KERJA PRAKTIK - IF184801

Implementasi dan Penanganan Breaking Changes dalam Pembaruan Backend Service di PT SRC Indonesia Sembilan

PT SRC Indonesia Sembilan
One Pacific Place Lantai 18, Jl. Jendral Sudirman, Kav 52 - 53,
DKI Jakarta, 12190
Periode: 1 September 2024 – 10 Januari 2025

Oleh:

Ketut Arda Putra Mahotama Sadha 5025211235

Pembimbing Jurusan
Ilham Gurat Adillion, S.Kom., M.Eng.

Pembimbing Lapangan
Katherine Limanu

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

KERJA PRAKTIK – IF184801

Implementasi dan Penanganan Breaking Changes dalam Pembaruan Backend Service di PT SRC Indonesia Sembilan

PT SRC Indonesia Sembilan
One Pacific Place Lantai 18, Jl. Jendral Sudirman, Kav 52 - 53, DKI Jakarta, 12190
Periode: 1 September 2024 – 10 Januari 2025

Oleh:

Ketut Arda Putra Mahotama Sadha 5025211235

Pembimbing Jurusan
Ilham Gurat Adillion, S.Kom., M.Eng.

Pembimbing Lapangan
Katherine Limanu

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SEMU

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Implementasi dan Penanganan Breaking Changes dalam Pembaruan Backend Service di PT SRC Indonesia Sembilan

Oleh:

Ketut Arda Putra Mahotama Sadha 5025211235

Disetujui oleh Pembimbing Kerja Praktik:

1. Ilham Gurat Adillion, S.Kom., M.Eng.
   NIP. 1995202411009

(Pembimbing Departemen)

2. Katherine Limanu

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Implementasi dan Penanganan Breaking Changes dalam Pembaruan Backend Service di PT SRC Indonesia Sembilan

Nama Mahasiswa	:	Ketut Arda Putra Mahotama Sadha
NRP	:	5025211235
Departemen	:	Teknik Informatika FTEIC-ITS
Pembimbing Departemen	:	Ilham Gurat Adillion, S.Kom., M.Eng.
Pembimbing Lapangan	:	Katherine Limanu

ABSTRAK

Laporan kerja praktik ini membahas implementasi dan penanganan breaking changes yang dilakukan untuk mengatasi technical debt dalam pembaruan backend service di PT SRC Indonesia Sembilan. Pembaruan ini bertujuan untuk meningkatkan performa, kompatibilitas, dan keamanan aplikasi melalui migrasi ke versi terbaru PHP dan Laravel. Tantangan utama yang dihadapi adalah breaking changes, seperti perubahan signifikan pada konfigurasi middleware dan routing dalam Laravel 11, yang merupakan bagian dari upaya untuk mengurangi technical debt yang terakumulasi. Laporan ini menjelaskan proses identifikasi, analisis, dan solusi implementasi untuk menangani perubahan tersebut, termasuk pengelolaan konfigurasi pada file bootstrap/app.php. Dengan pembaruan ini, sistem backend menjadi lebih optimal dan sesuai standar teknologi terkini.

Kata kunci: Backend, Laravel, Tech Debt, Microservices.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Dengan rasa syukur yang mendalam, penulis sebagai mahasiswa memanjatkan puji dan terima kasih kepada Tuhan Yang Maha Esa, yang telah memberikan penulis kekuatan, kesabaran, serta kesempatan untuk menyelesaikan kerja praktik dan laporan kerja praktik ini tepat waktu. Pengalaman selama kerja praktik di PT. SRC Indonesia Sembilan merupakan sebuah perjalanan yang penuh dengan pembelajaran dan penemuan baru, khususnya dalam pengembangan dan implementasi pembaruan *backend service* serta penanganan *technical debt* pada aplikasi AYO yang berbasis arsitektur *microservices*.

Laporan Kerja Praktik ini disusun untuk memenuhi salah satu syarat penyelesaian program studi Teknik Informatika di Fakultas Teknologi Elektro dan Informatika Cerdas (FTEIC) Institut Teknologi Sepuluh Nopember. Kegiatan kerja praktik ini telah memberikan penulis wawasan praktis mengenai aspek-aspek teknis dan profesional dalam peran sebagai seorang *Software Developer*, terutama dalam pengelolaan ekosistem aplikasi berbasis arsitektur *microservices* dan *framework* Laravel.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Melalui buku laporan ini, penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik ini, baik secara langsung maupun tidak langsung, antara lain:

1. Kepada kedua orang tua penulis, yang telah memberikan doa dan dukungan tanpa henti sepanjang hidup penulis. Terima kasih telah memenuhi segala kebutuhan penulis sehingga dapat menyelesaikan kerja praktik dan laporan kerja praktik ini tepat waktu.

2. Bapak Ilham Gurat Adillion, S.Kom., M.Eng., selaku dosen pembimbing dan dosen penguji yang telah membantu penulis dalam penyusunan laporan kerja praktik serta evaluasi kerja praktik.
3. Ibu Katherine Limanu, yang bertindak sebagai pembimbing lapangan selama kerja praktik dan memberikan banyak pengalaman berharga selama periode tersebut.
4. Bapak Ary Mazharuddin, S.Kom., M.Comp.Sc., sebagai koordinator kerja praktik yang telah mempermudah urusan administrasi serta bimbingan terkait ekivalensi dan konversi SKS selama periode kerja praktik.
5. Yudhi Purwananto, S.Kom., M.Kom., selaku dosen wali yang selalu memberikan dukungan yang memudahkan proses pendidikan penulis agar lebih lancar.
6. Koko Indrawan Wibisono, Koko Singgih Soepomo, Koko Gilbert Khomaro, dan Koko Andreas Iskandar, para Software Engineer yang senantiasa memberikan bimbingan serta arahan teknis yang sangat membantu dalam memperdalam pemahaman penulis dalam melaksanakan tugas kerja praktik.
7. Hanun Shaka Puspa, Java Kanaya Prada, Rizky Alifiyah Rahma, Fadilla Rizky Nurhidayah, dan Alfan Lukeyan Rizki, teman-teman seperjuangan dalam tim yang sama selama kerja praktik. Terima kasih atas kerjasama, diskusi, serta arahan yang bermanfaat yang telah dibagikan selama bersama-sama bekerja.
8. Seluruh anggota tim di divisi Inhouse Development di PT. SRC Indonesia Sembilan, yang telah memberikan dukungan, kolaborasi, serta pertukaran pengetahuan yang berharga selama penulis bergabung dalam tim. Kerja sama tim yang solid dan semangat yang selalu ada sangat mendukung penulis selama melaksanakan kerja praktik.
9. Semua pihak yang memberikan dukungan, baik secara langsung maupun tidak langsung, termasuk rekan-rekan

dan senior di kampus serta di tempat kerja praktik yang telah memberikan motivasi, bantuan, dan saran yang sangat membangun selama penulis melaksanakan kerja praktik.

Akhir kata, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, baik secara langsung maupun tidak langsung sepanjang perjalanan ini. Semoga laporan kerja praktik ini tidak hanya mencerminkan pengalaman dan pembelajaran yang penulis dapatkan, tetapi juga dapat menjadi sumber inspirasi dan wawasan bagi pembacanya. Penulis dengan terbuka menerima segala bentuk saran dan kritik yang membangun untuk perbaikan laporan ini. Semoga apa yang telah penulis pelajari dan bagikan melalui laporan ini dapat memberikan manfaat bagi banyak orang dan memberikan kontribusi terhadap perkembangan teknologi informasi di masa depan.

Surabaya, 10 Januari 2025

Ketut Arda Putra Mahotama Sadha

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

PT SRC Indonesia Sembilan (SRCIS) mengelola ekosistem aplikasi AYO yang dibangun menggunakan paradigma *microservices architecture*. Sebagian besar *microservice* dalam ekosistem ini dikembangkan menggunakan framework Laravel. Namun, dengan berjalannya waktu, versi PHP dan Laravel yang digunakan telah menjadi usang akibat perkembangan teknologi yang pesat. Kondisi ini menimbulkan *technical debt* (*tech debt*), yang dapat memengaruhi kinerja, keamanan, dan efisiensi sistem.

Untuk mengurangi *tech debt* tersebut, PT SRCIS berencana memperbarui versi PHP dan *framework* Laravel yang digunakan pada aplikasi mereka. Langkah ini bertujuan untuk meningkatkan kompatibilitas dengan teknologi terkini, memastikan performa sistem yang lebih optimal, serta memperkuat aspek keamanan. Dengan melakukan pembaruan ini, SRCIS juga berharap dapat memfasilitasi pengembangan fitur baru secara lebih efisien dan memberikan pengalaman yang lebih baik kepada pengguna ekosistem AYO.

1.2 Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 2 SKS dan membantu PT SRC Indonesia Sembilan (SRCIS) dalam mengurangi *technical debt* pada ekosistem aplikasi AYO. Kontribusi dilakukan dengan mendukung pembaruan versi PHP dan framework Laravel yang digunakan, guna meningkatkan performa, keamanan, dan efisiensi sistem aplikasi berbasis *microservices*.

1.3 Manfaat

Manfaat yang diperoleh dari pembaruan versi PHP dan *framework* Laravel pada ekosistem aplikasi AYO antara lain adalah meningkatkan performa dan keamanan sistem, serta mempermudah pengembangan fitur baru. Dengan sistem yang lebih mutakhir dan efisien, ekosistem AYO dapat memberikan pengalaman yang lebih baik kepada pengguna, mendukung operasional toko kelontong dalam jaringan SRC, dan memperkuat digitalisasi UMKM di Indonesia.

1.4 Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana cara memperbarui versi PHP yang digunakan oleh *microservice*?
2. Bagaimana cara memperbarui versi *framework* Laravel yang digunakan oleh *microservice*?
3. Bagaimana mengevaluasi hasil dari pembaruan versi PHP dan *framework* Laravel?

1.5 Lokasi dan Waktu Kerja Praktik

Kerja praktik dilakukan secara luring (offline) di Head Office PT. HM Sampoerna Tbk., yang berlokasi di Jl. Rungkut Industri Raya No. 18, Kali Rungkut, Kecamatan Rungkut.

Pelaksanaan kerja praktik berlangsung mulai 1 September 2024 hingga 10 Januari 2025. Kegiatan ini dijalankan selama lima hari kerja setiap minggunya, dengan jam operasional dari pukul 09.00 hingga 18.00 WIB.

