JABATAN PERDANA MENTERI
UNIT PEMODENAN TADBIRAN DAN PERANCANGAN PENGURUSAN MALAYSIA
(MAMPU)

DevOps Sektor Awam

PANDUAN

PELAKSANAAN DEVOPS DALAM PEMBANGUNAN SISTEM APLIKASI SEKTOR AWAM

Unit Pemodenan Tadbiran dan Perancangan
Pengurusan Malaysia (MAMPU)
Jabatan Perdana Menteri

Panduan Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam

(Versi Beta)

KANDUNGAN

SENARAI JADUAL

SENARAI RAJAH

PRAKATA

KETUA PENGARAH MAMPU

Assalamualaikum Warahmatullahi Wabarakatuh dan Salam Sejahtera.

Dengan nama Allah yang Maha Pemurah lagi Maha Mengasihani.

Alhamdulillah, syukur ke hadrat Ilahi kerana dengan limpah kurnia dan izin-Nya, buku **Panduan Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam** dapat diterbitkan dengan jayanya. Pembangunan sistem aplikasi merupakan cabang utama bidang ICT yang menjadi tonggak dalam merealisasikan pendigitalan penyampaian perkhidmatan kerajaan. Justeru, DevOps merupakan pendekatan bagi memendekkan tempoh kitar hayat pembangunan dalam usaha meningkatkan kualiti pembangunan sistem aplikasi sektor awam.

Selaras dengan aspirasi pendigitalan serta inisiatif kerajaan seperti Malaysia MADANI, Rangka Tindakan (Blueprint) Eknonomi Digital Malaysia (MyDigital), Pelan Strategik Pendigitalan Sektor Awam (PSPSA) 2021-2025 dan Pelan Strategik MAMPU 2021-2025, antara lain adalah dengan memberi fokus kepada tambah baik mutu perkhidmatan kepada rakyat, peluasan dan pembabitan masyarakat, perniagaan dan kerajaan dalam ekonomi digital serta penumpuan terhadap perkhidmatan inklusif dan bersepadu amatlah ditekankan. Pelaksanaan DevOps dapat membantu proses pembangunan dan operasi menjadi lebih cekap dan mesra sumber di samping mewujudkan mekanisme perancangan, penyelarasan, pembangunan dan pelaksanaan dalam pembangunan sistem aplikasi sektor awam.

Selain itu, pelaksanaan DevOps dapat menggalak penerokaaan idea baharu bagi pemodenan perkhidmatan awam dengan menggabungkan teknologi baharu seperti kecerdasan buatan (AI), pembelajaran mesin (ML), *internet of things* (IoT) dan pengkomputeran awan.

Sebagai sebuah agensi pusat, MAMPU bertanggungjawab sebagai perancang dan peneraju agenda pendigitalan dan pemodenan sektor awam perlu sentiasa peka terhadap perubahan dan penggunaan teknologi mempertingkatkan pencapaian dan seterusnya menambah baik kualiti perkhidmatan awam. Peralihan kepada pendekatan DevOps perlu dilaksanakan secara berperingkat kerana proses tersebut memerlukan masa yang sesuai, tahap ketersediaan dan persekitaran semasa agensi bagi memastikan perubahan dapat diterima dan diamalkan secara berkesan.

Seterusnya, saya ingin mengucapkan syabas dan tahniah kepada ahli pasukan projek yang terlibat secara langsung atau tidak langsung dalam menghasilkan buku Panduan Pelaksanaan DevOps ini. Buku ini merupakan kesinambungan dan kelangsungan dokumen **Rangka Kerja Pelaksanaan DevOps dalam Pembangunan Sistem Aplikasi Sektor Awam**. Adalah menjadi harapan saya, agar buku panduan ini dapat dimanfaatkan sepenuhnya sebagai rujukan utama oleh semua agensi kerajaan dalam pengadaptasian DevOps terutama dalam pembangunan sistem aplikasi yang lebih mantap dan efisien. Saya yakin dan percaya melalui pendekatan baharu yang digariskan dalam dokumen ini akan berupaya meningkatkan produktiviti dan kualiti pembangunan sistem aplikasi secara khusus seterusnya meningkatkan lagi mutu penyampaian perkhidmatan awam.

Sekian, terima kasih.

Datuk Haji Rodzi Bin Md Saad

KETUA PENGARAH MAMPU

PRAKATA

TIMBALAN KETUA PENGARAH (ICT)

Assalamualaikum Warahmatullahi Wabarakatuh dan Salam Sejahtera.

Dengan nama Allah yang Maha Pemurah lagi Maha Mengasihani.

Segala puji dan syukur bagi Allah kerana dengan limpah izin-Nya, dokumen Panduan Pelaksanaan DevOps bagi Pembangunan Sistem Aplikasi Sektor Awam telah dapat disiapkan dan diterbitkan dengan jayanya. Dokumen ini akan menjadi rujukan yang komprehensif kepada pasukan pembangun sistem aplikasi kerajaan khususnya dalam pengadaptasian pendekatan DevOps dalam aktiviti pembangunan sistem.

Pembangunan sistem aplikasi menggunakan pendekatan DevOps merupakan satu budaya kerja baharu yang menggabungkan pasukan pembangun dan pasukan operasi serta penggunaan *tools* yang berupaya membantu pengautomasian proses pembangunan sistem. Terdapat lapan peringkat dalam kitar hayat pelaksanaan DevOps iaitu peringkat perancangan, pengekodan, pembangunan, pengujian, pelepasan, penempatan, pengoperasian dan pemantauan. Selain daripada itu, metodologi *agile* turut diterapkan dalam pelaksanaan DevOps sebagai suatu anjakan budaya bagi pembangunan sistem aplikasi di sektor awam.

Di samping itu, budaya kerja baharu ini juga memudahkan dan mempercepatkan proses kerja serta meningkatkan hubungan kerjasama antara pasukan berlainan

dalam organisasi dengan bekerja sebagai satu pasukan. Secara tidak langsung, pendekatan ini menyokong usaha murni kerajaan dalam meningkatkan produktiviti pembangunan sistem dan seterusnya menjadi pemangkin kecekapan kepada penyampaian perkhidmatan awam melalui pendekatan teknologi terkini.

Syabas dan tahniah kepada semua pihak yang terlibat termasuk Pasukan Projek DevOps dan Pasukan Perunding ICT dari Bahagian Perundingan ICT, MAMPU yang komited dalam menggembleng tenaga, idea dan pandangan bagi merealisasikan buku Panduan Pelaksanaan DevOps Sektor Awam ini. Penghasilan dokumen ini membuktikan kerajaan amat mengutamakan penambahbaikan berterusan seiring dengan kepesatan perkembangan teknologi ICT. Semoga panduan ini dapat memberi manfaat sepenuhnya pada pelaksanaan pembangunan sistem aplikasi di agensi sektor awam.

Sekian, terima kasih.

Dr. Fazidah Binti Abu Bakar

TIMBALAN KETUA PENGARAH (ICT) MAMPU

AKRONIM

ACT- IAC	*American Council for Technology-Industry Advisory Council*
DASA	*DevOps Agile Skills Association*
IaC	*Infrastructure as Code*
ICT	*Information and Communication Technology*
MVP	*Minimum Viable Product*
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 *iterative* 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.
Minimum Viable Product (MVP)	MVP adalah versi produk dengan fungsi minimum yang mencukupi serta memberikan nilai kepada pengguna.

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.
Pasukan Operasi	Merupakan sebahagian daripada pasukan DevOps dan bekerjasama rapat dengan pasukan pembangun bagi memastikan penyampaian produk tersedia kepada pengguna.
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 lima teras yang telah diperkenalkan iaitu Penerapan Prinsip dan Budaya, Pengadaptasian Metodologi, Pengukuhan Tadbir Urus, Pemantapan Pelaksanaan dan Pemantauan, serta Pemerkasaan Teknologi. Panduan Pelaksanaan DevOps Sektor Awam ini akan menumpu dan memperincikan kepada teras Pemerkasaan Teknologi serta kaedah penggunaan *tools* bagi setiap peringkat dalam kitar hayat DevOps. Selain daripada itu, metodologi *Agile Scrum* juga akan diperkenalkan untuk melengkapi proses kerja DevOps. Aktiviti seperti *sprint zero, sprint planning meeting, daily scrum, product backlog refinement, sprint review* dan *sprint retrospective* akan diterangkan bersama artifak-artifak *agile* yang menyokong pelaksanaan pendekatan DevOps di sektor awam.

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, perancangan pelaksanaan *pipeline* CI/CD dan 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 serta imej *container*. Peringkat pengujian menjelaskan dua pengujian utama iaitu pengujian keperluan fungsian dan 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, pengemasan *container registry* dan pemulihan sistem aplikasi. 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 dengan menggunakan 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, MAMPU dan pegawai teknikal Bahagian Pembangunan Aplikasi, MAMPU untuk memastikan kandungan dokumen adalah lengkap, komprehensif dan bersesuaian dengan konteks pelaksanaan DevOps di agensi sektor awam.

Panduan ini terdiri daripada 6 bab iaitu:

a. **Bab 1** menjelaskan tujuan penghasilan dokumen panduan.

b. **Bab 2** menerangkan pengenalan kepada rangka kerja DevOps, amalan penambahbaikan berterusan, pelaksanaan DevOps sektor awam dan pengenalan kajian kes.

c. **Bab 3** menjelaskan konsep *agile scrum*, ahli pasukan, aktiviti dan artifak *scrum* serta langkah pelaksanaannya di agensi sektor awam.

d. **Bab 4** menerangkan *tools* automasi secara generik yang boleh digunakan oleh agensi dalam melaksanakan DevOps.

e. **Bab 5** menjelaskan berkenaan kitar hayat DevOps yang merangkumi peringkat perancangan, pengekodan, pembangunan, pengujian, pelepasan, penempatan, pengoperasian dan pemantauan.

f. **Bab 6** merumuskan kandungan dokumen panduan ini dan jangkaan pengguna terhadap dokumen ini.

Dokumen ini perlu dibaca bersama dengan dokumen berkaitan yang telah diterbitkan oleh MAMPU, 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 MAMPU 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

Objektif penyediaan dokumen ini adalah seperti berikut:

a. Memperkenalkan 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 MAMPU.

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

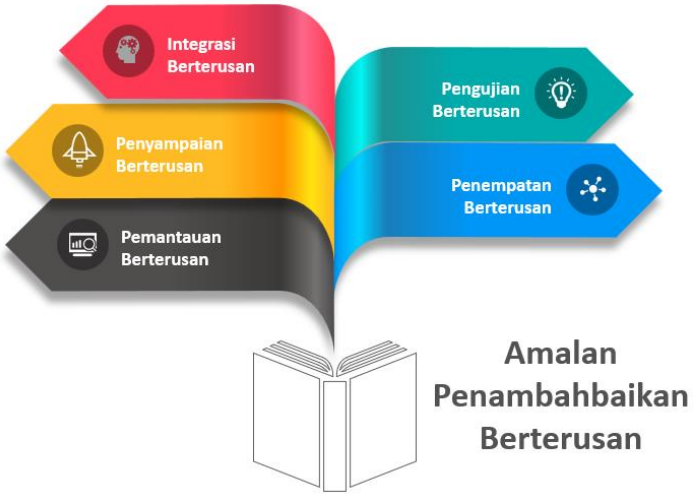
Rajah 2-1 memaparkan model Rangka Kerja Pelaksanaan DevOps Sektor Awam. Terdapat lima teras yang diperkenalkan dalam model rangka kerja ini iaitu Penerapan Prinsip dan Budaya, Pengadaptasian Metodologi, Pengukuhan Tadbir Urus, Pemantapan Pelaksanaan dan Pemantauan, serta Pemerkasaan Teknologi.

Manakala lima pemboleh daya yang menyokong kejayaan pelaksanaan teras 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. Amalan Penambahbaikan Berterusan

Amalan penambahbaikan berterusan merupakan pendekatan dalam pengurusan kualiti yang bertujuan meningkatkan kecekapan, keberkesanan dan kebolehpercayaan sistem aplikasi melalui penambahbaikan berterusan. Dengan amalan penambahbaikan berterusan, sistem aplikasi yang dibangunkan akan ditambah baik, diuji dan dilepaskan ke persekitaran produksi dalam kitaran yang berulang (iterative). Kaedah pengulangan (iteration) ini dapat mengurangkan risiko kegagalan sistem melalui perubahan kecil dan pelepasan secara kerap (frequent release). Konsep integrasi dan penyampaian berterusan (CI/CD) melibatkan pelaksanaan automasi merentas peringkat DevOps bagi meminimumkan proses manual dari pengekodan sehingga penempatan kod ke persekitaran yang bersesuaian.



Rajah 2-2: Amalan Penambahbaikan Berterusan

Lima pendekatan utama dalam amalan penambahbaikan berterusan seperti di Rajah 2-2:

a. Integrasi berterusan

Integrasi berterusan atau *continuous integration* (CI) adalah pendekatan dalam DevOps yang melibatkan pengintegrasian perubahan kod sumber secara automatik dan berterusan. Proses integrasi berterusan ini berlaku apabila pembangun membuat perubahan pada kod sumber dan mengintegrasikan kod secara berkala ke repositori pengurusan kod sumber berpusat. Proses ini membolehkan pembangun mengesan masalah dengan lebih awal dalam proses pembangunan, mempercepat pengesahan perubahan dan meminimumkan risiko yang terlibat apabila dilepaskan ke persekitaran produksi. Integrasi berterusan merupakan elemen penting dalam penyampaian dan penempatan berterusan supaya sistem aplikasi dapat dilepaskan dengan selamat, berulang dan cepat.

b. Pengujian berterusan

Pengujian berterusan atau *continuous testing* merupakan satu pendekatan DevOps yang melibatkan pengujian secara automatik dan berterusan daripada sistem aplikasi yang sedang dibangunkan. Tujuan pengujian berterusan ini adalah untuk memastikan bahawa setiap perubahan pada kod sumber diuji

secara automatik dan diintegrasikan dengan kod sumber yang sedia ada serta memastikan bahawa sistem aplikasi yang dihasilkan memenuhi keperluan bisnes dan mempunyai kualiti yang baik.

c. Penyampaian berterusan

Penyampaian berterusan atau *continuous delivery* (CD) merupakan pendekatan dalam DevOps yang melibatkan penghantaran sistem aplikasi ke persekitaran yang berkaitan secara automatik dan berterusan. Tujuan penyampaian berterusan ini adalah untuk memastikan setiap perubahan pada kod sumber diintegrasikan, diuji secara automatik dan sedia untuk dihantar ke persekitaran yang berkaitan.

d. Penempatan berterusan

Penempatan berterusan atau *continuous deployment* (CD) merupakan pendekatan dalam DevOps yang melibatkan penempatan sistem aplikasi di persekitaran yang berkaitan secara automatik dan berterusan. Tujuan penempatan berterusan ini adalah untuk mempercepatkan penghantaran dan meminimumkan waktu pelepasan perubahan kod sumber yang telah diuji ke persekitaran produksi.

e. Pemantauan berterusan

Pemantauan berterusan atau *continuous monitoring* adalah pendekatan dalam DevOps yang melibatkan pemantauan dan analisis secara berterusan daripada *tools* pemantauan. Tujuan proses pemantauan berterusan ini adalah supaya sistem aplikasi berjalan lancar dan pasukan DevOps dapat memberikan maklum balas dengan pantas terhadap masalah atau ralat. Pemantauan berterusan juga dapat mengurangkan atau meminimumkan gangguan fungsi produk kepada pengguna akhir dan memastikan kebolehcapaian produk berada pada tahap optimum.

2.2. 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.

2.3. 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 *AGILE SCRUM*

3.1. PENGENALAN *AGILE*

Pembangunan sistem aplikasi dengan metodologi *agile* merujuk kepada pendekatan pembangunan secara peningkatan (incremental) dan pengulangan (iterative) iaitu penambahbaikan keperluan pengguna adalah selari dengan aktiviti pembangunan melalui kerjasama di antara pasukan pembangun dan pemilik produk serta semua pemegang taruh yang berkaitan. Matlamat utama pembangunan *agile* adalah untuk membolehkan pasukan pembangun menghasilkan produk yang lebih pantas, berkualiti dan bertindak balas secara cepat terhadap perubahan.

Falsafah *agile* ini disokong oleh 4 nilai dan 12 prinsip yang didokumenkan dalam manifesto *agile*[1]. Empat nilai *agile* menggalakkan proses pembangunan produk yang memberi tumpuan kepada kualiti dan penghasilan produk yang memenuhi keperluan dan jangkaan pengguna. Manakala 12 prinsip *agile* yang digariskan bertujuan menyediakan persekitaran kerja yang tertumpu kepada keperluan pengguna dan sejajar dengan objektif penghasilan produk.

Rajah 3-1: Nilai-nilai Utama *Agile*

[1] Manifesto for Agile Software Development. (n.d.). Diakses pada April 4, 2023, dari https://agilemanifesto.org/

Pelaksanaan manifesto *agile* memberikan nilai yang lebih baik untuk pengguna dan meningkatkan kecekapan serta keberkesanan dalam proses pembangunan sistem aplikasi. Berikut merupakan penerangan terhadap 4 nilai utama *manifesto agile* seperti di Rajah 3-1:

a. Individu dan interaksi melebihi proses dan alatan

Menekankan kepentingan komunikasi, kerjasama dan kerja berpasukan di kalangan ahli pasukan. Interaksi antara ahli pasukan membantu meningkatkan kerjasama dan faktor terpenting dalam pembangunan sistem aplikasi yang berjaya.

b. Produk yang berfungsi melebihi dokumentasi komprehensif

Menekankan kepentingan menyampaikan produk yang berfungsi kepada pengguna seawal mungkin dan meminimumkan dokumentasi yang tidak diperlukan.

c. Kerjasama pelanggan melebihi rundingan kontrak

Menekankan kepentingan melibatkan pengguna akhir dalam proses pembangunan dan bekerjasama rapat untuk memahami keperluan dan kehendak pengguna. Perubahan keperluan pengguna dari semasa ke semasa dapat di peroleh secara segera seterusnya membawa kepada hasil produk yang lebih baik.

d. Bertindak balas terhadap perubahan melebihi perancangan

Menyesuaikan perancangan dan pembangunan sistem aplikasi terhadap perubahan keadaan, keutamaan dan keperluan pengguna secara fleksibel.

Terdapat 12 prinsip *agile* yang terkandung dalam manifesto *agile* iaitu:

a. Penyampaian produk yang memenuhi keperluan dan jangkaan pengguna melalui hasil kerja yang lebih cepat serta berterusan.

b. Perubahan keperluan pengguna diutamakan dan disesuaikan dengan cepat sepanjang tempoh pembangunan.

c. Penyampaian produk yang berfungsi pada setiap lelaran secara kerap.

d. Pemilik bisnes dan pembangun produk bekerjasama sepanjang tempoh pembangunan.

e. Produk dibangunkan dalam kalangan individu yang bermotivasi, saling memberi sokongan dan kepercayaan agar tugasan boleh diselesaikan.

f. Perbincangan secara bersemuka.

g. Produk yang berfungsi adalah ukuran utama kemajuan.

h. Persekitaran pembangunan yang mampan membantu pasukan bekerja dengan cekap dan produktif.

i. Perhatian yang berterusan kepada kecemerlangan teknikal.

j. Penumpuan kepada perkara yang penting dan perlu.

k. Pasukan yang berdikari.

l. Penilaian prestasi pasukan secara berkala ke arah penambahbaikan.

3.2. AGILE SCRUM

Terdapat beberapa pendekatan pembangunan sistem aplikasi berdasarkan metodologi *agile* seperti *Scrum*, *Kanban, Lean Software Development* dan *Extreme Programming*. Setiap pendekatan tersebut mengandungi konsep dan prinsipnya yang tersendiri. Namun begitu, metodologi *scrum* merupakan metodologi yang paling terkenal dan berkesan turut digunakan oleh organisasi besar seperti Google, Apple, Facebook, Adobe, AirBnB, Spotify dan Yahoo dalam mengurus operasi harian mereka.

Pendekatan *scrum* merupakan salah satu pendekatan *agile* yang menggalakkan kolaborasi yang efektif dan berterusan dalam kalangan ahli pasukan DevOps. Pendekatan ini dilengkapi dengan aktiviti, peranan dan artifak yang boleh dipraktikkan oleh pasukan DevOps bagi membolehkan mereka bekerja secara adaptif dan responsif terhadap perubahan keperluan pengguna. Gabungan falsafah *agile* dan pendekatan *scrum* membentuk metodologi *agile scrum* yang memfokuskan kepada pembangunan secara meningkat dan berulang (incremental and iterative development) melalui kaedah pelaksanaan yang berstruktur.

Berikut merupakan 4 manfaat utama pelaksanaan DevOps menggunakan metodologi *agile scrum*:

a. Berupaya mengurus perubahan keperluan pengguna dengan cepat

Bertepatan dengan prinsip *agile* yang menggalakkan perubahan di sepanjang tempoh pembangunan, sebarang perubahan pada keperluan pengguna dapat diurus dengan lebih efektif dan berkesan.

b. Ketelusan yang lebih tinggi

Pasukan pembangun melaporkan dan mengemas kini status aktiviti *sprint* setiap hari melalui artifak *agile scrum*. Ini menjadikan kandungan dan status kemajuan pembangunan semasa termasuk keputusan ujian, dapat dilihat oleh pasukan, pengurusan dan semua pihak yang berkepentingan.

c. Kolaborasi yang berkesan

Menggalakkan kolaborasi yang berterusan dengan menghapuskan *silo* melalui komunikasi terbuka dalam kalangan ahli pasukan DevOps.

d. Pelepasan produk yang pantas

Minimum viable product (MVP) boleh dilepas dan digunakan oleh pemilik produk dengan pantas pada setiap akhir *sprint*. Melalui maklum balas yang diterima, penambahbaikan berterusan akan dibangun dan dilepaskan pada *sprint* seterusnya.

Agile scrum juga menggalakkan penerapan nilai positif di sepanjang pelaksanaannya yang akan menjadi nilai tambah kepada pasukan DevOps. Berikut merupakan nilai-nilai positif yang selalu diterapkan menerusi pelaksanaan metodologi *agile scrum*:

a. Komitmen

Ahli pasukan perlu memberi komitmen sepenuhnya dan membantu satu sama lain dalam menghasilkan produk yang berkualiti kepada pengguna sepanjang *sprint*.

b. Keberanian

Ahli pasukan perlu berani membuat keputusan dalam menyelesaikan sebarang masalah sepanjang *sprint*.

c. Fokus

Ahli pasukan perlu fokus dalam menyelesaikan tugasan dalam mencapai sasaran pada setiap *sprint*.

d. Keterbukaan

Ahli pasukan perlu bersikap terbuka terhadap cabaran yang dihadapi.

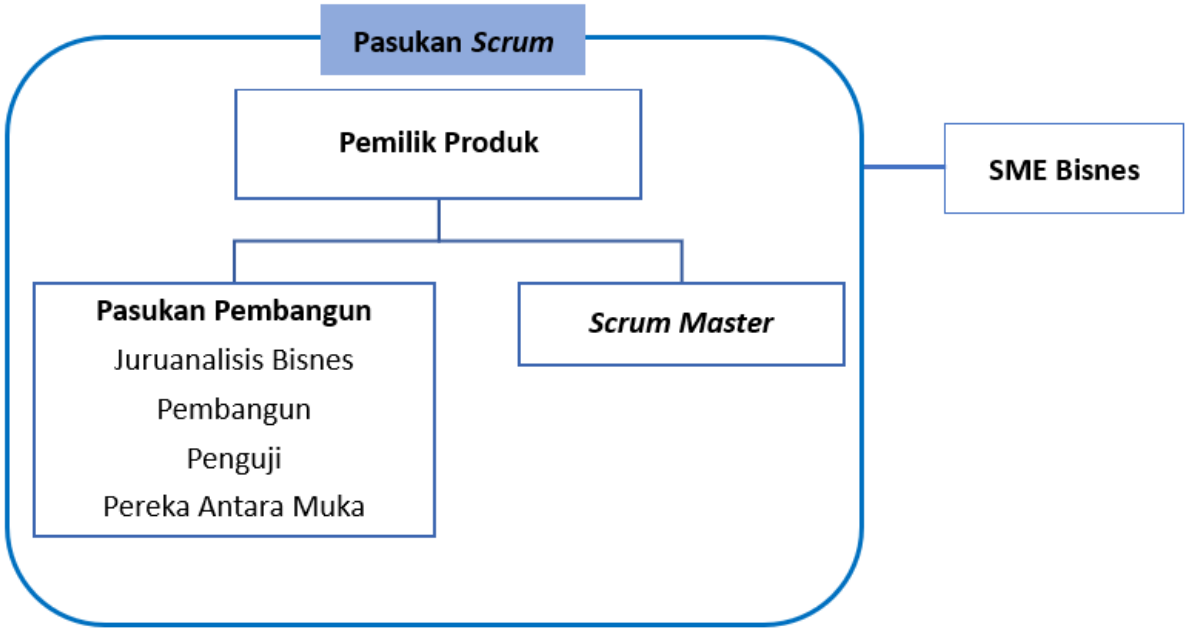
e. Hormat-menghormati

Ahli pasukan perlu saling menghormati antara satu sama lain dan bersedia menerima pandangan ahli pasukan.

3.3. AHLI PASUKAN *SCRUM*

Ahli pasukan *scrum* terdiri daripada pemilik produk, SME bisnes, pasukan pembangun dan *scrum master.* Peranan pasukan *scrum* memfokuskan kepada pembangunan produk untuk memenuhi spesifikasi dan keperluan pengguna dalam tempoh masa yang ditetapkan. Pasukan *scrum* juga merupakan sebahagian daripada ahli pasukan DevOps.

Rajah 3-2 memaparkan struktur organisasi keahlian pasukan scrum.



Rajah 3-2: Struktur Pasukan *Scrum dan* SME Bisnes

3.3.1. Pemilik Produk

Pemilik produk merupakan individu yang mengetuai dan bertanggungjawab untuk membuat keputusan berkenaan hala tuju produk yang ingin dibangunkan. Pemilik produk boleh terdiri daripada pemilik bisnes atau pegawai teknikal.

Berikut adalah beberapa peranan dan tanggungjawab pemilik produk dalam *agile scrum*:

a. Menentukan visi dan hala tuju produk

Berkolaborasi dengan SME Bisnes untuk memahami keperluan pengguna, mengumpulkan maklum balas dan memastikan produk memenuhi keperluan pengguna. Pemilik produk akan berkomunikasi dengan pasukan pembangun, SME Bisnes dan pengguna dalam menentukan visi produk seperti tujuan, matlamat dan objektif produk.

b. Menentukan *user story* dan *acceptance criteria*

Menentukan *user story* dan *acceptance criteria* yang menerangkan fungsi produk mengikut keperluan pengguna. Pemilik produk perlu membuat keputusan bagi pasukan seperti menerima atau menolak tugasan yang diselesaikan oleh pasukan pembangun.

c. Mengurus *product backlog*

Bertanggungjawab untuk menguruskan *product backlog* dan menyusun keutamaan ciri-ciri produk mengikut keperluan pengguna. Pemilik produk secara berkala perlu meninjau dan menyesuaikan *product backlog* berdasarkan maklum balas dan keutamaan yang berubah.

d. Memastikan kualiti produk pada tahap tertinggi

Memastikan produk yang dihasilkan berkualiti tinggi, memenuhi keperluan pengguna dan disampaikan tepat pada waktunya mengikut kesesuaian.

e. Bertindak sebagai pemimpin dan menyediakan arahan yang jelas

Bertindak sebagai pemimpin dalam menyediakan arahan dan penjelasan kepada pasukan pembangun bagi memastikan bahawa mereka memahami keperluan pengguna. Ini bagi memastikan produk yang terhasil mengikut kehendak pengguna.

3.3.2. SME Bisnes

SME ialah seseorang yang mempunyai kepakaran dan pengetahuan dalam proses bisnes produk yang dibangunkan. SME bisnes tidak terlibat secara langsung dalam aktiviti *scrum*, tetapi perlu bekerja rapat dengan pasukan pembangunan dan pemilik produk untuk memberikan panduan dan pandangan berkenaan dengan bidang kepakarannya.

SME bisnes bertanggungjawab untuk memberikan input, spesifikasi dan keperluan sebenar produk. Penglibatan SME yang berkemahiran dan kompeten dalam proses bisnes diperlukan dalam melaksanakan aktiviti pembangunan produk.

Berikut adalah beberapa peranan dan tanggungjawab SME bisnes:

a. Menyediakan panduan mengenai keperluan dan skop produk.
b. Membantu pasukan memahami dan melaksanakan amalan terbaik yang berkaitan dengan bidang kepakarannya.
c. Membantu dalam pembangunan *user story* dan *acceptance criteria*.
d. Menyertai *sprint planning meeting* untuk membantu mengutamakan dan menganggarkan tugasan *product backlog*.
e. Menyediakan maklum balas mengenai tugasan yang selesai dan mencadangkan penambahbaikan untuk *sprint* berikutnya.

3.3.3. Pasukan Pembangun

Pasukan pembangun merupakan sebuah pasukan yang bertanggungjawab membangunkan produk yang dikehendaki oleh pemilik produk. Secara idealnya, pasukan ini terdiri daripada lima hingga 10 orang yang mempunyai pelbagai kemahiran dalam menghasilkan produk untuk memenuhi *definition of done* (DoD) pada setiap *sprint*. Cadangan peranan ahli pasukan pembangun adalah terdiri daripada juruanalisis bisnes, pembangun, penguji dan pereka antara muka. Ahli pasukan boleh memainkan beberapa peranan bergantung kepada kemahiran, saiz pasukan dan kompleksiti produk.

3.3.4. *Scrum Master*

Scrum master merupakan individu yang bertanggungjawab membantu pasukan memudahkan, mempraktikkan dan mematuhi pelaksanaan prinsip serta amalan *scrum*. Antara tugasan utama *scrum master* adalah seperti berikut:

a. Mengurus dan melancarkan setiap aktiviti *scrum seperti scrum planning meeting, daily scrum, sprint review* dan *sprint restropective*.

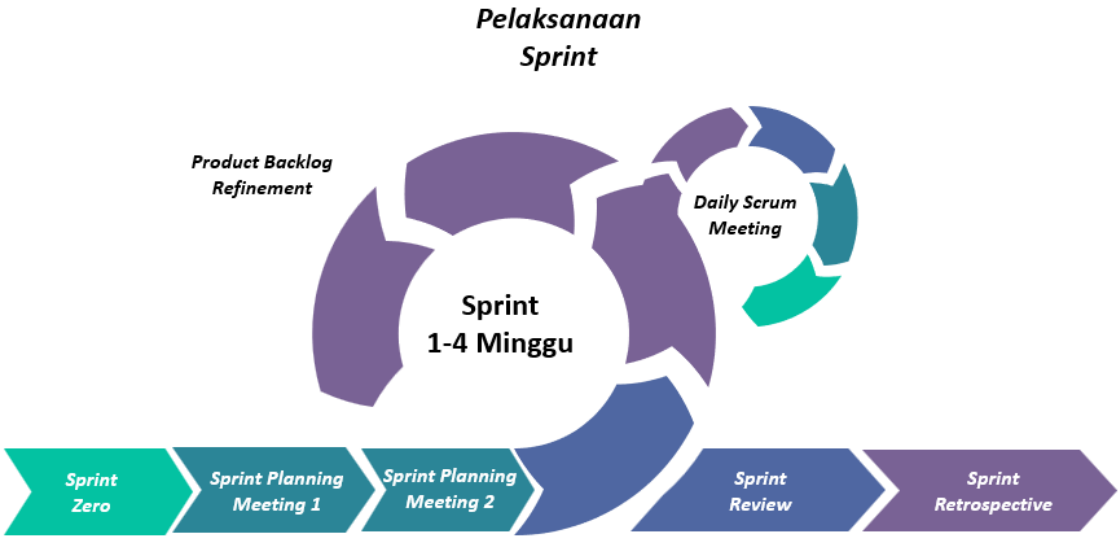
b. Menjadi pemudah cara kepada permasalahan yang dihadapi oleh ahli pasukan yang memberi kesan kepada kemajuan aktiviti pembangunan.

c. Memudahkan serta menggalakkan komunikasi dan kerjasama dalam pasukan juga dengan pihak berkepentingan luar.

d. Membantu menambah baik peranan dan tugas ahli pasukan dalam melaksanakan aktiviti pembangunan.

