem aplikasi yang digunakan. Bagi tujuan kes, imej Docker *Container* untuk panduan ini telah dikonfigurasi seperti berikut:

i. Sistem Pengoperasian: Linux VM

ii. Apache *Server*

iii. MySQL

iv. PHP

Tatacara penempatan sistem aplikasi menggunakan imej dari Docker *Container* boleh dirujuk pada **Lampiran G-3**.

c. Penempatan ke Kubernetes

GitLab mempunyai *feature* di bawah *Infrastructure Management* yang membolehkan Kluster Kubernetes digunakan bersama GitLab. *Feature* ini membolehkan penempatan, pengurusan dan pengemaskinian imej sistem aplikasi dijalankan secara automasi pada Kluster Kubernetes dengan menggunakan *pipeline* CI/CD.

Tatacara penggunaan *tools* berikut dibangunkan dengan mengambil kira persekitaran Kubernetes telah disediakan dan di integrasikan kepada GitLab. Langkah tambahan berikut diperlukan jika pasukan ingin menyediakan persekitaran Kubernetes pada persekitaran premis.

i. Menghubungkan Kluster Kubernetes kepada GitLab dan

ii. Mendaftar GitLab *Agent* untuk Kubernetes.

Tatacara penempatan sistem aplikasi ke Kubernetes boleh dirujuk pada **Lampiran G-4**.

4.6.5. Pengujian Penembusan

Pengujian penembusan dilaksanakan pada sistem aplikasi yang telah ditempatkan pada persekitaran *staging* untuk memastikan sistem aplikasi bebas dari sebarang ralat dan isu keselamatan sebelum ditempatkan pada persekitaran produksi.

Diantara pengujian yang dilaksanakan semula adalah seperti berikut:

a. Pengujian Sistem. Rujuk para 4.4.3.3

b. Pengujian DAST. Rujuk para 4.4.4.4

c. Pengujian IAST. Rujuk para 4.4.4.5

4.6.6. Serahan/Output

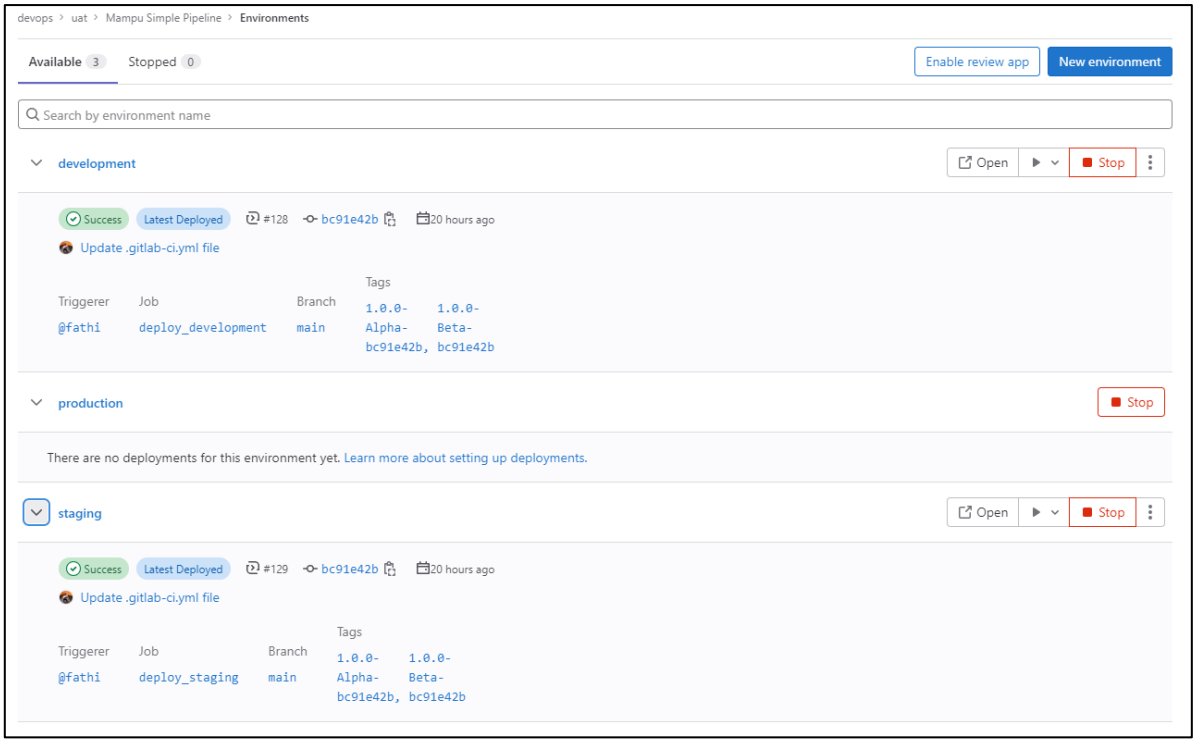
Status senarai persekitaran penempatan sistem aplikasi yang telah dijalankan adalah seperti paparan pada Rajah 4-17.