1.6 Metodologi Kerja Praktik

Metodologi yang diterapkan dalam pelaksanaan kerja praktik untuk mendukung pembaruan versi PHP dan *framework* Laravel pada ekosistem aplikasi AYO milik PT SRC Indonesia Sembilan mencakup beberapa tahapan. Tahapan-tahapan tersebut dirancang untuk memastikan proses pembaruan berjalan efektif dan sesuai dengan kebutuhan sistem:

1.6.1 Perumusan Masalah

Langkah awal dalam proses pembaruan versi PHP dan *framework* Laravel adalah mengidentifikasi kebutuhan utama sistem. Kebutuhan ini diperoleh dari permintaan pihak User yang disampaikan melalui tim *Software Engineer*. Setelah permintaan diterima oleh tim *Software Engineer*, informasi tersebut disampaikan kepada penulis melalui sesi pemaparan.

Pemaparan yang dilakukan oleh tim *Software Engineer* mencakup latar belakang perlunya pembaruan, tujuan pembaruan, serta persyaratan teknis yang harus dipenuhi. Berdasarkan hasil pemaparan, diperoleh kesimpulan bahwa pembaruan ini bertujuan untuk meningkatkan kompatibilitas sistem dengan teknologi terbaru, memperkuat keamanan, serta mempermudah pengembangan dan pemeliharaan fitur-fitur aplikasi di masa depan.

1.6.2 Studi Literatur

Studi literatur dilakukan untuk mempelajari alat-alat yang digunakan dalam proses pembaruan versi PHP dan framework Laravel pada ekosistem AYO. Beberapa alat utama yang digunakan dalam ekosistem ini antara lain adalah microservice architecture, yang menjadi paradigma utama dalam pengembangan aplikasi AYO, serta Laravel, framework yang digunakan untuk membangun berbagai microservice. Selain itu, ekosistem AYO juga memanfaatkan Docker, sebuah platform kontainerisasi yang

memudahkan pengelolaan aplikasi dan layanan, serta AWS sebagai *platform cloud* untuk menyokong infrastruktur dan skalabilitas aplikasi. Untuk mendukung proses integrasi dan pengiriman kode secara berkelanjutan, ekosistem AYO menggunakan Jenkins sebagai platform CI/CD.

1.6.3 Analisis dan Perancangan Sistem

AYO SRC adalah ekosistem digital yang dikembangkan oleh PT SRC Indonesia Sembilan untuk mendukung transformasi digital toko kelontong di Indonesia. AYO SRC terdiri dari beberapa aplikasi utama, seperti AYO SRC Mitra untuk grosir, AYO SRC Toko untuk pemilik toko kelontong, dan AYO Kelontong untuk konsumen. AYO SRC juga berkolaborasi dengan berbagai pihak untuk meningkatkan layanannya, seperti integrasi dengan GrabExpress dan opsi pembayaran digital melalui QRIS OVO. AYO SRC menggunakan arsitektur *microservices* untuk backend-nya, yang memungkinkan pengembangan layanan yang modular dan fleksibel. Namun, beberapa backend services masih menggunakan versi PHP dan Laravel yang sudah usang, yang berpotensi menimbulkan tantangan, seperti masalah keamanan dan keterbatasan fitur. Oleh karena itu, pembaruan *framework* dan teknologi diperlukan untuk meningkatkan efisiensi dan menjaga stabilitas sistem.

Dalam perancangan infrastruktur sistem, AYO SRC memanfaatkan *platform cloud* AWS, termasuk layanan seperti Route53, CloudFront, S3 Bucket, EC2, dan RDS Aurora, untuk mendukung operasi *backend*. Selain itu, untuk mendukung CI/CD, AYO SRC menggunakan Jenkins dan Bitbucket. Pembaruan PHP dan Laravel dilakukan dengan memperbarui Docker *image* dan menggunakan *Elastic Container Registry* (ECR) dan *Elastic Container Service* (ECS) di AWS. Pembaruan Laravel dilakukan secara bertahap mulai dari versi 5.x ke versi 11, dengan mengatasi perubahan signifikan yang terjadi di setiap versi, seperti perubahan pada sistem *middleware*, autentikasi, dan *query builder*. Setelah

pembaruan selesai, pengujian dan *debugging* dilakukan untuk memastikan aplikasi berjalan dengan baik tanpa gangguan pada fungsionalitasnya.

1.6.4 Implementasi Sistem

Pada tahap ini, pembaruan versi PHP dan *framework* Laravel dimulai berdasarkan perencanaan yang telah disepakati sebelumnya. Proses implementasi melibatkan pembaruan kode untuk memastikan bahwa layanan backend yang menggunakan PHP dan Laravel dapat berjalan dengan versi terbaru, termasuk pembaruan Docker image dan penyesuaian di Elastic Container Service (ECS). Kemudian dilakukan pengujian awal dalam lingkungan pengembangan untuk memverifikasi fungsionalitas sistem setelah pembaruan, guna memastikan kompatibilitas dan kinerja yang optimal sebelum diterapkan pada lingkungan produksi.

1.6.5 Pengujian dan Evaluasi

Setelah website yang telah direncanakan telah jadi, perlu adanya evaluasi untuk menguji apakah website sesuai dengan harapan client. Jika masih belum sesuai atau perlu menambah fitur, rapat akan dilakukan lagi untuk mem-*floor*-kan fitur - fitur apa saja yang perlu diperbaiki atau ditambah.

1.6.6 Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7 Sistematika Laporan

1.7.1 Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2 Bab II Profil Perusahaan

Bab ini berisi gambaran umum PT SRC Indonesia Sembilan mulai dari profil, lokasi perusahaan.

1.7.3 Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4 Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi mengenai tahap analisis sistem aplikasi dalam menyelesaikan proyek kerja praktik.

1.7.5 Bab V Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi aplikasi.

1.7.6 Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil uji coba dan evaluasi dari aplikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7 Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

BAB II
PROFIL PERUSAHAAN

2.1 Profil PT SRC Indonesia Sembilan

PT SRC Indonesia Sembilan (SRCIS) memainkan peran penting dalam mendukung pemberdayaan dan transformasi UMKM di Indonesia, khususnya pada sektor toko kelontong. Sebagai bagian dari PT Hanjaya Mandala Sampoerna Tbk, SRCIS berkomitmen untuk meningkatkan daya saing UMKM melalui inovasi dan digitalisasi. Program unggulan mereka, Sampoerna Retail Community (SRC), telah berkembang menjadi jaringan besar yang mencakup lebih dari 243.000 toko kelontong hingga kuartal ketiga 2023. Program ini memberikan dukungan kepada toko kelontong di seluruh Indonesia dengan menghadirkan ekosistem digital bernama AYO SRC, yang dilengkapi berbagai fitur inovatif, termasuk Pojok Lokal, untuk memasarkan produk-produk UMKM lokal.

Berkantor pusat di Jakarta, SRCIS menjadi pusat berbagai kegiatan yang dirancang untuk mendorong pertumbuhan UMKM. Program-program yang mereka jalankan tidak hanya berfokus pada peningkatan kapasitas bisnis, tetapi juga memperkuat komunitas melalui kolaborasi dengan mitra strategis dan pemerintah. Kolaborasi ini memungkinkan SRCIS terus menghadirkan inovasi, seperti tercermin dari tingginya tingkat adopsi digital di kalangan toko-toko SRC.

Keberadaan SRC di tengah masyarakat memberikan dampak positif, tidak hanya dari sisi ekonomi, tetapi juga dalam meningkatkan kesejahteraan sosial. Hal ini terlihat dari kenaikan omzet yang signifikan di kalangan anggota SRC serta peningkatan distribusi produk lokal berkualitas melalui inisiatif seperti Pojok Lokal. Ekosistem digital AYO SRC juga memiliki pengaruh besar

dengan lebih dari 243.000 toko yang tergabung dalam jaringan ini di seluruh Indonesia.

2.2 Lokasi

Kantor pusat PT SRC Indonesia Sembilan terletak di One Pacific Place, Lantai 18, Jl. Jendral Sudirman, Kav 52 - 53, DKI

BAB III
TINJAUAN PUSTAKA

3.1 Microservices Architecture

Microservices Architecture adalah pendekatan desain perangkat lunak yang menyusun aplikasi sebagai kumpulan layanan yang saling terhubung secara longgar. Setiap layanan dirancang untuk menjalankan fungsi bisnis tertentu dan dapat dikembangkan, diterapkan, serta diskalakan secara mandiri. Gaya arsitektur ini berbeda dengan arsitektur monolitik tradisional, di mana semua komponen terintegrasi erat dalam satu aplikasi (Yin & Du, 2019).

3.1.1 Karakteristik Utama Microservices Architecture

Berikut adalah uraian dari karakteristik utama *Microservices Architecture*:

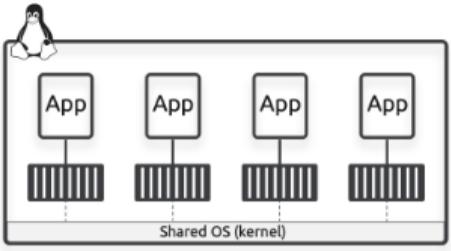
1. Layanan yang Terhubung Secara Longgar: *Microservice* beroperasi secara mandiri, memungkinkan perubahan pada satu layanan tanpa memengaruhi layanan lainnya. Kemandirian ini meningkatkan fleksibilitas dan kemampuan pemeliharaan.
2. Prinsip Tanggung Jawab Tunggal: Setiap *microservice* berfokus pada kemampuan bisnis tertentu, sehingga memudahkan pemahaman dan kejelasan.
3. Skalabilitas: *Microservice* dapat diskalakan secara mandiri sesuai permintaan, mengoptimalkan penggunaan sumber daya dan kinerja.
4. Bebas Teknologi: Setiap layanan dapat dibangun menggunakan bahasa pemrograman atau teknologi yang berbeda, memungkinkan tim memilih alat terbaik untuk kebutuhan spesifik mereka.
5. Ketahanan: Arsitektur ini dirancang untuk menangani kegagalan dengan baik, sehingga kegagalan satu layanan

tidak akan menyebabkan seluruh sistem mengalami kerusakan.

3.2 Docker

Docker adalah platform kontainerisasi yang memungkinkan pengembang untuk membangun, mengirim, dan menjalankan aplikasi dalam lingkungan yang terisolasi yang disebut kontainer. Kontainer ini berfungsi sebagai wadah yang mengemas aplikasi beserta semua dependensinya, sehingga aplikasi dapat dijalankan di berbagai lingkungan tanpa masalah kompatibilitas (Poulton, 2023).