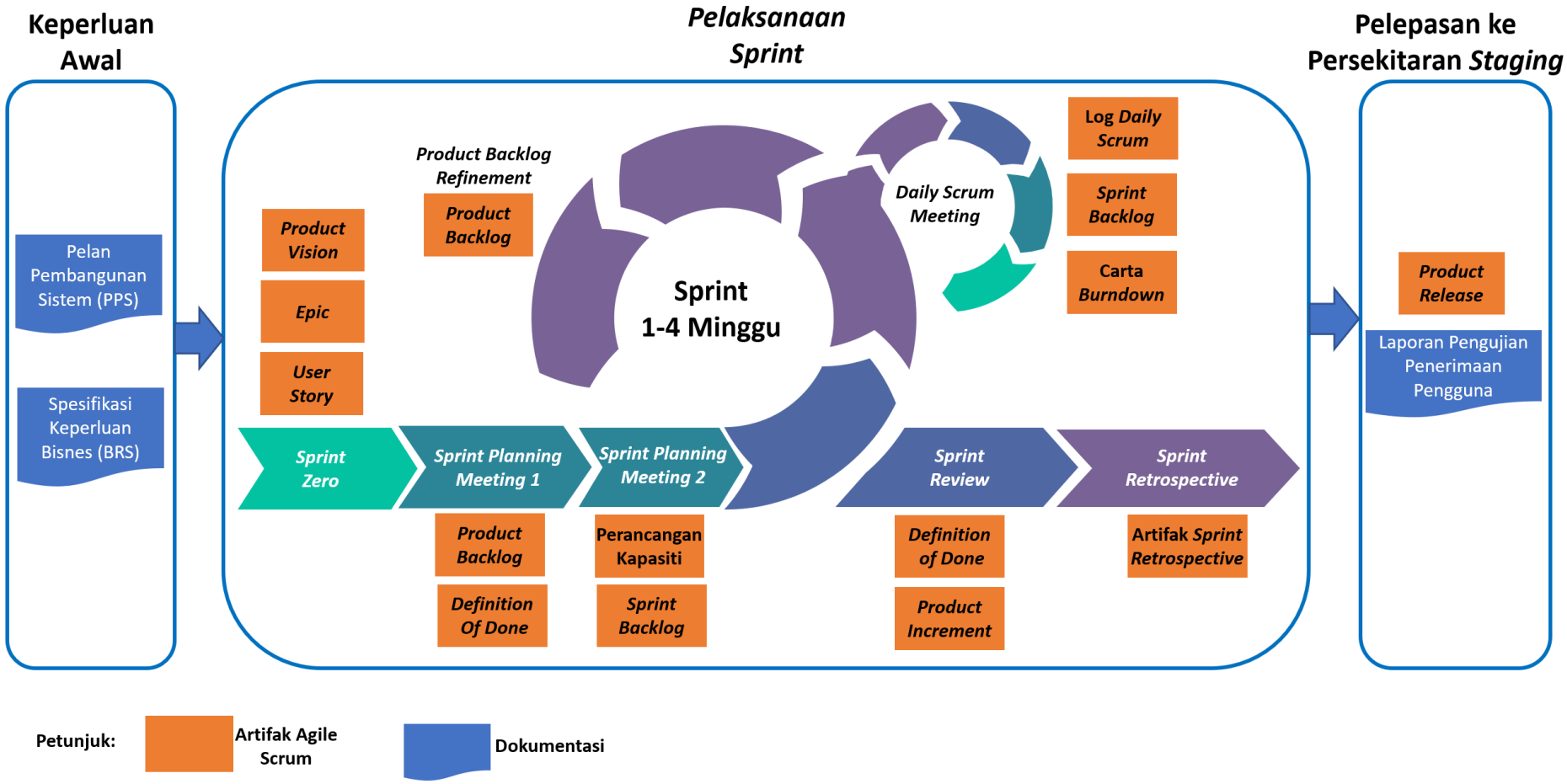
3.4. AKTIVITI *AGILE SCRUM*

Metodologi *agile scrum* terdiri daripada aktiviti berikut *iaitu Sprint Zero, Sprint Planning Meeting, Daily Scrum Meeting, Product Backlog Refinement, Sprint Review* dan *Sprint Retrospective*. Rajah 3-3 memaparkan aktiviti *scrum* yang perlu dilaksanakan pada satu *sprint*. Setiap aktiviti ini melibatkan semua ahli pasukan *scrum* iaitu pemilik produk, pasukan pembangun dan *scrum master* yang bertindak sebagai fasilitator.



Rajah 3-3: Aktiviti *Agile Scrum* dalam Satu *Sprint*

Rajah 3-4 memaparkan aktiviti *agile scrum* yang disokong oleh artifak yang dihasilkan daripada setiap aktiviti tersebut. Artifak merujuk kepada sumber informasi dan komunikasi antara pasukan *scrum*. Artifak juga membantu pasukan dalam mengurus tugasan bagi memastikan produk dapat dihasilkan mengikut keperluan pengguna dan perlu dikemas kini di sepanjang aktiviti *scrum.*



Rajah 3-4: Aktiviti dan Artifak *Agile Scrum*

Panduan ini menggunakan Sistem Tempahan Bilik Mesyuarat sebagai kajian kes untuk pelaksanaan pembangunan produk menggunakan metodologi *agile scrum* untuk memudahkan pemahaman pembaca. Aktiviti kajian kes ini melibatkan tempoh 10 hari bekerja bagi satu *sprint*. Hari pertama diperuntukkan untuk aktiviti *sprint planning meeting* pertama dan *sprint planning meeting* kedua. Manakala satu hari terakhir diperuntukkan untuk aktiviti *sprint review* dan *sprint retrospective*.

3.4.1. Aktiviti *Sprint Zero*

Sprint zero adalah aktiviti yang dilaksanakan sebagai persiapan sebelum *sprint* pertama bermula. Tempoh masa pelaksanaan aktiviti ini adalah dari satu hari hingga satu minggu mengikut kesesuaian pasukan. Tujuan *sprint zero* adalah untuk memastikan bahawa pasukan bersedia untuk memulakan pembangunan, mempunyai pemahaman yang jelas tentang tugasan yang perlu dilakukan, jadual pelaksanaan dan peranan serta tanggungjawab untuk setiap ahli pasukan.

Aktiviti *sprint zero* melibatkan perkara seperti berikut:

a. Menyiapkan persekitaran pembangunan.
b. Menentukan skop dan matlamat pembangunan produk.
c. Mengenal pasti keperluan pembangunan.
d. Menetapkan peranan dan tanggungjawab kepada ahli pasukan.

Dokumen Rujukan kepada aktiviti *sprint zero* adalah seperti berikut:

a. Dokumen BRS dan keperluan pengguna adalah daripada Hierarki Fungsi Bisnes.
b. Dokumen PPS digunakan untuk penetapan visi pembangunan produk.

Artifak hasil daripada aktiviti *sprint zero* adalah seperti berikut:

a. *Product vision*.
b. *Epic*.
c. *User story*.

Penglibatan ahli pasukan adalah seperti berikut:

a. Pemilik Produk.

b. *Scrum Master*.

c. SME Bisnes.

d. Pasukan Pembangun.

3.4.1.1. Artifak *Product Vision*

Product vision merupakan artifak yang dihasilkan oleh pemilik produk. *Product vision* memberi gambaran jelas tentang tujuan dan matlamat utama produk dibangunkan. Artifak ini bertujuan membantu pasukan *scrum* mencapai persefahaman dan jelas dengan hala tuju produk.

a. Komponen Artifak *Product Vision*

Jadual 3-1 menerangkan komponen yang membentuk artifak *product vision*.

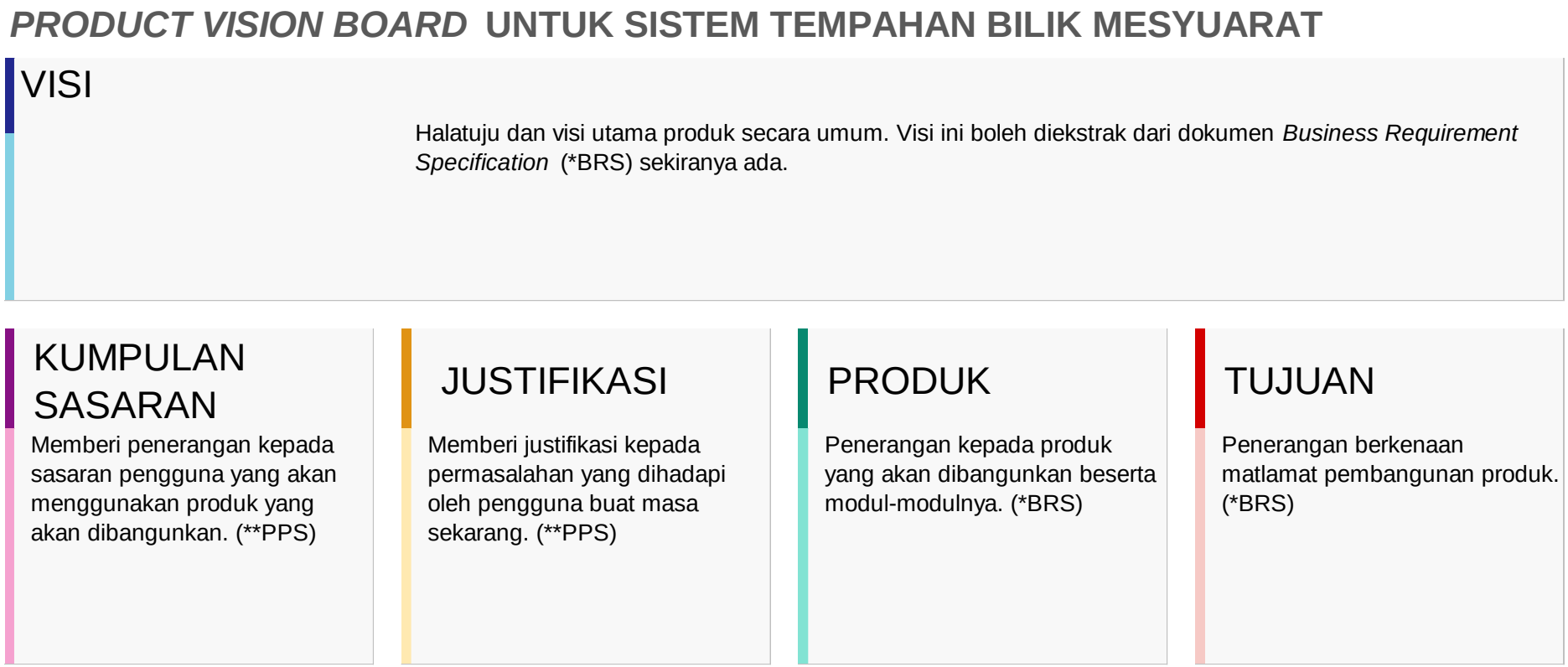
Jadual 3-1: Komponen Artifak *Product Vision*

No.	Komponen	Penerangan	Dokumen Rujukan
1.	Visi	Menerangkan hala tuju produk.	BRS
2.	Kumpulan Sasaran	Sasaran pengguna yang akan menggunakan produk.	PPS
3.	Justifikasi	Memberi justifikasi kepada permasalahan yang dihadapi oleh pengguna sebelum produk dibangunkan.	PPS
4.	Produk	Memberi penerangan kepada produk yang akan dibangunkan berserta modul-modulnya.	BRS
5.	Tujuan	Memberi penerangan berkenaan matlamat pembangunan produk dan manfaat produk kepada agensi.	BRS

Templat artifak *product vision* boleh dirujuk dalam Lampiran A-1.

b. Contoh Pengisian Artifak *Product Vision*

Rajah 3-5 memaparkan contoh pengisian artifak *product vision*.



Rajah 3-5: Contoh Artifak *Product Vision*

3.4.1.2. Artifak *Epic* dan *User Story*

Epic merupakan fungsi utama produk atau elemen besar keperluan pengguna yang perlu dibangunkan bagi mencapai matlamat keseluruhan. *Epic* seterusnya dipecahkan kepada beberapa keperluan pengguna yang lebih kecil dikenali sebagai *user story*. *Epic* dan *user story* yang dikumpul akan digunakan pada aktiviti seterusnya bagi tujuan perancangan dan pelaksanaan pembangunan produk. Oleh itu, adalah digalakkan agar kesemua *user story* disenaraikan sebelum bermulanya *sprint.*

Berikut merupakan format penulisan *user story*:

<peranan> <keperluan> <impak>

Contoh:

Peranan – Pengguna (Warga MAMPU)

Keperluan – BOLEH mendaftar profil pengguna baharu

Impak – SUPAYA boleh mendaftar masuk sistem tempahan bilik mesyuarat

Sebagai pengguna (Warga MAMPU), saya **BOLEH** mendaftar profil pengguna baharu, **SUPAYA** boleh mendaftar masuk sistem tempahan bilik mesyuarat

a. Komponen Artifak *User Story*

Jadual 3-2 menerangkan komponen artifak *user story*.

Jadual 3-2: Komponen Artifak *User Story*

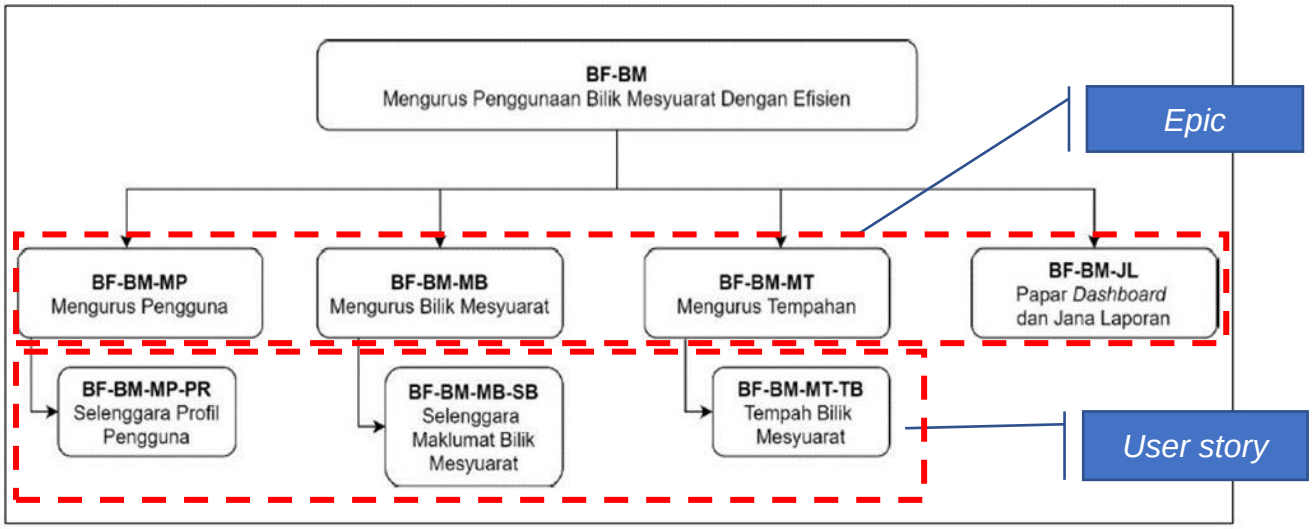
No.	Komponen	Penerangan
1.	*User story* ID	Merujuk kepada perwakilan ID unik (naming convention) untuk setiap *user story.*
2.	Peranan	Merujuk kepada jenis-jenis pengguna yang akan menggunakan produk tersebut.

No.	Komponen	Penerangan
3.	Keperluan	Keperluan produk yang ingin dibangunkan.
4.	Impak	Kesan ketara atau manfaat yang dapat digunakan hasil daripada pembangunan ciri-ciri tersebut.
5.	*Acceptance criteria*	Ciri-ciri keperluan fungsian yang perlu dipenuhi untuk melengkapkan *user story*.

Templat artifak *user story* boleh dirujuk dalam Lampiran A-2.

b. Contoh Pengisian Artifak *Epic* dan *User Story*

Hierarki Fungsi Bisnes yang terdapat pada dokumen BRS Sistem Tempahan Bilik Mesyuarat boleh dijadikan sebagai panduan untuk menterjemahkan *epic* dan *user story*. Rajah 3-6 memaparkan *epic* dan *user story* yang diekstrak daripada Hierarki Fungsi Bisnes Sistem Tempahan Bilik Mesyuarat.



Rajah 3-6: Hierarki Fungsi Bisnes Sistem Tempahan Bilik Mesyuarat

Epic yang telah dikenal pasti adalah **Mengurus Pengguna, Mengurus Bilik Mesyuarat, Mengurus Tempahan dan Papar *Dashboard*** serta **Jana Laporan**. Pemilik produk menterjemahkan keperluan pengguna ke dalam format *user story* untuk memberi kefahaman kepada pasukan *scrum*. Jadual 3-3 merupakan contoh pengisian artifak *user story*.

Jadual 3-3: Contoh Artifak *User Story*

***USER STORY* ID**	**PERANAN**	**KEPERLUAN**	**IMPAK**	***ACCEPTANCE CRITERIA***
Modul Mengurus Pengguna				
BF-BM-EP01-US01	Pengguna (warga MAMPU)	**BOLEH** mendaftar profil pengguna baharu	**SUPAYA** boleh mendaftar masuk sistem tempahan bilik mesyuarat	Semasa warga melengkapkan butiran pendaftaran perlu pastikan: i. Format nombor kad pengenalan 12 digit dimasukan dan tidak termasuk simbol '-'. ii. Hanya alamat e-mel rasmi agensi sahaja yang boleh diterima. iii. Semak dan pastikan tiada pertindihan untuk kad pengenalan dan e-emel.
BF-BM-EP01-US02	Pengguna (warga MAMPU)	**BOLEH** mengemas kini maklumat profil pengguna yang telah didaftarkan	**SUPAYA** akaun pengguna dapat disahkan dan selamat.	i. Saiz gambar dicadangkan tidak boleh melebih dari 800 x 800 piksel. ii. Format gambar yang dicadangkan adalah .jpeg dan .png sahaja.

3.4.2. Aktiviti *Sprint Planning Meeting* Pertama

Sprint planning meeting pertama merupakan aktiviti setelah bermulanya *sprint*. Aktiviti ini diadakan pada hari pertama pada setiap *sprint*. Tujuan *sprint planning meeting* pertama adalah seperti berikut:

a. Membincangkan dan menganalisis artifak yang telah dihasilkan daripada *sprint zero*.

b. Menetapkan keutamaan dan saiz *user story* menggunakan kaedah MoSCOw dan turutan Fibonacci.

c. Merancang *sprint* dan pelepasan produk.

d. Membincangkan DoD.

Artifak rujukan kepada aktiviti *sprint planning meeting* pertama adalah *product vision*, *epic* dan *user story* yang telah dihasilkan pada *Sprint Zero*.

Artifak hasil daripada aktiviti *sprint planning meeting* pertama adalah *product backlog* dan DoD*.*

Penglibatan ahli pasukan adalah seperti berikut:

a. Peranan pemilik produk adalah menjelaskan hala tuju produk dan memberi gambaran yang lebih jelas terhadap keperluan pengguna melalui lukisan, model atau *mockup.* Selain daripada itu, pemilik produk boleh menambahbaik, mengeluarkan dan memberi keutamaan semula *product backlog* di sepanjang pembangunan.

b. SME Bisnes digalakkan untuk menyertai aktiviti ini bagi membantu pasukan *scrum* menganggar dan mengutamakan *user story* yang telah disenaraikan.

c. *Scrum master* bertindak sebagai fasilitator bagi mengurus dan melancarkan sepanjang aktiviti ini berlangsung.

d. Pasukan pembangun memainkan peranan dalam menentukan *story point* bagi setiap *user story* yang disenaraikan. *Story point* merupakan skor yang menggambarkan saiz dan kerumitan *user story* tersebut.

3.4.2.1. Artifak *Product Backlog*

Product backlog merupakan senarai keperluan yang terdiri daripada ciri-ciri baharu, penambahbaikan, eksperimentasi, keperluan prestasi, sekuriti atau ralat. Artifak ini boleh ditambah baik, dikeluarkan dan diberi keutamaan semula di sepanjang pembangunan sekiranya perlu.

a. Komponen Artifak *Product Backlog*

Jadual 3-4 menerangkan komponen artifak *product backlog.*

Jadual 3-4: Komponen Artifak *Product Backlog*

No.	Komponen	Penerangan
1.	*User story* ID	Pengenalan unik ID untuk setiap *user story.*
2.	Keutamaan	Penetapan keutamaan untuk setiap *user story* berdasarkan kepentingan dan nilai kepada pengguna. Penetapan keutamaan dilakukan oleh pemilik produk dan pasukan pembangun dengan menggunakan kaedah MoSCoW.
3.	Turutan Keutamaan	Turutan keutamaan yang ditetapkan oleh pemilik produk bergantung kepada nilai keutamaan dan *story point*.
4.	*User story*	Penerangan ringkas tentang ciri atau fungsi dari perspektif pengguna yang merangkumi peranan, fungsi dan impak.
5.	*Story point*	Anggaran bagi masa, usaha, atau kompleksiti yang diperlukan untuk menyelesaikan *user story.* Kaedah turutan Fibonacci digunakan oleh pemilik produk dan pasukan pembangun pasukan untuk menetapkan saiz *user story*.
6.	Status	Status semasa *user story* tersebut sama ada Baharu, Dalam Tindakan atau Selesai.

Templat artifak *product backlog* boleh dirujuk dalam Lampiran A-3.

Keutamaan setiap *user story* boleh ditentukan dengan menggunakan kaedah MoSCow untuk mengurus keperluan pengguna.

Dalam pembangunan sistem aplikasi, kaedah MoSCoW merujuk kepada pendekatan menetapkan keutamaan keperluan pengguna berdasarkan pembahagian keperluan kepada empat kategori iaitu *must have, should have,*

could have dan *will not have.* Jadual 3-5 memberi penerangan setiap kategori berdasarkan kaedah MosCow.

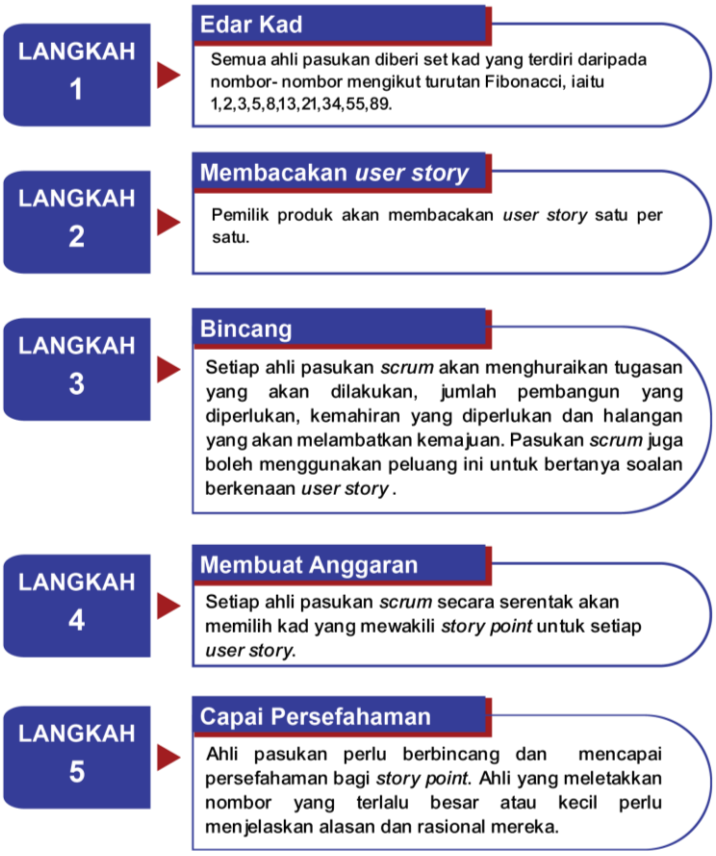
Jadual 3-5: Kategori Penetapan Keutamaan *User story*

No.	Kategori	Penerangan
1.	**Perlu Ada (Must Have)**	Keperluan yang wajib ada dan jika tiada produk tidak berfungsi
2.	**Patut Ada (Should Have)**	Keperluan yang kurang penting namun akan memberi impak yang tinggi
3.	**Mungkin Ada (Could Have)**	Keperluan yang kurang penting dan akan memberi impak yang rendah
4.	**Tidak Sepatutnya Ada (Will Not Have)**	Keperluan yang tidak menjadi keutamaan dalam tempoh pembangunan

Penentuan *story point* boleh ditentukan menggunakan kaedah ***playing poker.*** *Playing poker* adalah teknik yang digunakan oleh pasukan untuk menganggarkan nilai kompleksiti bagi setiap *user story*.

Turutan Fibonacci digunakan sebagai nilai kompleksiti untuk menganggarkan usaha yang diperlukan dalam menyelesaikan setiap *user story*. Turutan Fibonacci adalah siri nombor di mana setiap nombor adalah jumlah dua nombor sebelumnya, bermula dengan 0 dan 1. Turutan tersebut adalah 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 dan seterusnya dengan hasilnya membawa kepada kenaikan konsisten 60%. Adalah digalakkan skor tertinggi *story point* dihadkan sehingga 89 kerana nilai yang lebih besar akan menyebabkan anggaran saiz dan kompleksiti *user story* menjadi kurang tepat.

Proses ini membantu pasukan pembangun menganggar tempoh masa pembangunan dengan lebih tepat, mencapai persetujuan dan merancang tugasan dengan lebih strategik. Jika terdapat perbezaan pendapat antara ahli pasukan berkenaan *story point*, pemilik produk selaku ketua pasukan perlu meminta setiap ahli pasukan untuk mengusulkan pendapat supaya persefahaman dapat dicapai dalam penentuan *story point*. Rajah 3-7 menerangkan langkah-langkah kaedah *playing poker*.



Rajah 3-7: Kaedah *Playing Poker*

b. Contoh Pengisian Artifak *Product Backlog*

Jadual 3-6 memaparkan contoh pengisian artifak *product backlog*.

Jadual 3-6: Contoh Artifak *Product Backlog*

PRODUCT BACKLOG ID	KEUTAMAAN	TURUTAN KEUTAMAAN	*USER STORY*		*STORY POINT*	*SPRINT NO*	STATUS
***Epic* 1 : Pengurusan Pengguna**							
BF-BM-EP01-PB01	Mesti Ada	1	Pengguna (warga MAMPU)	**BOLEH** mendaftar profil pengguna baru	**SUPAYA** boleh mendaftar masuk sistem tempahan bilik mesyuarat	3	Sprint 1	Dalam Tindakan
BF-BM-EP01-PB02	Mesti Ada	2	Pengguna (warga MAMPU)	**BOLEH** mengemaskini maklumat profil pengguna yang telah didaftarkan	**SUPAYA** akaun dapat disahkan dan selamat.	5	Sprint 1	Baharu
***Epic* 2 : Pengurusan Bilik Mesyuarat**							
BF-BM-EP02-PB01	Mesti Ada	4	Pentadbir Bilik Mesyuarat	**BOLEH** menyelenggara maklumat bilik mesyuarat dengan mewujudkan rekod bilik yang baharu atau mengemaskini rekod sedia ada.	**SUPAYA** mendapatkan maklumat bilik mesyuarat yang tepat.	13		
BF-BM-EP02-PB02	Mesti Ada	3	Pentadbir Bilik Mesyuarat	**BOLEH** menyemak maklum balas penggunaan bilik mesyuarat yang diterima daripada pengguna (warga Agensi)	**SUPAYA** mendapatkan sumber rujukan untuk melaporkan kerosakan dan perlukan pembaikan yang berkaitan.	8	Sprint 1	

3.4.2.2. Artifak *Definition of Done* (DoD)

DoD merupakan senarai kriteria yang mesti dipenuhi untuk item *product backlog* dianggap lengkap dan produk sedia untuk pelepasan pada akhir *sprint.* DoD harus ditakrifkan dengan jelas, difahami dan dipersetujui oleh semua ahli pasukan *scrum* untuk memastikan produk yang dibangunkan adalah berkualiti tinggi dan memenuhi jangkaan pemilik produk.

a. Komponen Artifak *Definition of Done*

Artifak DoD terdiri daripada komponen seperti keperluan fungsian, keperluan bukan fungsian, penempatan dan dokumentasi. Jadual 3-7 menerangkan setiap komponen yang membentuk artifak DoD.

Jadual 3-7: Komponen Artifak *Definition of Done*

No.	Komponen	Penerangan
1.	Keperluan Fungsian	Senarai semak yang berkaitan pengujian fungsian.
2.	Keperluan Bukan Fungsian	Senarai semak yang berkaitan pengujian bukan fungsian.
3.	Penempatan	Persekitaran penempatan (deployment) produk sama ada pembangunan, *staging* atau produksi.
4.	Dokumentasi	Jenis dokumentasi dan dokumen serahan yang perlu disediakan semasa *sprint.*
5.	Status	Senarai semak telah diselesaikan atau tidak.
6.	Catatan	Penerangan tambahan mengenai komponen senarai semak seperti pelaksanaan pada *sprint* ke berapa.

Templat artifak DoD boleh dirujuk dalam Lampiran A-4.

b. Contoh Pengisian Artifak *Definition of Done*

Jadual 3-8 merupakan DoD yang telah dikenal pasti dan dipersetujui oleh pasukan *scrum*.

Jadual 3-8: Contoh Senarai *Definition of Done*

Komponen	Senarai Semak	Status	Catatan
Pengujian Keperluan Fungsian	1. Kod sumber selesai dibangunkan		
	2. Kod sumber disemak		
	3. Kod sumber digabungkan (merged)		
	4. Melepasi pengujian unit		
	5. Melepasi pengujian integrasi		
	6. Melepasi pengujian sistem dan memenuhi *acceptance criteria* bagi setiap *user story*		
	7. Melepasi pengujian penerimaan pengguna		
Pengujian Keperluan Bukan Fungsian	1. Melepasi pengujian kualiti kod		
	2. Melepasi pengujian SAST		
	3. Melepasi pengujian prestasi		
Penempatan	1. Penempatan ke persekitaran *staging* pada *sprint* yang melibatkan *product release* sahaja		
Dokumentasi	1. Artifak *product vision*		
	2. Dokumen Laporan UAT		
	3. Artifak *sprint backlog*		
	4. Artifak Carta *burndown*		
	5. Artifak Perancangan Kapasiti		
	6. Artifak log *daily scrum*		
	7. Artifak *sprint retrospective*		

3.4.3. Aktiviti *Sprint Planning Meeting* Kedua

Sprint planning meeting kedua merupakan aktiviti seterusnya selepas *sprint planning meeting* pertama. Aktiviti ini dilaksanakan pada hari yang sama dengan *sprint planning meeting* pertama. Tempoh masa pelaksanaan aktiviti ini adalah dari dua hingga empat jam mengikut kesesuaian pasukan.

Tujuan aktiviti *sprint planning meeting* kedua adalah:

a. Melakukan perancangan kapasiti.

b. Menyediakan artifak *sprint backlog.*

Aktiviti *sprint planning meeting* kedua meliputi perkara seperti berikut:

a. Pasukan pembangun akan berkongsi ketersediaan mereka sepanjang *sprint*.

b. Pasukan pembangun akan menyemak *user story* yang telah dikenal pasti semasa *sprint planning meeting* pertama dan memutuskan cara pelaksanaannya.

c. Membahagikan *user story* tersebut kepada tugasan kecil dan membuat anggaran kompleksiti, masa dan usaha yang diperlukan untuk menyelesaikan setiap *user story*.

d. Mengenal pasti sebarang pergantungan antara tugas.

e. Menentukan susunan tugasan berdasarkan keutamaannya.

Artifak Rujukan kepada aktiviti *Sprint Planning Meeting* Kedua adalah *product backlog* dan DoD yang telah dihasilkan pada *Sprint Planning Meeting* Pertama.

Artifak Hasil daripada aktiviti *Sprint Planning Meeting* Kedua adalah perancangan kapasiti dan artifak *sprint backlog*.

Penglibatan ahli pasukan adalah seperti berikut:

a. Pemilik produk bertanggungjawab untuk memastikan pasukan pembangun jelas tentang item *product backlog* yang perlu dilengkapkan dan merekodkan perkara-perkara yang dibincangkan ke dalam *sprint backlog*,

b. *Scrum master* bertindak sebagai fasilitator bagi mengurus dan melancarkan sepanjang aktiviti ini berlangsung.

c. Pasukan pembangun berperanan untuk mengenal pasti tugasan-tugasan kecil yang perlu dilaksanakan serta ketersediaan mereka semasa *sprint*.

3.4.3.1. Artifak Perancangan Kapasiti

Kapasiti pasukan dikira mengikut ketersediaan ahli pasukan dalam setiap *sprint*. Aktiviti ini melibatkan menganalisis kapasiti pasukan, bilangan ahli pasukan, ketersediaan dan kemahiran ahli pasukan. Maklumat ini digunakan untuk merancang jumlah tugas yang boleh dilaksanakan semasa *sprint*. Namun, perancangan kapasiti ini perlu mengambil kira hari ahli pasukan itu bercuti, jumlah hari menghadiri latihan, komitmen pada tugasan lain, hal peribadi dan lain-lain.

a. Komponen Artifak Perancangan Kapasiti

Jadual 3-9 menerangkan artifak perancangan kapasiti.