Untuk melihat senarai persekitaran penempatan:

a. Pilih Menu utama > *Project* dan pilih projek.

b. Pilih *Deployment* > *Environments*

c. Paparan *Environments* yang memaparkan status senarai persekitaran penempatan akan muncul seperti Rajah 4-17.



Rajah 4-17: Paparan *Deployment Environment* pada GitLab

4.7. PERINGKAT PENGOPERASIAN

Pengoperasian merujuk kepada proses dan aktiviti yang dilakukan untuk mengurus dan menyelenggara infrastruktur sistem aplikasi secara automasi melalui pendekatan *Infrastructure as a Code* (IaC). Penerangan berkaitan peringkat pengoperasian boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.8.

Matlamat pengoperasian adalah untuk memastikan infrastruktur sistem aplikasi beroperasi pada tahap optimum melalui aktiviti berikut:

a. Penyandaran pangkalan data (database backup).

b. Pengemasan *container registry* (housekeeping).

c. Pemulihan sistem aplikasi (system recovery).

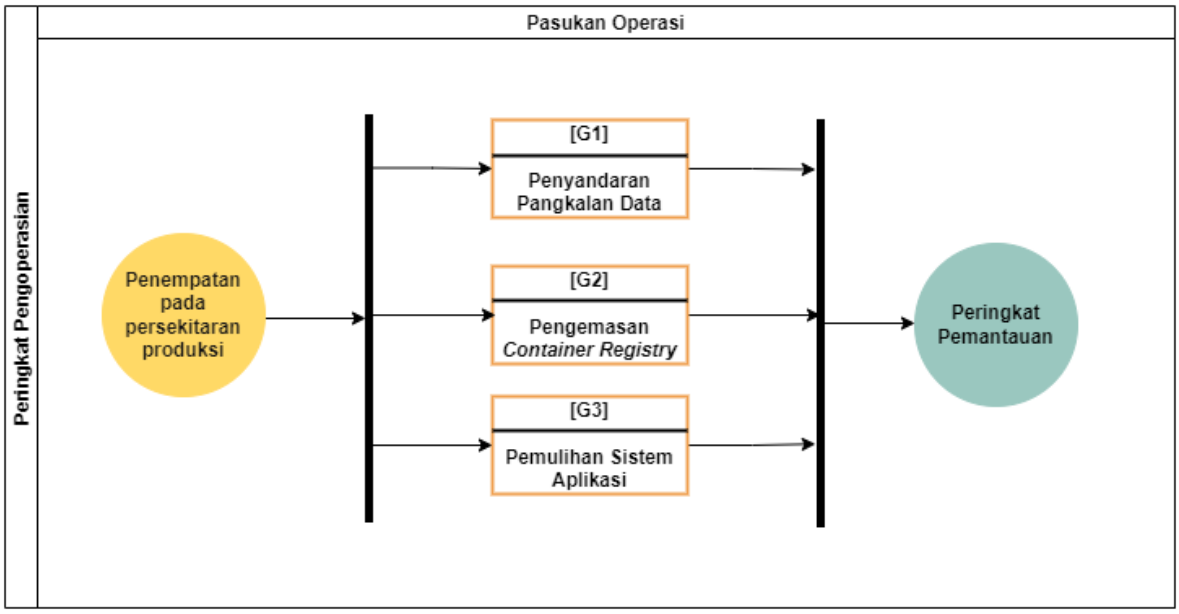
4.7.1. Prasyarat

Perkara yang diperlukan dalam peringkat pengoperasian adalah seperti berikut:

a. Sistem aplikasi telah ditempatkan pada persekitaran produksi.

b. *Tools* pengoperasian dan pemantauan telah dipasang pada persekitaran produksi.

4.7.2. Aliran Proses Kerja Peringkat Pengoperasian



Rajah 4-18: Aliran Kerja Peringkat Pengoperasian

Berikut adalah penerangan proses yang berlaku pada peringkat pengoperasian berdasarkan Rajah 4-18.

4.7.3. Penyandaran Pangkalan Data

Penyandaran merujuk kepada proses membuat salinan data atau fail data untuk digunakan sekiranya data asal atau fail data mengalami kerosakan. Penyandaran bagi panduan ini merujuk kepada aktiviti proses membuat sandaran arkitektur dan data tersimpan bagi pangkalan data. Penyandaran boleh dibuat sama ada secara berkala atau mengikut keperluan agensi.