Kontainerisasi berbeda dengan virtualisasi. Virtualisasi adalah teknologi yang memungkinkan pembuatan versi virtual dari sumber daya fisik, seperti server, penyimpanan, jaringan, atau sistem operasi. Dengan virtualisasi, satu perangkat keras fisik dapat menjalankan beberapa lingkungan virtual yang terisolasi, yang dikenal sebagai mesin virtual (VM). Mesin virtual (VM) menjalankan sistem operasi lengkap di atas *hypervisor*, yang berfungsi sebagai lapisan antara perangkat keras dan sistem operasi. Setiap VM memiliki kernel dan sistem operasinya sendiri, sehingga lebih berat dan memerlukan lebih banyak sumber daya. Di sisi lain, Kontainer berbagi kernel sistem operasi yang sama dan berjalan di atas sistem operasi *host*. Mereka mengisolasi aplikasi dalam lingkungan yang terpisah, tetapi tidak memerlukan sistem operasi lengkap untuk setiap instansi. Ini membuat kontainer lebih ringan dan lebih cepat untuk diluncurkan.



Gambar 3.1 Lingkunan Eksekusi Terisolasi (Poulton, 2023)

3.3 Amazon Web Services

Amazon Web Services (AWS) adalah platform komputasi awan yang komprehensif yang disediakan oleh Amazon. *Platform* ini menawarkan berbagai layanan yang memungkinkan bisnis dan pengembang untuk membangun, menerapkan, dan mengelola aplikasi serta layanan di cloud (Dubey dkk., 2023). Secara umum, berikut adalah fitur-fitur yang disediakan AWS:

Layanan Komputasi Awan: AWS menyediakan berbagai layanan seperti daya komputasi, opsi penyimpanan, dan kemampuan jaringan. Layanan ini memungkinkan pengguna menjalankan aplikasi dan menyimpan data tanpa memerlukan perangkat keras fisik.

1. Skalabilitas dan Fleksibilitas: Salah satu keunggulan utama AWS adalah kemampuannya untuk menyesuaikan sumber daya secara dinamis sesuai permintaan. Elastisitas ini membantu bisnis mengelola biaya secara efektif sambil menjaga kinerja.
2. Infrastruktur Global: AWS beroperasi di berbagai wilayah geografis di seluruh dunia, memungkinkan pengguna untuk menerapkan aplikasi lebih dekat ke pelanggan mereka untuk meningkatkan latensi dan kinerja.

3.3.1 Layanan Utama AWS

Amazon Web Services (AWS) menawarkan berbagai layanan komputasi awan yang memenuhi berbagai kebutuhan bisnis. Berikut adalah gambaran dari layanan utama yang disebutkan:

1. Amazon RDS (*Relational Database Service*): Layanan basis data terkelola yang mendukung berbagai mesin basis data, termasuk MySQL, PostgreSQL, dan Oracle. Layanan ini menyederhanakan pengaturan, pengoperasian, dan penskalaan basis data relasional di cloud.
2. Amazon S3 (*Simple Storage Service*): Layanan penyimpanan objek yang dirancang untuk skalabilitas, ketersediaan data, keamanan, dan kinerja. Umumnya digunakan untuk pencadangan, pengarsipan, dan analitik data besar.
3. Amazon Route 53: Layanan Domain Name System (DNS) yang skalabel, memberikan pendaftaran domain yang andal dan hemat biaya, serta routing pengguna akhir ke aplikasi Internet.
4. Amazon EC2 (*Elastic Compute Cloud*): Layanan web yang menyediakan kapasitas komputasi yang dapat disesuaikan di cloud. Pengguna dapat menjalankan server virtual dan menskalakan aplikasi sesuai kebutuhan.
5. Amazon ECS (*Elastic Container Service*): Layanan orkestrasi container yang sepenuhnya dikelola, memungkinkan pengguna menjalankan dan mengelola container Docker pada cluster.
6. Amazon ECR (*Elastic Container Registry*): Registry container Docker yang sepenuhnya dikelola, memudahkan pengembang untuk menyimpan, mengelola, dan menerapkan gambar container Docker.
7. Amazon ALB (*Application Load Balancer*): Layanan penyeimbang beban untuk lalu lintas HTTP dan HTTPS yang secara otomatis mendistribusikan lalu lintas aplikasi

masuk ke berbagai target, seperti *instance* EC2 atau container.

8. Amazon SES (*Simple Email Service*): Layanan pengiriman email berbasis cloud yang dirancang untuk bisnis mengirimkan email pemasaran, notifikasi, dan transaksional.
9. Amazon VPC (*Virtual Private Cloud*): Layanan yang memungkinkan pengguna membuat jaringan terisolasi dalam AWS cloud. Memberikan kontrol atas konfigurasi jaringan dan pengaturan keamanan.
10. AWS IAM (*Identity and Access Management*): Layanan yang membantu pengguna mengontrol akses ke layanan dan sumber daya AWS dengan aman. Memungkinkan pembuatan pengguna, grup, dan izin untuk mengelola hak akses secara efektif.
11. AWS *Systems Manager Parameter Store*: Solusi penyimpanan aman untuk manajemen data konfigurasi dan manajemen rahasia. Memungkinkan pengguna menyimpan data seperti kata sandi, *string* basis data, dan kode lisensi dengan aman.

Dengan memanfaatkan layanan-layanan AWS seperti yang telah disebutkan, perusahaan dapat membangun sistem yang andal, aman, dan skalabel untuk berbagai kebutuhan bisnis. Amazon RDS memungkinkan pengelolaan basis data yang efisien, sementara Amazon S3 menawarkan solusi penyimpanan data yang tangguh. Amazon EC2 dan Amazon ECS memberikan fleksibilitas dalam penyediaan kapasitas komputasi dan pengelolaan kontainer, mendukung aplikasi yang membutuhkan skalabilitas tinggi. Integrasi dengan Amazon ALB memastikan distribusi lalu lintas yang optimal, sehingga meningkatkan kinerja aplikasi. Selain itu, layanan seperti Amazon VPC dan AWS IAM memastikan keamanan dan kontrol penuh atas sumber daya serta jaringan. Untuk pengelolaan konfigurasi dan rahasia, AWS *Systems Manager Parameter Store* menyediakan solusi yang aman dan

mudah digunakan. Dengan kombinasi layanan ini, perusahaan dapat membangun infrastruktur sistem yang tidak hanya memenuhi kebutuhan operasional, tetapi juga mampu beradaptasi dengan perubahan bisnis secara dinamis (AWS, 2024).

3.4 Jenkins

Jenkins adalah alat otomatisasi *open-source* yang banyak digunakan untuk mengotomatisasi berbagai aspek dalam pengembangan perangkat lunak, terutama dalam proses *Continuous Integration* (CI) dan *Continuous Delivery* (CD). Jenkins membantu pengembang mengintegrasikan perubahan kode pada repositori, serta menguji perubahan tersebut secara otomatis.

3.4.1 Fitur Utama Jenkins

1. Integrasi dengan Berbagai Sistem Kontrol Versi: Jenkins mendukung integrasi dengan sistem kontrol versi seperti Git, Subversion, dan Mercurial. Hal ini memudahkan pengembang dalam mengelola kode sumber secara efisien.
2. *Pipeline* Jenkins: memungkinkan pengguna untuk membuat pipeline yang mendefinisikan proses *build*, *test*, dan *deployment* dalam bentuk skrip. *Pipeline* ini mempermudah pengelolaan proses pengembangan secara otomatis dan terstruktur.
3. *Plugin*: Dengan ketersediaan banyak *plugin*, Jenkins dapat diperluas untuk integrasi dengan berbagai alat, seperti Docker, Kubernetes, dan alat pengujian lainnya, guna meningkatkan fungsionalitasnya.
4. Notifikasi dan Pelaporan: Jenkins menyediakan fitur notifikasi melalui email atau alat komunikasi lainnya untuk memberikan informasi tentang *status build* dan hasil pengujian, yang membantu tim pengembang memantau proses secara *real-time*.

Secara keseluruhan, Jenkins adalah alat yang sangat berguna dalam dunia DevOps, mendukung tim pengembang dalam meningkatkan produktivitas, efisiensi, dan kualitas perangkat lunak (Banyu Adil dkk., 2024).

3.5 Laravel

Laravel adalah *framework* PHP *open-source* yang dirancang untuk mempermudah pengembangan aplikasi web. Diciptakan oleh Taylor Otwell dan pertama kali dirilis pada tahun 2011, Laravel bertujuan untuk memberikan alternatif yang lebih baik dibandingkan *framework* PHP lainnya seperti CodeIgniter. *Framework* ini menggunakan pola desain *Model-View-Controller* (MVC), yang membantu memisahkan logika aplikasi dari tampilan, sehingga memudahkan pengembang dalam mengelola dan mengembangkan aplikasi.

Kelebihan Menggunakan Laravel dibandingkan *framework* lainnya antara lain:

1. Kecepatan Pengembangan: Mempercepat proses pembuatan aplikasi dengan menyediakan banyak fitur bawaan.
2. Keamanan: Menawarkan perlindungan terhadap serangan umum seperti SQL *injection* dan *cross-site request forgery* (CSRF).
3. Komunitas Aktif: Memiliki dukungan komunitas yang besar, sehingga banyak sumber daya dan dokumentasi tersedia untuk membantu pengembang.

Dengan semua fitur dan keunggulannya, Laravel telah menjadi salah satu *framework* PHP paling populer di dunia, termasuk di Indonesia, dan banyak digunakan oleh perusahaan untuk membangun aplikasi *web* yang kompleks dan dinamis (Dese Fitriani dkk., 2024).

3.6 Technical Debt

Technical debt adalah istilah dalam pengembangan perangkat lunak yang menggambarkan biaya implisit dari pekerjaan tambahan akibat memilih solusi cepat atau mudah dibandingkan dengan pendekatan yang lebih baik tetapi memakan waktu lebih lama. Konsep ini mengacu pada *trade-off* antara manfaat jangka pendek dan konsekuensi jangka panjang terhadap kualitas dan keterawatan kode (Caglayan & Özcan-Top, 2024). Berikut adalah uraian jenis *technical debt*:

1. *Framework* Usang: Terjadi ketika aplikasi perangkat lunak bergantung pada versi *framework* lama yang tidak lagi didukung atau kekurangan fitur modern. *Framework* usang dapat menghambat kecepatan pengembangan dan meningkatkan risiko kerentanan keamanan.
2. Masalah Kualitas Kode: Kode yang ditulis dengan buruk atau tidak dioptimalkan menyulitkan pengembang untuk melakukan perubahan di masa depan.
3. Kurangnya Dokumentasi: Dokumentasi yang tidak memadai dapat menyebabkan kesalahpahaman dan meningkatkan waktu onboarding bagi pengembang baru.
4. Pengujian yang Tidak Memadai: Melewatkan proses pengujian untuk mengejar tenggat waktu dapat menghasilkan bug yang mahal untuk diperbaiki di kemudian hari.