Jadual 3-9: Komponen Artifak Perancangan Kapasiti

No.	Komponen	Penerangan
1.	Nama	Nama ahli pasukan yang telah ditugaskan untuk menyelesaikan keperluan pengguna.
2.	Jumlah Hari Bekerja *sprint* semasa (Hari)	Bilangan hari bekerja setiap ahli pasukan untuk *sprint* semasa. Perlu mengambil kira hari pegawai itu bercuti, menghadiri latihan, hal peribadi dan cuti umum yang dirancang.
3.	Jumlah Masa Bekerja Sehari (Jam)	Bilangan jam bekerja yang boleh diperuntukkan oleh ahli pasukan tersebut dalam sehari untuk menyiapkan tugasan.
4.	Jumlah Masa yang Diperuntukkan (Jam)	Jumlah jam bekerja keseluruhan yang ada untuk setiap ahli pasukan.

No.	Komponen	Penerangan
		Jumlah Masa yang Diperuntukkan = Jumlah Hari Bekerja setiap *sprint* x Jumlah Waktu Bekerja Sehari
5.	Jumlah Masa Keseluruhan (Jam)	Merujuk kepada penambahan keseluruhan jumlah masa yang diperuntukkan dalam *sprint* semasa.
6.	5-10% *interrupt buffer* (Jam)	*Interrupt buffer* 5% hingga 10% adalah masa kerja untuk isu luar jangka yang telah diambil kira daripada Jumlah Masa Keseluruhan yang diperuntukkan bagi setiap *sprint* dan perlu ditolak semasa kiraan Jumlah Masa Bersih.
7.	5% -10% *Product Backlog Refinement* (Jam)	5% hingga 10% masa yang telah diambil kira bagi memperhalusi *product backlog*.
8.	Jumlah Masa Bersih (Jam)	Jumlah Masa Keseluruhan ditolak dengan masa *interrupt buffer* dan *product backlog refinement.*

Templat artifak perancangan kapasiti boleh dirujuk dalam Lampiran A-4.

b. Contoh Pengisian Artifak Perancangan Kapasiti

Contoh artifak perancangan kapasiti yang telah diisi berdasarkan lima orang ahli pasukan pembangun bagi 10 hari bekerja adalah seperti Jadual 3-10.

Jadual 3-10: Jadual Pengiraan Perancangan Kapasiti *Sprint* 1

Perancangan Kapasiti *Sprint* 1			
Tempoh *Sprint*, 2 minggu = 10 hari bekerja			
Nama	Jumlah Hari Bekerja Semasa *Sprint* (Hari) (a)	Jumlah Masa Bekerja Sehari (Jam) (b)	Jumlah Masa yang Diperuntukkan (Jam) (a*b)
Ali	8	5	40
Azhim	9	5	45
Ahmad	7	5	35
Raziman	6	2	12
Fauzi	4	4.5	18
Jumlah Masa Keseluruhan (Jam) (c)			150
(-)	5-10% *Interrupt Buffer* (Jam) (d)		15
(-)	5-10% *PB Refinement* (Jam) (e)		15
Jumlah Masa Bersih (Jam) = (c - (d + e))			120

Berdasarkan perancangan kapasiti yang dihasilkan, jumlah masa bersih yang diperuntukkan oleh ahli pasukan pembangun bagi *sprint* 1 adalah sebanyak 120 jam.

3.4.3.2. Artifak *Sprint Backlog*

Sprint backlog adalah artifak yang mengandungi senarai tugasan yang perlu diselesaikan oleh pasukan pembangun semasa *sprint. Sprint backlog* bersifat dinamik yang boleh berkembang di sepanjang *sprint* apabila tugasan ditambah, dialih keluar atau diubah suai.

Sprint backlog berfungsi sebagai panduan yang boleh digunakan pasukan pembangun untuk membantu mereka kekal fokus kepada matlamat *sprint.* Artifak ini juga membantu pasukan pembangun mengurus tugas yang sedang dilaksana dan memantau kemajuan tugas mereka sepanjang *sprint*.

a. Komponen Artifak *Sprint Backlog*

Jadual 3-11 menerangkan artifak *sprint backlog.*

Jadual 3-11: Komponen Artifak *Sprint Backlog*

No.	Komponen	Penerangan
1.	*User Story* ID	Mewakili ID unik untuk setiap *product backlog.*
2.	*Product Backlog Item*	Senarai *user story* yang telah dikenal pasti dalam setiap *sprint.*
3.	ID Tugasan	Mewakili ID unik untuk setiap tugasan *sprint.*
4.	Tugasan	Tugasan bagi setiap *sprint backlog* yang telah dikenal pasti oleh pasukan *scrum.*
5.	Ahli Pasukan	Nama ahli pasukan yang ditugaskan untuk menyelesaikan tugasan.
6.	Status	Status semasa tugasan tersebut sama ada Baharu, Dalam Tindakan atau Selesai.
7.	*Story Point*	Anggaran saiz yang ditetapkan oleh pasukan *scrum* menerusi kaedah turutan fibonacci.
8.	Anggaran Masa (Jam)	Masa yang diperlukan untuk menyelesaikan tugasan tersebut.
9.	Masa Sebenar (jam)	Jumlah jam yang telah digunakan untuk menyelesaikan tugasan setiap hari.

10.	*Remaining Effort*	Jumlah masa yang berbaki untuk menyelesaikan tugasan setiap hari. ***Remaining effort* = Jumlah Anggaran Masa (Hari Sebelumnya) – Masa Sebenar**
11.	*Ideal Trend*	Merujuk kepada trend penurunan ideal untuk mencapai *zero effort remaining* pada penghujung *sprint.* ***Ideal Trend* = Jumlah Anggaran Masa – (Jumlah Anggaran Masa / Jumlah Hari *Sprint* * Hari)**

Templat artifak *sprint backlog* boleh dirujuk dalam Lampiran A-5.

b. Contoh Pengisian Artifak *Sprint Backlog*

Jadual 3-12 memaparkan contoh pengisian artifak *sprint backlog*. Terdapat dua *epic* dan tiga *product backlog* yang disenaraikan dalam *sprint* 1. Setiap *product backlog* dipecahkan kepada beberapa tugasan. Setiap pembangun bertanggungjawab untuk menyelesaikan tugasan tersebut berdasarkan anggaran masa yang ditetapkan. Hari 1 melibatkan aktiviti s*print planning meeting* pertama dan kedua manakala Hari 10 melibatkan aktiviti *sprint review* dan *sprint retrospective*. Pembangun akan mula membangunkan produk bermula dari Hari 2 sehingga Hari 9.

Jadual 3-12: Contoh *Sprint Backlog* Hari 1 bagi *Sprint* 1

PRODUCT BACKLOG ID	*PRODUCT BACKLOG ITEM*	ID TUGASAN	TUGASAN	AHLI PASUKAN	STATUS	*STORY POINT*	ANGGARAN MASA (JAM)	HARI 1	HARI 2	HARI 3	HARI 4	HARI 5	HARI 6	HARI 7	HARI 8	HARI 9	HARI 10
***Epic* 1: Pengurusan Pengguna**																	
BF-BM-EP01-PB01	**Pengguna (warga MAMPU)BOLEH mendaftar profil pengguna baru SUPAYA boleh mendaftar masuk sistem tempahan bilik mesyuarat**	**US-001-ST01**	***new table format***	**Azhim**	**Selesai**	**3**	**8**										
		US-001-ST02	***new table UI***	**Ahmad**	**Selesai**		**4**										
		US-001-ST03	***implement new db format***	**Fauzi**	**Selesai**		**6**										
		US-001-ST04	***template setup***	**Raziman**	**Selesai**		**2**										
		US-001-ST05	***creation of auto approve***	**Ali**	**Selesai**		**4**										
BF-BM-EP01-PB02	**Pengguna (warga MAMPU)BOLEH mengemaskini maklumat profil pengguna yang telah didaftarkan SUPAYA akaun dapat disahkan dan selamat.**	**US-002-ST01**	***Install contact page.***	**Ali**	**Selesai**	**5**	**5**										
		US-002-ST02	***new table UI***	**Raziman**	**Selesai**		**6**										
		US-002-ST03	***implement new db format***	**Ahmad**	**Selesai**		**8**										
		US-002-ST04	***template setup***	**Azhim**	**Selesai**		**8**										
		US-002-ST05	***export cofiguration***	**Ahmad**	**Selesai**		**7**										
		US-002-ST06	***creation of auto approve***	**Ali**	**Selesai**		**8**										
		US-002-ST07	***audit trail***	**Azhim**	**Selesai**		**5**										
		US-002-ST08	***creation of new table***	**Raziman**	**Selesai**		**5**										
***Epic* 2: Pengurusan Bilik Mesyuarat**																	
BF-BM-EP02-PB02	**Pentadbir Bilik Mesyuarat BOLEH menyemak maklum balas penggunaan bilik mesyuarat yang diterima daripada pengguna (warga Agensi)SUPAYA mendapatkan sumber rujukan untuk melaporkan kerosakan dan perlukan pembaikan yang berkaitan.**	**US-004-ST01**	***Install contact page***	**Azhim**	**Selesai**	**8**	**8**										
		US-004-ST02	***new table UI***	**Raziman**	**Selesai**		**6**										
		US-004-ST03	***implement new db format***	**Ali**	**Selesai**		**7**										
		US-004-ST04	***template setup***	**Ahmad**	**Selesai**		**7**										
		US-004-ST05	***table implementation***	**Ali**	**Selesai**		**6**										
		US-004-ST06	***export cofiguration***	**Fauzi**	**Selesai**		**5**										
		US-004-ST07	***creation of auto approve***	**Raziman**	**Selesai**		**6**										
					JUMLAH *STORY POINT*	16											
					MASA SEBENAR (Jam) (a)		**0**										
					***REMAINING EFFORT* (Jam) (b) (b=b hari sebelum – a hari semasa)**		**121**										
					***IDEAL TREND* (jam) (c) (c = c rancang – (c rancang/jumlah hari x hari semasa))**		**121**										

3.4.4. Aktiviti *Daily Scrum Meeting*

Aktiviti *daily scrum meeting* diadakan selepas *sprint planning meeting* kedua. Aktiviti ini merupakan aktiviti harian yang diadakan maksimum 15 minit setiap hari. Menerusi aktiviti ini, setiap ahli pasukan *scrum* berpeluang untuk berkongsi kemajuan masing-masing, mengkoordinasi kerja dan melaporkan perkara berikut:

a. Tugasan yang telah diselesaikan hari sebelum.
b. Tugasan yang bakal diselesaikan hari ini.
c. Isu yang dihadapi dalam menyelesaikan tugasan (sekiranya ada).

Artifak Rujukan kepada aktiviti *daily scrum meeting* adalah *sprint backlog* dan carta *burndown* hari sebelumnya.

Artifak Hasil daripada aktiviti *daily scrum meeting* adalah *sprint backlog* dan carta *burndown* yang dikemas kini setiap hari.

Penglibatan ahli pasukan adalah seperti berikut:

a. Pemilik produk bertanggungjawab untuk merekod perkara-perkara yang dikemas kini pasukan pembangun,
b. *Scrum master* bertindak sebagai fasilitator bagi mengurus dan melancarkan sepanjang aktiviti ini berlangsung di samping membantu mengkoordinasikan isu yang dihadapi oleh pasukan pembangun. Sekiranya isu tidak dapat diselesaikan dan berlanjutan, mesyuarat lanjutan bersama SME Bisnes dan pemegang taruh boleh diadakan.
c. Pasukan pembangun berperanan untuk berkongsi status kemajuan kerja masing-masing.

3.4.4.1. Artifak Log *Daily Scrum*

Log *daily scrum* merekodkan kemajuan pembangunan dan isu yang menghalang perancangan untuk hari seterusnya. Log ini digunakan sebagai rujukan kepada pasukan pembangun untuk menjejaki kemajuan mereka dan mengenal pasti mana-mana tugasan yang memerlukan perhatian.

a. Komponen Artifak Log *Daily Scrum*

Jadual 3-13 menerangkan komponen artifak log *daily scrum.*

Jadual 3-13: Log *Daily Scrum*

No.	Komponen	Penerangan
1.	Hari	Mewakili hari dalam *sprint.*
2.	Ahli Pasukan	Nama ahli pasukan yang ditugaskan untuk menyelesaikan tugasan.
3.	Kemas kini Kemajuan	i. Tugasan yang telah diselesaikan hari sebelum. ii. Tugasan yang bakal diselesaikan hari ini. iii. Isu yang dihadapi dalam menyelesaikan tugasan (sekiranya ada).

Templat artifak log *daily scrum* boleh dirujuk dalam Lampiran A-5.

b. Contoh Pengisian Log *Daily Scrum*

Jadual 3-14 menunjukkan contoh artifak log *daily scrum* yang telah diisi bagi seorang pembangun untuk tempoh lima hari bekerja.

Jadual 3-14: Log *Daily Scrum* bagi Tempoh Lima Hari

LOG *DAILY SCRUM*						
Ahli Pasukan	Kemajuan	Hari 1 Isnin	Hari 2 Selasa	Hari 3 Rabu	Hari 4 Khamis	Hari 5 Jumaat
Ali	Tugasan yang telah diselesaikan pada hari sebelum	*Sprint Planning Meeting* 1 & 2	sambungan penyediaan fungsi *auto approve* (30% siap)	bercuti	bercuti	penyediaan fungsian *auto approve* (80% siap)
	Tugasan yang bakal diselesaikan pada hari ini	penyediaan fungsian *auto approve*	sambungan penyediaan fungsian *auto approve*	bercuti	sambungan penyediaan fungsian *auto approve* (50% siap)	sambungan penyediaan fungsian *auto approve*
	Isu yang dihadapi semasa menyelesaikan tugasan (sekiranya ada)	memerlukan maklumat lanjut berkenaan db *table auto approve*	tiada	bercuti	tiada	Tiada

3.4.4.2. Pengemaskinian Artifak *Sprint Backlog*

Artifak *sprint backlog* perlu dikemas kini bagi mencatatkan masa yang diperuntukkan oleh pasukan pembangun bagi menyelesaikan tugasan yang diberikan. Masa yang diperuntukkan bagi setiap tugasan akan dicatatkan pada artifak ini. Contoh pengemaskinian artifak *sprint backlog* bagi tempoh 10 hari bekerja adalah seperti Jadual 3-15.

Jadual 3-15: Contoh Artifak *Sprint Backlog* sehingga Hari 10

PRODUCT BACKLOG ID	*PRODUCT BACKLOG ITEM*	ID TUGASAN	TUGASAN	AHLI PASUKAN	STATUS	*STORY POINT*	ANGGARAN MASA (JAM)	HARI 1	HARI 2	HARI 3	HARI 4	HARI 5	HARI 6	HARI 7	HARI 8	HARI 9	HARI 10
***Epic* 1: Pengurusan Pengguna**																	
BF-BM-EP01-PB01	**Pengguna (warga MAMPU)BOLEH mendaftar profil pengguna baru SUPAYA boleh mendaftar masuk sistem tempahan bilik mesyuarat**	**US-001-ST01**	***new table format***	**Azhim**	**Selesai**	**3**	**8**		3	0	2	2	1	0	0	0	
		US-001-ST02	***new table UI***	**Ahmad**	**Selesai**		**4**		3	1	0	0	0	0	0	0	
		US-001-ST03	***implement new db format***	**Fauzi**	**Selesai**		**6**		2	0	0	4	0	0	0	0	
		US-001-ST04	***template setup***	**Raziman**	**Selesai**		**2**		0	0	2	1	0	0	0	0	
		US-001-ST05	***creation of auto approve***	**Ali**	**Selesai**		**4**		2	3	0	0	0	0	0	0	
BF-BM-EP01-PB02	**Pengguna (warga MAMPU)BOLEH mengemaskini maklumat profil pengguna yang telah didaftarkan SUPAYA akaun dapat disahkan dan selamat.**	**US-002-ST01**	***Install contact page.***	**Ali**	**Selesai**	**5**	**5**		0	3	0	3	0	0	0	0	
		US-002-ST02	***new table UI***	**Raziman**	**Selesai**		**6**		0	0	0	2	2	0	0	2	
		US-002-ST03	***implement new db format***	**Ahmad**	**Selesai**		**8**		0	2	3	0	3	1	0	0	
		US-002-ST04	***template setup***	**Azhim**	**Selesai**		**8**		0	0	4	2	3	0	0	0	
		US-002-ST05	***export cofiguration***	**Ahmad**	**Selesai**		**7**		0	0	0	0	4	2	0	0	
		US-002-ST06	***creation of auto approve***	**Ali**	**Selesai**		**8**		0	0	0	0	4	4	1	0	
		US-002-ST07	***audit trail***	**Azhim**	**Selesai**		**5**		0	0	0	1	2	3	1	0	
		US-002-ST08	***creation of new table***	**Raziman**	**Selesai**		**5**		0	0	0	0	2	0	0	2	
***Epic* 2: Pengurusan Bilik Mesyuarat**																	
BF-BM-EP02-PB02	**Pentadbir Bilik Mesyuarat BOLEH menyemak maklum balas penggunaan bilik mesyuarat yang diterima daripada pengguna (warga Agensi)SUPAYA mendapatkan sumber rujukan untuk melaporkan kerosakan dan perlukan pembaikan yang berkaitan.**	**US-004-ST01**	***Install contact page***	**Azhim**	**Selesai**	**8**	**8**		0	0	0	0	0	0	2	3	
		US-004-ST02	***new table UI***	**Raziman**	**Selesai**		**6**		0	0	0	0	0	0	0	3	
		US-004-ST03	***implement new db format***	**Ali**	**Selesai**		**7**		1	0	0	0	0	2	3	1	
		US-004-ST04	***template setup***	**Ahmad**	**Selesai**		**7**		0	2	0	0	0	2	0	0	
		US-004-ST05	***table implementation***	**Ali**	**Selesai**		**6**		3	0	0	0	0	0	3	1	
		US-004-ST06	***export cofiguration***	**Fauzi**	**Selesai**		**5**		0	0	0	2	0	3	0	3	
		US-004-ST07	***creation of auto approve***	**Raziman**	**Selesai**		**6**		0	0	0	3	2	0	0	0	
					JUMLAH *STORY POINT*	16											
					MASA SEBENAR (Jam) (a)		**121**		**14**	**11**	**11**	**20**	**23**	**17**	**10**	**15**	
					***REMAINING EFFORT* (Jam) (b) (b=b hari sebelum – a hari semasa)**		**121**		**107**	**96**	**85**	**65**	**42**	**25**	**15**	**0**	
					***IDEAL TREND* (jam) (c) (c = c rancang – (c rancang/jumlah hari x hari semasa))**		**121**		**105.875**	**90.75**	**75.625**	**60.5**	**45.375**	**30.25**	**15.125**	**0**	

3.4.4.3. Artifak Carta *Burndown*

Carta *burndown* merupakan *tool* visual yang dijana hasil daripada pengemaskinian *sprint backlog* yang dilaksanakan setiap hari untuk menjejak kemajuan *sprint.* Carta ini berbentuk menurun dan menuju kepada *zero effort remaining* pada akhir *sprint.*

a. Komponen Artifak Carta *Burndown*

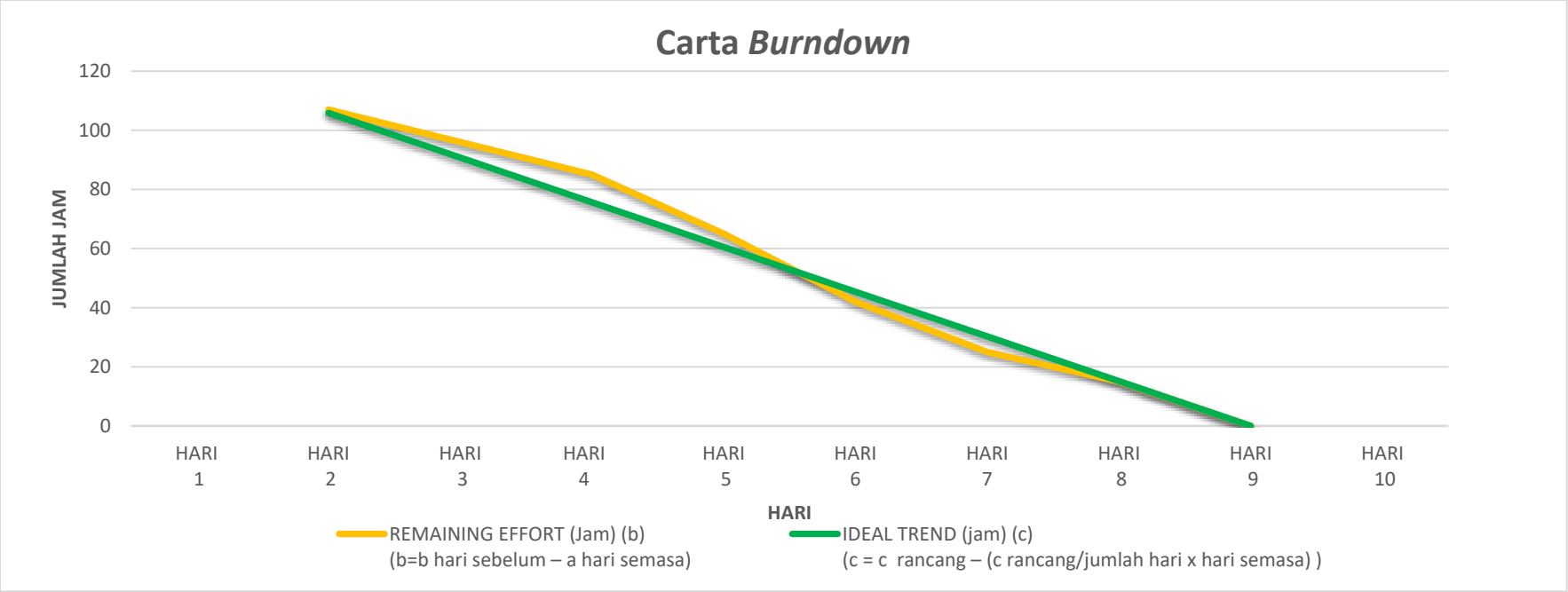
Jadual 3-16 menerangkan komponen yang terdapat berdasarkan artifak carta *burndown*.

Jadual 3-16: Komponen Carta *Burndown*

No.	Elemen	Penerangan
1.	Paksi Menegak (Paksi-Y)	Jumlah jam dalam *sprint.*
2.	Paksi Mendatar (Paksi-X)	Hari dalam *sprint.*
3.	*Ideal Trend*	Trend penurunan ideal untuk mencapai *zero effort remaining* pada penghujung *sprint.*
4.	*Remaining Effort*	Jumlah masa (jam) yang berbaki untuk menyelesaikan tugasan setiap hari.

b. Contoh Pengisian Artifak Carta *Burndown*

Rajah 3-8 menunjukkan contoh artifak carta *burndown* yang dihasilkan daripada artifak *sprint backlog* yang telah diisi bagi tempoh 10 hari bekerja.



	Jumlah Jam	HARI 1	HARI 2	HARI 3	HARI 4	HARI 5	HARI 6	HARI 7	HARI 8	HARI 9	HARI 10
REMAINING EFFORT (Jam) (b) (b=b hari sebelum – a hari semasa)	**121**		**107**	**96**	**85**	**65**	**42**	**25**	**15**	**0**	
IDEAL TREND (jam) (c) (c = c rancang – (c rancang/jumlah hari x hari semasa))	**121**		**105.875**	**90.75**	**75.625**	**60.5**	**45.375**	**30.25**	**15.125**	**0**	

Rajah 3-8: Carta *Burndown* sehingga Hari ke 10

3.4.5. Aktiviti *Product Backlog Refinement*

Product backlog refinement adalah aktiviti menyemak tugasan dalam senarai *product backlog,* menilai keutamaan *user story* berdasarkan nilai kepada pengguna dan kompleksiti, serta mengemas kini *product backlog.* Ini bertujuan bagi memastikan item tersebut ditakrifkan dengan baik, difahami, relevan dan memenuhi keperluan pengguna. Panduan *Scrum* mencadangkan bahawa tempoh aktiviti ini dilaksanakan tidak melebihi 10% daripada kapasiti pasukan pembangun[2].

Aktiviti yang dilaksanakan semasa sesi *product backlog refinement* adalah seperti berikut:

a. Menyemak *produk backlog* untuk memastikan item adalah terkini dan relevan.
b. Menambah *user story* baharu ke dalam *produk backlog* berdasarkan maklum balas pengguna atau keperluan baru.
c. Menganggarkan masa, usaha, atau kompleksiti yang diperlukan untuk menyelesaikan setiap *user story* dalam *produk backlog*. Ini membantu pasukan merancang dan mengutamakan setiap *user story*.
d. Membahagikan *user story* yang lebih besar menjadi tugasan yang lebih kecil dan mudah diuruskan. Ini membantu memastikan setiap *user story* adalah difahami dengan baik dan boleh diselesaikan dalam satu *sprint*.
e. Mengemas kini keperluan, butiran item dan menambah kriteria penerimaan setiap *user story*.
f. Mengutamakan *user story* dalam *produk backlog* berdasarkan nilai kepada pengguna, maklum balas dan faktor lain.
g. Mengemas kini status setiap *user story* dalam *produk backlog* berdasarkan kemajuan, perubahan, atau maklum balas.

Artifak Rujukan kepada aktiviti *product backlog refinement* adalah *product backlog*, *sprint backlog* dan DoD.

[2] Product Backlog Refinement Time. (n.d.). Scrum.org. Diakses pada April 11, 2023, from https://www.scrum.org/

Artifak Hasil daripada aktiviti *product backlog refinement* adalah pengemaskinian artifak *product backlog* yang mengandungi perkara berikut:

a. Item baharu yang ditambah ke dalam *produk backlog.*

b. Pengemaskinian yang dibuat pada item sedia ada.

c. Keutamaan yang ditetapkan pada setiap item.

d. Anggaran *story point* yang diberikan pada setiap item.

Penglibatan ahli pasukan adalah seperti berikut:

a. Pemilik produk bertanggungjawab untuk merekod, menyemak dan menetapkan keutamaan keatas senarai item *product backlog.*

b. Pasukan pembangun perlu bekerja sama dengan pemilik produk dengan memberikan input dan maklum balas kepada perkara berikut:

    i. Memahami keutamaan item yang ditetapkan.

    ii. Memberikan anggaran *story point* untuk setiap item.

    iii. Memberikan maklum balas untuk memastikan *product backlog* sentiasa dikemas kini.

c. *Scrum master* bertindak sebagai fasilitator bagi mengurus dan melancarkan sepanjang aktiviti ini berlangsung.

Product backlog refinement melibatkan pasukan *scrum* sahaja bagi memastikan ahli pasukan fokus dan mengelakkan perbincangan yang tidak berkaitan. Walau bagaimanapun, SME Bisnes boleh dijemput jika boleh memberikan input dan penyelesaian yang diperlukan oleh pasukan.

a. Pengemaskinian Artifak *Product Backlog*

Senarai *product backlog* pada Jadual 3-17 digunakan semasa sesi *product backlog refinement* sebagai artifak perbincangan. Lajur Catatan *Product Backlog Refinement* digunakan sebagai penerangan mengenai kemaskini yang dilaksanakan pada *product backlog*.

Jadual 3-17: *Product Backlog* Sebelum Aktiviti *Product Backlog Refinement*

PRODUCT BACKLOG								
USER STORY ID	KEUTAMAAN	TURUTAN KEUTAMAAN	*USER STORY*			*STORY POINT*	*SPRINT* NO	*CATATAN PRODUCT BACKLOG REFINEMENT*
Modul Mengurus Pengguna								
PB-BF-BM-MP-PR-01	Mesti Ada	1	Pengguna (warga MAMPU)	**BOLEH** mendaftar profil pengguna baharu	**SUPAYA** boleh mendaftar masuk sistem tempahan bilik mesyuarat	3	Sprint 1	
PB-BF-BM-MP-PR-02	Mesti Ada	2	Pengguna (warga MAMPU)	**BOLEH** mengemas kini maklumat profil pengguna yang telah didaftarkan	**SUPAYA** akaun dapat disahkan dan selamat.	5	Sprint 1	

Contoh *product backlog* yang dikemas kini hasil daripada aktiviti *Product Backlog Refinement* adalah seperti Jadual 3-18.

a. *Product backlog* PB-BF-BM-MP-PR-01 memerlukan fungsi tambahan, iaitu notifikasi melalui email perlu dihantar kepada pengguna setelah pendaftaran pengguna berjaya. *Story point* dikemaskini untuk pertambahan aktiviti pembangunan API.

b. *Product backlog* PB-BF-BM-MP-PR-03 ditambah dalam *product backlog sebagai* item baharu untuk dilaksanakan pada *sprint* kedua.

Jadual 3-18: Pengemaskinian *Product Backlog* Selepas Aktiviti *Product Backlog Refinement*

PRODUCT BACKLOG								
USER STORY ID	KEUTAMAAN	TURUTAN KEUTAMAAN	*USER STORY*			*STORY POINT*	*SPRINT* NO	*CATATAN PRODUCT BACKLOG REFINEMENT*
Modul Mengurus Pengguna								
PB-BF-BM-MP-PR-01	Mesti Ada	1	Pengguna (warga MAMPU)	**BOLEH** mendaftar profil pengguna baharu	**SUPAYA** boleh mendaftar masuk sistem tempahan bilik mesyuarat	5	Sprint 1	Dipecahkan kepada *product backlog* baharu, PB-BF-BM-MP-PR-03
PB-BF-BM-MP-PR-02	Mesti Ada	2	Pengguna (warga MAMPU)	**BOLEH** mengemas kini maklumat profil pengguna yang telah didaftarkan	**SUPAYA** akaun dapat disahkan dan selamat.	5	Sprint 1	
PB-BF-BM-MP-PR-03	Mesti Ada	3	Pengguna (warga MAMPU)	**BOLEH** menghantar notifikasi melalui email apabila profil pengguna baharu didaftarkan	**SUPAYA** pengguna menerima notifikasi pendaftaran selesai	5	Sprint 2	Pecahan fungsi daripada PB-BF-BM-MP-PR-01

3.4.6. Aktiviti *Sprint Review*

Sprint review adalah aktiviti yang berlangsung pada akhir setiap *sprint* semasa. Tujuan *sprint review* adalah untuk menilai kerja yang telah dilakukan dalam *sprint* semasa, mendapatkan maklum balas dari SME Bisnes dan memastikan bahawa kerja yang dilakukan selaras dengan *product vison* dan keperluan SME Bisnes. Penglibatan *scrum master* adalah amat penting bertindak sebagai fasilitator aktiviti.

Aktiviti yang dilaksanakan semasa sesi *sprint review* adalah seperti berikut:

a. Pasukan pembangun mempamerkan kerja yang telah dilaksanakan kepada SME Bisnes dan menerima maklum balas daripada mereka.
b. Pemilik produk membentangkan kemajuan pembangunan dalam mencapai matlamat projek secara keseluruhan.
c. Pasukan pembangun memberikan maklum balas mengenai kerja yang telah dilakukan, yang boleh digunakan untuk meningkatkan produk dalam *sprint* seterusnya.
d. Pemilik produk mengemas kini *product backlog* berdasarkan maklum balas yang diterima dan merancang untuk *sprint* seterusnya.
e. Pasukan boleh menggunakan peluang untuk meraikan kemajuan yang telah dicapai dan menghargai usaha ahli pasukan dalam pelaksanaan pembangunan produk.

Artifak Rujukan kepada aktiviti *sprint review* adalah *product backlog*, *sprint backlog* dan DoD.

Artifak Hasil daripada aktiviti *sprint review* adalah artifak DoD dan *product increment*.

3.4.6.1. Contoh Pengisian Artifak *Definition of Done*

Contoh artifak DoD yang telah lengkap diisi adalah seperti Jadual 3-9.

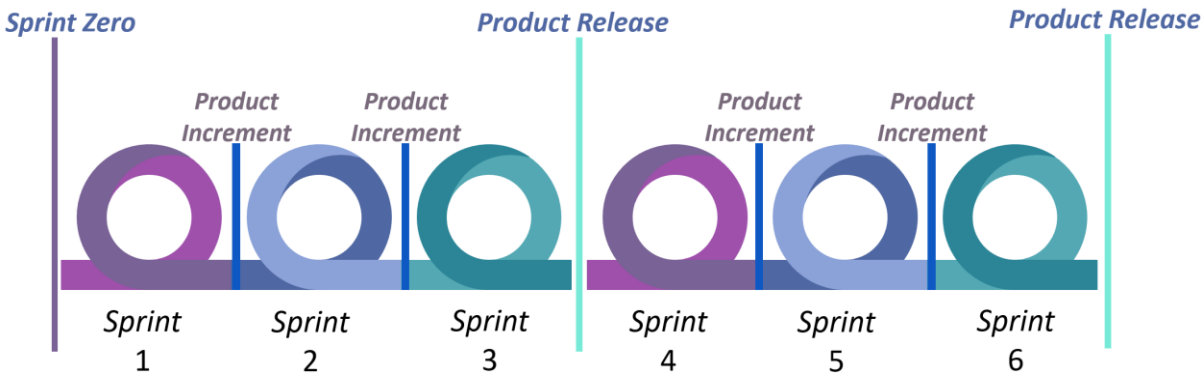
.