4.7.3.1. Penggunaan *Tools*

Penyandaran pangkalan data dilaksanakan secara automasi iaitu menggunakan *pipeline* CI/CD. Bagi tujuan kajian kes, persekitaran telah dikonfigurasi seperti berikut:

a. Sistem Pengoperasian: Linux Ubuntu

b. Pangkalan Data: PostgreSQL

c. Skrip *library*: Liquibase

4.7.3.2. Ringkasan Tatacara penggunaan *Tools*

Tatacara penyandaran pangkalan data boleh dirujuk pada **Lampiran H-1.**

4.7.4. Pengemasan *Container Registry*

Pengemasan adalah proses pengoptimuman dan pengemasan sistem yang melibatkan aktiviti seperti pengalihan dan pengemasan ruang storan, fail dan program. Hasil daripada pengemasan memastikan sistem dan storan dapat berfungsi pada keadaan prestasi yang optimum.

Pengemasan imej *container* melalui aktiviti pemadaman imej yang tidak digunakan atau dari versi terdahulu daripada *container registry* bertujuan memastikan penggunaan ruang storan yang optimum. Polisi pemadaman imej yang disarankan adalah mengikut *tools* pengurusan imej yang digunakan atau mengikut keperluan agensi. Pengemasan boleh dibuat sama ada secara berkala atau secara mengikut keperluan agensi.

4.7.4.1. Penggunaan *Tools*

Pemadaman imej *container* dilaksanakan melalui dua kaedah dan *tools* berikut:

a. Pemadaman secara automasi iaitu menggunakan *pipeline* CI/CD untuk melaksanakan kerja pemadaman imej *container* dari *container registry* iaitu Harbor.

b. Pemadaman secara manual iaitu imej *container* dipadam secara manual melalui akses ke Harbor.

4.7.4.2. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pengemasan *container registry* boleh dirujuk pada Lampiran H-2.

4.7.5. Pemulihan Sistem Aplikasi

Pemulihan sistem aplikasi adalah proses memulihkan sistem aplikasi kepada keadaan operasi asal apabila terjadi kegagalan atau kerosakan. Proses ini memainkan peranan penting untuk memastikan sistem aplikasi memiliki tahap ketersediaan yang tinggi, meskipun terjadi masalah teknikal atau kegagalan sistem.

Proses pemulihan sistem aplikasi adalah seperti berikut:

a. Pasukan mendapat notifikasi kegagalan sistem melalui *tools* pemantauan.
b. Proses penempatan semula imej sistem aplikasi dari *container registry* ke persekitaran yang terkesan yang dijalankan secara automasi oleh *pipeline* CI/CD.

4.7.5.1. Penggunaan *Tools*

Pipeline CI/CD digunakan untuk melaksanakan kerja penempatan semula imej *container* sistem aplikasi pada persekitaran yang terkesan.

4.7.5.2. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemulihan sistem aplikasi boleh dirujuk pada **Lampiran H-3**.

4.7.6. Serahan/Output

Serahan untuk peringkat pengoperasian adalah seperti berikut:

a. Artifak GitLab hasil penyandaran pangkalan data.

b. Artifak GitLab dan paparan hasil pengemasan *container registry.*

c. Paparan log hasil pemulihan sistem aplikasi.

4.8. PERINGKAT PEMANTAUAN

Pemantauan adalah proses menjejak, menganalisis dan memaparkan laporan prestasi sistem aplikasi untuk membantu pasukan DevOps mengenal pasti dan menyelesaikan isu dengan lebih efisien Penerangan berkaitan peringkat pemantauan boleh dirujuk pada dokumen Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam di para 3.3.9.