Framework usang merupakan salah satu bentuk technical debt yang signifikan karena dapat membatasi kemampuan tim dalam mengimplementasikan fitur baru secara efisien. Selain itu, *framework* yang usang sering memerlukan upaya *refactoring* atau penulisan ulang kode yang substansial, yang dapat mengganggu jadwal proyek dan menghabiskan sumber daya.

Mengelola technical debt, termasuk *framework* usang, memerlukan pertimbangan matang terhadap trade-off antara

mempertahankan sistem lama dengan investasi untuk memperbarui atau menggantinya. Pendekatan strategis meliputi:

1. Melakukan pembaruan framework secara berkala.
2. Memprioritaskan *technical debt* berdasarkan dampaknya terhadap produktivitas tim dan keamanan aplikasi.
3. Mengalokasikan waktu dalam siklus pengembangan untuk mengurangi *technical debt* yang ada.

Dengan manajemen yang baik, *technical debt* dapat dikurangi untuk mendukung pengembangan perangkat lunak yang lebih berkelanjutan dan efisien (Oliveira dkk., 2015).

[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN INFRASTRUKTUR SISTEM

4.1 Analisis Sistem

Pada bab ini akan dijelaskan mengenai tahapan dalam pembaruan versi PHP dan versi *framework* Laravel yang digunakan yaitu analisis dari infrastruktur sistem yang akan dibangun. Hal tersebut dijelaskan ke dalam dua bagian, definisi umum aplikasi dan analisis kebutuhan.

4.2 Definisi Umum Aplikasi

AYO SRC adalah ekosistem digital yang dikembangkan oleh PT SRC Indonesia Sembilan (SRCIS) untuk mendukung transformasi digital toko kelontong di Indonesia. Melalui aplikasi AYO SRC, pemilik toko kelontong dapat mengelola bisnis mereka dengan lebih efisien, termasuk dalam hal manajemen stok, pemesanan produk, dan pembuatan promosi digital. Ekosistem AYO SRC terdiri dari beberapa komponen utama:

- **AYO SRC Mitra**: Aplikasi yang ditujukan untuk grosir (wholesaler) yang memasok produk ke toko kelontong SRC. Dengan AYO SRC Mitra, grosir dapat mengelola bisnis mereka secara digital, mulai dari manajemen stok hingga pembuatan promosi untuk retailer.
- **AYO SRC Toko**: Aplikasi untuk pemilik toko kelontong yang memungkinkan mereka memesan produk secara digital dari grosir SRC, menjual produk digital, dan membuat promosi mandiri untuk pelanggan mereka.
- **AYO Kelontong**: Aplikasi yang ditujukan untuk konsumen, memberikan pengalaman berbelanja yang lebih menyenangkan dengan berbagai fitur menarik, serta mendukung toko kelontong UMKM di Indonesia.

Selain itu, AYO SRC juga telah berkolaborasi dengan berbagai pihak untuk meningkatkan layanan digitalnya. Misalnya, integrasi dengan layanan GrabExpress memungkinkan toko kelontong menawarkan layanan pengiriman kepada konsumen melalui aplikasi AYO SRC. Selain itu, tersedia juga opsi pembayaran digital melalui QRIS OVO, memudahkan konsumen dalam melakukan transaksi non-tunai.

Dengan lebih dari 225.000 toko kelontong yang tergabung dalam jaringan SRC di seluruh Indonesia, AYO SRC berperan penting dalam mendukung digitalisasi UMKM dan meningkatkan daya saing mereka di era ekonomi digital.

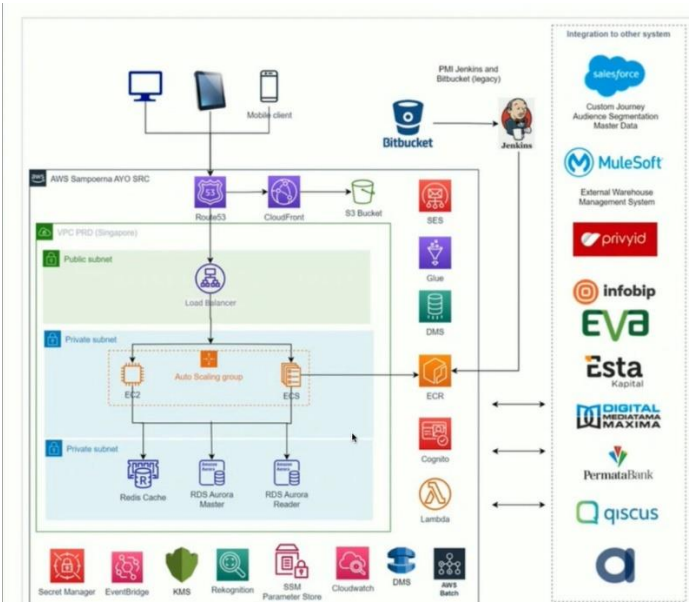
AYO SRC memanfaatkan arsitektur *microservices* untuk backend-nya. Paradigma arsitektur ini memungkinkan pengembangan dan pemeliharaan layanan yang modular, sehingga setiap layanan dapat diimplementasikan, diuji, dan dikelola secara terpisah untuk mendukung skalabilitas dan fleksibilitas sistem. Namun, saat ini terdapat sejumlah backend *services* yang masih menggunakan PHP dengan *framework* Laravel dalam versi yang sudah outdated, yang berpotensi menghadirkan tantangan, seperti kerentanan keamanan, keterbatasan fitur, performa yang kurang optimal, dan kesulitan kompatibilitas dengan teknologi terbaru. Untuk memastikan keberlanjutan dan daya saing ekosistem, diperlukan pembaruan *framework* ke versi yang lebih baru atau migrasi ke teknologi lain yang lebih modern, guna meningkatkan efisiensi operasional dan menjaga stabilitas serta keamanan layanan bagi ribuan pengguna yang bergantung pada platform ini.

4.3 Perancangan Infrastruktur Sistem

4.3.1 Desain Sistem

Infrastruktur sistem *microservices* AYO SRC menggunakan platform cloud AWS untuk menjalankan setiap

backend *services*-nya. Gambar 4.1 merupakan ilustrasi infrastruktur AYO SRC.



Gambar 4.1 Infrastruktur AYO SRC

Flow client pada sistem AYO SRC dimulai ketika client, baik dari perangkat desktop maupun mobile, mengakses aplikasi melalui jaringan internet. Permintaan pertama kali diterima oleh *Route53*, layanan DNS yang mengarahkan traffic ke lokasi yang sesuai. Selanjutnya, permintaan diteruskan ke CloudFront, layanan *Content Delivery Network* (CDN) yang mempercepat pengiriman konten statis maupun dinamis. Jika dibutuhkan, file statis seperti gambar atau dokumen akan diambil dari *S3 Bucket*. Permintaan kemudian diarahkan ke *Load Balancer* yang bertugas mendistribusikan beban kerja secara merata ke *EC2 instances* yang berada dalam subnet privat dan dikelola oleh *Auto Scaling Group*, yang memastikan kapasitas server selalu optimal. Backend akan

memproses permintaan dengan bantuan database *RDS Aurora* (Master untuk operasi write dan Reader untuk operasi read), serta *Redis Cache* untuk mempercepat pengambilan data yang sering diakses. Setelah diproses, respon dikembalikan ke client melalui jalur yang sama.

Sementara itu, flow Bitbucket berfokus pada pengelolaan kode sumber dan proses deployment. Kode sumber aplikasi disimpan di Bitbucket, dan setiap perubahan atau commit dapat memicu Jenkins, alat CI/CD, untuk melakukan proses build, testing, dan deployment otomatis. Hasil build disimpan di *Elastic Container Registry* (ECR) untuk digunakan dalam lingkungan produksi. Jenkins kemudian melakukan deployment ke EC2 instances melalui Auto Scaling Group atau ke layanan seperti ECS (Elastic Container Service) atau *Lambda* jika menggunakan arsitektur serverless. Flow ini memastikan pengembangan, pengujian, dan deployment aplikasi berjalan secara terorganisir, cepat, dan aman, dengan integrasi ke berbagai sistem lain seperti *Salesforce* dan *Mulesoft* untuk mendukung pengelolaan data dan alur kerja yang kompleks.

4.3.2 Metode Pembaruan versi PHP dan *framework* Laravel

Berdasarkan desain infrastruktur pada subbab 4.3.1, perlu melakukan perubahan pada layanan-layanan berikut:

1. **Docker**:
   - *Base Docker Image*, yaitu *docker image* yang akan digunakan oleh semua *services* dalam AYO SRC sebagai *image* dasar yang berisi *dependencies* umum *services* termasuk versi PHP.
   - *Service Docker Image*, yaitu *docker image* yang akan digunakan masing-masing *services* berisi *dependencies* spesifik yang digunakan oleh *service* tersebut.
2. **AWS**:

- *Elastic Container Registry* (ECR), yaitu layanan untuk menyimpan *docker image* yang telah di *build*.
- *Elastic Container Service* (ECS), yaitu layanan untuk menjalankan *docker image* sebagai service di AWS.

3. **Jenkins:**
   - *Pipeline*, yaitu alur kerja otomatis untuk CI/CD yang mencakup proses *build*, *testing*, hingga *deployment*. *Pipeline* mendukung alur kerja kompleks dan didefinisikan secara terstruktur dalam bentuk skrip.
   - *Jenkinsfile*, yaitu file skrip yang menyimpan definisi *pipeline*, biasanya disimpan di *repository* kode. File ini memuat tahap-tahap seperti *build*, *test*, dan *deploy*, serta mempermudah pengelolaan *pipeline* secara kolaboratif.

Versi *framework* Laravel paling kecil yang digunakan adalah versi 5, sedangkan versi laravel terbaru saat ini adalah versi 11. Pembaruan Laravel dari versi 5 ke versi 11 merupakan suatu tugas yang memerlukan perencanaan yang matang dan pelaksanaan yang cermat. Langkah pertama yang perlu dilakukan adalah evaluasi kebutuhan dan persiapan, yang mencakup pembacaan dokumentasi resmi untuk setiap versi Laravel yang akan dilakukan pembaruan, pemahaman terhadap perubahan signifikan yang diperkenalkan, serta pembuatan cadangan aplikasi dan database untuk menghindari kehilangan data atau kerusakan kode. Selanjutnya, disarankan untuk melakukan pembaruan secara bertahap dari Laravel 5.x ke 6.x, karena Laravel 6 merupakan versi LTS (Long-Term Support) yang stabil. Setiap versi baru Laravel memperkenalkan perubahan besar pada berbagai aspek sistem, seperti autentikasi, *routing*, *factory*, dan *dependency injection*. Oleh karena itu, proses pembaruan sebaiknya dilakukan secara bertahap, mulai dari versi 5.x ke 6.x, kemudian dari 6.x ke 7.x, 7.x ke 8.x, 8.x ke 9.x, dan seterusnya hingga mencapai versi 11.

