BAB 1
USULAN GAGASAN

1.1 Latar Belakang Masalah

Perkembangan teknologi yang sangat pesat membuat peningkatan yang luar biasa dalam industri *e-commerce*. *E-commerce* adalah penjualan dan pembelian barang secara *online* untuk konsumen *Business to Consumer* (B2C) dan *Business to Business* (B2B) [1]. Hal tersebut secara tidak langsung menuntut pemilik *platform e-commerce* untuk selalu meningkatkan dan mengembangkan layanannya agar bisnis terus berkembang. *Platform e-commerce* merupakan teknologi digital untuk memudahkan proses *e-commerce*. Lowe's merupakan salah satu *platform e-commerce* berbasis *mobile application* dan *web* yang beralih dari arsitektur *monolithic* menjadi arsitektur *microservices* [2]. Pergantian arsitektur tersebut dikarenakan arsitektur *monolithic* membuat pengembangan fitur *platform* tersebut menjadi kompleks dan lambat yang akhirnya dapat menjadikan perkembangan bisnis perusahaan menurun.

Penggunaan arsitektur *microservices* menyederhanakan pengembangan *platform* dengan memisahkan bagian-bagian independen yang kecil dimana setiap bagian akan memiliki tanggung jawab masing-masing [3][4] [5]. Namun, *microservices* akan efektif ketika hanya dikembangkan dalam skala kecil atau satuan. Jika suatu *platform* akan mengembangkan puluhan, ratusan, ataupun ribuan *microservices* baik itu untuk alasan pengembangan lebih lanjut ataupun penambahan fitur dalam *platform* tersebut, maka hal itu akan menjadi masalah. Masalah tersebut adalah ketergantungan antar *services* yang tinggi. Hal ini tentu berbanding terbalik dengan konsep *microservices*, bahwasannya setiap *services* itu dapat berdiri sendiri [5].

Penggunaan *service mesh* dapat membantu mengatasi permasalah tersebut. *Service mesh* merupakan lapisan *platform* di atas *infrastructure layer* yang memungkinkan komunikasi antar layanan dapat dikelola, diamati, dan aman [5]. Dengan penggunaan konfigurasi *service mesh*, maka pengembang dapat lebih mudah untuk membangun ataupun mengembangkan aplikasi yang terdiri dari banyak *microservices* dalam suatu infrastruktur. Hal ini berbanding terbalik saat menggunakan konfigurasi tanpa *service mesh* yang dapat menyebabkan tahapan berulang, kompleksitas, dan membutuhkan waktu yang lebih lama. *Service mesh* menggunakan alat yang konsisten untuk memperhitungkan semua masalah umum dalam menjalankan suatu *service*, seperti monitoring, *networking*, dan *security* [5].

Untuk meminimalisir *downtime* saat *end-user* mengakses *platform e-commerce*, maka diperlukan monitoring pada *resources metrics* seperti CPU dan *memory*. Proses monitoring dilakukan dengan mengambil *metrics* pada indikator yang telah ditentukan dari *services platform e-commerce*. Pengumpulan *metrics* tersebut menggunakan Prometheus. Hal ini dikarenakan Prometheus bersifat *open-source* sehingga banyak tersedia modifikasi dan kecocokan dengan *tools open-source* lainnya. Selain itu, biaya langganan dan pemeliharaannya relatif lebih murah dibandingkan dengan penerapan *Software as a Service* (SaaS) *all-in-one* seperti Datadog. Fungsionalitasnya pun lebih menyeluruh dibandingkan dengan menggunakan perintah dalam kodingan. Dalam penggunaanya nanti, Prometheus sebagai *time-series database* akan memonitor dan mengumpulkan *metrics.* Lalu, Prometheus dihubungkan dengan Grafana untuk analisis dan visualisasi.