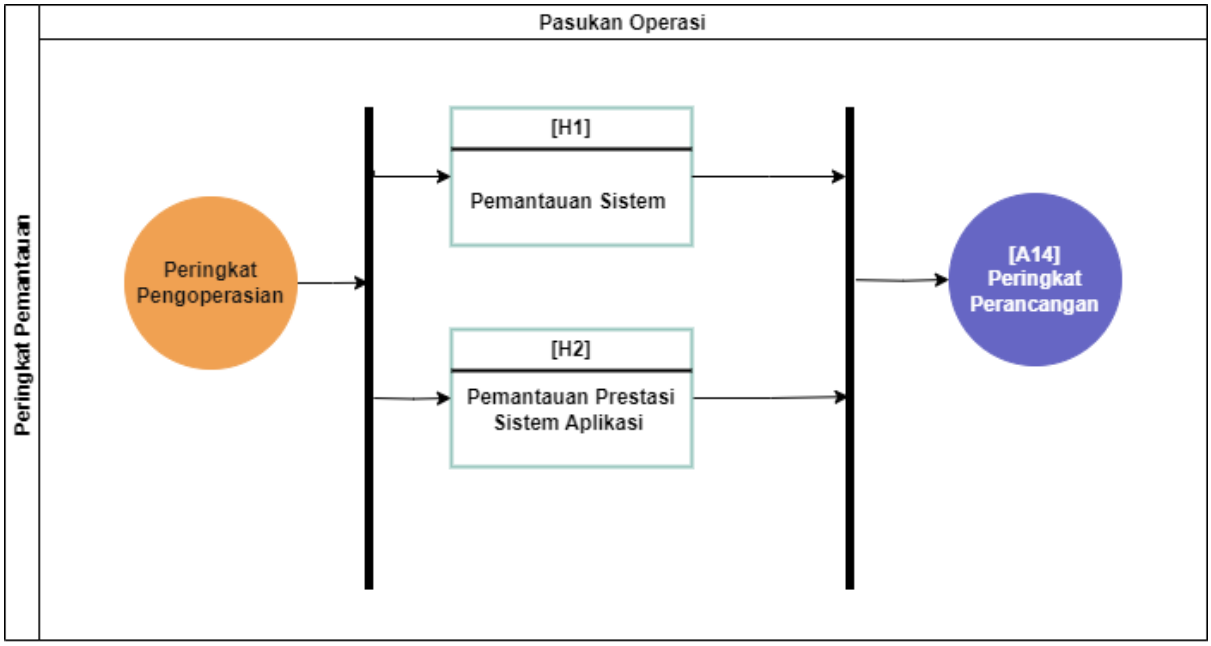
4.8.1. Prasyarat

Perkara yang diperlukan dalam peringkat pemantauan adalah seperti berikut:

a. Sistem aplikasi yang telah ditempatkan pada persekitaran produksi.

b. *Tools* pemantauan telah dipasang pada persekitaran produksi.

4.8.2. Aliran Proses Kerja Peringkat Pemantauan



Rajah 4-19: Aliran Proses Kerja Peringkat Pemantauan

Berikut adalah penerangan peringkat pemantauan yang berlaku berdasarkan Rajah 4-19.

4.8.3. Pemantauan Sistem [H1]

Pemantauan sistem adalah proses untuk memantau log, penggunaan sumber dan kadar ketersediaan sistem. Aktiviti dalam pemantauan sistem adalah seperti berikut:

a. Pemantauan Log [H1.1],
b. Pemantauan Infrastruktur [H1.2], dan
c. Pemantauan *Uptime* [H1.3].

4.8.3.1. Pemantauan Log [H1.1]

Pemantauan log adalah proses memantau log output sistem untuk menentukan peristiwa yang berlaku pada sistem. Dengan perisian pemantauan log, pasukan boleh mengumpul maklumat, mendapat notifikasi dan melaksanakan tindakan penyelesaian berdasarkan log yang diperolehi.

Contoh log yang boleh dipantau adalah seperti berikut:

a. Log *container* seperti Docker dan Kubernetes,
b. Log pangkalan data seperti MySQL dan PostgreSQL,
c. Log web *server* seperti Nginx dan Apache Tomcat,
d. Log sistem pengoperasian seperti Linux dan Windows dan
e. Log dari pelbagai sumber sistem lain.

a. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pemantauan log adalah Elastic Observability.

b. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemantauan log boleh dirujuk pada **Lampiran I-1**.

4.8.3.2. Pemantauan Infrastruktur [H1.2]

Pemantauan infrastruktur adalah proses untuk mengumpul data keadaan dan prestasi dari *server*, mesin maya, *container*, pangkalan data dan komponen *backend* yang lain.

a. Penggunaan *Tools*

Tools yang digunakan pada aktiviti pemantauan infrastruktur adalah Elastic Observability.

b. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemantauan infrastruktur boleh dirujuk pada **Lampiran I-2**.

4.8.3.3. Pemantauan *Uptime* [H1.3]

Pemantauan *uptime* adalah pemeriksaan berkala untuk melihat ketersediaan perkhidmatan seperti laman web atau sistem aplikasi. Apabila perkhidmatan mengalami gangguan (downtime), pemantauan *uptime* mengesan isu dan memberi notifikasi untuk tindakan susulan.

a. Penggunaan *Tools*

Tools yang digunakan pada aktiviti pemantauan *uptime* adalah Elastic Observability.

b. Ringkasan Tatacara Penggunaan *Tools*

Penetapan konfigurasi untuk pengumpulan data *uptime* dari server adalah melibatkan langkah berikut:

i. Penyemakan ketersediaan Agent untuk penerimaan data uptime.

ii. Pemasangan *Heartbeat* pada server yang dipantau.

iii. Penetapan dan konfigurasi protokol pada server yang dipantau.

iv. Penghantaran data *uptime* ke Elasticsearch.

v. Paparan data *uptime.*

Tatacara penetapan pemantauan *uptime* boleh dirujuk pada **Lampiran I-3**.