Selama proses pembaruan, perlu diperhatikan untuk mengatasi breaking changes yang mungkin muncul. Contohnya

adalah perubahan pada sistem *middleware* antara Laravel 5 dan 6, yang mempengaruhi pengaturan *middleware* global yang sebelumnya dapat didefinisikan di satu tempat, namun mulai Laravel 5.2, pengaturan ini dipindahkan ke grup *middleware* di dalam file app/Http/Kernel.php. Selain itu, pada Laravel 7, perubahan besar terjadi pada sistem autentikasi, dimana Laravel UI digantikan dengan Jetstream, yang mempengaruhi cara pengelolaan otentikasi pengguna dan registrasi di *backend* aplikasi. Pada Laravel 8, penggunaan Model Factory yang sebelumnya menggunakan metode factory() kini harus mengimplementasikan trait HasFactory di model, yang mempengaruhi pengelolaan data uji dan pembuatan entitas dalam aplikasi backend. Perubahan lainnya yang perlu diperhatikan adalah peningkatan pada Query Builder dan Eloquent mulai dari Laravel 9, yang memperkenalkan sintaksis dan fitur baru yang lebih ketat dan membutuhkan penyesuaian kode pada bagian query database di backend aplikasi.

Untuk penjelasan lebih detail mengenai breaking changes dan solusi terkait, pembaca dapat merujuk pada dokumentasi resmi Laravel yang tersedia pada Laravel Upgrade Guide untuk setiap versi yang relevan. Dokumentasi ini memberikan panduan lengkap tentang perubahan yang perlu diperhatikan serta solusi dan langkah-langkah yang diperlukan untuk mengatasi masalah kompatibilitas.

Setelah pembaruan selesai, langkah berikutnya adalah melakukan pengujian dan debugging aplikasi untuk memastikan bahwa tidak ada fitur yang terganggu akibat perubahan yang dilakukan. Pengujian unit dan pengujian end-to-end sangat penting untuk memastikan seluruh fungsionalitas aplikasi berjalan dengan baik.

BAB V
IMPLEMENTASI SISTEM

Bab ini membahas tentang implementasi dari pembaruan versi PHP dan *framework* Laravel untuk salah satu *backend service* yang ditugaskan kepada penulis. Pembaruan ini dilakukan untuk memastikan bahwa *backend service* yang ada dapat memanfaatkan fitur-fitur terbaru dari PHP dan Laravel, serta untuk meningkatkan kinerja dan keamanan aplikasi. Proses implementasi meliputi pembaruan versi PHP yang digunakan, serta peningkatan versi Laravel dari yang sebelumnya digunakan, dengan melakukan penyesuaian pada kode dan konfigurasi untuk mengatasi *breaking changes* yang terjadi. Penulis juga merujuk pada dokumentasi resmi Laravel untuk memahami perubahan yang diperkenalkan pada setiap versi dan untuk mendapatkan solusi yang tepat dalam mengatasi masalah kompatibilitas.

5.1 Implementasi Base Docker Image

Implementasi *Base Docker Image* berfokus pada pembuatan *docker image* dasar yang akan digunakan oleh semua *service* yang akan dilakukan perbaruan versi PHP dan *framework* Laravel. Pada kode sumber 5.1, bisa dilihat bahwa terdapat tahapan utama yaitu, instalasi dependency umum *service*, definisi *environtment variable* serta memasukan *template* konfigurasi NGINX serta PHP-FPM.

```
FROM alpine:3.20.3

# Penambahan user untuk NGINX
RUN adduser -S www-data -G www-data -u 82 -D

RUN apk update
# instalasi dependency umum service
RUN apk add SEMUA DEPENDENCY

# penghapuusan konfigurasi default nginx
RUN rm -rf KONFIGURASI DEFAULT NGINX
```

```
# definisi environtment variable
ENV TZ=Asia/Jakarta

# copy konfigurasi nginx serta php-fpm
COPY folder_konfigurasi folder_konfigurasi_tujuan

# pastikan entrypoint kontainer bisa dieksekusi
RUN chmod +x /run.sh && chown www-data:www-data /run.sh

# change working directory
WORKDIR /app

# expose container port
EXPOSE 443/tcp 80/tcp

CMD ["/run.sh"]
```

Kode Semu 5.1 Implementasi *Dockerfile Base Docker Image*

5.2 Implementasi Service Docker Image

Implementasi *Service Docker Image* berfokus pada pembuatan *docker image* yang menggunakan *Base Docker Image* sebagai dasar dan menambahkan *dependencies* spesifik yang diperlukan masing-masing *service*. Kode sumber 5.2 adalah contoh *Dockerfile* dari salah satu service yaitu ayo-raffle yang di *build* menjadi *Service Docker Image*.

```
# build vendor
FROM base_docker_image as vendor

ARG GITU
ARG GITPW

WORKDIR /app
COPY composer.json .
COPY database database
COPY app/Helpers app/Helpers
COPY tests tests

RUN apk update && apk add git
```

```
RUN       composer       config       --global       bitbucket-
oauth.source.app.pconnect.biz $GITU $GITPW
RUN composer install \
    --ignore-platform-reqs \
    --no-interaction \
    --no-plugins \
    --no-scripts \
    --prefer-dist

# production stage
FROM   454418302049.dkr.ecr.ap-southeast-1.amazonaws.com/docker-
repo:websrv-c2-p8.3-aws

# install dependencies untuk service
RUN apk update && apk add SEMUA DEPENDENCY SERVICE

# install NewRelic agent
ARG NEWRELIC_AGENT_VERSION
RUN apk update && apk add --no-cache gcompat
# Create the configuration directory
RUN mkdir -p /etc/conf.d && chown -R www-data:www-data /etc/conf.d
&& \
    apk update && apk add --no-cache gcompat && \
    curl                                                          -L
https://download.newrelic.com/php_agent/archive/${NEWRELIC_AGENT
_VERSION}/newrelic-php5-${NEWRELIC_AGENT_VERSION}-linux-
musl.tar.gz | tar -C /tmp -zx && \
       export NR_INSTALL_USE_CP_NOT_LN=1 && \
         export NR_INSTALL_SILENT=1 && \
         /tmp/newrelic-php5-*/newrelic-install install && \
           rm -rf /tmp/newrelic-php5-* /tmp/nrinstall*

COPY    --from=vendor    --chown=www-data:www-data    /app/vendor
/app/vendor
COPY --chown=www-data:www-data . .
RUN chmod -R 777 /app/storage
RUN composer dump-autoload

CMD ["/run.sh"]
```

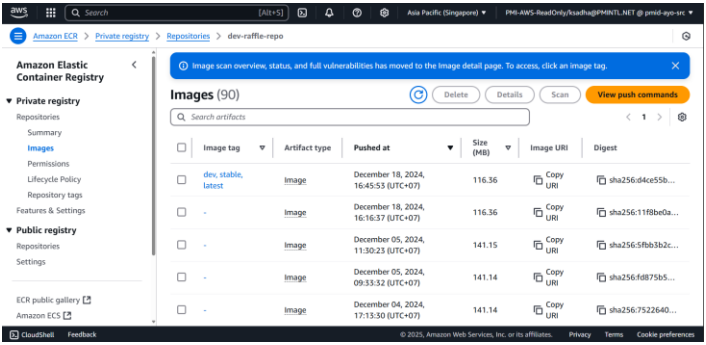
Kode Semu 5.2 Implementasi *Dockerfile Service Docker Image*

Pada Kode sumber 5.2, bisa dilihat bahwa *Service Docker Image* ini menggunakan *Base Docker Image* yang telah di *build* sebelumnya dan di unggah ke dalam ECR. Kemudian,

ditambahkan *dependencies* spesifik pada *service* tersebut seperti; Git, OpenSSL, libiconv, AWS-CLI, Curl, NewRelic, dan lainnya.

5.3 Implementasi Repositori ECR

Sebagian besar *backend service* dalam AYO SRC memiliki repositori AWS *Elastic Container Registry* (ECR). Repositori ini berfungsi sebagai tempat penyimpanan *docker image* dari setiap *service* yang berhasil di *build*. Salah satu repositori di ECR adalah untuk *backend service* ayo-raffle, yang menyimpan docker image dari service tersebut. Gambar 5.1 menunjukkan tampilan dari repositori untuk *backend service* ayo-raffle, di mana setiap *docker image* disusun berdasarkan tag yang mencerminkan versi atau status *build*-nya.



Gambar 5.1 Repositori *Service* ayo-raffle pada ECR

5.4 Implementasi Service ECS

Layanan AWS Elastic Container Service (ECS) memiliki peran penting dalam menjalankan sebagian besar backend services dalam AYO SRC. ECS dirancang untuk mengelola dan mengorkestrasi container secara efisien, memungkinkan backend services untuk berjalan dalam lingkungan yang terisolasi dan terstandarisasi. Layanan ini menggunakan docker image yang

sebelumnya telah berhasil di *build* dan disimpan di AWS Elastic Container Registry (ECR).

Docker image tersebut memuat semua dependensi, konfigurasi, dan kode aplikasi yang diperlukan untuk menjalankan backend services. Dengan menggunakan ECS, proses *deployment* menjadi lebih sederhana karena ECS dapat menarik *docker image* langsung dari ECR, memastikan bahwa setiap service berjalan dengan versi image yang sesuai.

Terdapat dua komponen utama dalam ECS yang diperlukan sebelum bisa melakukan deployment yaitu; *Task Definition* dan *Service*.

- *Task Definition* adalah templat yang mendeskripsikan bagaimana container akan dijalankan. Task Definition mencakup pengaturan seperti nama image, jumlah CPU dan memori yang dialokasikan, port yang diekspos, variabel lingkungan, serta konfigurasi lainnya yang diperlukan untuk menjalankan container.
- *Service* adalah komponen yang mengelola dan mempertahankan tugas (*tasks*) yang berjalan pada cluster ECS. Service memastikan bahwa jumlah instans container yang diinginkan tetap aktif dan berjalan, bahkan jika salah satu container mengalami kegagalan.