Sebelum *platform e-commerce* dimonitoring, maka fungsionalitas *platform e-commerce* harus diperiksa terlebih dahulu agar dapat mengoptimalkan pemenuhan kebutuhan *end-user*. Pemeriksaan tersebut merupakan *testing* sebelum dan sesudah proses *production*. *End-user* seringkali menemukan *platform e-commerce* yang *not responding* atau *down* sehingga menghambat akses dan penurunan *rating platform*. Penggunaan *manual testing* memang dapat menghemat biaya, namun sering terjadi *human error* sehingga kurang optimal dalam pengecekan *bug.* Oleh karena itu, diperlukan *automated testing* untuk memperluas pengecekan *bug* dengan cepat dan efektif serta bersifat *reusable*.

Dapat disimpulkan bahwa terdapat empat rumusan masalah yang akan dijawab oleh penulis. Pertama, perlunya automasi dalam menjaga *availability platform e-commerce* untuk meminimalisir *downtime*. Kedua, perlunya automasi pengujian *platform e-commerce* berbasis *microservices*. Ketiga, perlunya *automated monitoring* dalam menjaga *reliability platform* pada *cloud*. Keempat, perlunya automasi pembuatan infrastruktur *cloud* agar lebih efektif, efisien, dan terhindari dari *human error*.

Dari permasalahan yang telah dijabarkan di atas, maka penulis mengusulkan solusi berupa Perancangan *Platform E-Commerce* Menggunakan Arsitektur *Microservices* dan Sistem Aplikasi Monitoring Berbasis *Service Mesh*. Happy merupakan *platform e-commerce* yang menjual *voucher games* dan dapat diakses melalui *browser*. *Platform* berbeda dengan *website* karena terdapat komunikasi dua arah dan data personalisasi artinya terdapat proses autentikasi pengisian data dari *end-user* [6]. *Platform e-commerce* ini dibuat berbasis *web* artinya dapat dijalankan dan dibuka melalui *browser* [7].

Pemilihan *voucher games* tersebut dikarenakan bisnis *game online* dapat beradaptasi dan mengalami peningkatan yang signifikan dari sisi peminat serta keuntungan setiap tahunnya [8]. Sedangkan, Anbu dibuat menggunakan *tools* Prometheus dan Grafana dengan tujuan untuk mengoptimasi proses monitoring *platform* Happy serta menjaga *reliability*-nya. Jika terdapat anomali pada sistem, maka Anbu akan mengirimkan *alert notification* melalui Slack.

Terdapat batasan masalah yang menjadi limitasi dalam pengerjaan *capstone design* untuk pembuatan *platform e-commerce* dan sistem aplikasi monitoring. Pertama, pembuatan *platform e-commerce* hanya difokuskan pada implementasi arsitektur *microservices*. Kedua, fitur pembayaran pada *platform e-commerce* Happy tidak dihubungkan pada API *payment* sehingga pengecekan validasi pembayaran dilakukan manual oleh admin melalui WA dan persetujuan pembelian *voucher* akan di-*acc* melalui *platform e-commerce* Happy milik admin. Ketiga, fitur pembelian *voucher games* menggunakan sistem reservasi dan tidak menyediakan fitur keranjang. Keempat, sistem aplikasi monitoring Anbu hanya memonitoring *metrics* berupa CPU, *memory*, dan *request duration*. Kelima, *alerting* pada sistem aplikasi monitoring Anbu hanya mengaplikasikan 2 *alert* yang tersedia di dalam *alertmanager* milik Prometheus.

1.2 Informasi Pendukung Masalah

Terdapat 90% *startup* baru di Amerika menggunakan *cloud computing services* karena mempertimbangkan ketersediaan sistem yang tinggi, rasionalitas perencanaan sumber daya, dan efisiensi energi [9]. Peningkatan kebutuhan *consumer* pada *platform* yang menggunakan teknologi *cloud computing* menyebabkan penambahan *interface* untuk dimonitoring secara periodik dalam interval yang singkat [10]. Proses monitoring secara manual pada *platform* yang kompleks dapat memakan waktu yang lama dan membutuhkan sumber daya manusia yang lebih banyak.