4.8.4. Pemantauan Prestasi Sistem Aplikasi [H2]

Pemantauan prestasi sistem aplikasi (APM) adalah proses mengumpul data untuk membantu pasukan mengesan ralat, memantau penggunaan sumber dan mengesan perubahan prestasi yang berlaku dalam sistem aplikasi seterusnya memberi kesan terhadap pengalaman pengguna. Aktiviti dalam pemantauan sistem adalah seperti berikut:

a. Pemantauan Prestasi Aplikasi (APM) [H2.1].
b. Pemantauan Pengguna Sebenar (RUM) [H2.2].
c. Pemantauan Sintetik [H2.3].

4.8.4.1. Pemantauan Prestasi Aplikasi (APM) [H2.1]

Pemantauan prestasi aplikasi melibatkan pemerhatian perkhidmatan perisian dan sistem aplikasi pada masa nyata dengan mengumpulkan maklumat terperinci berkaitan prestasi mengenai masa tindak balas untuk permintaan masuk, transaksi permintaan (query) pangkalan data dan banyak lagi.

Pemantauan prestasi sistem aplikasi memfokuskan pada pemantauan lima komponen utama berikut terhadap prestasi aplikasi.

a. *Runtime* seni bina aplikasi.
b. Pemantauan pengguna sebenar.
c. Transaksi aplikasi.
d. Pemantauan komponen.
e. Analisis dan pelaporan.

a. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pemantauan adalah Elastic Observability.

b. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemantauan prestasi sistem aplikasi boleh dirujuk pada **Lampiran I-4**.

4.8.4.2. Pemantauan Pengguna Sebenar (RUM) [H2.2]

Pemantauan pengguna sebenar (RUM) adalah pemantauan pasif yang merekodkan semua interaksi pengguna terhadap sistem aplikasi berasaskan laman web. *Tools* pemantauan akan mengumpul data berkaitan maklumat berikut:

a. Peranti pengguna.
b. Versi pelayar.
c. Tindakan yang diambil oleh pengguna.
d. Maklum balas daripada sistem aplikasi.
e. Pengenalan unik pengguna.

a. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pemantauan adalah Elastic.

b. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemantauan pengguna sebenar boleh dirujuk pada **Lampiran I-5**.

4.8.4.3. Pemantauan Sintetik [H2.3]

Pemantauan sintetik adalah kaedah mensimulasikan interaksi pengguna terhadap penggunaan sistem aplikasi untuk memberikan maklumat tambahan berkaitan prestasi sistem aplikasi berdasarkan senario yang ditetapkan.

Semasa RUM, sebarang kemerosotan sistem aplikasi hanya dapat diperoleh apabila digunakan oleh pengguna sebenar. Masalah kekurangan data transaksi juga akan menjadi kekangan dalam mengenal pasti masalah sistem aplikasi pada peringkat awal.

Mewujudkan pemantauan sintetik sistem aplikasi dalam persekitaran yang ditetapkan dan mensimulasikan tindakan pengguna boleh membantu mengatasi kelemahan RUM seterusnya mengenal pasti sebarang permasalahan pada peringkat awal sebelum sampai kepada pengguna.

a. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pemantauan sintetik adalah seperti berikut:

i. Sistem Pengoperasian: Linux Ubuntu.

ii. *Tools* Pemantauan Sintetik: Sitespeed.io.

iii. *Tools Dashboard* Paparan Data: Grafana.

b. Ringkasan Tatacara Penggunaan *Tools*

Pasukan perlu memastikan ketersediaan persekitaran dan penetapan untuk pelaksanaan pemantauan sintetik seperti berikut:

i. Sitespeed.io dipasang dan dikonfigurasi pada sistem pengoperasian.

ii. Penetapan fail konfigurasi Sitespeed.io untuk menjalankan imej *container* sistem aplikasi.

iii. Penciptaan *dashboard* menggunakan Grafana untuk memaparkan hasil pemantauan sintetik.

Tatacara pemantauan sintetik boleh dirujuk pada **Lampiran I-6**.

4.8.5. Serahan/Output

Serahan untuk peringkat pemantauan adalah seperti berikut:

a. Dashboard Kebolehperhatian
    i. Paparan Pemantauan Log
    ii. Paparan Pemantauan Infrastruktur
    iii. Paparan Pemantauan *Uptime*
    iv. Paparan Pemantauan Prestasi Aplikasi (APM)
    v. Paparan Pemantauan Pengguna Sebenar (RUM)
    vi. Paparan Pemantauan Sintetik
b. Laporan perancangan kapasiti
c. Hasil pemantauan maklum balas pengguna

BAB 5 : PENUTUP

Dokumen ini secara keseluruhannya telah memberi penjelasan mengenai Panduan Pelaksanaan DevOps Dalam Pembangunan Sistem Aplikasi Sektor Awam.