Jadual 3-19: Artifak *Definition of Done*

Komponen	Senarai Semak	Status	Catatan
Pengujian Keperluan Fungsian	1. Kod sumber selesai dibangunkan	✓	*Sprint* 1
	2. Kod sumber disemak	✓	*Sprint* 1
	3. Kod sumber digabungkan (merged)	✓	*Sprint* 1
	4. Melepasi pengujian unit	✓	*Sprint* 1
	5. Melepasi pengujian integrasi	✓	*Sprint* 1
	6. Melepasi pengujian sistem dan memenuhi *acceptance criteria* bagi setiap *user story*	✓	*Sprint* 1
	7. Melepasi pengujian penerimaan pengguna		*Sprint* 3
Pengujian Keperluan Bukan Fungsian	1. Melepasi pengujian kualiti kod	✓	*Sprint* 1
	2. Melepasi pengujian SAST	✓	*Sprint* 1
	3. Melepasi pengujian prestasi	✓	*Sprint* 1
Penempatan	1. Penempatan ke persekitaran *staging* pada *sprint* yang melibatkan *product release* sahaja		*Sprint* 3
Dokumentasi	1. Artifak *product vision*	✓	*Sprint* 1
	2. Dokumen laporan UAT		*Sprint* 3
	3. Artifak *sprint backlog*	✓	*Sprint* 1
	4. Artifak carta *burndown*	✓	*Sprint* 1
	5. Artifak perancangan kapasiti	✓	*Sprint* 1
	6. Artifak log *daily scrum*	✓	*Sprint* 1
	7. Artifak *sprint retrospective*	✓	*Sprint* 1

3.4.6.2. *Product Increment*

Product increment adalah sejumlah *product backlog* yang telah diselesaikan pada *sprint* semasa. *Product increment* merupakan produk yang telah dibangunkan oleh pasukan *scrum* dan memenuhi DoD yang telah dipersetujui. Rajah 3-9 menunjukkan gambaran bahawa *product increment* akan dihasilkan pada setiap *sprint* manakala *product release* akan dilepaskan ke persekitaran produksi setelah berakhirnya *sprint* 3. Penerangan berkaitan *product release* akan diterangkan pada para 5.5.3.1.



Rajah 3-9: Aktiviti *Sprint* dan *Product Increment*

3.4.7. Aktiviti *Sprint Retrospective*

Sprint retrospective merupakan aktiviti yang diadakan pada akhir *sprint* semasa. Semasa aktiviti ini, pasukan *scrum* menyemak perkara yang berjalan dengan lancar semasa *sprint* semasa, perkara yang tidak berjalan seperti dijangka dan perkara yang perlu ditambah baik dalam *sprint* berikutnya.

Tujuan *sprint retrospective* adalah untuk mengenal pasti perkara untuk penambahbaikan dan membuat perubahan untuk meningkatkan prestasi pasukan.

Aktiviti yang dilaksanakan semasa *sprint retrospective* adalah untuk menggalakkan komunikasi dan maklum balas terbuka adalah seperti berikut:

a. Menjelaskan tujuan retrospektif dan menetapkan peraturan asas seperti hanya seseorang dibenarkan bercakap pada satu-satu masa, tidak menyalahkan sesama ahli pasukan dan memberikan maklum balas secara membina.

b. Membincang dan mengumpul maklumat berkaitan *sprint* semasa seperti perkara yang berjalan lancar dan perkara yang tidak berjalan seperti dijangka.

c. Menganalisis maklumat untuk mengenal pasti perkara yang boleh ditambah baik.

d. Menentukan tindakan yang boleh diambil untuk memperbaiki proses pelaksanaan pembangunan.

e. Menyimpulkan perbincangan, bersetuju tentang item tindakan dan menetapkan rancangan untuk *sprint* seterusnya.

f. Memastikan tindak lanjut pada item tindakan dilaksanakan dan memberi impak positif.

Artifak Rujukan kepada aktiviti *sprint retrospective* adalah artifak DoD dan *product increment*.

Artifak Hasil daripada aktiviti *sprint retrospective* adalah artifak *sprint retrospective* yang mengandungi senarai penambahbaikan untuk *sprint* seterusnya.

Penglibatan ahli pasukan adalah seperti berikut:

a. Pemilik produk boleh menyertai retrospektif, memberikan maklum balas tentang *product backlog* dan membantu mengenal pasti perkara yang perlu untuk peningkatan produk.

b. *Scrum master* berperanan sebagai fasilitator, memastikan ahli pasukan berpeluang memberikan idea dan maklum balas.

c. Pasukan pembangun perlu terlibat secara aktif dengan memberikan input dan maklum balas.

Sprint retrospective melibatkan pasukan *scrum* sahaja untuk memastikan ahli pasukan bebas mengutarakan idea dan maklum balas. Walau bagaimanapun, SME Bisnes boleh dijemput jika boleh memberikan input dan penyelesaian yang diperlukan oleh pasukan.

3.4.7.1. Artifak *Sprint Retrospective*

Artifak *sprint retrospective* merujuk kepada templat yang mengandungi soalan-soalan *post mortem* yang perlu dibincang dan dipersetujui oleh ahli pasukan *scrum* pada penghujung *sprint*. Artifak ini akan menjadi input penambahbaikan yang boleh dilaksanakan pada *sprint* seterusnya.

a. Komponen Artifak *Sprint Retrospective*

Jadual 3-20 menerangkan komponen di dalam artifak *sprint retrospective*.

Jadual 3-20: Komponen Artifak *Sprint Retrospective*

No.	Komponen	Penerangan
1.	Perkara yang Berjalan dengan Lancar	Merujuk kepada perkara positif yang perlu dilakukan lebih lagi.
2.	Perkara yang Tidak Berjalan seperti Dijangka	Merujuk kepada perkara negatif yang sedang dilakukan dan perlu diberhentikan.
3.	Perkara yang Perlu Ditambah baik	Merujuk kepada penambahbaikan yang perlu dilaksanakan.

Templat artifak *sprint retrospective* boleh dirujuk dalam Lampiran A-8.

b. Contoh Pengisian Artifak *Sprint Retrospective*

Contoh pengisian artifak *sprint retrospective* adalah seperti Jadual 3-21.

Jadual 3-21: Artifak *Sprint Retrospective*

ID	Perkara Yang Berjalan Dengan Lancar	Perkara Yang Tidak Berjalan Seperti Dijangka	Perkara Yang Perlu Ditambah Baik
SP 001	Nota *daily scrum meeting* telah diedarkan sebaik selesai aktiviti *daily scrum meeting*	Tempoh masa *daily scrum meeting* kerap melebihi 15 minit	Menyediakan pelan penempatan di akhir setiap *sprint*
SP 001	Komunikasi dan kolaborasi yang konsisten dengan pemilik produk	Sesetengah ralat mengambil masa yang agak lama untuk diselesaikan	

BAB 4 PENGENALAN *TOOLS* DEVOPS SEKTOR AWAM

4.1. PENGENALAN *TOOLS* DEVOPS

Rajah 4-1: Teras 5 Pemerkasaan Teknologi

Merujuk kepada Teras 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 5.

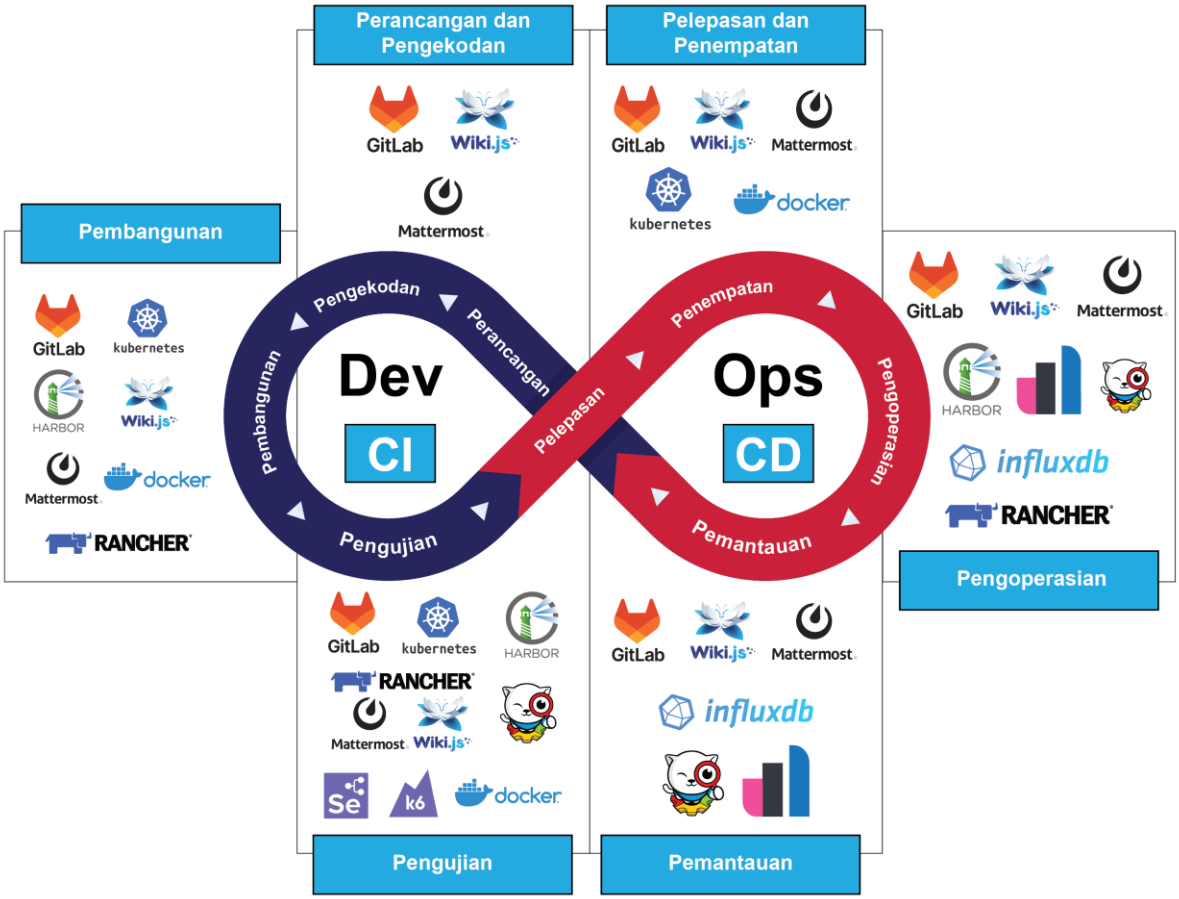
4.2. SENARAI *TOOLS* PELAKSANAAN DEVOPS SEKTOR AWAM

Berikut merupakan senarai dan penggunaan *tools* pada pelaksanaan peringkat DevOps dalam pembangunan sistem aplikasi sektor awam. *Tools* yang disenaraikan pada Jadual 4-1 merupakan senarai *tools* yang boleh didapati daripada platform sumber terbuka.

Jadual 4-1: 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.	InfluxDB							✓	✓
13.	Elastic Observability							✓	✓

Rajah 4-2 memaparkan infografik padanan *tools* pada setiap peringkat pada kitar hayat DevOps.



Rajah 4-2: Senarai *Tools* DevOps Sektor Awam.

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

4.2.1. 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.

4.2.2. 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 4-2 merupakan padanan di antara artifak *agile* dan *features* GitLab yang akan digunakan pada Bab 5.

Jadual 4-2: 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.
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.

Artifak *Agile*	*Features* GitLab	Penjelasan
Carta *Burndown*	Burndown *charts*	Fungsi GitLab Burndown *Chart* adalah memaparkan perkembangan pembangunan sistem aplikasi.
Agile board	*Issues Boards*	Fungsi GitLab *Issues Boards* adalah memaparkan keseluruhan *issues* dibawah *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 telah dibincangkan pada para 2.1 iaitu integrasi, pengujian, penyampaian dan penempatan 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 yang terakhir iaitu pemantauan berterusan akan dilakukan dengan menggunakan *tools* yang berbeza.

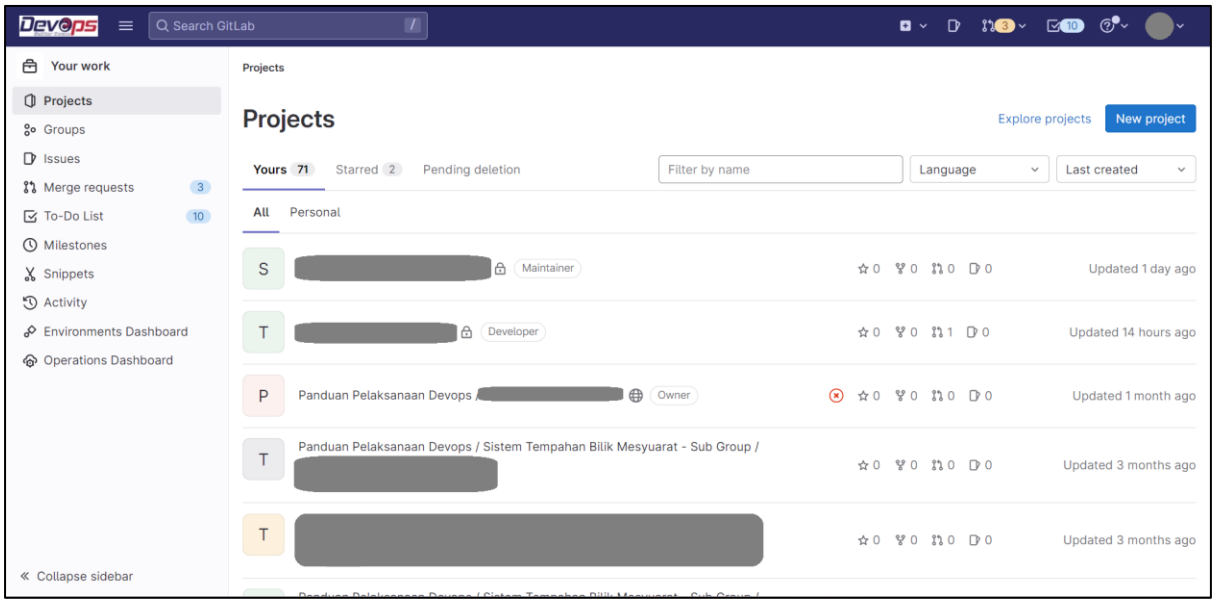
b. GitLab *Runner*

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 4-3 menjelaskan kategori *runner* yang boleh dipasang.

Jadual 4-3: Kategori GitLab *Runner*

Kategori Runner	Penjelasan
Specific Runner	i. Dikhaskan hanya untuk satu projek repositori git sahaja dan tidak dapat digunakan oleh repositori projek Git yang lain. ii. Boleh dijadikan sebagai *shared runner* sekiranya *maintainer* atau pemilik *runner* memutuskan agar *runner* tersebut boleh digunakan oleh repositori yang lain.
Shared Runner	i. Ditujukan untuk seluruh projek repositori Git. ii. 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	i. Ditujukan untuk beberapa projek repositori Git yang terdaftar di dalam satu kumpulan repositori.

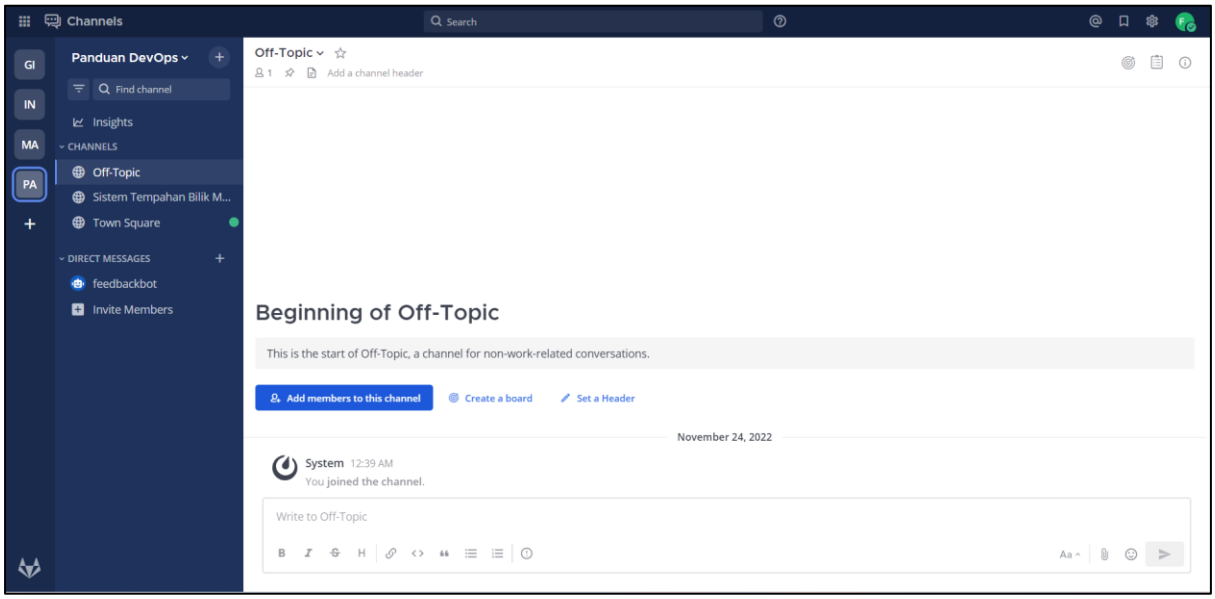
Rajah 4-3 memaparkan contoh paparan *tools* GitLab.



Rajah 4-3: Contoh paparan GitLab

4.2.3. Mattermost

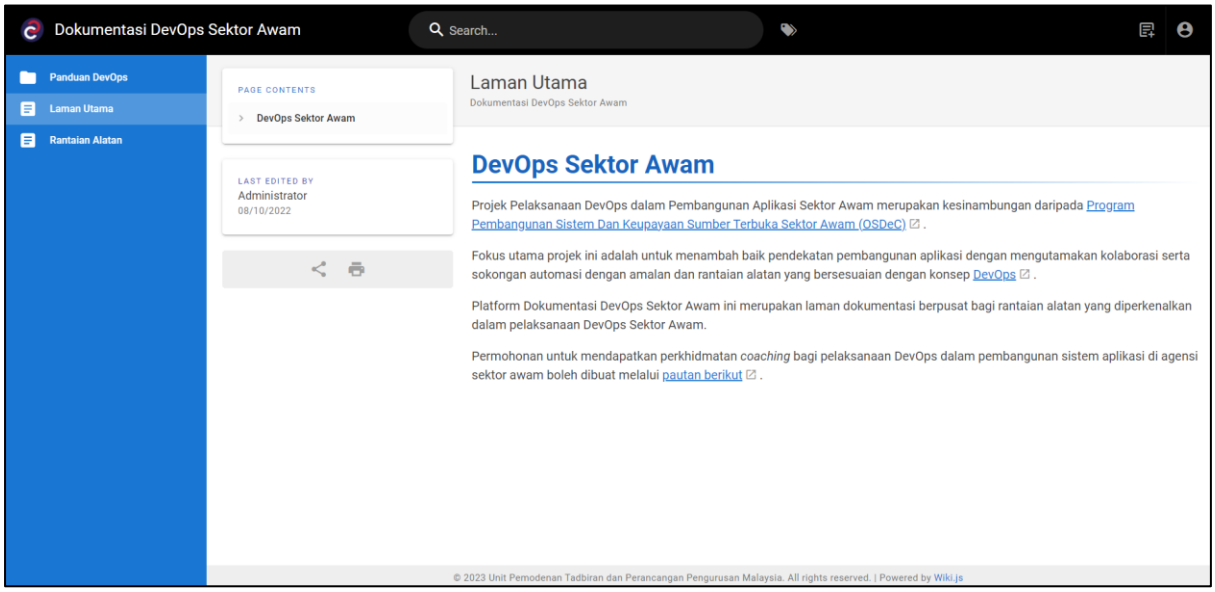
Mattermost merupakan platform yang menyediakan perkhidmatan komunikasi dan kolaborasi antara ahli pasukan. Integrasi Mattermost dengan GitLab boleh dilaksanakan bagi tujuan mewujudkan *Issues* dan memulakan CI/CD *jobs*. Tatacara penggunaan Mattemost *ChatOps* pada dokumen ini adalah menggunakan Mattermost Team Edition versi 7.5.1. Rajah 4-4 memaparkan contoh paparan *tools* Mattermost *ChatOps.*



Rajah 4-4: Contoh Paparan Mattermost *ChatOps*

4.2.4. Wiki.js

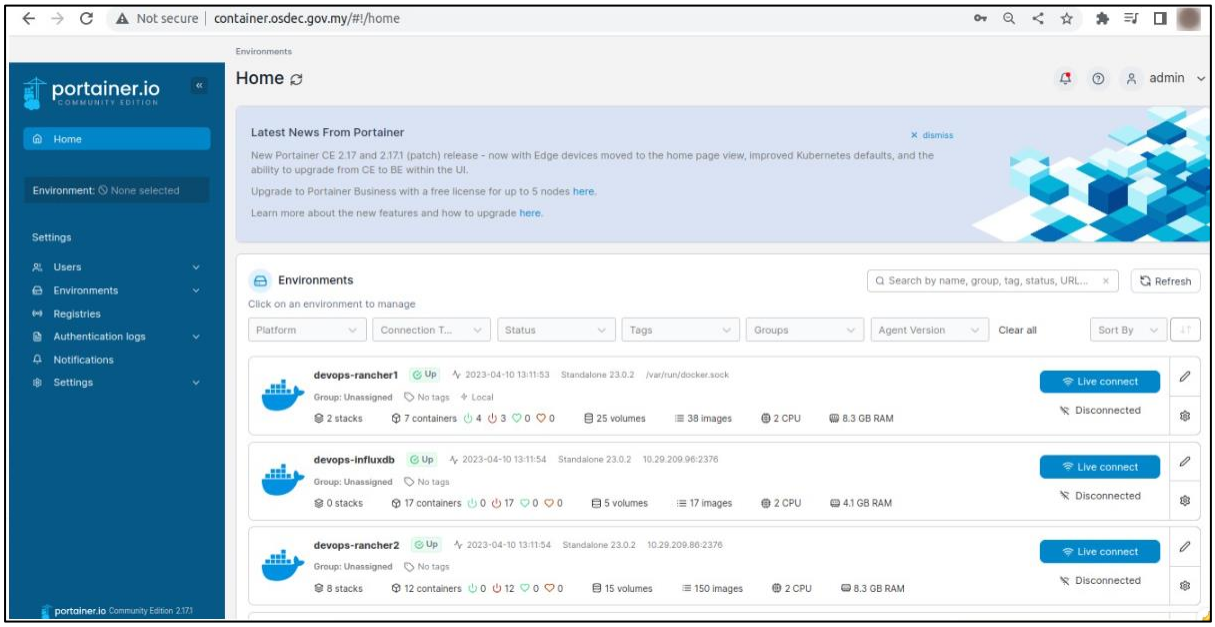
Wiki.js merupakan perisian kolaboratif yang membolehkan pengguna mencipta dan mengemas kini dokumentasi secara dalam talian. Kandungan dokumentasi ditulis menggunakan format Markdown atau fungsi *visual editor.* Wiki.js boleh juga digunakan sebagai repositori dokumentasi sepanjang pelaksanaan pembangunan sistem aplikasi. Tatacara penggunaan Wiki.js pada dokumen ini adalah menggunakan Wiki.js versi 2.5.295. Rajah 4-5 memaparkan contoh paparan *tools* Wiki.js.



Rajah 4-5: Contoh Paparan Wiki.js

4.2.5. Docker

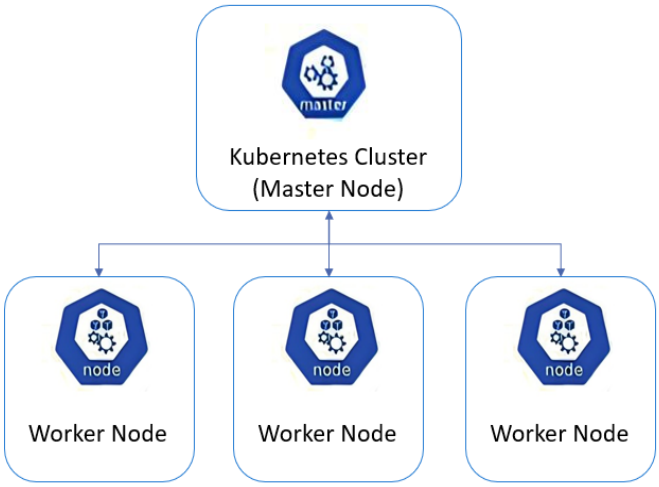
Docker adalah platform yang membolehkan sistem aplikasi dipakejkan kepada satu format standard *Open Container Initiative* (OCI) dipanggil imej *container* yang mengandungi *library*, *runtime* dan kod sumber sistem aplikasi. Platform ini membolehkan pasukan membina, menguji dan menempatkan sistem aplikasi 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 4-6 memaparkan contoh paparan *tools* Docker.



Rajah 4-6: Contoh Paparan Docker

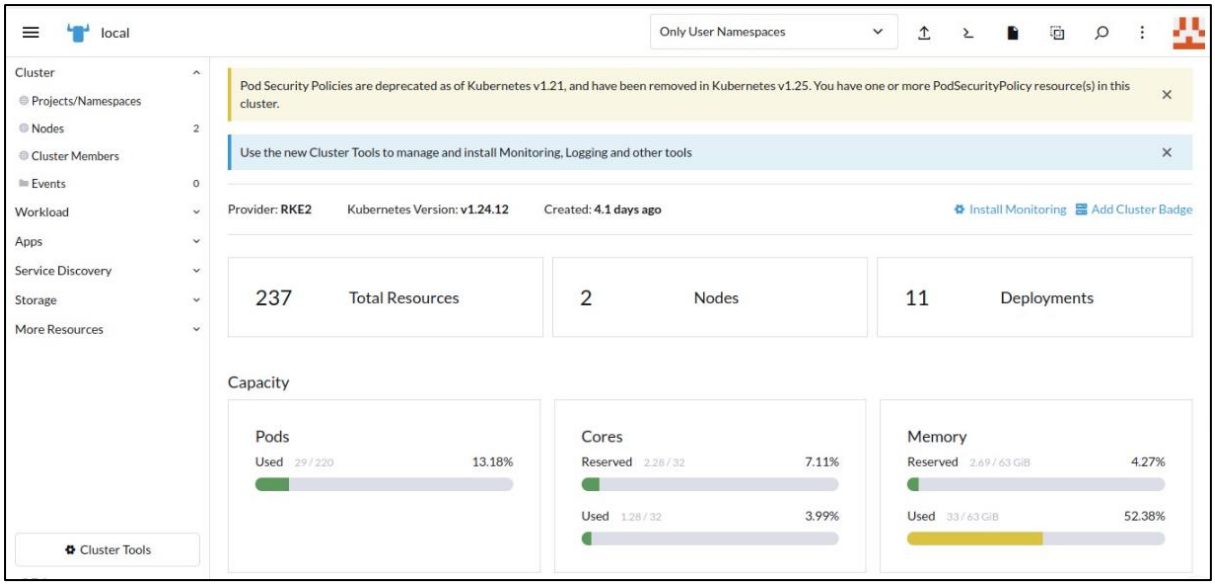
4.2.6. Kubernetes

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 4-7: Komponen Kluster Kubernetes

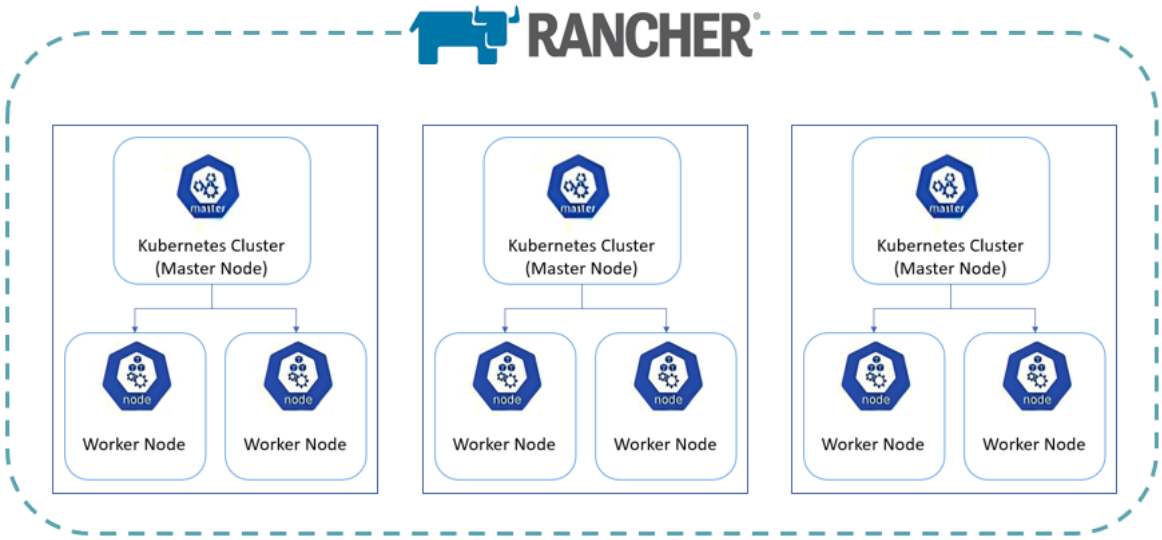
Rajah 4-7 menunjukkan Kluster Kubernetes yang berfungsi sebagai *master node* berupaya menguruskan beberapa *worker node* yang terdiri daripada pelayan *virtual* ataupun fizikal. *Worker node* berfungsi untuk menempatkan dan menjalankan aplikasi *container*. Rajah 4-8 memaparkan contoh paparan *tools* Kubernetes.



Rajah 4-8: Contoh Paparan Kubernetes

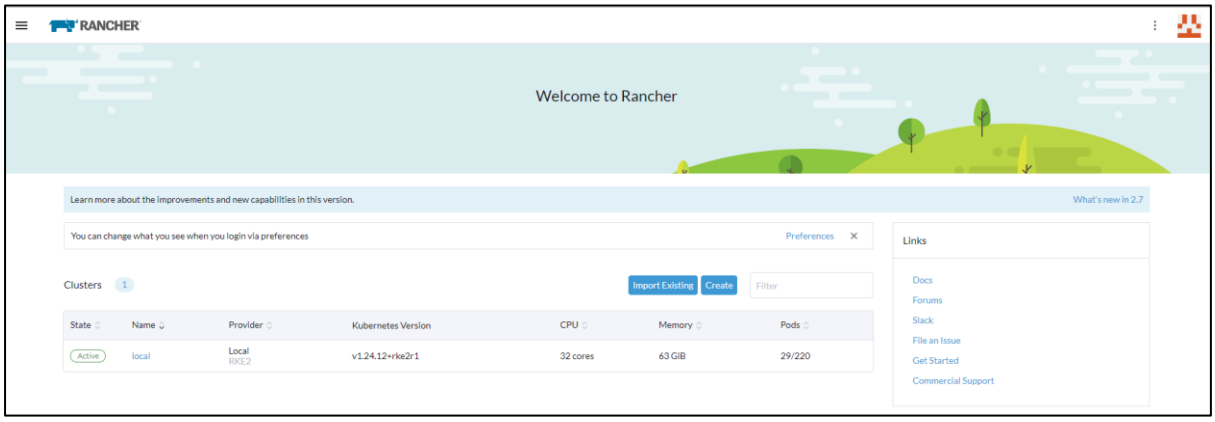
4.2.7. Rancher

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



Rajah 4-9: Arkitektur Platform Rancher

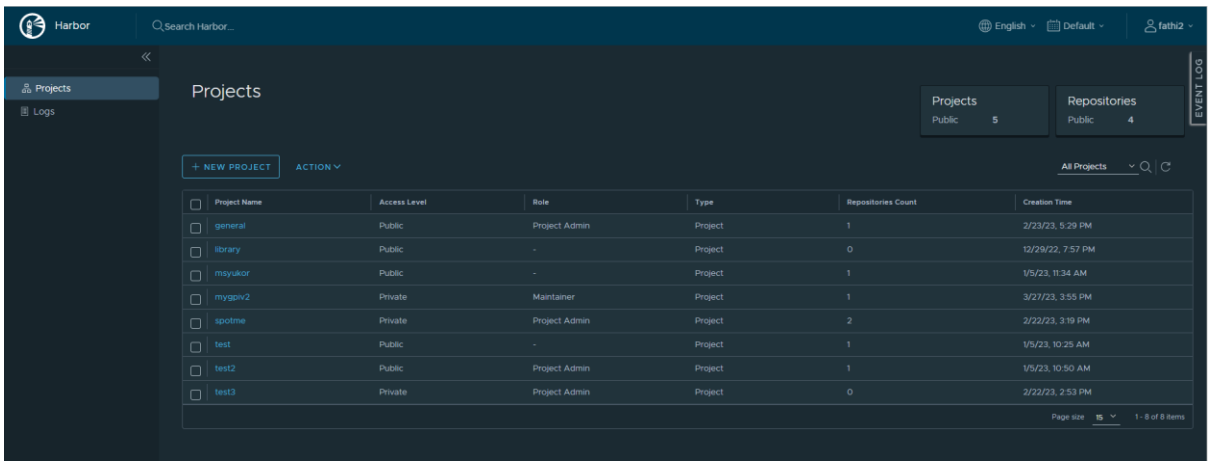
Rajah 4-9 menunjukkan arkitektur platform Rancher yang berfungsi sebagai antara muka pengguna dalam memantau dan menguruskan satu atau lebih kluster Kubernetes. Rajah 4-10 memaparkan contoh paparan *tools* Rancher.