Sedangkan, setiap perusahaan ingin memuaskan konsumen dan mempermudah pengembang dalam mengelola sebuah *platform* dengan memperkecil minimum *downtime* dan melakukan *frequent update* tanpa gangguan [11]. Hal tersebut melandasi penggunaan arsitektur *microservices* yang dioptimalkan dengan konfigurasi *service mesh* dan *Infrastructure as a Code* (IaC). Di sisi lain, terdapat 6% yang melakukan pengujian *platform* secara otomatis dan 28% sisanya menggunakan *manual testing* dari 1725 *executives reported* [12]. Padahal penggunaan *automated testing* lebih unggul dibandingkan *manual testing*.

Terdapat penelitian sebelumnya yang membahas *automation monitoring* untuk aplikasi *microservice* tetapi tidak melakukan kombinasi *autoscaling* HPA dan VPA. Penelitian tersebut berjudul "*Auto-scaling Microservices on IaaS under SLA with Cost-Effective Framework*". Penelitian ini berfokus pada pembuatan *new autoscaling framework* berbasis prediksi *workload* [13]. Selain itu, terdapat juga penelitian tentang pengiriman *alert notification* dari Prometheus ke Telegram [14]. Penelitian tersebut berjudul "*On-Premise Server Monitoring with Prometheus and Telegram Bot*". Pentingnya *alert notification* dalam memberitahukan perilaku anomaly pada *server*, maka penulis menggabungkan kedua penelitian tersebut dengan membuat sistem aplikasi monitoring yang mengkombinasikan HPA dan VPA serta penambahan *alert notification*.

1.3 Analisis Umum

Dalam penelitian ini, terdapat beberapa permasalahan yang dapat dianalisis dari berbagai aspek untuk implementasi sistem pada *platform* Happy dan sistem aplikasi monitoring Anbu.

1.3.1 Aspek Ekonomi

Berikut merupakan aspek ekonomi yang dapat dianalisis.

- Penggunaan Kubernetes pada arsitektur *microservices* dapat mengatasi permasalahan optimasi biaya untuk penambahan serta pengurangan skalabilitas.
- Penggunaan *cloud computing platform* seperti Google Cloud Platform (GCP) dapat memangkas biaya yang cukup besar dikarenakan seluruh perangkat keras serta *service* yang digunakan telah tersedia pada layanan tersebut tanpa harus membeli secara pribadi atau secara *on-premises*.

1.3.2 Aspek Manufakturabilitas

Berikut merupakan aspek manufakturabilitas yang dapat dianalisis.

- Jika dilihat dari aspek Sumber Daya Manusia (SDM), sistem otomatisasi monitoring dapat mengefisiensi *human resource* untuk mencegah berbagai kesalahan yang dapat dilakukan manusia. Dengan adanya perangkat lunak yang bekerja dalam memonitoring *metrics* pada *platform*, maka waktu yang diperlukan untuk proses monitoring menjadi lebih efisien.
- Kerangka kerja seperti Kubernetes, Prometheus, Grafana, dan mesin analitik secara proaktif dalam memonitoring dan mengelola operasi *data center* mampu menskalakan untuk mengakomodasi heterogenitas serta kompleksitas *next-generation systems*.