Beberapa metodologi telah dilaksanakan sepanjang penghasilan dokumen ini antaranya adalah penganjuran bengkel, sesi *onboarding* sistem aplikasi Spot-Me dan semakan oleh pasukan pakar. *Output* dalam penghasilan dokumen ini adalah panduan pembangunan sistem aplikasi berdasarkan metodologi *Agile* dan penggunaan *tools* dalam membantu proses automasi serta penyesuaiannya terhadap pelaksanaan DevOps di sektor awam.

Dokumen ini secara asasnya merangkumi proses pelaksanaan DevOps dan kajian kes serta menjelaskan berkenaan kitar hayat DevOps yang merangkumi peringkat perancangan, pengekodan, pembangunan, pengujian, pelepasan, penempatan, pengoperasian dan pemantauan. Pembangunan sistem aplikasi berdasarkan konsep *Agile Scrum* serta panduan pelaksanaan dan penggunaan *tools* automasi merupakan komponen utama bagi dokumen panduan ini yang menjelaskan secara terperinci langkah-langkah untuk melaksanakan DevOps dalam pembangunan sistem aplikasi di sektor awam.

Kejayaan pelaksanaan DevOps di sektor awam bergantung kepada faktor-faktor berikut:

a. Pengurusan atasan sektor awam perlu komited dalam menyokong pelaksanaan DevOps melalui pengadaptasian panduan yang dibangunkan dan aktiviti-aktiviti yang digariskan di dalamnya. Pemahaman yang jelas tentang visi dan objektif bagi pelaksanaan DevOps amat penting bagi memastikan pengurusan atasan agensi dapat merealisasikan pembangunan produk melalui pendekatan ini dengan lebih berkesan,

b. Penglibatan sumber manusia yang kompeten dan mencukupi meliputi pelbagai disiplin adalah penting untuk meningkatkan kecekapan dalam pembangunan produk melalui pendekatan DevOps. Dengan adanya tahap kemahiran dan pengetahuan yang tinggi dalam penggunaan *tools* DevOps, pembangunan

produk dapat dilaksanakan dengan efisien, seterusnya berupaya menghasilkan produk yang berkualiti tinggi,

c. Pembudayaan DevOps perlu dilaksanakan secara berterusan dan tidak terhad kepada tempoh masa tertentu. Langkah pembudayaan ini sentiasa perlu diterapkan dalam amalan kerja pembangunan produk di agensi dengan melaksanakan penambahbaikan berterusan melalui pendekatan yang lebih kolaboratif dan menekankan aspek komunikatif yang tinggi sepanjang proses pembangunan produk,

d. Pemilihan infrastruktur dan *tools* DevOps yang bersesuaian adalah penting bagi memastikan proses pembangunan produk dapat dilaksanakan secara automasi dan

e. Pemantauan yang berterusan pada setiap peringkat dalam kitaran DevOps untuk memastikan kejayaan pelaksanaan DevOps di agensi.

Dokumen ini diharap dapat menjadi rujukan kepada agensi Sektor Awam dalam memantapkan pembangunan sistem aplikasi dan meningkatkan tahap penyampaian perkhidmatan melalui pendekatan DevOps. Inisiatif pelaksanaan DevOps ini juga menyokong Pelan Strategik Pendigitalan Sektor Awam (PSPSA) 2021 – 2025 dengan memperkukuhkan penggunaan teknologi baharu bagi mencipta inovasi dan nilai.

SUMBER RUJUKAN

Amaradri, A. S., & Nutalapati, S. B. (2016). Continuous Integration, Deployment and Testing in DevOps Environment.

Artac, M., Borovssak, T., Di Nitto, E., Guerriero, M., & Tamburri, D. A. (2017, May). DevOps: introducing infrastructure-as-*code*. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 497-498). IEEE.

B. S. Farroha and D. L. Farroha, "A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment," in 2014 IEEE Military Communications Conference, 2014, pp. 288–293.

Bahadori, K., & Vardanega, T. (2018, March). DevOps meets dynamic orchestration. In *International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment* (pp. 142-154). Springer, Cham.

Bolhuis, W. T. C. (2021). *How Can (Large Scale) Agile be Effectively Adopted and Scaled Up in Dutch Public Sector Organisations* (Master's thesis, University of Twente).

Bucena, I., & Kirikova, M. (2017). Simplifying the DevOps Adoption Process. *BIR Workshops*, 1-15.