Rajah 4-10: Contoh Paparan Rancher

4.2.8. 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 4-11 memaparkan contoh paparan *tools* Harbor.



Rajah 4-11: Contoh Paparan Harbor

4.2.9. Selenium Grid

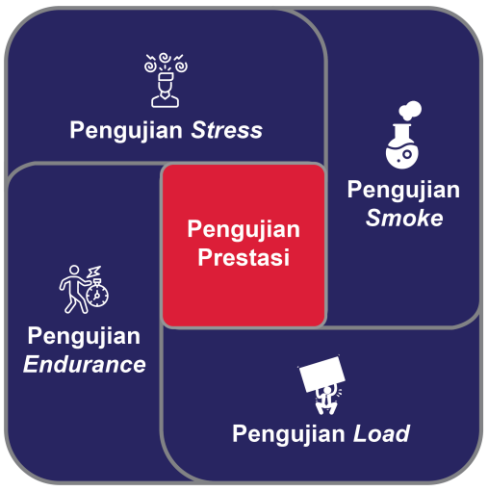
Selenium Grid adalah *tools* untuk melaksanakan pengujian fungsian aplikasi web secara automasi di pelbagai pelayar dan *remote machine* secara serentak dalam persekitaran berbeza. Jadual 4-4 menyenaraikan sistem pengoperasian, bahasa pengaturcaraan dan pelayar web yang disokong oleh Selenium Grid untuk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

Jadual 4-4: 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.

4.2.10. K6

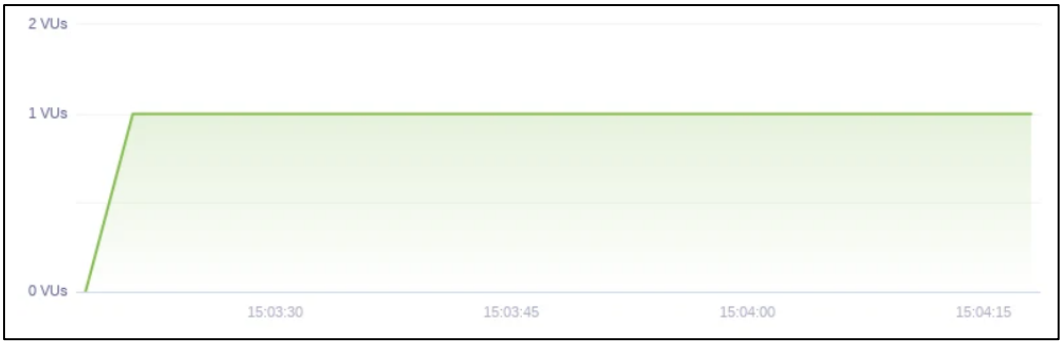
Grafana K6 merupakan *tools* pengujian prestasi *load testing* untuk sistem berasaskan web dan web API. K6 dapat melaksanakan pengujian terhadap *reliability* dan prestasi sistem aplikasi seterusnya mengesan masalah berkaitan prestasi lebih awal. K6 boleh menjalankan pelbagai jenis pengujian prestasi bergantung kepada skop pengujian yang ingin dilaksanakan. Rajah 4-12 menunjukkan jenis pengujian prestasi yang boleh dilaksanakan oleh K6.



Rajah 4-12: 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 4-13 memaparkan contoh graf hasil daripada pengujian *smoke*.



Rajah 4-13: Contoh Paparan Graf bagi Pengujian *Smoke*

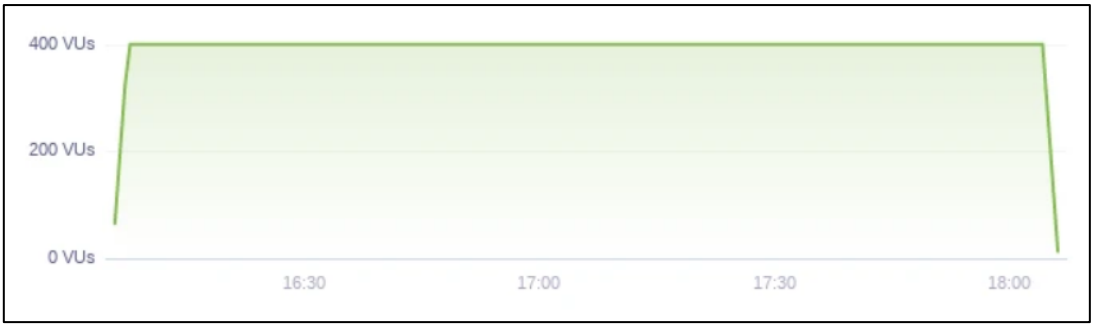
b. Pengujian *Load* menilai prestasi sistem dari segi akses pengguna secara serentak atau berdasarkan permintaan sesaat (requests per second). Rajah 4-14 memaparkan contoh graf hasil daripada pengujian *load*.

Rajah 4-14: Contoh Paparan Graf bagi Pengujian *Load*

c. Pengujian *Stress* menilai had dan kestabilan sistem dalam keadaan yang maksimum berdasarkan had yang ditetapkan. Rajah 4-15 memaparkan contoh graf hasil daripada pengujian *stress*.

Rajah 4-15: Contoh Paparan Graf bagi Pengujian *Stress*

d. Pengujian *Endurance* menilai dan mengesan aktiviti berkaitan prestasi sistem dalam persekitaran produksi secara berterusan. Rajah 4-16 memaparkan graf hasil daripada pengujian *endurance*.



Rajah 4-16: Contoh Paparan Graf bagi Pengujian *Endurance*

4.2.11. Sitespeed.io

Sitespeed.io merupakan *tools* yang membantu dalam memantau, menganalisis dan mengoptimumkan prestasi laman web. Sitespeed.io mempunyai 3 keupayaan utama berikut:

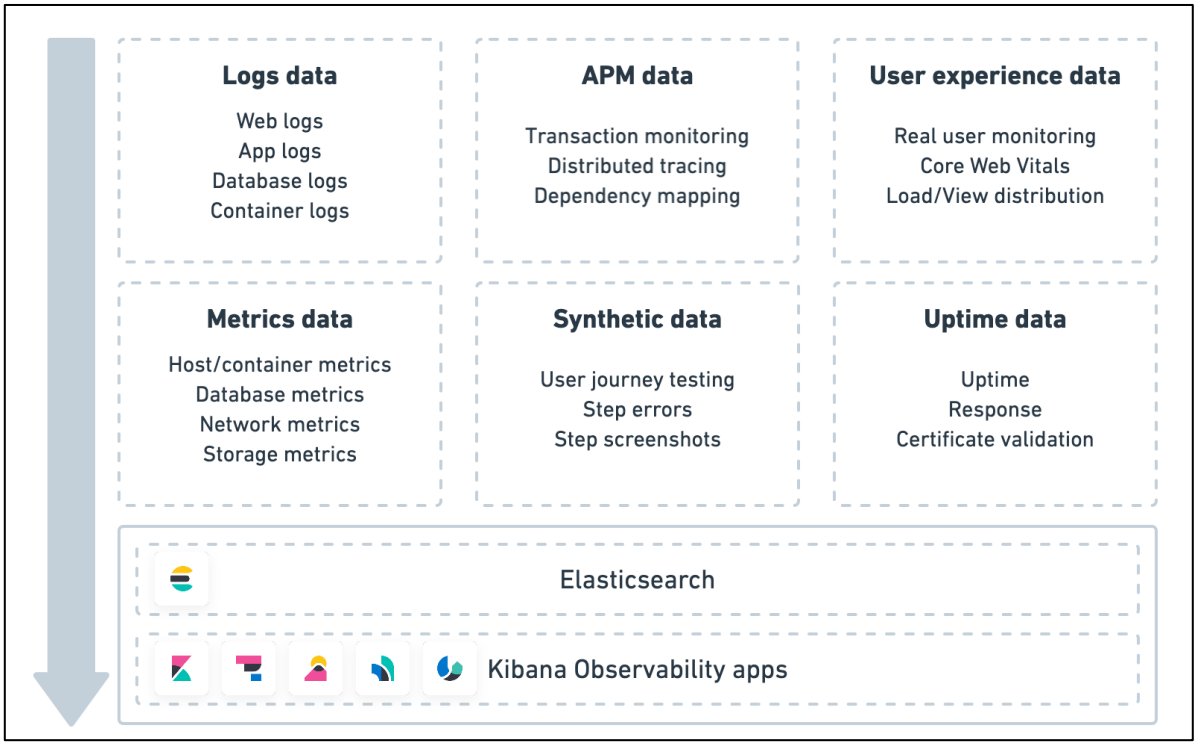
a. Menjalankan pengujian laman web menggunakan metrik penyemak imbas *Navigation Timing API, User Timings and Visual Metrics* (FirstVisualChange, SpeedIndex & LastVisualChange).
b. Mensimulasikan ketersambungan pengguna sebenar dan mengumpulkan metrik yang penting seperti *Speed Index* dan *First Visual Render*.
c. Menganalisis cara laman web dibina dan mencadangkan penambahbaikan yang bersesuaian bagi meningkat prestasi laman web.

4.2.12. 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.

4.2.13. 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 4-17 menunjukkan jenis data dan metrik yang diterima oleh Elastic Observability.



Rajah 4-17: Komponen yang terdapat dalam Elastic Observability. Sumber: Elastic Observability

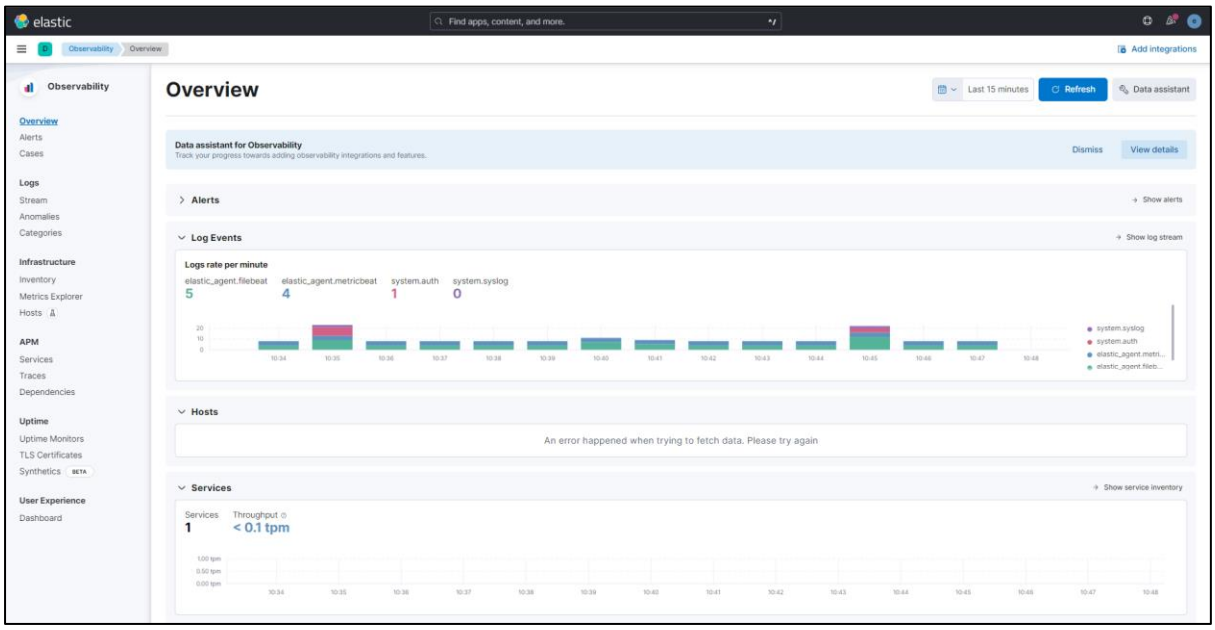
Penerangan berikut menjelaskan aliran data yang dihantar dari sumber ke Elastic Observability.

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.

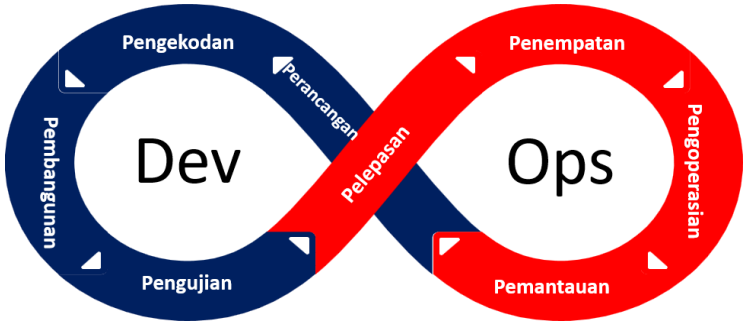
Rajah 4-18 memaparkan contoh paparan *tools* Elastic Observability.



Rajah 4-18: Contoh Paparan Elastik Observability

BAB 5 KITAR HAYAT DEVOPS

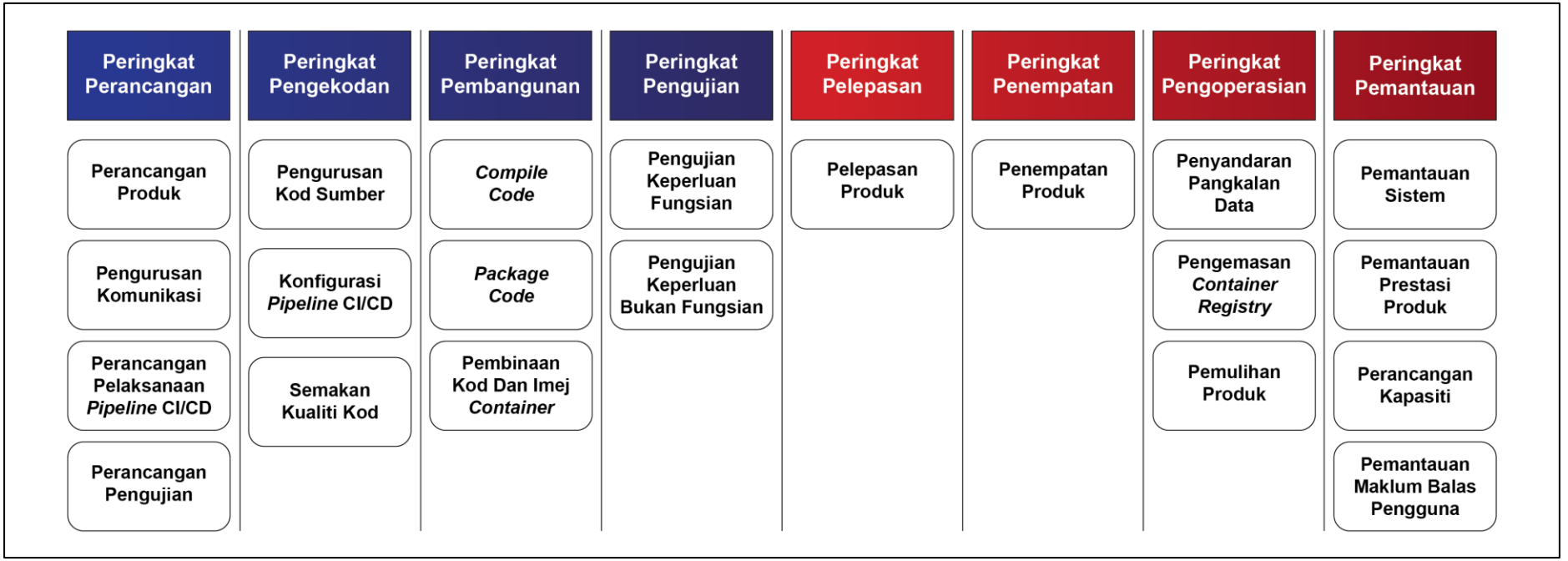
DevOps merupakan penerapan prinsip dan budaya yang membolehkan komunikasi dan kerjasama di antara pihak-pihak berkepentingan menjadi lebih baik terutamanya dalam menentukan hala tuju, membangunkan dan mengendalikan sistem aplikasi serta melaksanakan penambahbaikan berterusan dalam semua aspek kitaran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



Rajah 5-1: 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 5-2: Aktiviti di Peringkat DevOps

Rajah 5-2 memaparkan senarai aktiviti utama yang terdapat pada setiap peringkat DevOps. Penerangan setiap aktiviti tersebut akan diterangkan pada setiap peringkat berikut.

5.1. PERINGKAT PERANCANGAN

Peringkat perancangan merujuk kepada peringkat awal dalam pembangunan produk. Pada peringkat ini, keperluan dan maklum balas pengguna dikumpulkan daripada pihak berkepentingan serta perancangan perubahan perlu dilakukan untuk penambahbaikan produk. Peringkat perancangan penting kerana boleh membantu memastikan matlamat dan hala tuju produk yang dibangunkan difahami oleh setiap ahli pasukan DevOps. Matlamat yang jelas membolehkan pasukan DevOps merancang dan membangunkan produk yang memenuhi keperluan pengguna.

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.

Peringkat perancangan terdiri daripada empat aktiviti utama iaitu:

a. Perancangan Produk,
b. Pengurusan Komunikasi,
c. Perancangan Pelaksanaan *Pipeline* CI/CD dan
d. Perancangan Pengujian.

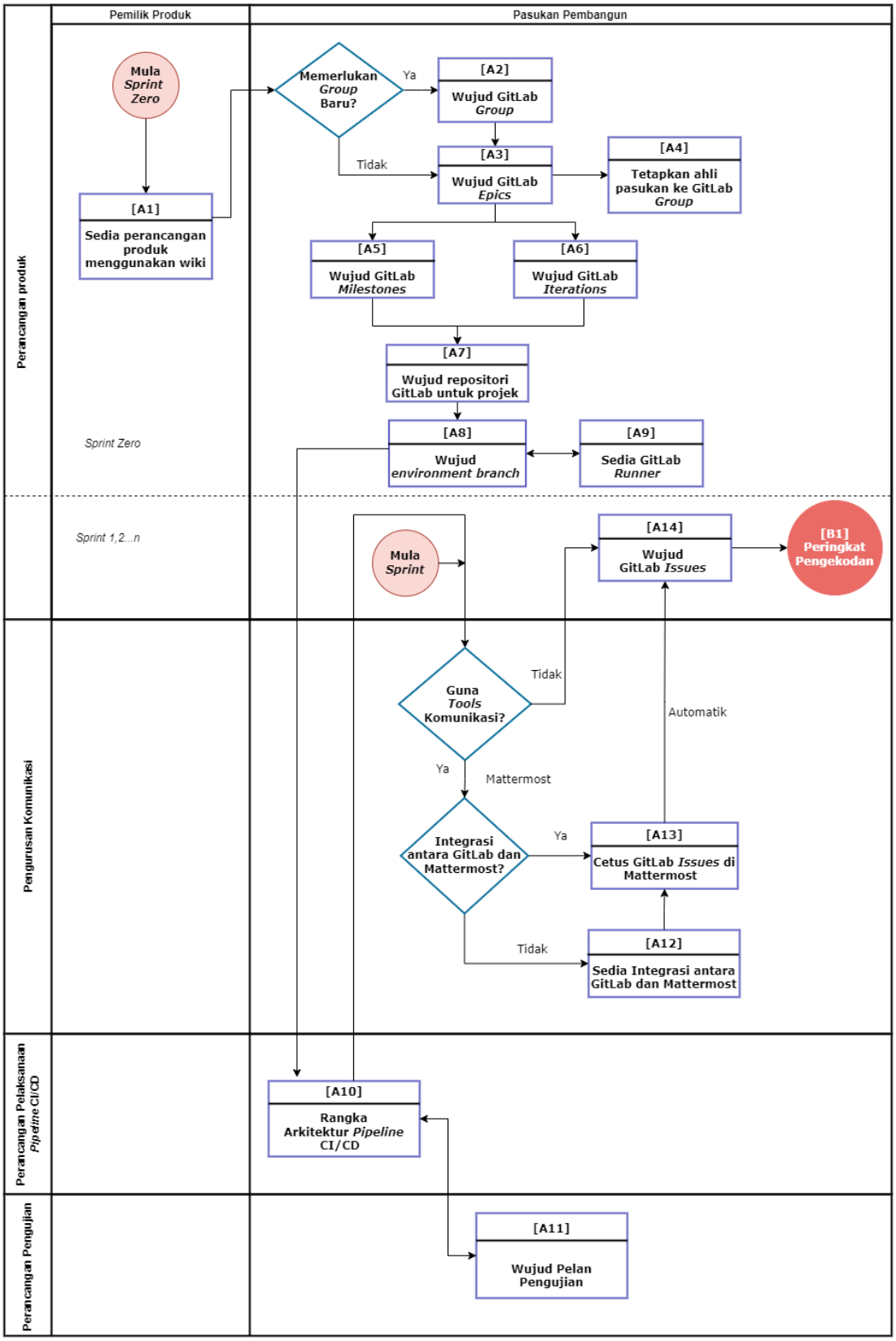
5.1.1. Prasyarat

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

5.1.2. 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 5-3 memaparkan aliran proses kerja yang terlibat dalam peringkat perancangan.

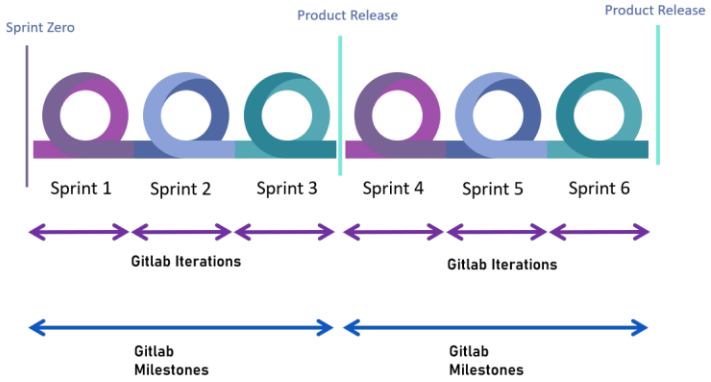


Rajah 5-3: Aliran Kerja Peringkat Perancangan

Jadual 5-1 menerangkan proses yang berlaku dalam peringkat perancangan berdasarkan Rajah 5-3.

Jadual 5-1: 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.
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

Aktiviti	Penerangan
	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 5-4 menunjukkan gambaran tempoh masa bagi GitLab *Iterations* dan *Milestones*.  **Rajah 5-4: Tempoh Masa bagi GitLab *Iterations* dan *Milestones*** **Rajah 5-4: Tempoh Masa bagi GitLab *Iterations* dan *Milestones***

Aktiviti	Penerangan
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.*

5.1.3. Perancangan Produk

Perancangan produk menjadi asas kepada langkah kerja yang akan dilaksanakan bagi semua ahli pasukan. 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.

Antara perkara yang dilakukan dalam aktiviti perancangan produk adalah seperti berikut:

a. Penyediaan pelan hala tuju produk

Artifak *product vision, epic* dan *user story* akan dikumpul sebagai pelan hala tuju produk dan didokumenkan dalam bentuk wiki.

b. Penyediaan *tools* bagi persekitaran pembangunan

Tools seperti GitLab yang akan digunakan sepanjang pembangunan produk perlu disediakan.

c. Pengumpulan keperluan pengguna

Keperluan pengguna dan tugasan dikumpulkan dalam artifak *product backlog* dan *sprint backlog* akan dimasukkan ke *tools* GitLab. Manakala DoD boleh didokumenkan dalam bentuk wiki.

d. Membuat perancangan kapasiti

Perancangan kapasiti yang dibuat pada *sprint planning meeting* kedua boleh didokumenkan ke bentuk wiki.

5.1.3.1. Penggunaan *Tools*

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

a. Wiki.js

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*

5.1.3.2. Ringkasan Tatacara Penggunaan *Tools*

a. Sedia Perancangan Produk menggunakan Wiki.js [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.

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-2.

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-3.

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-4.

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-5.

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-6.

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-7.

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 5-2 menerangkan penetapan *environment branch*.

Jadual 5-2: 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-8.

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*.*

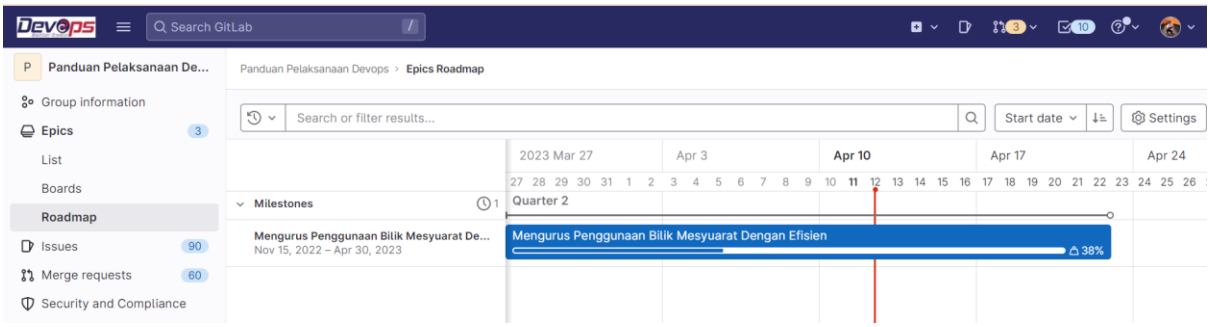
GitLab *runners* yang menyokong binari bagi arkitektur x86, AMD64, ARM64, ARM, s390x dan ppc64le. *Official Packages* boleh didapati bagi Linux *distributions* seperti CentOS, Debian, Ubuntu, RHEL, Fedora, Mint, Oracle dan Amazon. Tatacara untuk menyediakan *runners* boleh dirujuk pada Lampiran B-9.

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-10.

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 5-5 memaparkan contoh paparan GitLab *Roadmap*. Tatacara untuk melihat paparan GitLab *Roadmap* boleh dirujuk pada Lampiran B-11.



Rajah 5-5: Paparan GitLab *Roadmap*

5.1.4. 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.

5.1.4.1. Penggunaan *Tools*

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

5.1.4.2. 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-12.

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-13.

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-14.

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-15.

5.1.5. 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.

5.1.5.1. *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:

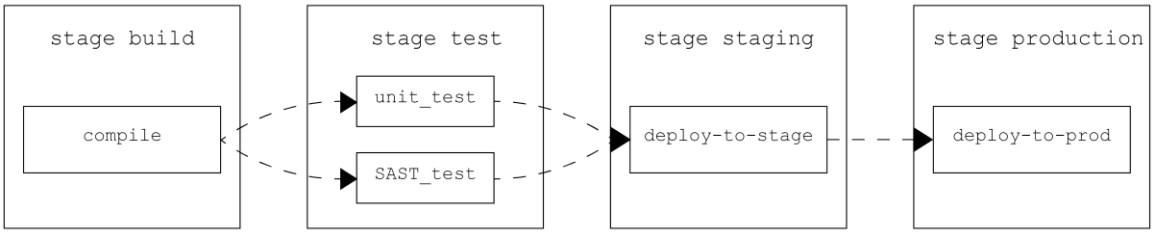
a. *Stage* `build` dengan *job* yang dipanggil `compile`.

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 5-6 memaparkan contoh *pipeline* yang lazim digunakan.



Rajah 5-6: Contoh *Pipeline*

5.1.5.2. 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.

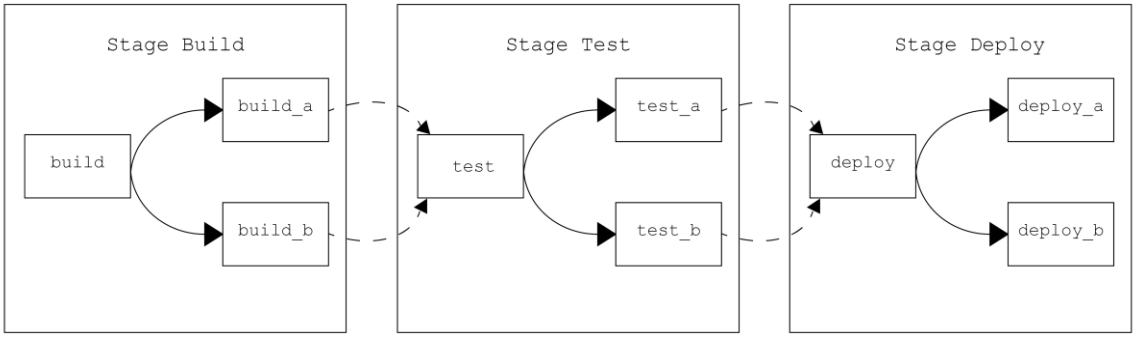
d. ***Pipeline Multi-project*** menggabungkan dua atau lebih projek yang berlainan di dalam satu *pipeline* yang sama.

5.1.5.3. 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*

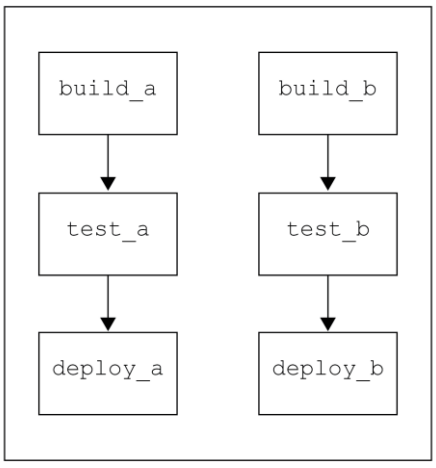
Sesuai bagi projek yang kecil apabila semua konfigurasi berada pada lokasi yang sama. Rujuk Rajah 5-7 bagi gambaran arkitektur *pipeline basic.*



Rajah 5-7: Arkitektur *Pipeline Basic*

b. *Pipeline* DAG

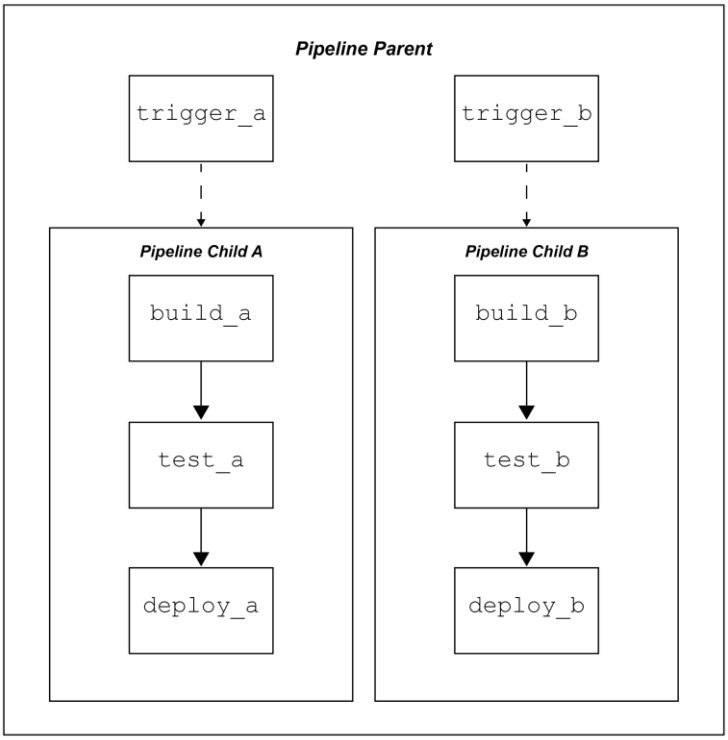
Sesuai bagi projek yang besar dan lebih kompleks yang memerlukan pelaksanaan yang lebih cekap. Rujuk Rajah 5-8 bagi gambaran arkitektur *pipeline* DAG.



Rajah 5-8: Arkitektur *Pipeline* DAG

c. *Pipeline Parent-child*

Sesuai bagi monorepo dan projek yang mempunyai komponen yang tersendiri. Rujuk Rajah 5-9 bagi arkitektur *pipeline parent-child*.



Rajah 5-9: 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.