Kombinasi *Task Definition* dan *Service* memungkinkan ECS untuk mengelola dan menjalankan *backend services* dengan cara yang terorganisir dan skalabel. Berikut adalah implementasi ECS *Task Definition* dan *Service* untuk salah satu *backend service*.

5.4.1 ECS Task Definition

Seperti yang telah dibahas pada subbab 5.4, *Task Definition* bagaimana sebuah kontainer akan dijalankan dalam

ECS. Kode semu 5.3 adalah contoh implementasi *Task Definition* dalam bentuk JSON. Bagian containerDefinitions berisi informasi utama untuk kontainer yang akan dijalankan seperti;

- *name* : Nama *Task*.
- *image* : *Docker image* yang digunakan.
- *cpu* : Limit cpu untuk container.
- *portMappings* : Pemetaan port dari host ke kontainer.
- *essential* : Menandai bahwa kontainer tersebut harus berhasil dijalankan.
- *environtment* : Variabel *Environtment* kontainer.
- *mountPoints* : Definisi *mount volume* untuk kontainer.
- *volumesFrom* : Menggunakan *mount* dari kontainer lain.
- *secrets* : Variabel *Environtment* yang tersimpan dalam AWS *Secrets Manager* atau *Parameter Store*.

```
{
    "taskDefinitionArn": "ARN TASK DEFINITION"
    "containerDefinitions": [
        {
            "name": "dev-raffle-task",
            "image": "ARN DOCKER IMAGE",
            "cpu": 0,
            "portMappings": [
                {
                    "name": "dev-raffle-task-80-tcp",
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "environment": [
                {
                    "name": "SERVICE_NAME",
                    "value": "raffle-api-dev"
```

```
            },
            {
                "name": "DD_SERVICE_NAME",
                "value": "raffle-api-dev"
            },
            {
                "name": "DD_TRACE_APP_NAME",
                "value": "areas-api-dev"
            },
            {
                "name": "DD_AGENT_HOST",
                "value": "localhost"
            },
            {
                "name": "WEBPATH",
                "value": "/app/public"
            },
            {
                "name": "ENV",
                "value": "laravel"
            },
            {
                "name": "DD_TRACE_AGENT_PORT",
                "value": "8126"
            },
            {
                "name": "ENV_PATH",
                "value": "Path of ENV"
            }
        ],
        "mountPoints": [],
        "volumesFrom": [],
        "secrets": [
            {
                "name": "DB_AYO_B2C_PASSWORD",
                "valueFrom": "ARN SSM PARAMETER"
            },
            {
                "name": "DB_AYO_B2C_USERNAME",
                "valueFrom": "ARN SSM PARAMETER"
            }
        ],
        "logConfiguration": // Log Config
        "systemControls": []
    }
],
```

```
    "family": "Dev-Raffle-Task",
    "taskRoleArn": "ARN TASK ROLE
    "executionRoleArn": "ARN EXECUTION ROLE"
    "networkMode": "awsvpc",
    "revision": 6,
    "volumes": [],
    "status": "ACTIVE",
    "requiresAttributes": [
        // Attributes
    ],
    "placementConstraints": [],
    "compatibilities": [
        "EC2",
        "FARGATE"
    ],
    "requiresCompatibilities": [
        "FARGATE"
    ],
    "cpu": "256",
    "memory": "512",
    "registeredAt": "2024-10-23T02:23:59.407Z",
    "registeredBy": "ARN AWS USER",
    "enableFaultInjection": false,
    "tags": [
        {
            "key": "Environment",
            "value": "DEV"
        }
    ]
}
```

Kode Semu 5.3 ECS *Task Definition* Dalam Bentuk JSON

5.4.2 ECS Service

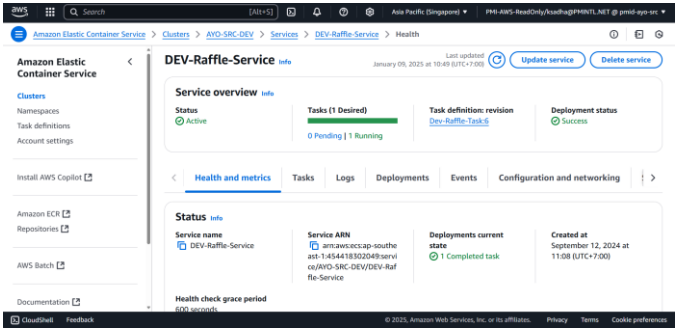
ECS *Service* bertugas untuk menjalankan ECS *Task Definition* yang telah didefinisikan pada Kode Semu 5.3. ECS *Service* berisikan konfigurasi utama seperti:

- *Launch type* yaitu tipe jalannya service. Saat ini terdapat dua *launch type* dalam ECS; EC2 atau *Elastic Compute 2* yang menggunakan instansi EC2 sebagai *host* untuk container, FARGATE yaitu layanan *serverless compute*

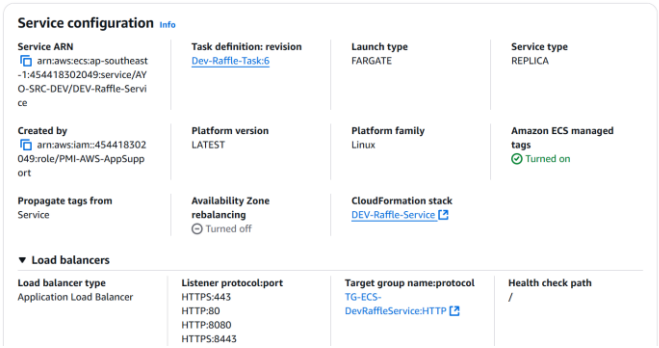
AWS yang memungkinkan untuk tidak menggunakan EC2.

- *Load balancers* yaitu *load balancer* yang digunakan oleh ECS *Service* berisi informasi seperti protokol dan *port* yang digunakan.
- *Network Configuration* yaitu pengaturan jaringan untuk ECS *Service* yang berisi informasi serperti *subnet*, *security group*, *load balancer*, dan lainnya.

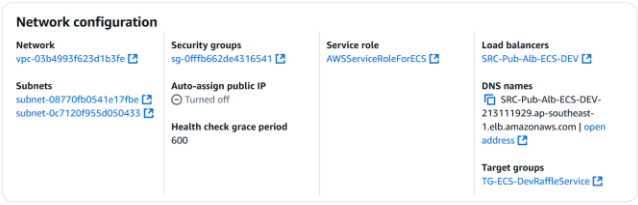
Gambar 5.2, 5.3, dan 5.4 adalah tampilan ECS Service untuk salah satu *backend service* yaitu ayo-raffle. Bisa dilihat bahwa *launch type* yang digunakan adalah FARGATE sehingga tidak perlu menggunakan instansi EC2.



Gambar 5.2 Tampilan Utama ECS Service



Gambar 5.3 Tampilan Pengaturan ECS Service



Gambar 5.4 Tampilan Pengaturan Jaringan ECS Service

5.5 Implementasi Pembaruan *Framework* Laravel

Langkah pertama untuk melakukan pembaruan *framework* Laravel yaitu dengan mengubah versi *framework* Laravel pada composer.json. Perlu diingat bahwa dengan mengubah versi *framework* Laravel dapat menyebabkan kode yang ada sebelumnya menjadi rusak atau muncul *bug*. Hal ini karena tiap kenaikan versi *framework* Laravel terdapat *breaking changes* yang disebabkan karena perubahan *syntax*, parameter, tipe data, format data dan lainnya. Selain itu, perlu dipastikan bahwa *dependencies service* lainnya yang terdapat pada composer.json kompatibel dengan versi *framework* Laravel baru. Dokumentasi resmi Laravel memiliki panduan untuk pembaruan *framework* Laravel untuk setiap versinya sehingga dapat mengatasi setiap *breaking changes* yang

ditemukan. Laporan ini akan menyajikan contoh-contoh implementasi penanganan breaking changes yang ditemukan pada salah satu layanan backend yang menjadi tanggung jawab penulis untuk diperbarui yaitu ayo-raffle.

5.5.1 Implementasi Composer.json

Composer.json adalah file yang menentukan modul-modul composer yang terinstall sesuai versi yang ditentukan serta *dependencies*-nya. Kode semu 5.4 dan 5.5 menunjukkan sebelum dan sesudah menaikkan versi dari laravel 8 ke laravel 9.

```
// ...
"require": {
        "php": "^7.3|^8.0",
        "fruitcake/laravel-cors": "x.x",
        "guzzlehttp/guzzle": "x.x",
        "laravel/framework": "^8.75",
        "laravel/sanctum": "x.x",
        "laravel/tinker": "x.x",
        "fideloper/proxy": "x.x",
        // Dependencies lainnya
    },
    "require-dev": {
        "facade/ignition": "x.x",
        "fakerphp/faker": "x.x",
        "laravel/sail": "x.x",
        "mockery/mockery": "x.x",
        "nunomaduro/collision": "x.x",
        "phpunit/phpunit": "x.x"
        // Dependencies lainnya
    },
// ...
```

Kode Semu 5.4 Composer.json Laravel 8

```
// ...
"require": {
        "php": "^8.0.2",
        "guzzlehttp/guzzle": "x.x",
        "laravel/framework": "^9.19",
        "laravel/sanctum": "x.x",
        "laravel/tinker": "x.x"
```

```
        // Dependencies lainnya
    },
    "require-dev": {
        "fakerphp/faker": "x.x",
        "laravel/pint": "x.x",
        "laravel/sail": "x.x",
        "mockery/mockery": "x.x",
        "nunomaduro/collision": "x.x",
        "phpunit/phpunit": "x.x",
        "spatie/laravel-ignition": "x.x"
        // Dependencies lainnya
    },
// ...
```

Kode Semu 5.5 Composer.json Laravel 9

Bisa dilihat bahwa file composer.json pada Laravel 9 tidak memiliki modul fruitcake/laravel-cors dan filedoper/proxy. Hal itu karena Laravel 9 telah mengintegrasi kedua *composer module* tersebut secara langsung di *framework*. Selain itu juga, terdapat perubahan versi minimum PHP menjadi 8.0.2 dikarenakan Laravel 9 memerlukan versi minimum tersebut.

5.5.2 Implementasi App Bootstrap

File bootstrap/app.php dalam Laravel berperan penting dalam proses inisialisasi aplikasi. File ini bertugas membuat instance aplikasi melalui kelas Illuminate\Foundation\Application, yang berfungsi sebagai *container service* (*IoC container*) untuk mengelola berbagai komponen dan dependensi aplikasi. Dengan demikian, file bootstrap/app.php menjadi pintu masuk utama yang menyiapkan seluruh elemen dasar yang diperlukan agar aplikasi Laravel dapat berjalan dengan baik.