Cagle, R., Rice, T., & Kristan, M. (2018). *DevOps for federal acquisition*. MITRE CORP BEDFORD MA.

Cherinka, R., Foote, S., Burgo, J., & Prezzama, J. (2022). The Impact of Agile Methods and "DevOps" on Day 2+ Operations for Large Enterprises. In *Intelligent Computing* (pp. 1068-1081). Springer, Cham.

Chris, 10 Essential Steps to Mapping Your DevOps Journey

Chrissis MB, Konrad M, Shrum S. CMMI for development: guidelines for process integration and product improvement. Pearson Education 2011.

CNCF. (Diakses Mac 01, 2022). *CNCF Cloud Native Interactive Landscape.* Retrieved from CNCF Cloud Native Interactive Landscape: https://landscape.cncf.io/

D. K. Taft, "Rackspace Survey Spotlights DevOps Business Benefits: Top 6 Findings," eWeek, pp. 1–1, Nov. 2014.

E. Diel, S. Marczak, and D. S. Cruzes, "Communication Challenges and Strategies in Distributed DevOps," in 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE), 2016, pp. 24–28.

Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. *Ieee Software*, 33(3), 94-100.

Faustino, J., Pereira, R., Alturas, B., & Da Silva, M. M. (2020). Agile information technology service management with DevOps: An incident management case study. *Agile information technology service management with DevOps: An incident management case study*, (4), 339-389.

Garcia VC. RiSE reference model for software reuse adoption in Brazilian companies. From web site http://ivanmachado.com.br/research/rise/thesis/files/2010_ViniciusGarcia_phd.pdf..

Government Accountability Office. Powner, D. (2012). Software Development: Effective Practices and Federal Challenges in Applying Agile Methods, (GAO Publication No. 12-681). Washington, D.C.: U.S. Government Printing Office.

Government Accountability Office. (2020). Agile Assessment Guide : Best Practices for Agile Adoption and Implementation, (GAO Publication No. 20-590G). Washington, D.C.: U.S. Government Printing Office.

Gruver, G. (2016). Starting and Scaling DevOps in the Enterprise. In G. Gruver, *Starting and Scaling DevOps in the Enterprise* (pp. 7-45).

Hannah Moss, Francesca El-Attrash, Joshua Hill - Your Guide to DevOps in Government [Report]. https://docs.broadcom.com/doc/your-guide-to-devops-in-government

Jabbari, R., bin Ali, N., Petersen, K., & Tanveer, B. (2016, May). What is DevOps? A systematic mapping study on definitions and practices. In *Proceedings of the Scientific Workshop Proceedings of XP2016* (pp. 1-11).

Johannes Wettinger, Uwe Breitenbücher, and Frank Leymann, DevOpSlang – Bridging the Gap between Development and Operations

J. Wettinger, U. Breitenbücher, and F. Leymann, "DevOpSlang–bridging the gap between development and operations," in European Conference on Service-Oriented and Cloud Computing,2014, pp. 108–122.

J. Wettinger, U. Breitenbücher, and F. Leymann, "Standards-based devops automation and integration using tosca," in Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014, pp. 59–68.

Khan SU. Software outsourcing vendors' readiness model (SOVRM). Ph.D. dissertation, School Comput. Math., Keele Univ., Keele, U.K: 2011.

Kupiainen, E., Mäntylä, M. V., & Itkonen, J. (2015). Using metrics in Agile and Lean Software Development–A systematic literature review of industrial studies. *Information and software technology*, *62*, 143-163.

L. Evenstad, "Delivering Success with DevOps," Computer Weekly, pp. 23–26, Dec. 2015.

Lappi, T., Karvonen, T., Lwakatare, L. E., Aaltonen, K., & Kuvaja, P. (2018). Toward an improved understanding of agile project governance: A systematic literature review. *Project Management Journal*, *49*(6), 39-63.

Leite, L., Kon, F., Pinto, G., & Meirelles, P. (2020, June). Platform teams: An organizational structure for continuous delivery. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops* (pp. 505-511).

M. A. McCarthy, L. M. Herger, S. M. Khan, and B. M. Belgodere. Composable devops: Automated ontology based devops maturity analysis. In Services Computing (SCC), 2015 IEEE International Conference on, pages 600–607, June 2015.

M. Hüttermann, DevOps for Developers. Apress, 2012.

Mantovani Fontana, R., & Marczak, S. (2020). Characteristics and Challenges of Agile Software Development Adoption in Brazilian Government. *Journal of technology management & innovation*, *15*(2), 3-10.

Merkow, M. S. (2022). *Practical Security for Agile and DevOps*. Auerbach Publications.