5.1.5.4. 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).

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

5.1.5.5. 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 5.2.4.

5.1.5.6. Ringkasan Tatacara Penggunaan *Tools*

Penjelasan penggunaan *pipeline* CI/CD boleh dirujuk pada para 5.2.4. Tatacara untuk mewujud, mengakses dan mengemas kini fail `.gitlab-ci.yml` boleh dirujuk pada Lampiran B-16.

5.1.6. Perancangan Pengujian

Perancangan pengujian dilaksanakan bagi membolehkan pemilik produk, pasukan pembangun dan penguji merancang, menjadual dan melaksana aktiviti pengujian termasuklah serahan yang akan digunakan untuk mengesahkan bahawa aplikasi itu memenuhi keperluan pengguna yang ditetapkan.

Bermula pada *sprint zero*, pemilik produk dan pasukan pembangun akan membincangkan perkara yang perlu dicapai dalam aktiviti pengujian bagi setiap *sprint*. Pelan Pengujian disediakan dengan tujuan untuk:

a. Merancang dan mengurus aktiviti pengujian sistem aplikasi secara menyeluruh.
b. Menentukan maklumat berkaitan keperluan *tools* pengujian, data pengujian dan persekitaran pengujian yang digunakan untuk menyediakan keperluan pengujian.
c. Memahami peranan dan tanggungjawab pihak yang terlibat yang akan melaksanakan proses dan prosedur pengujian.

5.1.6.1. Pelan Pengujian

Pelan Pengujian mengandungi perkara berikut:

a. Penetapan jenis dan skop aktiviti pengujian yang akan dilaksanakan.
b. Pendekatan pelaksanaan pengujian.
c. Penetapan tahap kritikal untuk pengendalian ralat.
d. Penetapan *traceability matrix* untuk liputan kes pengujian.

5.1.6.2. Jenis dan Skop Aktiviti Pengujian

Pendekatan bagi aktiviti pengujian ini dikenali sebagai *incremental testing* iaitu pengujian dilakukan secara berperingkat bermula daripada pengujian unit/komponen terkecil sistem aplikasi seperti fungsi, kelas, prosedur dan antara muka (Pengujian Unit); seterusnya menguji dua atau lebih modul/sistem/elemen yang disepadukan

(Pengujian Integrasi); dan akhirnya semua modul yang terlibat diuji secara menyeluruh (Pengujian Sistem).

Skop pengujian dalam panduan ini memfokuskan kepada jenis dan tahap pengujian sistem aplikasi seperti berikut:

a. Pengujian keperluan fungsian (functional test)
    i. Pengujian Unit
    ii. Pengujian Integrasi
    iii. Pengujian Sistem
    iv. Pengujian Penerimaan Pengguna (UAT)
b. Pengujian keperluan bukan fungsian (non-functional test)
    i. Kualiti Kod
    ii. Pengujian SAST
    iii. Pengujian Prestasi

Terdapat 7 jenis pengujian sistem aplikasi yang akan diguna pakai dalam pelaksanaan panduan ini seperti Jadual 5-3.

Jadual 5-3: Jenis Pengujian Sistem Aplikasi berdasarkan Kuadran Pengujian *Agile*

Pengujian Keperluan Fungsian			
No.	**Jenis Pengujian**	**Keterangan**	**Skop Pengujian**
1.	Pengujian Unit	Pengujian bagi menilai unit atau komponen terkecil (lowest level) sistem aplikasi seperti fungsi, kelas, prosedur dan antara muka.	i. Dilaksanakan secara automasi menggunakan *pipeline* CI/CD dengan bantuan *tools* pengujian unit atau secara manual. ii. Skrip automasi pengujian unit disediakan oleh pembangun dengan bantuan *tools* pengujian unit.
2.	Pengujian Integrasi	Pengujian dua atau lebih modul/sistem/elemen yang disepadukan dan telah berjaya melepasi pengujian unit.	i. Dilaksanakan secara automasi menggunakan *pipeline* CI/CD atau secara manual. ii. Menguji keupayaan di antara modul/sistem/elemen untuk menerima, memproses dan mengeluarkan output dalam sistem atau persekitaran yang sama. iii. Menguji keupayaan komunikasi API. iv. Menguji semula modul/sistem/elemen dan API yang telah selesai pada *sprint* sebelumnya bersama fungsi baharu pada *sprint* semasa.

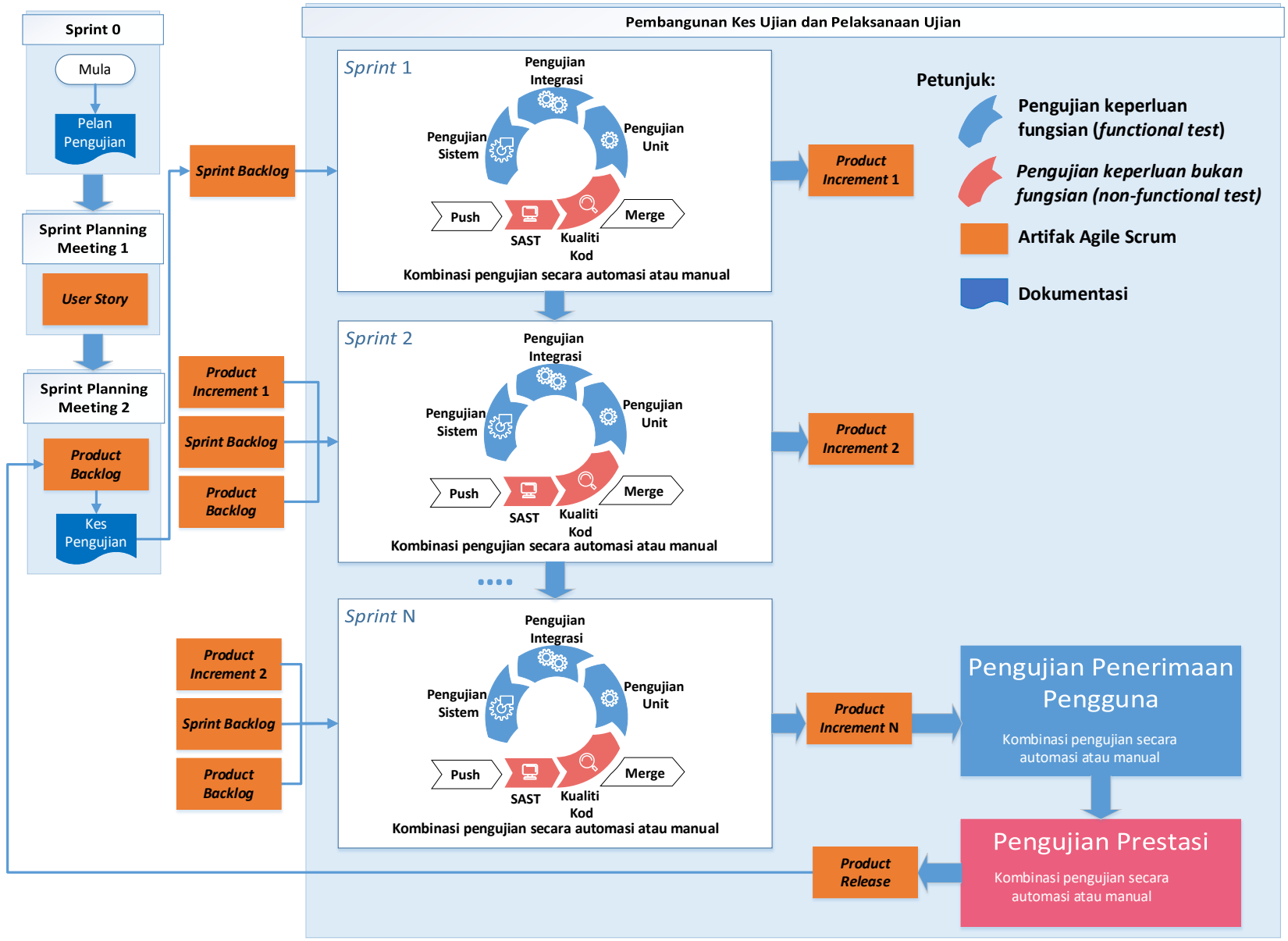
Pengujian Keperluan Fungsian			
No.	**Jenis Pengujian**	**Keterangan**	**Skop Pengujian**
3.	Pengujian Sistem	Pengujian semua modul yang terlibat pada *sprint* semasa secara menyeluruh.	i. Dilaksanakan secara automasi menggunakan *pipeline* CI/CD dengan bantuan *tools* pengujian atau secara manual. ii. Menguji keperluan fungsian (functional requirements) semua modul yang terlibat melalui antara muka pengguna mengikut kes pengujian yang telah ditetapkan. iii. Menguji semula keperluan fungsian yang telah selesai pada *sprint* sebelumnya bersama fungsi baharu pada *sprint* semasa.
4.	Pengujian Penerimaan Pengguna	Pengujian keperluan fungsian sistem aplikasi secara menyeluruh untuk memastikan sistem aplikasi memenuhi spesifikasi keperluan yang ditetapkan oleh pemilik produk.	i. Dilaksanakan secara manual oleh pemilik produk berdasarkan kes pengujian yang telah dipersetujui. ii. Menguji dan mengesahkan keperluan fungsian yang dibangunkan memenuhi spesifikasi keperluan yang telah ditetapkan oleh pemilik produk.

Pengujian Keperluan Bukan Fungsian			
No.	**Jenis Pengujian**	**Keterangan**	**Skop Pengujian**
5.	Kualiti Kod	Pengawalan kualiti kod sumber dengan memastikan pengekodan kod sumber mengikut standard yang ditetapkan.	i. Dilaksanakan secara automasi menggunakan *pipeline* CI/CD. ii. Dilaksanakan dengan menganalisis kod sumber untuk mengesan ralat dan kualiti berkaitan pengekodan kod sumber.
6.	Pengujian SAST	Pengujian kod sumber untuk mengenal pasti kelemahan atau ralat dari segi keselamatan ke atas sistem aplikasi.	i. Dilaksanakan secara automasi menggunakan *pipeline* CI/CD. ii. Menganalisis kod sumber sistem aplikasi dari segi reka bentuk dan pengekodan untuk mengenal pasti kelemahan atau ralat dari segi keselamatan yang boleh menjadi ancaman ke atas sistem aplikasi.
7.	Pengujian Prestasi	Pengujian terhadap keupayaan sistem dari segi kelajuan tindak balas dan kestabilan berdasarkan tahap bebanan yang telah ditetapkan mengikut sasaran pengguna sistem aplikasi.	i. Dilaksanakan untuk mengenal pasti bagaimana sistem bertindak di bawah beban kerja dan kekangan tertentu untuk melaksanakan sesuatu fungsi. ii. Dilaksanakan menggunakan *tools* secara automasi menggunakan *pipeline* CI/CD atau secara manual dengan bantuan *tools* pengujian prestasi.

5.1.6.3. Pelaksanaan Pengujian

Aktiviti pengekodan serta pengujian dilakukan secara *incremental* dan *iterative* semasa pembangunan dapat memastikan penghasilan produk yang berkualiti.

Setiap *sprint* menghasilkan *product increment* dan maklum balas yang diperolehi akan menjadi input kepada *sprint* seterusnya. Aktiviti pelaksanaan pengujian secara *iterative* mengikut *sprint* melalui pembangunan produk berdasarkan metodologi *Agile Scrum* adalah seperti Rajah 5-10.



Rajah 5-10: Aktiviti Pengujian Berdasarkan Metodologi *Agile Scrum*

Aktiviti pengujian berdasarkan *Agile Scrum* diterangkan seperti berikut:

a. Pembangunan Kes Pengujian

Pembangunan kes pengujian dilaksanakan bersama semasa pembangunan produk pada setiap *sprint*. Reka bentuk kes pengujian memastikan bahawa pengujian kualiti merangkumi keseluruhan keperluan pengguna seperti yang terdapat pada *user story, product backlog* dan DoD. Kes pengujian di dokumentasikan pada templat kes pengujian seperti Lampiran B-17.

Kes pengujian yang didokumenkan kemudiannya diserahkan kepada pemilik produk untuk semakan. Seterusnya pasukan pembangun akan memutuskan kes pengujian yang boleh diautomasikan.

Berikut merupakan komponen dan contoh pembangunan kes pengujian.

i. Penyediaan Templat Kes Pengujian

Kes pengujian dibangunkan berpandu pada Templat Kes Pengujian seperti di Lampiran B-17 dengan komponen seperti berikut:

Jadual 5-4: Komponen Templat Kes Pengujian

Komponen	Keterangan
ID Kes Ujian	ID unik bagi kes ujian yang merujuk kepada senario pengujian fungsi bisnes.
Nama Kes Ujian	Nama bagi kes ujian.
Keterangan Kes Ujian	Keterangan ringkas bagi kes ujian.
ID *Product Backlog*	ID unik bagi *product backlog* yang merujuk kepada senario pengujian fungsi bisnes.
Prasyarat	Prasyarat bagi membolehkan kes ujian dilaksanakan.
Input/ Langkah-Langkah Ujian	Turutan langkah-langkah dalam melaksanakan kes ujian.
Jangkaan Hasil	Jangkaan hasil bagi kes ujian.
Hasil Sebenar	Hasil sebenar yang diperolehi setelah kes ujian dilaksanakan.

Komponen	Keterangan
Status	Keputusan/status kes ujian sama ada lulus atau gagal.

ii. Contoh Pengisian Templat Kes Pengujian

Contoh Templat Kes Pengujian yang telah diisi adalah seperti Jadual 5-5.

Jadual 5-5: Contoh Pengisian Templat Kes Pengujian

ID Kes Ujian	BF-BM-EP01-PB01-TC01		
Nama Kes Ujian	Mendaftar profil pengguna baharu.		
Keterangan Kes Ujian	Sebagai Pengguna (warga MAMPU), saya BOLEH mendaftar profil pengguna baharu SUPAYA pengguna boleh mendaftar masuk sistem tempahan bilik mesyuarat.		
ID *Product Backlog*	BF-BM-EP01-PB01		
Prasyarat	Nama pengguna dan alamat email yang sah untuk didaftarkan.		
Input/ Langkah-Langkah Ujian	**Jangkaan Hasil**	**Hasil Sebenar**	**Status (Lulus/Gagal)**
1. Daftar nama dan email pengguna.	a. Antara muka daftar pengguna baharu dipaparkan. b. Sistem akan memaparkan mesej “Pendaftaran pengguna baharu berjaya”.		

b. Pelaksanaan Pengujian

Semasa pembangunan produk berdasarkan *product backlog* pada setiap *sprint*, pengujian secara berulang dilakukan untuk mengenal pasti ralat pada peringkat awal supaya pembetulan dapat dilakukan dengan segera.

Aktiviti pengujian boleh dipecahkan kepada pelaksanaan dalam atau luar *sprint*. Walau bagaimanapun, aktiviti pada luar *sprint* boleh dilaksanakan dalam *sprint* menggunakan *tools* automasi dan disahkan pada *sprint review*.

i. Pengujian dalam *sprint* – pengujian yang dilakukan secara berulang dalam *sprint* adalah seperti berikut:
    a. Pengujian SAST
    b. Kawalan Kualiti Kod
    c. Pengujian Unit
    d. Pengujian Integrasi
    e. Pengujian Sistem

ii. Pengujian luar *sprint* – pengujian yang dilakukan luar *sprint* adalah seperti berikut:
    a. Pengujian Penerimaan Pengguna
    b. Pengujian Prestasi

Tatacara pelaksanaan dan penggunaan *tools* untuk setiap aktiviti pengujian diterangkan dengan lebih lanjut pada para 5.4.3 dan para 5.4.4.

5.1.6.4. Penetapan Tahap Kritikal (Severity Level)

Severity level bermaksud tahap kritikal terhadap kegagalan fungsian atau penemuan ralat semasa pengujian. Faktor dalam menentukan keutamaan (priority) boleh dirujuk pada Lampiran B-18 sebagai panduan.

Masalah yang diperolehi semasa proses pengujian akan dianalisis dan dikategorikan kepada berdasarkan Tahap Kritikal dan kesannya, rujuk Lampiran B-18. Pasukan DevOps boleh menggunakan Tahap Kritikal untuk menentukan pelan tindakan selanjutnya seperti berikut:

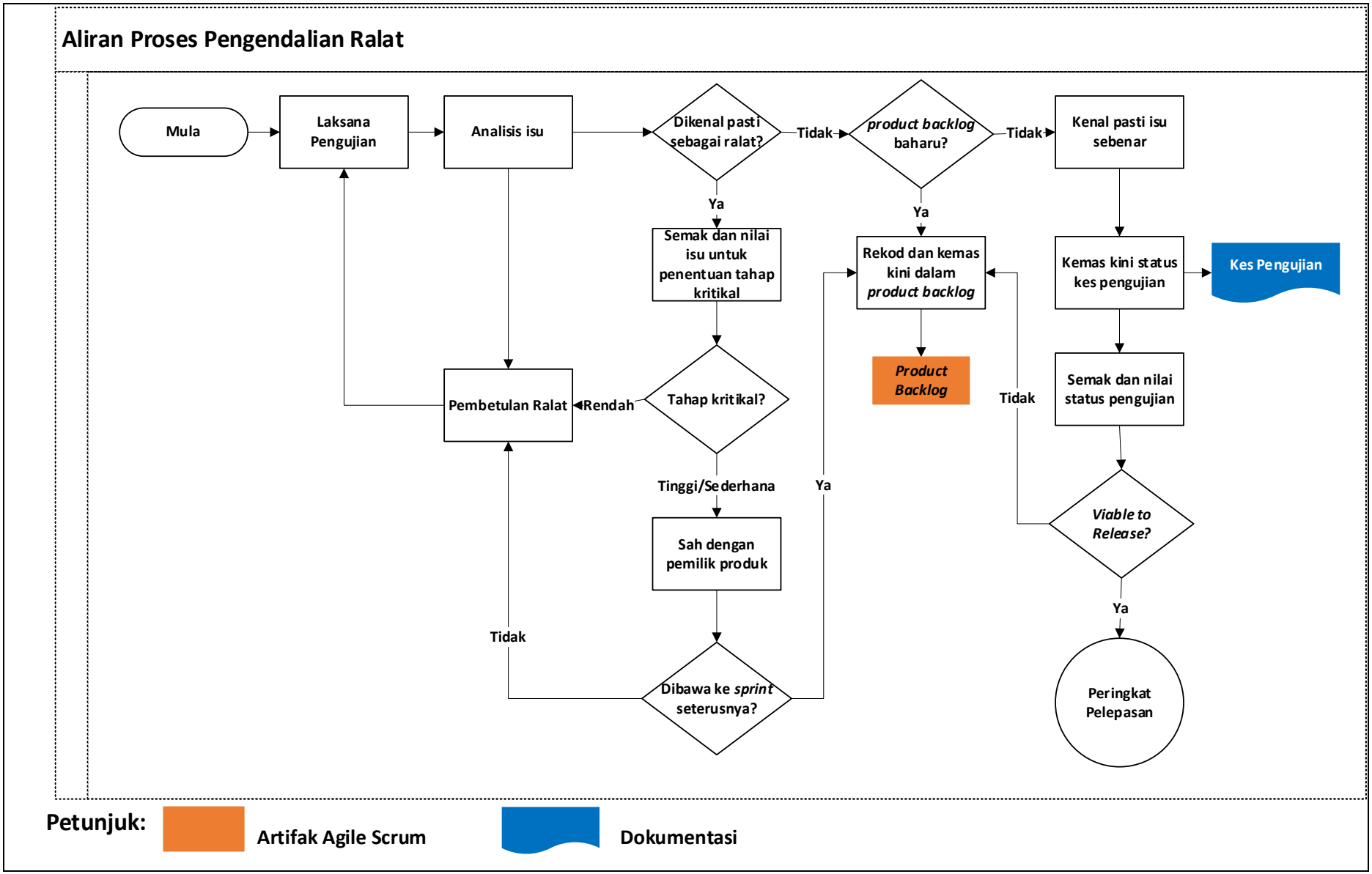
a. Penentuan keutamaan kerja terhadap item daripada senarai *product backlog*.

b. Penentuan impak ralat ke atas produk serta penetapan keutamaan kerja untuk pembetulan setiap ralat dalam pengujian.

c. Penentuan kriteria penerimaan dan kelulusan pengujian (viable to release).

a. Aliran Proses Pengendalian Ralat

Proses pengendalian ralat diterangkan dalam aliran proses pada Rajah 5-11.



Rajah 5-11: Aliran Proses Pengendalian Ralat

Berikut adalah penerangan proses dalam aliran pengendalian ralat.

Jadual 5-6: Keterangan Aliran Proses Pengendalian Ralat

Proses	Keterangan
Pelaksanaan Pengujian	Penguji melaksanakan pengujian dan melaporkan ralat yang ditemui semasa proses pengujian.
Menganalisis Isu	Pasukan pembangunan dan pemilik produk akan menganalisis isu yang dilaporkan.
Kenal pasti isu sebenar	Jika isu yang dilaporkan dikenal pasti sebagai bukan ralat, tetapi keperluan baharu, isu akan direkodkan semasa aktiviti *product backlog refinement*.
Kemas kini status kes pengujian	Wakil pasukan pembangun akan mengemas kini log isu berdasarkan ralat yang telah dikenal pasti.
Semakan dan penilaian isu bagi menentukan keutamaan dan menentukan tahap kritikal	Jika isu yang dilaporkan dikenal pasti sebagai ralat, pasukan pembangun dan penguji akan melakukan semakan dan penilaian isu bagi penentuan keutamaan dan menentukan tahap kritikal ralat.
Pembetulan ralat	Jika tahap kritikal ralat adalah rendah, pasukan pembangunan akan terus membetulkan ralat dan melaksanakan pengujian semula.
Sahkan dengan pemilik produk	Jika tahap kritikal ralat adalah tinggi atau sederhana, perbincangan di antara pasukan pembangun dan pemilik produk akan dilakukan untuk menentukan sama ada ralat harus diselesaikan pada *sprint* semasa atau *sprint* berikutnya.
Semakan dan penilaian status pengujian	Pasukan pembangun dan pemilik produk seterusnya akan memutuskan sama ada DoD dipenuhi berdasarkan status kes pengujian.

5.1.6.5. Penetapan *Traceability Matrix*

Traceability Matrix disediakan untuk menjejaki hubungan diantara keperluan (requirement) dengan kes pengujian sepanjang kitar hayat pembangunan produk. Ia berupaya memastikan:

a. Semua keperluan sistem telah dibangunkan.

b. Keperluan atau fungsi yang dibangunkan telah dilaksanakan pengujian.

c. Semua pindaan ke atas aktiviti yang berkaitan dilaksanakan.

Kelebihan penggunaan *Traceability Matrix* adalah seperti berikut:

a. Memastikan keseluruhan keperluan diuji dan dipenuhi.

b. Mengenal pasti keperluan yang tidak dinyatakan atau tidak konsisten.

Berikut merupakan komponen dan contoh *Traceability matrix.*

a. Penyediaan Templat *Traceability Matrix*

Traceability matrix disediakan berdasarkan templat seperti di Lampiran B-19 dengan komponen seperti berikut:

Jadual 5-7: Komponen Templat *Traceability Matrix*

No.	Komponen	Keterangan
1.	ID Fungsi Bisnes	ID unik bagi fungsi bisnes yang dirujuk dari Spesifikasi Keperluan Bisnes (BRS).
2.	ID User Story	ID unik bagi *user story* yang merujuk kepada ringkasan fungsi bisnes.
3.	ID *Product Backlog*	ID unik bagi *product backlog* yang merujuk kepada senario pengujian fungsi bisnes.
4.	ID Kes Ujian	ID unik bagi kes ujian yang merujuk kepada kes senario pengujian fungsi bisnes.
5.	Keterangan Kes Ujian	Keterangan ringkas bagi kes ujian.

b. Penetapan Konvensyen Nama dan Nombor *Traceability Matrix*

Contoh penetapan Konvensyen Nama dan Nombor Traceability Matrix adalah seperti Jadual 5-8.

Jadual 5-8: Contoh Penetapan Konvensyen Nama dan Nombor *Traceability Matrix*

ID Fungsi Bisnes	ID *User Story*	ID *Product Backlog*	ID Kes Ujian	Keterangan Kes Ujian
BF-BM-EP01	BF-BM-EP01-**US01**	BF-BM-EP01-**PB01**	BF-BM-EP01-PB01-**TC01**	Selenggara Profil Pengguna.
Keterangan Kandungan ID: **BF** : Ringkasan teknik fungsi bisnes **BM** : Ringkasan Fungsi Utama Nama Bilik Mesyuarat **EP** : Ringkasan *Epic* **01** : Bilangan *Epic*	Keterangan Kandungan ID: **US** : Ringkasan *User Story* **01** : Bilangan *User Story*	Keterangan Kandungan ID: **PB** : Ringkasan *Product Backlog* **01** : Bilangan *Product Backlog*	Keterangan Kandungan ID: **TC** : Ringkasan Kes Ujian **01** : Bilangan Kes Ujian	

c. Pengisian Templat *Traceability Matrix*

Contoh Templat Kes Pengujian yang telah diisi adalah seperti Jadual 5-9.

Jadual 5-9: Contoh Pengisian Templat *Traceability Matrix*

ID Fungsi Bisnes	ID *User Story*	ID *Product Backlog*	ID Kes Ujian	Keterangan Kes Ujian
BF-BM-EP01	BF-BM-EP01-US01	BF-BM-EP01-PB01	BF-BM-EP01-PB01-TC01	Mendaftar profil pengguna baharu.
			BF-BM-EP01-PB01-TC02	Memuat naik gambar profil pengguna baharu.
	BF-BM-EP01-US02	BF-BM-EP01-PB02	BF-BM-EP01-PB02-TC01	Pentadbir sistem boleh mengemas kini profil pengguna.
			BF-BM-EP01-PB02-TC02	Pengguna boleh mengemas kini profil sendiri.
BF-BM-EP02	BF-BM-EP02-US01	BF-BM-EP02-PB01	BF-BM-EP02-PB01-TC01	Pentadbir sistem boleh mewujudkan rekod bilik mesyuarat.
			BF-BM-EP02-PB01-TC02	Pentadbir sistem boleh mengemas kini rekod bilik mesyuarat sedia ada.

5.1.7. 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*.

5.2. 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).

Setelah pasukan pembangun menyelesaikan *sprint task*, mereka akan *commit* dan *push* kod sumber mereka 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 pantas, tetapi berkesan untuk mengenal pasti masalah lebih awal.

Terdapat 3 aktiviti utama pada peringkat ini:

a. Pengurusan Kod Sumber,

b. Konfigurasi *Pipeline* CI/CD dan

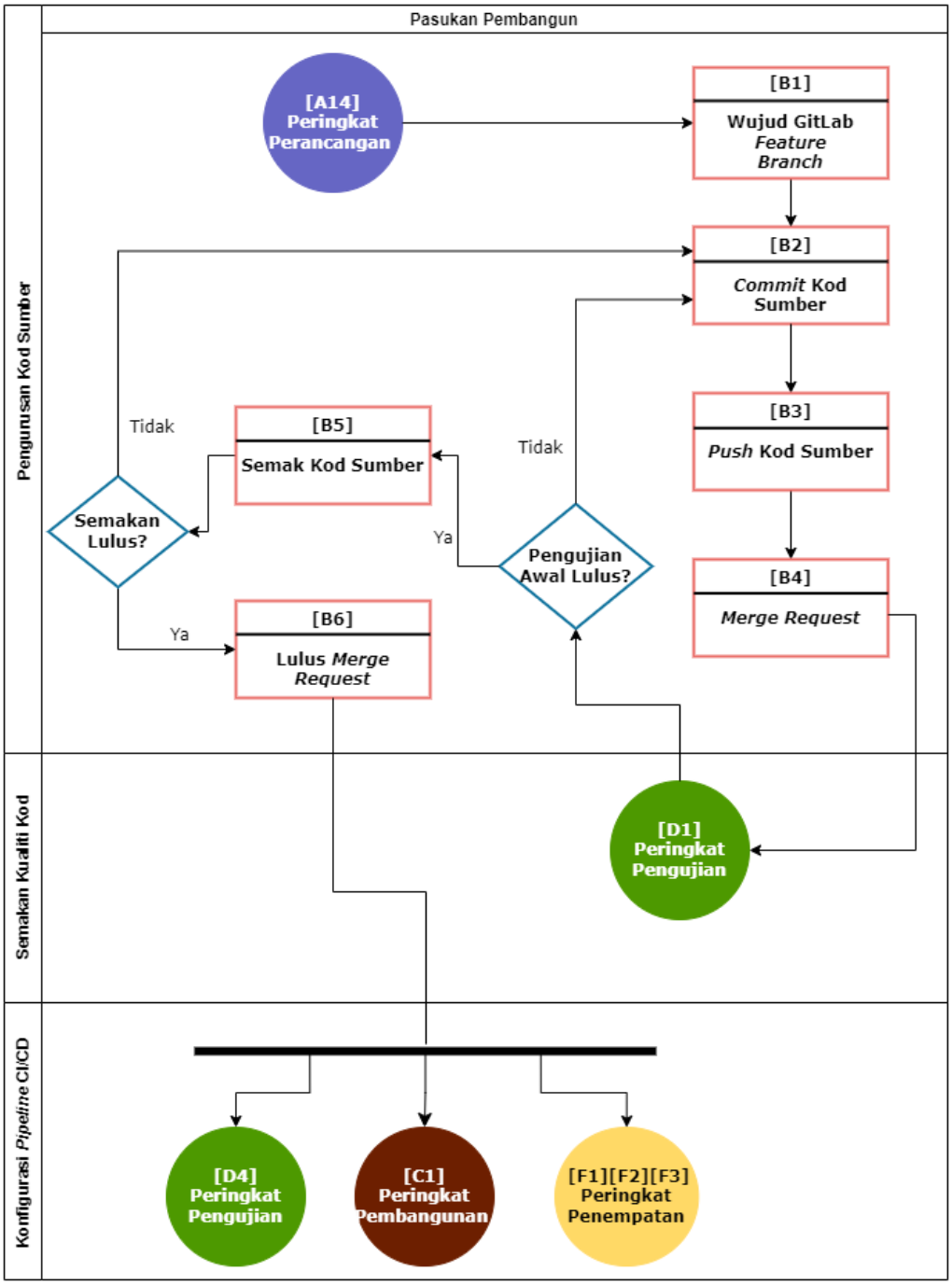
c. Semakan Kualiti Kod.

5.2.1. Prasyarat

Input bagi peringkat pengekodan melibatkan senarai perkara pada para 5.1.7.

5.2.2. 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 5-12 memaparkan aliran proses kerja yang terlibat dalam peringkat pengekodan.



Rajah 5-12: Aliran Kerja Peringkat Pengekodan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 5-12.

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.

5.2.3. 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.

5.2.3.1. 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

5.2.3.2. 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 Lampiran C-1.

b. *Commit* Kod Sumber [B2]

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.

5.2.4. 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 5.1.5.

5.2.4.1. Penggunaan *Tools*

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

5.2.4.2. Ringkasan Tatacara Penggunaan *Tools*

Rujuk Lampiran C-7 bagi contoh skrip `.gitlab-ci.yml` untuk *pipeline* CI/CD berdasarkan arkitektur yang diterangkan pada para 5.1.5.3.

5.2.5. 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.

5.2.5.1. Penggunaan *Tools*

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

5.2.5.2. 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.

5.2.6. 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.

5.3. 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.

Peringkat Pembangunan terdapat 3 aktiviti utama iaitu:

a. *Compile Code*,

b. *Package Code* dan

c. Pembinaan Kod dan Imej *Container*.