- Aspek *agility* (kelincahan) pada *microservices* terkait proses pelayanan yang lebih cepat karena suatu komponen tidak bergantung pada komponen lainnya. Selain itu, adanya kemudahan dalam penggunaan *cloud computing platform* saat konfigurasi sumber daya komputasi dalam *server* di *cloud* serta meminimalisir usaha manajemen seperti pengelolaan sumber daya penyimpanan, kecepatan komputasi, dan lain-lain. Lalu, dengan adanya teknologi *autoscaling* dapat memungkinkan teknologi *cloud computing* menjadi lebih stabil dengan respon waktu yang baik.
- Aspek *usability* dimana *Application Programming Interface* (API) membuat layanan *microservices* lebih mudah karena lebih dari satu sistem yang dapat berkomunikasi dengan API.
- Aspek *availability of tools* yakni terdapat banyak alat optimasi yang dapat digunakan untuk menghindari kesalahan konfigurasi dan *deployment*. Infrastruktur yang mendukung persyaratan keamanan aplikasi berbasis layanan *microservices*, salah satunya adalah *service mesh*.

1.3.3 Aspek Keberlanjutan

Berikut merupakan aspek keberlanjutan yang dapat dianalisis.

- Melakukan kerja sama dengan perusahaan *games* untuk menjual *voucher games* mereka.
- Melakukan kerja sama dengan *startup* yang memiliki *small business* dalam memonitoring aplikasi mereka.

1.4 Kebutuhan yang Harus Dipenuhi

Dalam pembuatan *platform e-commerce* bernama Happy dan sistem aplikasi monitoring bernama Anbu, diperlukan beberapa perangkat lunak seperti yang dijelaskan pada Tabel 1.1 Perangkat lunak merupakan suatu sistem komputer yang tidak memiliki bentuk fisik seperti aplikasi yang memiliki fungsi khusus masing-masing. Selain itu, diperlukan juga perangkat keras seperti laptop dengan spesifikasi yang dapat dilihat pada Tabel 1.2 Kemudian, terdapat kebutuhan fungsionalitas untuk setiap sistemnya seperti pada Tabel 1.3.

1.4.1 Perangkat Lunak

Tabel 1.1 Kebutuhan Perangkat Lunak

Perangkat Lunak	Fungsi
Next.js	React *framework* untuk membangun tampilan *platform e-commerce*
Kubernetes	*Platform orchestration* dalam mengelola beberapa kontainer
Google Kubernetes Engine (GKE)	Layanan dari *Google Cloud Platform* (GCP) untuk *deployment*, pengelolaan, dan penskalaan dalam *container*
Docker	Sebagai sistem operasi dalam menjalankan *container*
Go	Bahasa pemrograman untuk membuat aplikasi *back-end*
Figma	Layanan design serta purwarupa *platform*
Visual Studio Code	Sebuah *code editor* untuk membuat sebuah kode sumber
Lighthouse	Untuk mengukur *performance of platform e-commerce*
Go *Unit Test*	*Unit test* bawaan dari Go untuk melakukan pengujian unit
Cypress	*Software automation testing* untuk *End-to-End* (E2E) *testing*
Jest	Sebuah kerangka pengujian JavaScript dapat digunakan untuk *integration test*
Postman	Aplikasi yang berfungsi untuk melakukan RESTful *Application Programming Interface* (API) *Test*
Grafana K6	Alat *open source* untuk pengujian beban dengan mengirimkan *Virtual Users* (VUs)
Prometheus	Alat *open source* untuk *monitoring resources* dan *alert notifications*
Grafana	Alat *open source* untuk memvisualisasikan hasil *metrics* yang ditangkap oleh Prometheus
Terraform	Alat untuk membuat dan mengelola infrastruktur *server* secara automasi dengan kode
Ansible	Alat untuk melakukan konfigurasi manajemen secara automasi pada infrastruktur *server* dengan kode
Cloudflare	Salah satu layanan CDN dalam menyediakan konten dan didistribusikan pada *data center* di berbagai lokasi.
Istio	Layanan *open source* untuk penerapan *service mesh*
Terrascan	*Software* untuk memeriksa keamanan dan konfigurasi Terraform
Helm	*Open-source package manager* untuk mempermudah *deployment*
onlineyamltools.com	Alat *Yet Another Markup Language* (YAML) *validator* untuk pengujian konfigurasi *file* YAML pada manifest
Locust	Alat untuk melakukan *distributed load-testing*