Morales, J. A., Yasar, H., & Volkman, A. (2018, May). Implementing DevOps practices in highly regulated environments. In *Proceedings of the 19th International Conference on Agile Software Development: Companion* (pp. 1-9).

Niazi M, Wilson D, Zowghi D. A maturity model for the implementation of software process improvement: an empirical study. J Syst Softw. 2005;74:155-172.

Nicole Blake Johnson, Isaac Constans, Mark Hensch, Katie Malone, Catherine Andrews - Your Guide to DevOps in Government Today [Report]. https://go.govloop.com/rs/231-DWB-776/images/DevOps-in-Government-Today.pdf

Northern, C., Mayfield, K., Benito, R., & Casagni, M. (2010). *Handbook for implementing agile in department of defense information technology acquisition*. MITRE CORP MCLEAN VA.

Ogala, J. O. (2022). A Complete Guide to DevOps Best Practices. *International Journal of Computer Science and Information Security (IJCSIS)*, *20*(2).

Plant, O. H. (2019). *DevOps under control: development of a framework for achieving internal control and effectively managing risks in a DevOps environment* (Master's thesis, University of Twente).

Prestes, M., Parizi, R., Marczak, S., & Conte, T. (2020, June). On the use of design thinking: A survey of the Brazilian agile software development community. In *International Conference on Agile Software Development* (pp. 73-86). Springer, Cham.

Project Management Institute. (2021). *A guide to the Project Management Body of Knowledge (PMBOK guide)* (7th ed.). Project Management Institute.

Rafi S, Yu W, Akbar MA. RMDevOps: a road map for improvement in DevOps activities in context of software organizations. In Proceedings of the Evaluation and Assessment in Software Engineering. 2020:413-418.

Rafi S, Yu W, Akbar MA, Mahmood S, Alsanad A, Gumaei A (2020). Readiness model for DevOps implementation in software organizations. Journal of Software Evolution and Process. 33. 10.1002/smr.2323.

Riungu-Kalliosaari, L. M. (2016, November). DevOps adoption benefits and challenges in practice: A case study. *In International conference on product-focused software process improvement*, pp. 590-597.

Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2019). Advances in using agile and lean processes for software development. In *Advances in Computers* (Vol. 113, pp. 135-224). Elsevier.

S. Jones, J. Noppen, and F. Lettice, “Management Challenges for DevOps Adoption Within UK SMEs,” in Proceedings of the 2Nd International Workshop on Quality-Aware DevOps, New York, NY, USA, 2016, pp. 7–11.

S. W. Hussaini, “Strengthening harmonization of Development (Dev) and Operations (Ops) silos in IT environment through systems approach,” in 2014 IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 178–183.

Sharma, S. (2017). *The DevOps adoption playbook: a guide to adopting DevOps in a multi-speed IT enterprise.* John Wiley & Sons.

Skelton, M., & Pais, M. (2019). *Team topologies: organizing business and technology teams for fast flow*. IT Revolution.

S7 C. Preimesberger, “10 Essential Steps to Mapping Your DevOps Journey,” eWeek, pp. 1–1, Mar. 2016

Taft, Darryl K. Rackspace Survey Spotlights DevOps Business Benefits: Top 6 Findings

W. B.S. Farroha, D.L. Farroha, A Framework for Managing Mission Needs, Compliance and Trust in the DevOps Environment

Wiedemann, A., Wiesche, M., & Krcmar, H. (2019, June). Integrating development and operations in cross-functional teams-toward a devops competency model. In *Proceedings of the 2019 on Computers and People Research Conference* (pp. 14-19).

Yarlagadda, R. T. (2018). How Public Sectors Can Adopt the DevOps Practices to Enhance the System. *International Journal of Emerging Technologies and Innovative Research (www. jetir. org| UGC and issn Approved), ISSN*, 2349-5162.

DevOps Implementation Plan: Benefits, Guide, Definition [Report]. https://*code*it.us/blog/devops-implementation-plan#developing-a-devops-implementation-plan

Enterprise Architect Framework for DevOps Implementation Strategies [Report]. https://www.veritis.com/solutions/devops/implementation-strategy-tools-collaboration/

Dr. Gopala Krishna Behara, Enterprise Architect Framework for DevOps Adoption [Report]. https://www.wipro.com/blogs/dr-gopala-krishna-behara/ea-framework-for-devops-adoption/govloop - DevOps in Government [Report]. https://media.erepublic.com/document/DevOps_in_Government_ebook.pdf.

KEMENTERIAN DIGITAL
JABATAN DIGITAL NEGARA
Bangunan MKN Embasy Techzone,
Blok B, No. 3200 Jalan Teknokrat 2,
63000 Cyberjaya Sepang, Selangor
Darul Ehsan
Malaysia

www.osdec.gov.my