5.3.1. Prasyarat

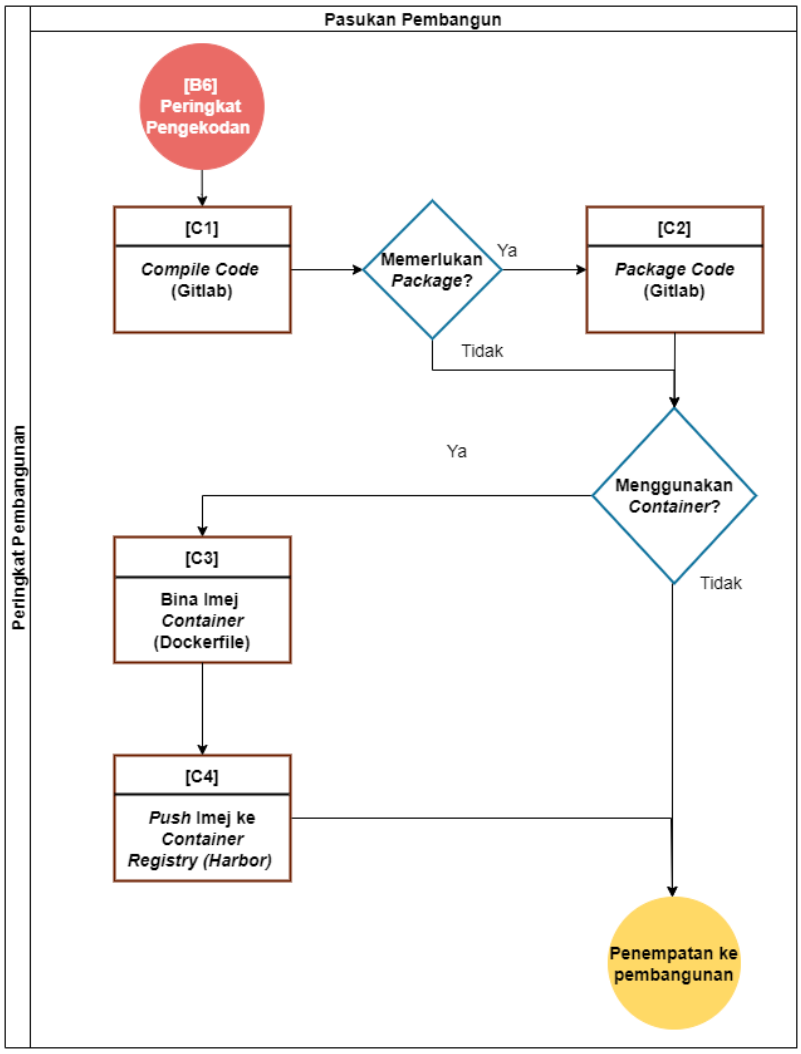
Untuk menjalankan aktiviti CI/CD berkaitan Docker *Container*, pasukan perlu:

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`.

5.3.2. 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 5-13 memaparkan aliran proses kerja yang terlibat dalam peringkat pembangunan.



Rajah 5-13: Aliran Kerja Peringkat Pembangunan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 5-13:

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.

5.3.3. *Compile Code*

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.

5.3.3.1. Penggunaan *Tools*

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

5.3.3.2. Ringkasan Tatacara Penggunaan *Tools*

a. ***Compile Code* [C1]**

   Code compiler yang digunakan bergantung kepada bahasa pengaturcaraan kod sumber. Lampiran D-1 menerangkan contoh skrip *code build* yang digunakan.

5.3.4. *Package Code*

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.

5.3.4.1. Penggunaan *Tools*

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

5.3.4.2. Ringkasan Tatacara Penggunaan *Tools*

a. *Package Code* [C2]

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*.

5.3.5. 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.

5.3.5.1. 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.

5.3.5.2. 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]

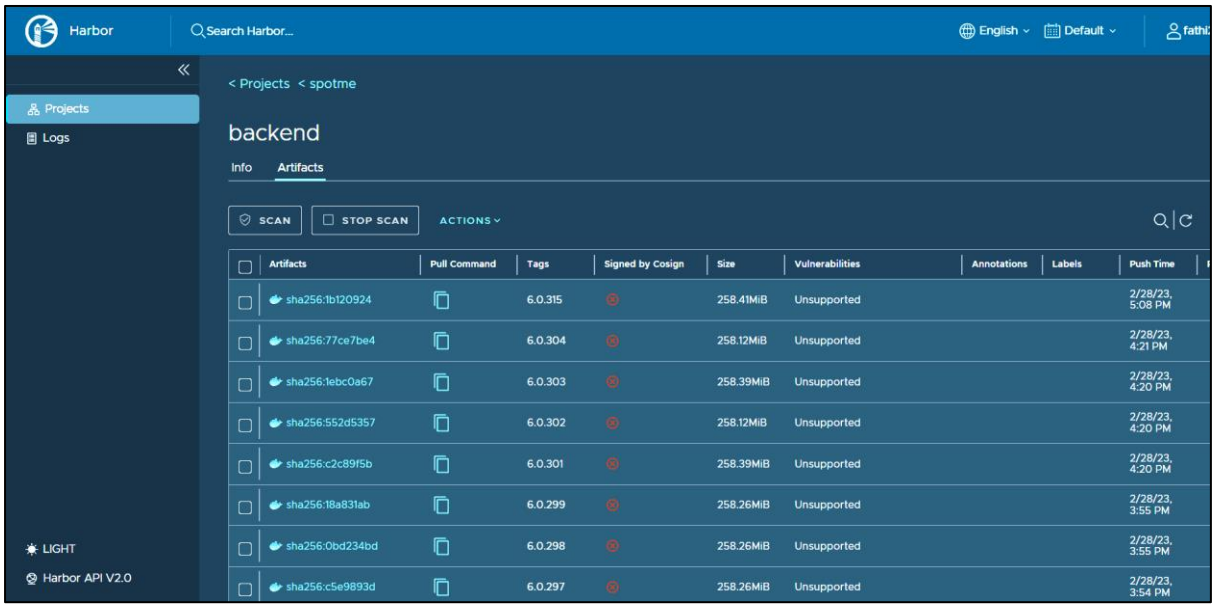
Setelah imej Docker dibina, imej tersebut perlu dihantar ke *container registry*. Rujuk Lampiran D-4 bagi contoh skrip untuk *push* imej Docker ke *container registry*.

5.3.6. 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 5-14.



Artifacts	Pull Command	Tags	Signed by Cosign	Size	Vulnerabilities	Annotations	Labels	Push Time
sha256:1b120924		6.0.315		258.41MiB	Unsupported			2/28/23, 5:08 PM
sha256:77ce7be4		6.0.304		258.12MiB	Unsupported			2/28/23, 4:21 PM
sha256:1ebc0a67		6.0.303		258.39MiB	Unsupported			2/28/23, 4:20 PM
sha256:552d5357		6.0.302		258.12MiB	Unsupported			2/28/23, 4:20 PM
sha256:c2c89f5b		6.0.301		258.39MiB	Unsupported			2/28/23, 4:20 PM
sha256:18a831ab		6.0.299		258.26MiB	Unsupported			2/28/23, 3:55 PM
sha256:0bd234bd		6.0.298		258.26MiB	Unsupported			2/28/23, 3:55 PM
sha256:c5e9893d		6.0.297		258.26MiB	Unsupported			2/28/23, 3:54 PM

Rajah 5-14: Paparan Kod Imej pada Harbor

5.4. 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.

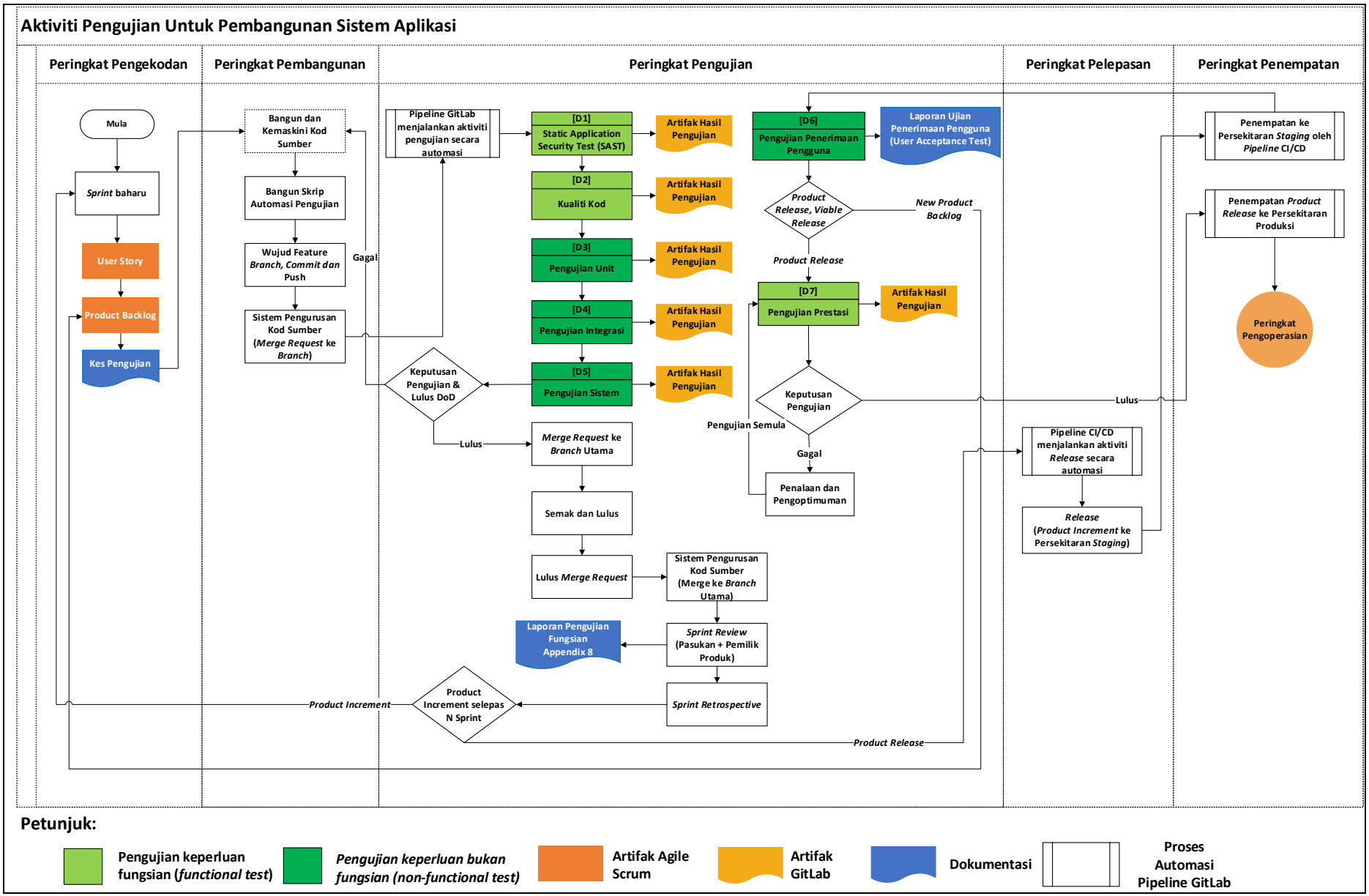
5.4.1. Prasyarat

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.
b. Persekitaran dan *tools*:
    i. Konfigurasi GitLab *Runner* supaya menyokong arahan kerja berkaitan aktiviti pengujian.
    ii. *Tools* pengujian yang menyokong bahasa pengaturcaraan yang digunakan.

5.4.2. Aliran Proses Kerja Peringkat Pengujian

Aliran proses aktiviti pengujian berdasarkan pendekatan pembangunan *Agile Scrum* pada Rajah 5-15 berikut telah diadaptasikan kepada proses pembangunan sistem aplikasi di sektor awam.



Rajah 5-15: 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 dan pengujian sistem akan dijalankan dan 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.

5.4.3. Pengujian Keperluan Fungsian (Functional Test)

5.4.3.1. Pengujian Unit [D3]

Jadual 5-10 menerangkan tatacara pelaksanaan pengujian unit.

Jadual 5-10: 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.

5.4.3.2. Pengujian Integrasi [D4]

Jadual 5-11 menerangkan tatacara pelaksanaan pengujian integrasi.

Jadual 5-11: 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.
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.

5.4.3.3. Pengujian Sistem [D5]

Jadual 5-12 menerangkan tatacara pelaksanaan pengujian sistem.

Jadual 5-12: 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-17: 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*. iv. Hasil pengujian akan dipaparkan pada log *pipeline* CICD.
Pengujian Semula	i. Mengenal pasti skrip kes pengujian yang terlibat pada *sprint*

(Regression Testing)	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.

5.4.3.4. 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 5-13 menerangkan tatacara pelaksanaan pengujian penerimaan pengguna.

Jadual 5-13: 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. 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.

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

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

Jadual 5-14 menerangkan tatacara pelaksanaan pengujian SAST.

Jadual 5-14: 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* 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 menjalankan pengujian SAST secara automasi. iii. Menganalisis dan menyemak hasil pengujian. *Tools* pengujian akan mengenal pasti dan mencadangkan penambahbaikan ke atas kod sumber.

Perkara	Keterangan
	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.

5.4.4.2. Kualiti Kod [D2]

Kualiti kod akan menganalisis kod sumber dari segi kualiti dan kerumitan struktur pengekodan. Jadual 5-15 menerangkan tatacara pelaksanaan kawalan kualiti kod.

Jadual 5-15: 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.

Perkara	Keterangan
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.

5.4.4.3. Pengujian Prestasi [D7]

Jadual 5-16 menerangkan tatacara pelaksanaan pengujian prestasi.

Jadual 5-16: 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.

Perkara	Keterangan
	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-8.

5.4.5. 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

5.5. 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.

Peringkat pelepasan melibatkan 3 persekitaran utama iaitu:

a. Pelepasan ke persekitaran pembangunan,

b. Pelepasan ke persekitaran *staging* dan

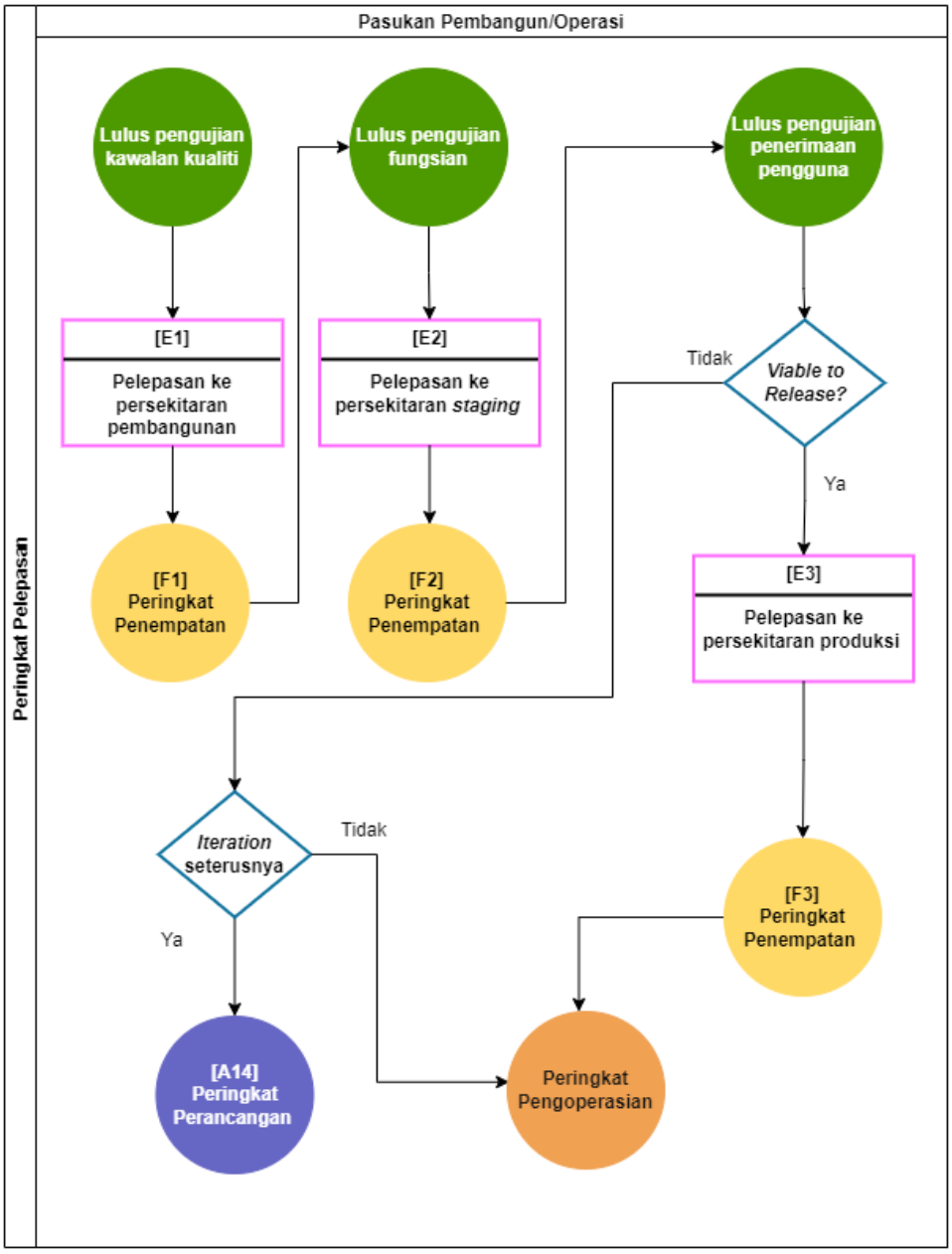
c. Pelepasan ke persekitaran produksi.

5.5.1. 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.

5.5.2. Aliran Proses Kerja Peringkat Pelepasan

Pelepasan produk bergantung kepada persekitaran yang ingin dilepaskan. Rajah 5-16 memaparkan aliran proses kerja yang terlibat dalam peringkat pelepasan.



Rajah 5-16: Aliran Kerja Peringkat Pelepasan

Berikut adalah penerangan proses yang berlaku dalam GitLab berdasarkan Rajah 5-16:

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.

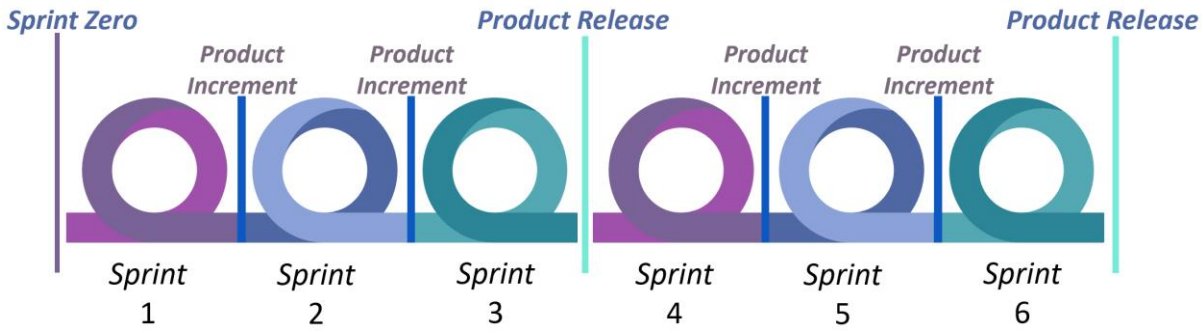
5.5.3. Pelepasan Sistem Aplikasi

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.

5.5.3.1. *Product Release*

Product release adalah *product increment* yang terdiri dari beberapa *sprint* di mana produk direka, dibangunkan, diuji dan ditempatkan seperti Rajah 5-17.

Product increment yang terdiri dari setiap *sprint* 1 hingga *sprint* 3 dan dari setiap *sprint* 4 hingga *sprint* 6 akan dilepaskan kepada pengguna sebagai *product release*. Penetapan aktiviti ini boleh berubah mengikut kesesuaian produk dan hasil perbincangan bersama ahli pasukan *scrum*.



Rajah 5-17: Aktiviti *Sprint* dan *Product Release*

Pelepasan membolehkan *snapshot* sistem aplikasi disediakan untuk pengguna. Antara perkara yang dijana bagi aktiviti ini adalah seperti berikut:

a. *Snapshot* kod sumber

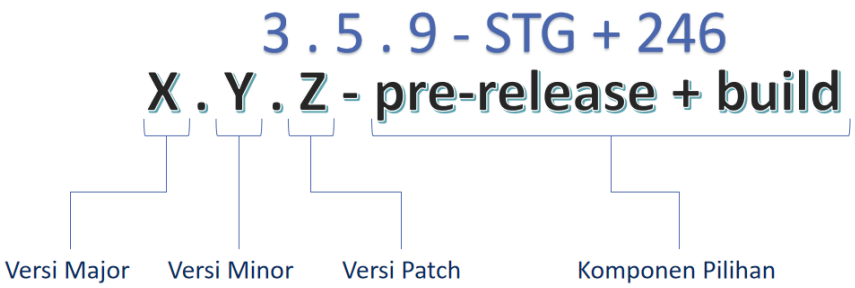
b. *Generic packages*

c. Fail metadata yang berkaitan dengan versi pelepasan

d. Nota pelepasan (release note)

5.5.3.2. Kaedah Penetapan Versi *Release*

Pelepasan sistem aplikasi ke setiap persekitaran memerlukan penetapan versi untuk menyelaras penempatan, penambahbaikan dan integrasi antara sistem dengan lebih efisien. Kaedah yang *Semantic Versioning* atau lebih dikenali sebagai SemVer merupakan antara kaedah yang biasa digunakan dalam penetapan versi pada sistem aplikasi.



Rajah 5-18: *Semantic Versioning*

Penggunaan SemVer merangkumi tiga komponen teras, ditulis dalam bentuk x,y,z dimana x,y dan z adalah nombor bulat. Manakala komponen pilihan terdiri daripada *pre-release* dan *build.*

Jadual 5-17 akan menerangkan penggunaan SemVer berdasarkan Rajah 5-18.

Jadual 5-17: Jadual Penerangan *Semantic Versioning*

Komponen	Contoh	Penerangan
Versi Major (X)	**3**.0.0	i. X merujuk kepada versi major. ii. Penambahan nombor berlaku apabila terdapat perubahan pada sistem aplikasi yang melibatkan *API* terjejas. iii. Apabila nombor versi major ditambah, nombor versi minor dan versi *patch* akan ditetapkan semula kepada sifar. iv. Contoh versi semasa 3.5.9, maka versi seterusnya adalah 4.0.0.
Versi Minor (Y)	3.**5**.0	i. Y merujuk kepada versi minor. ii. Penambahan nombor berlaku apabila fungsi baru dilepaskan ke persekitaran produksi tanpa melibatkan perubahan pada API. iii. Apabila nombor versi minor ditambah, nombor versi *patch* akan ditetapkan semula kepada sifar. iv. Contoh versi semasa 3.5.9, maka versi seterusnya adalah 3.6.0.
Versi *Patch* (Z)	3.5.**9**	i. Z merujuk kepada versi *patch.* ii. Penambahan nombor berlaku apabila terdapat pembetulan ralat dan tiada perubahan fungsi pada sistem aplikasi tersebut. iii. Tiada limit nombor bagi penambahan versi *patch.* iv. Contoh versi semasa 3.5.9, maka versi seterusnya adalah 3.5.10.
Pre-release	STG	i. *Pre-release* merupakan komponen pilihan dan merujuk kepada persekitaran penempatan bagi tujuan pengujian. ii. Sekiranya pelepasan untuk persekitaran produksi, komponen *pre-release* tidak perlu dinyatakan. iii. Contoh **DEV** untuk persekitaran pembangunan dan **STG** untuk persekitaran *staging.*
Build	246	i. *Build* merupakan komponen pilihan dan merujuk kepada *Pipeline* ID daripada GitLab CI/CD.

5.5.3.3. 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.

5.5.3.4. 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*

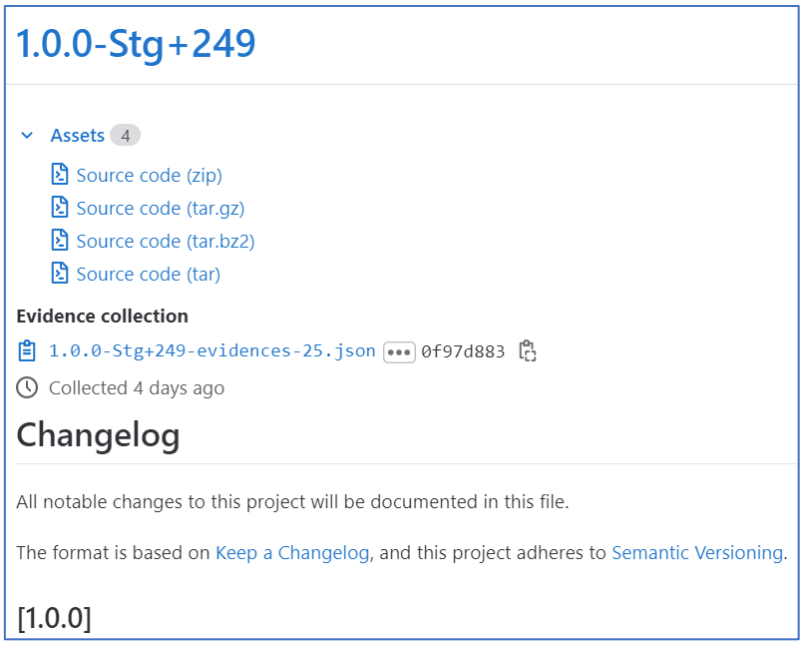
Pelepasan sistem aplikasi ke persekitaran *staging* boleh dilakukan dengan menggunakan *pipeline* CI/CD. Tatacara pelepasan ke persekitaran *staging* boleh dirujuk pada Lampiran F-2.

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.

5.5.4. Serahan/Output

Nota pelepasan boleh dilihat pada artifak GitLab *Releases*. Rajah 5-19 memaparkan artifak daripada GitLab *Releases.*



Rajah 5-19: Paparan Nota Pelepasan pada GitLab *Releases*

5.6. 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.

5.6.1. 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.

5.6.2. 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

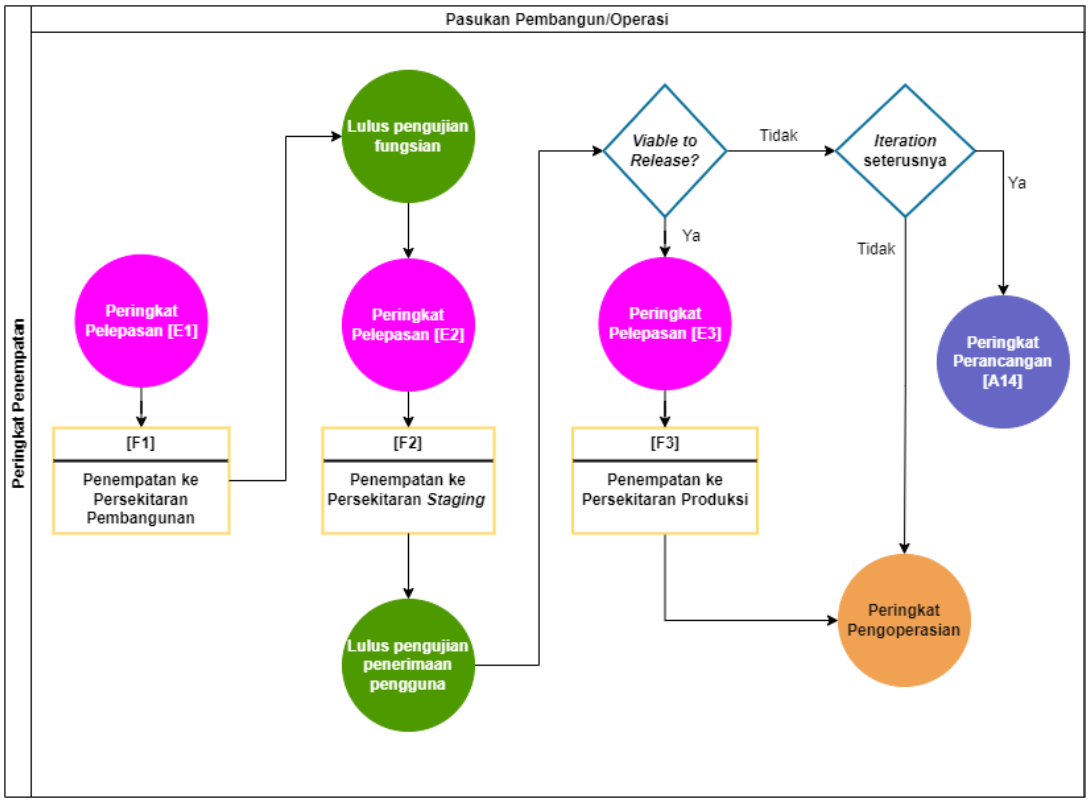
c. Produksi

Pelayan

a. *Virtual Machine* (VM)

b. Docker *Container*

c. Kubernetes



Rajah 5-20: Aliran Kerja Peringkat Penempatan

Berikut adalah penerangan proses penempatan yang berlaku pada *pipeline* CI/CD berdasarkan Rajah 5-20.

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.

5.6.3. 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.

5.6.3.1. 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.*

5.6.3.2. 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-1.

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-2.

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-3.

5.6.4. Serahan/Output

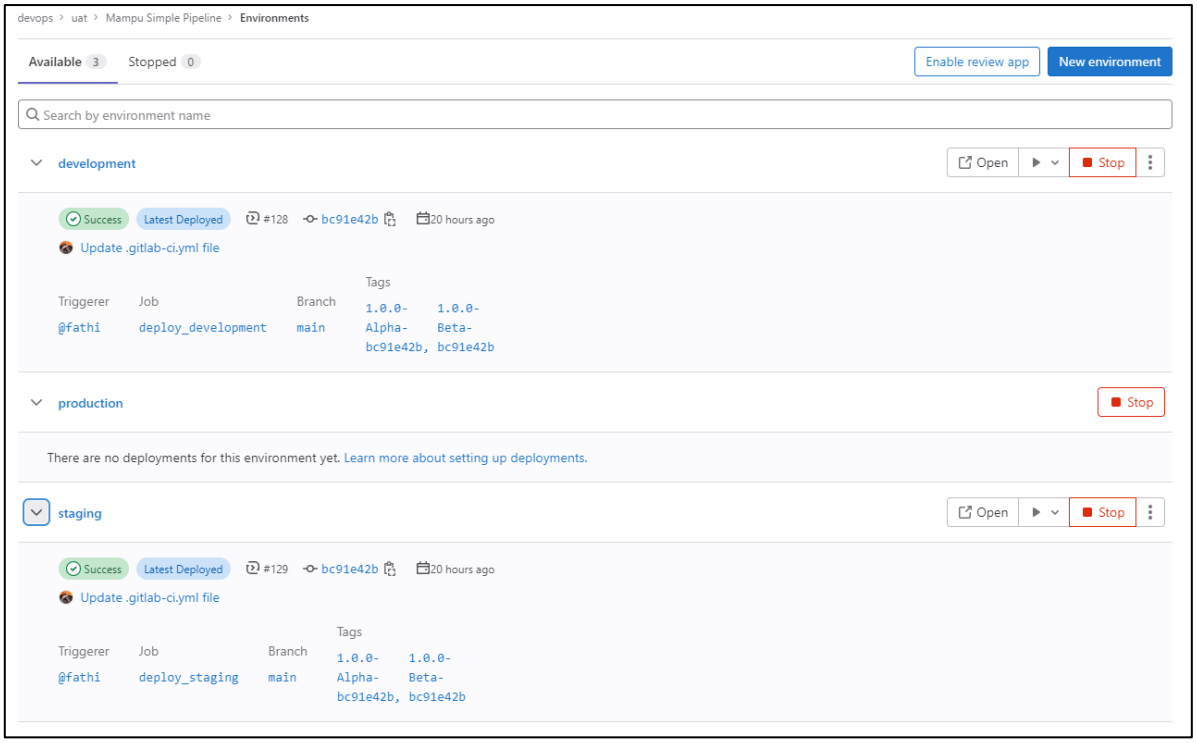
Status senarai persekitaran penempatan sistem aplikasi yang telah dijalankan adalah seperti paparan pada Rajah 5-21.

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 5-21.



Rajah 5-21: Paparan *Deployment Environment* pada GitLab

5.7. 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).

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).

5.7.1. Pengenalan Konsep *Infrastructure as a Code*

IaC adalah pendekatan pengurusan infrastruktur sistem aplikasi secara automasi dengan penyepaduan penggunaan *pipeline* CI/CD dan *tools* yang bersesuaian. Penerapan IaC memberi penekanan kepada aktiviti pengurusan infrastruktur sistem aplikasi yang dilaksanakan secara peningkatan (incremental) dan pengulangan (iterative) seperti pengurusan dan penyelenggaraan sistem pengoperasian, pelayan, pangkalan data dan penempatan sistem aplikasi.

5.7.1.1. Penggunaan IaC dalam DevOps

Contoh penggunaan IaC dalam DevOps adalah seperti berikut:

a. Pengautomasian aktiviti pengurusan dan penyelenggaraan berkala terhadap infrastruktur sistem aplikasi.

b. Pembangunan persekitaran penempatan dan konfigurasi dilaksanakan dengan lebih konsisten melalui pengautomasian.

c. Penyeragaman kemas kini versi dan sistem aplikasi merentasi pelbagai persekitaran penempatan dengan lebih efisien.