1.4.2 Perangkat Keras

Tabel 1.2 Kebutuhan Perangkat Keras

Perangkat Keras	Spesifikasi
Laptop	RAM minimal 8 GB
	Kapasitas *storage disk* minimal 256 GB
	OS minimal Windows/Linux/Mac OS
	CPU minimal Intel Core i3/ AMD Ryzen 3/Apple M1
	Telah ter-*install* perangkat lunak yang diperlukan

1.4.3 Fungsionalitas

Tabel 1.3 Kebutuhan Fungsionalitas

Sistem	Sub-Sistem	Fungsionalitas
Platform e-commerce	*Front-end*	Dapat menampilkan laman untuk *register, login, logout,* detail produk, detail pembelian, ulasan, dan laman terkait pembayaran
		Dapat menampilkan *dashboard* utama berisi katalog *voucher games*, dan *dashboard* utama di sisi admin
	Back-end	Dapat melakukan *register, login, logout,* membeli *voucher games* di sisi *end-user*, dan memberikan ulasan terkait *voucher games* yang dibeli
		Dapat mencari item penjualan dengan filter
		Dapat melakukan *acc* pembelian *voucer games* di sisi admin
	Arsitektur	Dapat di-*debug*, di-*deploy*, dan di-*manage* secara individual pada tiap *service*-nya
	Integration Test	Dapat berinteraksi dengan tepat antar modul yang ada di dalam *front-end*
Sistem Aplikasi Monitoring	Monitoring	Dapat terintegrasi dan mengambil *metrics* dari *platform e-commerce* dengan Prometheus.
		Dapat memberikan *alert notification* ke Slack
	Visualisasi	Dapat menampilkan visualisasi *metrics* ke bentuk grafis dalam *dashboard* Grafana
Infrastruktur	*Infrastructure as a Code*	Dapat melakukan otomasi pembuatan infrastruktur Cloud
	Service Mesh	Dapat membantu komunikasi antar *services* dan *observability*
	Container Orchestration	Dapat melakukan *autoscaling* secara *horizontal* dan *vertical*
	Helm	Dapat menyederhanakan *deployment* pada Kubernetes

1.5 Solusi Sistem yang Diusulkan

1.5.1 Karakteristik Produk

1.5.1.1 *Platform E-Commerce*

Platform e-commerce berbasis aplikasi *web* dibuat menggunakan arsitektur *microservices*. *Platform* ini bernama Happy dan ada tiga kategori pembagian fitur yaitu fitur utama dan fitur dasar pada *platform* Happy.

- Fitur Utama

  Arsitektur yang digunakan adalah *microservices* untuk memudahkan *developers* dalam pengembangan *platform*-nya. *End-user* dapat membeli *voucher games* yang disediakan pada *platform* Happy. Ketersediaan *voucher* ini harus bermitra terlebih dahulu dengan perusahaan *games* dan sejenisnya.

- Fitur Dasar

  End-user dapat melihat detail dari tiap *voucher games* serta dapat melihat *feedback* pada *item games* dari *end-user* lain.

1.5.1.2 Sistem Aplikasi Monitoring

Sistem Aplikasi Monitoring dibuat menggunakan Prometheus dan Grafana. Sistem Aplikasi Monitoring ini bernama Anbu dan terdapat dua kategori pembagian fitur yaitu fitur utama dan tambahan. Berikut merupakan penjelasannya.

- Fitur Utama

  Sistem Aplikasi Monitoring yang digunakan untuk memantau *platform e-commerce* berbasis *web* pada *cloud* menggunakan Prometheus untuk mengambil dan menyimpan *metrics* dari bagian *back-end platform* yang ada dalam Kubernetes. Kemudian, untuk dilakukan visualisasi *metrics* ke dalam bentuk grafik dan angka yang mudah dimengerti dengan menggunakan Grafana.