Laravel 11 membawa perubahan signifikan pada struktur proyek, khususnya pada konfigurasi *routing*. Pada Laravel 10, konfigurasi *middleware* berada di app/Http/Kernel.php, sedangkan konfigurasi *routing* dikelola di app/Providers/RouteServiceProvider.php. Namun, pada Laravel 11, kedua konfigurasi tersebut kini dipusatkan langsung di file

bootstrap/app.php, menjadikan proses pengelolaan *middleware* dan *routing* lebih terintegrasi.

Kode semu 5.6 menunjukkan isi pada *file* app/Http/Kernel.php dalam Laravel 10 dimana terdapat variabel yang menyimpan daftar *middleware* global dan daftar alias untuk *middleware*.

```
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    protected $middleware = [/* list global middleware */];

    protected $middlewareAliases = [/* list alias untuk
middleware */];
}
```

Kode Semu 5.6 File app/Http/Kernel.php Laravel 10

Kode semu 5.7 menunjukkan isi pada *file* app/Providers/RouteServiceProvider.php dalam Laravel 10 dimana di dalam fungsi routes() bisa didefinisikan *route* untuk *service*.

```
<?php

namespace App\Providers;

/* ... */

class RouteServiceProvider extends ServiceProvider
{
    public function boot(): void
    {
        /* ... */

        $this->routes(function () {
            /* Definisi-definisi route */
```

```
        });

        /* ... */
    }
}
```

Kode Semu 5.7 File app/Providers/RouteServiceProvider.php Laravel 10

Kode semu 5.8 menunjukkan isi pada *file* bootstrap/app.php dalam Laravel 10 dimana file ini menjalankan *singleton* untuk aplikasi.

```
<?php

$app = new Illuminate\Foundation\Application(
    $_ENV['APP_BASE_PATH'] ?? dirname(__DIR__)
);


$app->singleton(
    Illuminate\Contracts\Http\Kernel::class,
    App\Http\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Console\Kernel::class,
    App\Console\Kernel::class
);

$app->singleton(
    Illuminate\Contracts\Debug\ExceptionHandler::class,
    App\Exceptions\Handler::class
);


return $app;
```

Kode Semu 5.8 File bootstrap/app.php Laravel 10

Kode semu 5.9 menunjukkan isi pada *file* bootstrap/app.php setelah pembaruan ke Laravel 11 dimana semua informasi *middleware* serta *routing* pada *file* app/Http/Kernel.php

dan *file* app/Providers/RouteServiceProvider.php telah dipindahkan kedalam *file* tersebut.

```
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use Illuminate\Support\Facades\Route;

return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        api: __DIR__.'/../routes/api.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
        then: function() {
            /* Definisi-definisi route */
        }
    )
    ->withMiddleware(function (Middleware $middleware) {
        $middleware->use([\* list global middleware *\]);

        $middleware->alias([\* list alias untuk middleware *\]);
    })
    ->withExceptions(function (Exceptions $exceptions) {
        /* custom error handling */
    })->create();
```

Kode Semu 5.9 File bootstrap/app.php Laravel 11

5.5.3 Implementasi Service Providers

Service Provider dalam Laravel adalah kelas yang bertanggung jawab untuk menginisialisasi dan mengatur berbagai komponen atau layanan yang digunakan oleh aplikasi. *Service Provider* bertindak sebagai tempat utama untuk mengikat layanan ke *container service* (*IoC container*), mendaftarkan dependensi, serta melakukan *bootstrapping* fitur atau logika khusus. Pada Laravel 10 Semua *Service Provider* terdaftar di *file* konfigurasi config/app.php, pada bagian *providers*. Laravel menyediakan berbagai *Service Provider* bawaan, seperti untuk *database*,

routing, dan validasi, tetapi pengembang juga dapat membuat *Service Provider* kustom untuk kebutuhan spesifik aplikasi.

Laravel 11 membawa perubahan, dimana *Service Provider* didaftarkan di dalam *file* bootstrap/providers.php. Kode semu 5.10 menunjukkan isi *file* config/app.php pada Laravel 10.

```
<?php

use Illuminate\Support\Facades\Facade;
use Illuminate\Support\ServiceProvider;

return [
    /* ... */
    'providers' => ServiceProvider::defaultProviders()->merge([
        /*
         * Package Service Providers...
         */

        /*
         * Application Service Providers...
         */
        App\Providers\AppServiceProvider::class,
        App\Providers\AuthServiceProvider::class,
        // App\Providers\BroadcastServiceProvider::class,
        App\Providers\EventServiceProvider::class,
        App\Providers\RouteServiceProvider::class,
    ])->toArray(),
    /* ... */
];
```

Kode Semu 5.10 *File* config/app.php Laravel 10

Kode semu 5.11 menunjukkan isi pada *file* bootstrap/providers.php setelah pembaruan ke Laravel 11 dimana *Service Providers* telah dipindahkan ke dalam *file* tersebut. Bisa dilihat juga seperti pada suubbab 5.5.2 bahwa app/Providers/RouteServiceProvider.php sudah tidak digunakan.

```
<?php

return [
    App\Providers\AppServiceProvider::class,
```

```
    App\Providers\AuthServiceProvider::class,
    // App\Providers\BroadcastServiceProvider::class,
    App\Providers\EventServiceProvider::class,
    App\Providers\ValidationServiceProvider::class
];
```

Kode Semu 5.11 *File* bootstrap/providers.php Laravel 11

5.5.4 Implementasi CORS dan Trusted Proxies

Laravel 9 mengintegrasikan dua modul composer ke dalam *framework* yaitu fruitcake/laravel-cors yang sebelumnya dikenal dengan barryvdh/laravel-cors, dan fideloper/proxy. Konfigurasi untuk kedua modul tersebut juga ada perubahan, seperti pada laravel-cors yang menggunakan *snake case* dimana sebelumnya *camel case*, dan proxy dimana definisi *header* berubah.

Kode Semu 5.12 menunjukkan *file* config/cors.php pada Laravel 8 dengan modul fruitcake/laravel-cors yang konfigurasinya menggunakan *camel case*.

```
<?php

return [
    'supportsCredentials' => false,
    'allowedOrigins' => explode(',', env('CORS_ALLOWED_ORIGINS',
'*')),
    'allowedOriginsPatterns' => [],
    'allowedHeaders' => explode(',', env('CORS_ALLOWED_HEADERS',
'*')),
    'allowedMethods' => explode(',', env('CORS_ALLOWED_METHODS',
'*')),
    'exposedHeaders' => [],
    'maxAge' => 0,
];
```

Kode Semu 5.12 *File* config/cors.php Laravel 8

Kode Semu 5.13 menunjukkan *file* config/cors.php pada Laravel 9 yang sudah diubah menjadi *snake case*.

```
<?php
```

```
return [
    'supports_credentials' => false,
    'allowed_origins'              =>              explode(',',
env('CORS_ALLOWED_ORIGINS', '*')),
    'allowed_origins_patterns' => [],
    'allowed_headers'              =>              explode(',',
env('CORS_ALLOWED_HEADERS', '*')),
    'allowed_methods'              =>              explode(',',
env('CORS_ALLOWED_METHODS', '*')),
    'exposed_headers' => [],
    'maxAge' => 0,
];
```

Kode Semu 5.13 *File* config/cors.php Laravel 9

Kode Semu 5.14 menunjukkan *file* config/cors.php pada Laravel 8 dengan modul fideloper/proxy dengan definisi *header* lama.

```
<?php

namespace App\Http\Middleware;

use Illuminate\Http\Request;
use Fideloper\Proxy\TrustProxies as Middleware;

class TrustProxies extends Middleware
{
    protected $proxies;
    protected $headers = Request::HEADER_X_FORWARDED_ALL;
}
```

Kode Semu 0.14 *File* app/Http/Middleware/TrustProxies.php Laravel 8

Kode Semu 5.15 menunjukkan *file* config/cors.php pada Laravel 9 dengan definisi *header* baru.

```
<?php

namespace App\Http\Middleware;

use Illuminate\Http\Request;
```

```
use Illuminate\Http\Middleware\TrustProxies as Middleware;

class TrustProxies extends Middleware
{
    protected $proxies;
    protected $headers =
        Request::HEADER_X_FORWARDED_FOR |
        Request::HEADER_X_FORWARDED_HOST |
        Request::HEADER_X_FORWARDED_PORT |
        Request::HEADER_X_FORWARDED_PROTO |
        Request::HEADER_X_FORWARDED_AWS_ELB;
}
```

Kode Semu 5.15 *File* app/Http/Middleware/TrustProxies.php Laravel 9

5.5.5 Implementasi Konversi Format Datetime

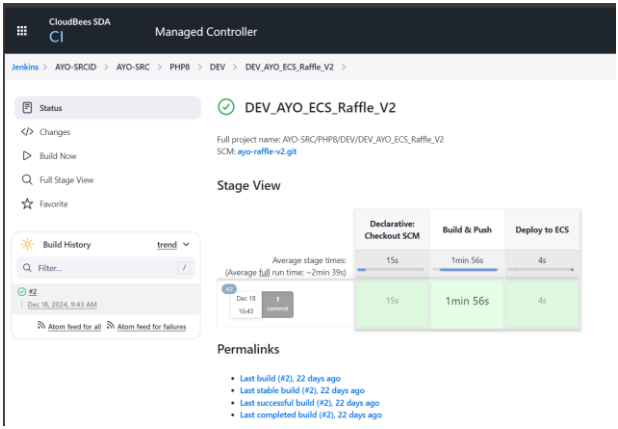
Pada Laravel 7, terdapat perubahan dari Laravel 6 yang cukup signifikan, yaitu format waktu dan tanggal. Laravel 6 menggunakan format yyyy-mm-dd hh:ii:ss. Di sisi lain, Laravel 7 menggunakan format ISO-8601 yang selalu dalam UTC dan berisi informasi zona waktu. Untuk mengatasi ini, perlu *override* fungsi serializeDate() pada setiap model. Kode semu 5.16 menunjukkan implementasi serializeDate() yang digunakan di setiap model.

```
<?php

/* ... */

use DateTimeInterface;

protected function serializeDate(DateTimeInterface $date)
{
    return $date->format('Y-m-d H:i:s');
}

/* ... */
```

Kode Semu 5.16 Implementasi serializeDate() pada *File* Model Laravel 7

5.5.6 Implementasi Jenkins Pipeline dan Jenkinsfile

Jenkins adalah platform CI/CD yang digunakan AYO SRC untuk *build*, *test*, dan *deploy* aplikasi. Setiap *backend service* memiliki *pipeline* masing-masing yang memerlukan detail *pipeline* di dalam sebuah *file* yang bernama *jenkinsfile*. Gambar 5.5 adalah tampilan Jenkins *Pipeline* untuk *backend service* ayo-raffle yang menunjukkan histori *build,* status *build*, commit yang *trigger* build tersebut, dan lainnya.