Beberapa contoh *tools* IaC adalah:

a. Ansible

b. Chef

c. Puppet

d. Terraform

e. Vagrant

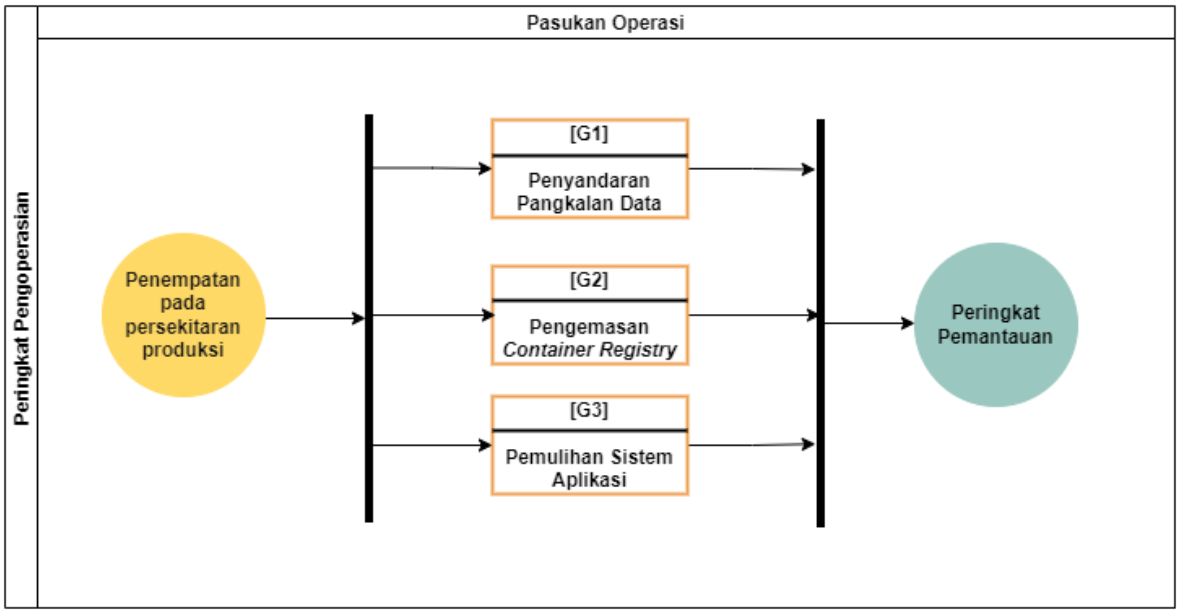
5.7.2. Prasyarat

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.

5.7.3. Aliran Proses Kerja Peringkat Pengoperasian



Rajah 5-22: Aliran Kerja Peringkat Pengoperasian

Berikut adalah penerangan proses yang berlaku pada peringkat pengoperasian berdasarkan Rajah 5-22.

5.7.4. 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.

5.7.4.1. 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

5.7.4.2. Ringkasan Tatacara penggunaan *Tools*

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

5.7.5. 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.

5.7.5.1. 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.

5.7.5.2. Ringkasan Tatacara Penggunaan *Tools*

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

5.7.6. 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.

5.7.6.1. Penggunaan *Tools*

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

5.7.6.2. Ringkasan Tatacara Penggunaan *Tools*

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

5.7.7. 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.

5.8. PERINGKAT PEMANTAUAN

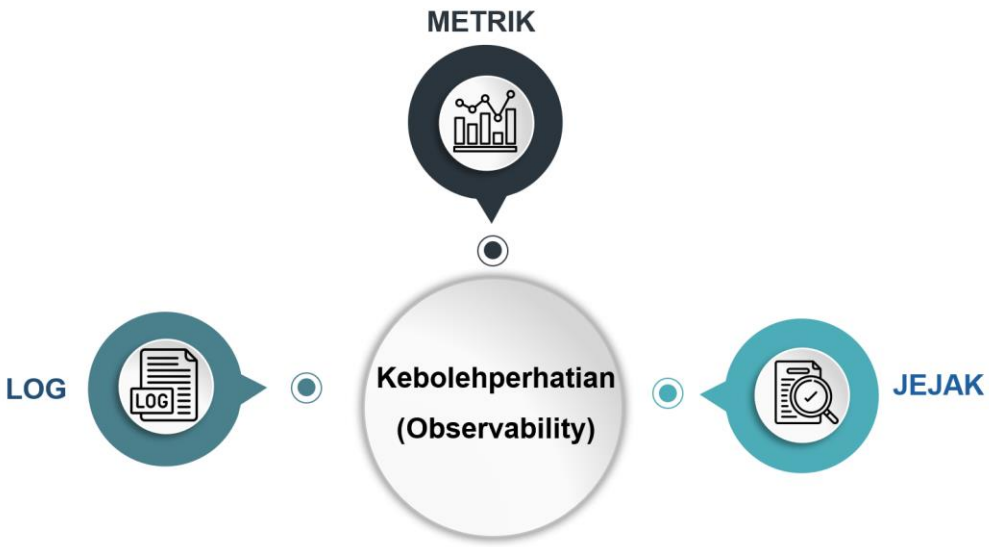
Pemantauan adalah proses menjejak, menganalisis dan memaparkan laporan prestasi sistem aplikasi untuk membantu pasukan operasi mengenal pasti dan menyelesaikan isu dengan lebih efisien. Aktiviti pemantauan secara berterusan melalui pengumpulan dan analisis data pemantauan adalah seperti berikut:

a. Pemantauan sistem
    i. Pemantauan log.
    ii. Pemantauan infrastruktur.
    iii. Pemantauan *uptime.*
b. Pemantauan prestasi sistem aplikasi
    i. Pemantauan Prestasi Aplikasi (APM).
    ii. Pemantauan Pengguna Sebenar (RUM).
    iii. Pemantauan sintetik.
c. Perancangan kapasiti (capacity planning).
d. Pemantauan maklum balas pengguna.

5.8.1. Pengenalan Konsep Kebolehperhatian (Observability)

Kebolehperhatian ditakrifkan sebagai bagaimana pasukan membuat tafsiran terhadap keadaan sistem berdasarkan output yang diperoleh daripada aktiviti pemantauan. Dengan penggunaan arkitektur dan infrastruktur *cloud-native* seperti *microservices* dan *container*, konsep kebolehperhatian membolehkan pasukan melaksanakan aktiviti berikut:

a. Pemantauan sistem dan infrastruktur yang kompleks dengan lebih berkesan,
b. Pemantauan dari pelbagai sumber data dapat di satukan untuk memberikan analisis yang lebih lengkap dan
c. Pemantauan secara automasi dapat ditingkatkan dengan penggunaan *tools* yang bersesuaian.



Rajah 5-23: Tiga Komponen yang Menyokong Konsep Kebolehperhatian

Terdapat 3 komponen utama yang menyokong konsep kebolehperhatian seperti Rajah 5-23 iaitu log, metrik dan jejak (trace). Tiga komponen ini memberikan data dan analisis yang berbeza berkaitan prestasi sistem. Apabila data dari tiga komponen ini dianalisis bersama, gambaran lengkap berkaitan keadaan sistem, sistem aplikasi dan infrastruktur dapat diperoleh.

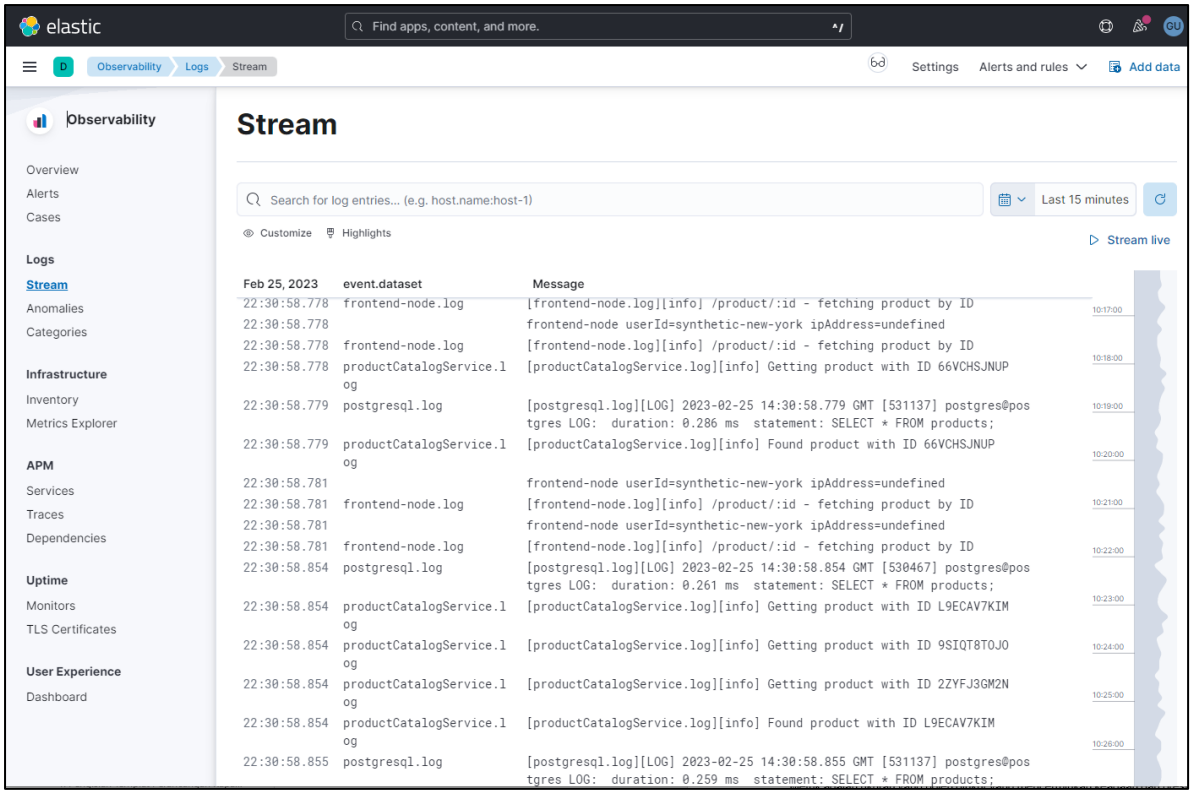
a. Log

Log adalah rekod peristiwa yang berlaku pada masa tertentu dan mengandungi maklumat seperti waktu dan peristiwa yang berlaku. Log terdiri daripada format berikut.

Jadual 5-18 : Format Log

Format Log	Penerangan
Log teks kosong	Mengandungi data berbentuk teks.
Log berstruktur	Mengandungi data tambahan seperti metadata.
Log binari	Mengandungi data berbentuk jujukan bait.

Log yang diperoleh dari pelbagai sumber akan dipaparkan seperti Rajah 5-24 menggunakan aplikasi pemantauan log iaitu Elastic Observability.



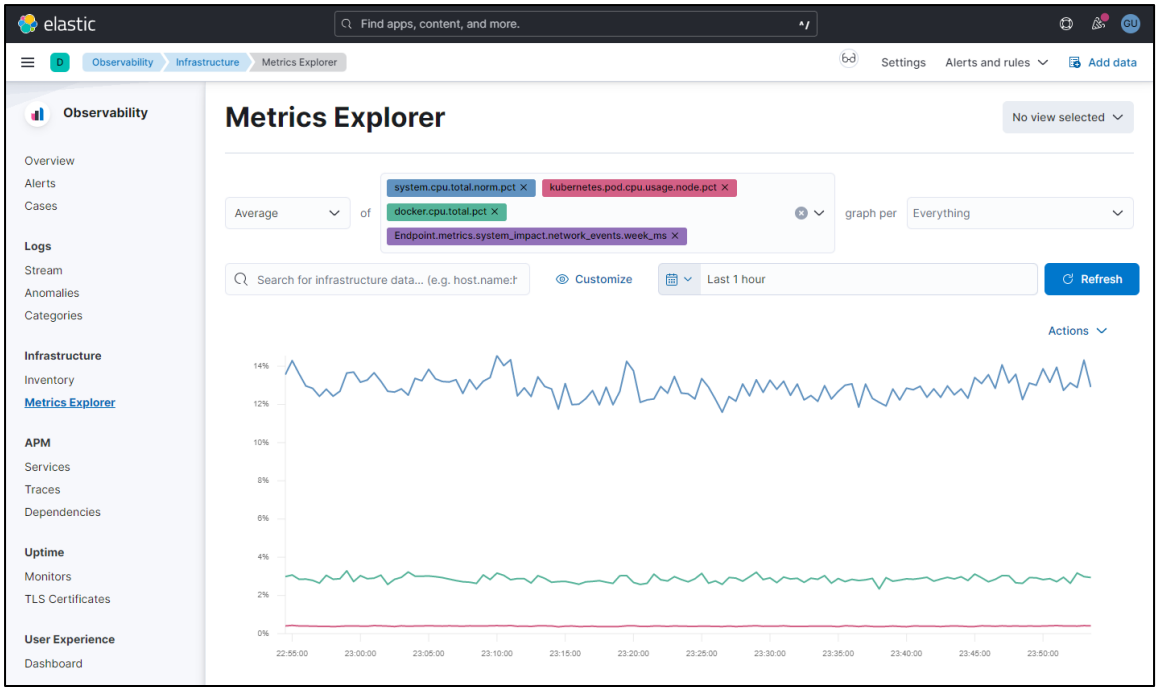
Rajah 5-24: Paparan Log daripada Elastic Observability

b. Metrik

Metrik adalah nilai angka berstruktur yang diukur dalam tempoh masa tertentu dan disimpan secara data siri masa. Metrik diperoleh dari prestasi sistem adalah seperti masa operasi sistem, masa tindak balas, bilangan permintaan sesaat dan jumlah kuasa pemprosesan atau memori yang digunakan sistem aplikasi. Pasukan boleh menggunakan pangkalan data siri masa sebagai pelan tindakan masa hadapan dan unjuran untuk memahami perkara yang berlaku pada sistem.

Sebagai contoh, pemantauan terhadap bilangan permintaan sesaat dalam perkhidmatan HTTP menunjukkan peningkatan trafik yang mendadak. Metrik memberikan data secara siri masa dan memaparkan perubahan trend bagi sebarang tindakan susulan untuk mengenal pasti punca lonjakan.

Rajah 5-25 merupakan metrik yang diperoleh daripada aplikasi pemantauan metrik iaitu Elastic Observability.



Rajah 5-25: Paparan Metrik daripada Elastic Observability

c. Jejak

Jejak mewakili kitaran perjalanan untuk tindakan atau permintaan semasa merentasi beberapa sistem seperti aplikasi *container* atau arkitektur *microservices*. Analisis terhadap data jejak membolehkan pasukan mengukur tahap prestasi sistem, menentukan punca perubahan prestasi sistem, mengenal pasti dan menyelesaikan isu dengan kadar segera.

Di antara jejak yang boleh digunakan untuk mengenal pasti punca perubahan prestasi sistem adalah seperti berikut:

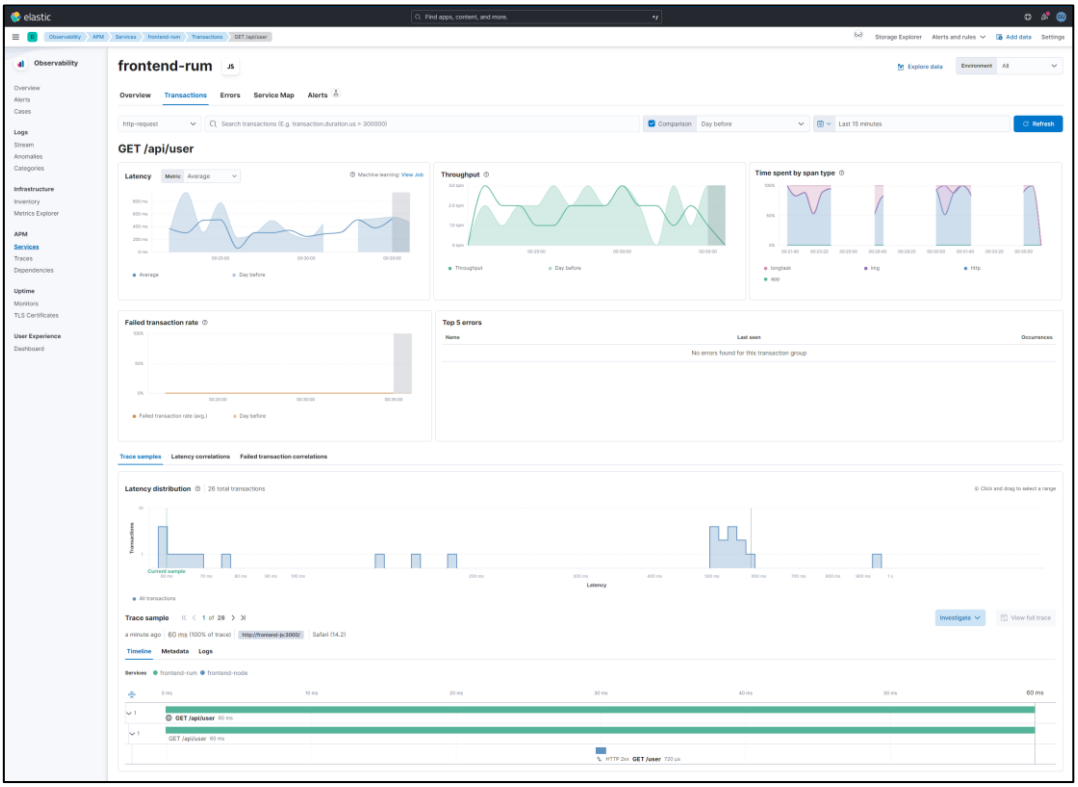
i. *API Queries*

ii. *Server-to-Server Workload*

iii. *Internal API Calls*

iv. *Frontend API Traffic*

Rajah 5-26 merupakan data jejak yang diperoleh daripada aplikasi pemantauan jejak iaitu Elastic Observability.



Rajah 5-26: Paparan Jejak *API Queries* daripada Elastic Observability

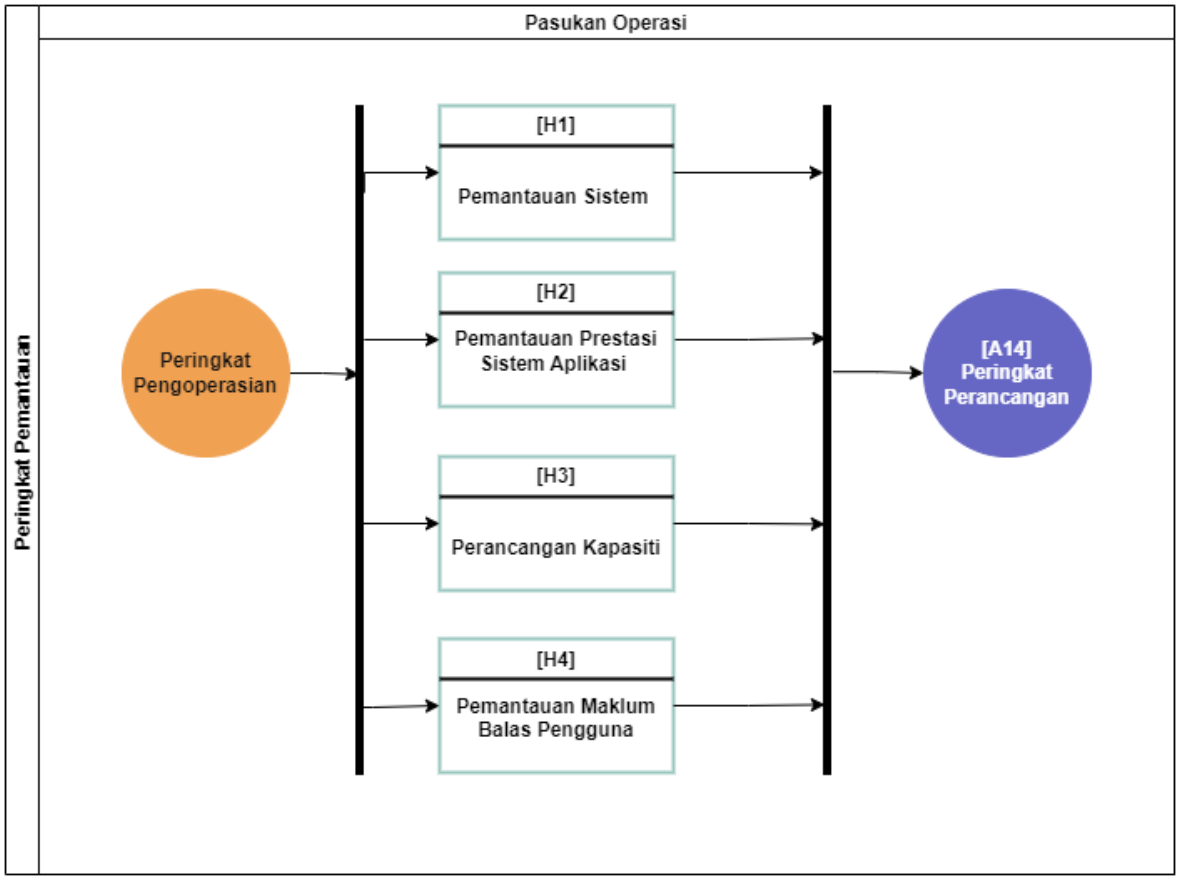
5.8.2. Prasyarat

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.

5.8.3. Aliran Proses Kerja Peringkat Pemantauan



Rajah 5-27: Aliran Proses Kerja Peringkat Pemantauan

Berikut adalah penerangan peringkat pemantauan yang berlaku berdasarkan Rajah 5-27.

5.8.4. Pemantauan Sistem

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

a. Pemantauan log,
b. Pemantauan infrastruktur dan
c. Pemantauan *uptime*.

5.8.4.1. Pemantauan Log

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:

i. Log *container* seperti Docker dan Kubernetes,
ii. Log pangkalan data seperti MySQL dan PostgreSQL,
iii. Log web *server* seperti Nginx dan Apache Tomcat,
iv. Log sistem pengoperasian seperti Linux dan Windows dan
v. 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.

5.8.4.2. Pemantauan Infrastruktur

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

a. Metrik Pemantauan Infrastruktur

Jadual 5-19: Metrik Pemantauan Infrastruktur

Perkara	Penerangan
Penggunaan CPU	Menunjukan peratusan kadar penggunaan CPU yang memberi kesan kepada prestasi.
Penggunaan Memori	Menunjukkan peratusan saiz memori yang digunakan semasa program dijalankan.
Penggunaan Storan	Menunjukkan peratusan saiz storan yang digunakan untuk menyimpan fail, gambar dan kandungan yang lain.

b. Penggunaan *Tools*

Tools yang digunakan pada aktiviti pemantauan infrastruktur adalah Elastic Observability.

c. Ringkasan Tatacara Penggunaan *Tools*

Tatacara pemantauan infrastruktur boleh dirujuk pada Lampiran I-2.

5.8.4.3. Pemantauan *Uptime*

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. Metrik Pemantauan *Uptime*

Aktiviti pemantauan *uptime* adalah berdasarkan metrik berikut.

Jadual 5-20: Metrik Pemantauan *Uptime*

Perkara	Penerangan
Memantau HTTP	Pemantauan ketersediaan dan tempoh masa tindak balas sistem aplikasi.
Memantau ICMP	Pemantauan kebolehcapaian dan ketersediaan server pada rangkaian.
Memantau TCP	Pemantauan kebolehcapaian dan ketersediaan port perkhidmatan sistem aplikasi.

b. Penggunaan *Tools*

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

c. 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.

5.8.5. Pemantauan Prestasi Sistem Aplikasi

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.

5.8.5.1. Pemantauan Prestasi Aplikasi

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 aplikasi memfokuskan pada pemantauan 5 komponen utama berikut terhadap prestasi aplikasi.

i. *Runtime* seni bina aplikasi.
ii. Pemantauan pengguna sebenar.
iii. Transaksi aplikasi.
iv. Pemantauan komponen.
v. Analisis dan pelaporan.

a. Penggunaan *Tools*

Tools yang akan digunakan pada aktiviti pemantauan adalah Elastic Observability.

b. Ringkasan Tatacara Penggunaan *Tools*

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

5.8.5.2. Pemantauan Pengguna Sebenar (RUM)

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:

i. Peranti pengguna.
ii. Versi pelayar.
iii. Tindakan yang diambil oleh pengguna.
iv. Maklum balas daripada sistem aplikasi.
v. 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.

5.8.5.3. Pemantauan Sintetik

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.

5.8.6. Perancangan Kapasiti Sistem (Capacity Planning)

Perancangan kapasiti bertujuan memastikan perkhidmatan sistem aplikasi dan kapasiti infrastruktur dapat menyokong penyampaian perkhidmatan berdasarkan sasaran tahap perkhidmatan yang telah ditetapkan. Selain itu, perancangan kapasiti boleh digunakan sebagai asas perancangan untuk permohonan sumber tambahan yang diperlukan untuk menyokong unjuran pertumbuhan dan pengoptimuman sistem aplikasi dan infrastruktur pada masa hadapan.

Templat berikut menyediakan panduan dan maklumat yang digunakan untuk merancang keperluan kapasiti.

a. Penyediaan Templat Perancangan Kapasiti

Membangunkan perancangan kapasiti dengan berpandukan maklumat-maklumat seperti Jadual 5-21.

Jadual 5-21: Kandungan Templat Perancangan Kapasiti

Item	Keterangan
Perkhidmatan/Item Dipantau	Penerangan berkenaan perkhidmatan atau item yang dianalisis.
Keperluan Kapasiti	Keperluan dan pengukuran kapasiti.
% Peningkatan Diperlukan Setiap Tahun	Unjuran peningkatan untuk setiap tahun.
Nilai Ambang	Nilai tahap ambang di mana perkhidmatan atau item melepasi had yang ditetapkan.
Pelan Tindakan	Strategi tindak balas kepada had kapasiti.

b. Analisis Perancangan Kapasiti

Analisis perancangan kapasiti adalah senario yang dianalisis dari segi impak proses bisnes supaya pasukan boleh menentukan keperluan kapasiti dengan optimum.

Berikut digariskan perkara yang menjadi pertimbangan untuk membangunkan analisis perancangan kapasiti terhadap penggunaan sistem aplikasi.

i. Keperluan kapasiti CPU.

ii. Keperluan memori.

iii. Kapasiti rangkaian.

iv. Keperluan kapasiti storan.

c. Pengisian Templat Perancangan Kapasiti

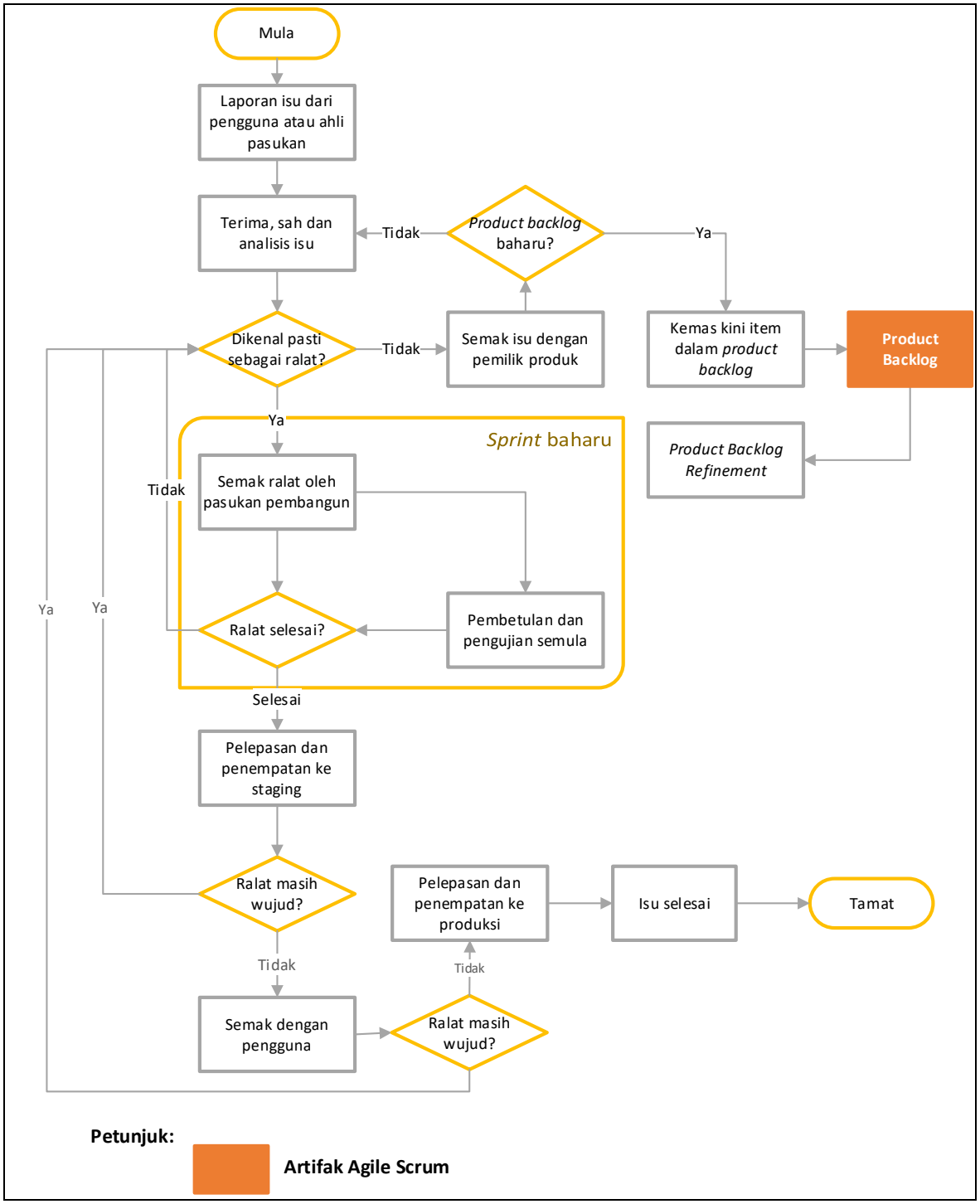
Contoh Templat Perancangan Kapasiti yang telah diisi adalah seperti Jadual 5-22.

Jadual 5-22: Contoh Pengisian Templat Perancangan Kapasiti

Perkhidmatan/Item Dipantau	Keperluan Kapasiti	% Peningkatan Diperlukan Setiap Tahun	Nilai Ambang	Pelan Tindakan
Storan Penyimpanan Data Sistem Aplikasi	10 MB per profil pengguna	5%	80%	Memantau peningkatan storan dari Elastic Observability dan juga pelan peluasan sistem aplikasi.

5.8.7. Pemantauan Maklum Balas Pengguna

Maklum balas pengguna adalah penting dalam membantu meningkatkan kualiti sistem aplikasi. Saluran maklum balas seperti meja bantuan boleh digunakan sebagai medium pengantara untuk pengguna memberikan maklum balas berkaitan sistem aplikasi.



Rajah 5-28: Aliran Proses Pemantauan Maklum Balas Pengguna

Berikut adalah penerangan proses dalam aliran proses pemantauan maklum balas pengguna berdasarkan Rajah 5-28.

Jadual 5-23: Keterangan Aliran Pemantauan Maklum Balas Pengguna

Proses	Keterangan
Laporan isu dari pengguna atau ahli pasukan.	Pengguna atau ahli pasukan akan melaporkan isu yang ditemui semasa proses penggunaan sistem aplikasi di persekitaran produksi.
Terima, sah dan analisis isu.	Meja bantuan akan mengesahkan penerimaan isu dan mendapatkan maklumat lanjut berkenaan isu yang dilaporkan dari pelapor.
Kenal pasti isu.	Jika isu yang dilaporkan dikenal pasti sebagai permintaan baharu, isu akan dihantar ke pemilik produk untuk pengesahan seterusnya direkodkan sebagai item baharu dalam *product backlog*.
Semak isu dengan pemilik produk.	Isu yang disahkan sebagai permintaan baru akan dibincangkan secara lanjut semasa sebagai *product backlog refinement* pada *sprint planning meeting*. Jika isu tersebut memerlukan penjelasan dan maklumat lanjut dari pelapor, isu akan di ajukan semula ke meja bantuan untuk semakan.
Semak ralat oleh pasukan pembangun.	Jika isu dikenal pasti sebagai ralat, ralat tersebut akan di ajukan ke pasukan pembangun untuk semakan. Pasukan pembangun memasukan ralat yang diterima ke dalam *product backlong* untuk pembetulan yang akan dilaksanakan dalam *sprint* ditetapkan.
Pembetulan dan pengujian semula.	Ralat yang telah dikemas kini dan diuji akan dilepaskan serta ditempatkan pada persekitaran *staging* untuk pengesahan dengan pengguna.
Semak dengan pengguna.	Semakan dengan pengguna dilaksanakan untuk pengesahan penerimaan. Jika ralat masih wujud, semakan semula akan dilakukan untuk mengenal pasti isu sebenar.
Penempatan ke produksi.	*Product release* akan ditempatkan ke persekitaran produksi setelah pengesahan penerimaan pengguna lulus.

5.8.8. 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 6 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://codeit.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.

UNIT PEMODENAN TADBIRAN
DAN PERANCANGAN PENGURUSAN
MALAYSIA
Aras 6, Setia Perdana 2,
Kompleks Setia Perdana,
Pusat Pentadbiran Kerajaan Persekutuan
62502 Putrajaya
Malaysia

OSDEC
OPEN SOURCE DEVELOPMENT AND CAPABILITIES PROGRAMME
MALAYSIA PUBLIC SECTOR

www.osdec.gov.my