- Fitur Tambahan

  Dapat terintegrasi dengan platform perpesanan yaitu Slack untuk mengirimkan notifikasi *alert* ketika terjadi kesalahan konfigurasi *deployment.* Terdapat dua *alert* yang diaktifkan berasal dari alertmanager di dalam Prometheus.

1.5.2 Skenario Penggunaan

1.5.2.1 *Platform E-Commerce*

End-user merupakan seorang *gamers* yang *mengunjungi platform e-commerce* Happy melalui *browser*. Kemudian, *end-user* melihat-lihat *item games* berupa *voucher games* yang tersedia lengkap dengan keterangan dan harganya. Ketika *end-user* ingin membeli *voucher* yang tersedia, *end-user* harus *login* ke *platform e-commerce* Happy. Jika *end-user* belum mempunyai akun Happy, maka *end-user* diarahkan untuk melakukan registrasi akun dengan memasukkan beberapa data penting seperti *username* dan *password*. Pada saat *end-user* melakukan *registrasi*, data *end-user* tersimpan di *database* agar saat *login* terjadi penyesuaian data yang dimasukkan oleh *end-user* dengan data di *database*. Jika datanya sesuai, maka *end-user* dapat masuk ke *platform* Happy. Jika datanya tidak sesuai, maka *end-user* diarahkan untuk mengulang kembali pengisian datanya. Ketika *end-user* berhasil masuk dan berada di halaman *catalog*, *end-user* dapat memilih dan membeli *voucher games* yang tersedia serta melakukan pembayaran secara *online*.

1.5.2.2 Sistem Aplikasi Monitoring

Insinyur infrastruktur maupun insinyur DevOps merupakan bagian yang bertanggung jawab dalam kinerja aplikasi ataupun *platform* agar dapat diakses dengan mudah oleh pelanggan atau *end-user*. Dalam pekerjaannya, kedua peran ini juga saling bekerjasama dengan pengembang untuk memastikan jika aplikasi maupun *platform* dapat bekerja sesuai spesifikasi dan ekspetasi pengembang. Untuk memenuhi hal tersebut, makad dibutuhkan monitoring secara *real-time* yaitu Anbu yang menggunakan Prometheus dan juga Grafana. Alat ini akan dapat diakses melalui mekanisme port-forward, nodeport, maupun IP eksternal.

Dalam penggunaannya, Prometheus dapat membantu untuk mengetahui target dan *metrics* apa saja yang terdapat dalam *platform* Happy. Grafana dalam penggunaannya dapat diakses dengan melakukan *login* pada *username* dan *password* yang sudah didaftarkan oleh administrator. Setelah *login* berhasil, dapat dilakukan pengamatan dan observasi lebih lanjut untuk mengetahui kinerja pada bagian sistem *platform* Happy.

1.6 Kesimpulan dan Ringkasan CD-1

Dokumen CD 1 berisi usulan gagasan untuk menjawab permasalahan yang ada. Saat ini perkembangan industri *platform e-commerce* sudah semakin berkembang dengan pesat. Seiring berjalannya waktu, perkembangan fitur pada *platform e-commerce* semakin banyak dan kompleks. Dengan mengimplementasikan arsitektur *microservices* dapat meminimalisir kelemahan pada arsitektur *monolithic* pada saat pengembangan aplikasi kompleks seperti *platform e-commerce*. Selain itu, dengan meningkatnya konsumen dalam mengakses *platform e-commerce* yang sangat tinggi, maka kebutuhan akan ketersediaan sistem harus memadai. Oleh karena itu, diperlukan suatu aplikasi sistem monitoring bernama Anbu yang menggunakan teknologi bernama Prometheus dan Grafana untuk mengoptimasi monitoring terhadap *platform e-commerce* bernama Happy untuk menjaga *availability* serta *reliability* pada *platform e-commerce* tersebut.