Gambar 5.5 Tampilan Jenkins Pipeline untuk ayo-raffle

Pipeline ini mendapatkan informasi detail untuk setiap proses dalam CI/CD pipeline dari *file* Jenkinsfile. Kode semu 5.17 adalah implementasi Jenkinsfile yang berisi proses untuk *build docker image*, *push image* tersebut ke ECR, dan *deploy*.

```
pipeline{
    agent {
        kubernetes {
            /* konfigurasi agen kubernetes */
        }
    }
```

```
    environment {
        AWS_CREDS = "CREDS"
        AWS_ACCOUNT_ID = "CREDS"
        AWS_REGION = "CREDS"
        AWS_ECR_PROJECT_NAME ="dev-raffle-repo"
        AWS_ECR_REPOSITORY_TARGET
="${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_REGION}.amazonaws.com/${AWS_EC
R_PROJECT_NAME}"
        AWS_ECS_CLUSTER_NAME = "CREDS"
        AWS_ECS_SERVICE_NAME = "DEV-Raffle-Service"
        GIT_AUTH = credentials('CREDS')
        SSM_PARAMSTORE_NRVERSION = "CREDS"
    }

    stages {
        stage('Build & Push') {
            steps {
                                withCredentials([[$class:
'AmazonWebServicesCredentialsBinding',        accessKeyVariable:
'AWS_ACCESS_KEY_ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY',
credentialsId: AWS_CREDS]]) {
                    script{
                        /* ... */

                        /* build image */
                        container('kaniko-executor') {
                            sh """
                            /kaniko/executor        --dockerfile
Dockerfile         --context         .         --force         --
destination=${AWS_ECR_REPOSITORY_TARGET}:dev                  --
destination=${AWS_ECR_REPOSITORY_TARGET}:stable               --
destination=${AWS_ECR_REPOSITORY_TARGET}:latest      --build-arg
NEWRELIC_AGENT_VERSION=\"${NEWRELIC_VERSION}\"       --build-arg
GITU=\"${GIT_AUTH_USR}\" --build-arg GITPW=\"${GIT_AUTH_PSW}\" -
-verbosity fatal
                            """
                        }
                    }
                }
            }
        }

        stage('Deploy to ECS') {
            steps {
                withCredentials([[$class:
'AmazonWebServicesCredentialsBinding',        accessKeyVariable:
```

```
'AWS_ACCESS_KEY_ID', secretKeyVariable: 'AWS_SECRET_ACCESS_KEY',
credentialsId: AWS_CREDS]]) {
                    script{
                        container('aws') {
                            sh """
                                aws ecs update-service \
                                --cluster
${AWS_ECS_CLUSTER_NAME} \
                                --service
${AWS_ECS_SERVICE_NAME} \
                                --force-new-deployment \
                                --region ${AWS_REGION}
                                """
                        }
                    }
                }
            }
        }
    }
}
```

Kode Semu 5.17 Implementasi *File* Jenkinsfile

BAB VI
PENGUJIAN DAN EVALUASI

Tujuan dari pengujian dan evaluasi adalah untuk memastikan bahwa pembaruan versi PHP dan Laravel telah berhasil diimplementasikan dan berjalan dengan optimal sesuai dengan kebutuhan sistem. Langkah ini dilakukan untuk memverifikasi bahwa semua layanan *backend service* dalam ekosistem AYO dapat berfungsi dengan baik, tanpa adanya gangguan pada fungsionalitas yang ada, serta untuk memastikan kompatibilitas dengan infrastruktur dan teknologi terbaru.

6.1. Tujuan Pengujian

Pengujian dilakukan terhadap *backend* ekosistem AYO untuk memastikan kemampuan arsitektur sistem dalam menangani pembaruan versi PHP dan Laravel. Proses ini bertujuan untuk memverifikasi kinerja layanan dalam melayani permintaan dari aplikasi, memastikan stabilitas, kompatibilitas, dan efisiensi sistem setelah pembaruan dilakukan.

6.2. Kriteria Pengujian

Pengujian terhadap sistem dilakukan untuk memastikan bahwa proses pembaruan versi PHP dan Laravel berjalan dengan sukses dan menghasilkan sistem yang stabil serta berkinerja optimal. Dua kriteria utama digunakan sebagai tolok ukur dalam pengujian ini, yaitu:

a. Performa service setelah melakukan pembaruan
b. Kesesuaian response terhadap request pada service

6.3. Skenario Pengujian

Pengujian dan evaluasi dilakukan menggunakan Postman, sebuah platform yang dirancang untuk mengembangkan dan menguji API secara efisien. Proses pengujian dimulai dengan membuat koleksi API yang berisi endpoint-endpoint layanan backend yang diperbarui. Setiap endpoint diuji dengan mengirimkan permintaan HTTP, seperti GET, POST, PUT, atau DELETE, untuk memverifikasi respons yang diberikan. Pengujian melibatkan validasi terhadap status kode HTTP, waktu respons, dan data yang dikembalikan untuk memastikan bahwa layanan berjalan sesuai spesifikasi. Postman juga digunakan untuk menulis skrip otomatisasi pada pengujian untuk mengevaluasi respons API secara lebih mendalam, seperti pengujian autentikasi atau pengolahan data kompleks. Hasil pengujian dianalisis untuk mengidentifikasi potensi masalah, memastikan performa optimal, dan memverifikasi bahwa pembaruan telah berhasil diterapkan tanpa mengganggu fungsionalitas sistem.

6.4. Evaluasi Pengujian

Hasil pengujian dilakukan berdasarkan pengamatan terhadap perilaku layanan backend ekosistem AYO setelah pembaruan versi PHP dan Laravel, dengan menggunakan Postman sebagai alat utama untuk mengembangkan dan menguji API. Setiap skenario uji coba dirancang untuk mengukur kriteria utama, yaitu performa layanan dan kesesuaian respons terhadap permintaan. Proses pengujian melibatkan pengiriman permintaan HTTP pada endpoint layanan, analisis waktu respons, validasi data yang dikembalikan, serta pemantauan stabilitas sistem. Hasil uji coba yang telah dilakukan dirangkum dalam Tabel 6.1, yang memberikan gambaran mengenai kinerja layanan dan memastikan bahwa pembaruan tidak mengganggu fungsionalitas sistem.

Tabel 6.1 Tabel Hasil Pengujian dan Evaluasi

Service	Kriteria Pengujian	Hasil Pengujian
ayo-newsfeed	Performa	Terpenuhi
	Kesesuaian Response	Terpenuhi
ayo-consumer-permainan	Performa	Terpenuhi
	Kesesuaian Response	Terpenuhi
ayo-raffle	Performa	Terpenuhi
	Kesesuaian Response	Terpenuhi

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Kesimpulan yang diperoleh setelah melakukan pembaruan *service backend* pada ekosistem AYO di PT SRC Indonesia Sembilan selama kegiatan kerja praktik adalah sebagai berikut:

a. Pembaruan versi PHP dan Laravel berhasil meningkatkan keamanan sistem, dengan adanya perbaikan pada patch keamanan dan fitur-fitur terbaru yang dapat melindungi aplikasi dari potensi kerentanannya.
b. Dengan menggunakan versi PHP dan Laravel yang lebih baru, AYO SRC kini lebih dapat diandalkan, memastikan bahwa layanan dapat berjalan lebih stabil dan aman, serta mendukung transformasi digital UMKM dengan risiko yang lebih rendah.

7.2. Saran

Saran untuk pembaruan *service backend* pada ekosistem AYO di PT SRC Indonesia Sembilan adalah sebagai berikut :

a. Mengimplementasi *automation testing* sehingga meringankan load QA *tester* dikarenakan jumlah *endpoint* yang banyak.
b. Penambahan komen-komen pada kode sehingga *programmer* lain dapat memahami kode yang ada lebih baik.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

AWS. (2024). *Amazon Web Services Documentation*. https://docs.aws.amazon.com/.

Banyu Adil, S., Richard Beeh, Y., Studi Teknik Informatika, P., Teknologi Informasi, F., Kristen Satya Wacana, U., Salatiga, K., & Jawa Tengah, P. (2024). E-P-Implementasi Monitoring Sistem Perusahaan On-Premises dan Cloud Menggunakan Teknologi Jenkins. Dalam *Jurnal Indonesia : Manajemen Informatika dan Komunikasi (JIMIK)* (Vol. 5, Nomor 2). https://journal.stmiki.ac.id

Caglayan, D., & Özcan-Top, Ö. (2024). Revisiting Technical Debt Types and Indicators for Software Systems. *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, 834–841. https://doi.org/10.1145/3605098.3636043

Dese Fitriani, W., Hidayat, A., Wayan Suardinata, I., Negeri Banyuwangi, P., Raya Jember NoKM, J., Kabat, K., Banyuwangi, K., & Timur, J. (2024). Perancangan Backend Pada Sistem Informasi Penerimaan Peserta Didik Baru (PPDB) SMK Bina Mandiri Al Qodiriyah Srono Backend Design In The New Student Acceptance Information System (PPDB) At Bina Mandiri Al Qodiriyah Srono Vocational School. Dalam *JIKOM: Jurnal Informatika dan Komputer* (Vol. 14, Nomor 2).

Dubey, P., Kumar Tiwari, A., & Raja, R. (2023). *Amazon Web Services: the Definitive Guide for Beginners and Advanced Users*. BENTHAM SCIENCE PUBLISHERS. https://doi.org/10.2174/9789815165821123010 1

Oliveira, F., Goldman, A., & Santos, V. (2015). Managing Technical Debt in Software Projects Using Scrum: An Action Research. *2015 Agile Conference*, 50–59. https://doi.org/10.1109/Agile.2015.7

Poulton, N. (2023). *Getting Started with Docker*. http://leanpub.com/gsd

Yin, K., & Du, Q. (2019). *On Representing Resilience Requirements of Microservice Architecture Systems*. http://arxiv.org/abs/1909.13096

BIODATA PENULIS I

Nama	: Ketut Arda Putra Mahotama Sadha
Tempat, Tanggal Lahir	: Denpasar, 6 Agustus 2003
Jenis Kelamin	: Laki-laki
Telepon	: +6287862278072
Email	: ardaputramahot@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 7 (Tujuh)