Available at https://ejournal3.undip.ac.id/index.php/jtk
DOI:10.14710/jtk.v2i2.38347

Jurnal Teknik Komputer, Vol. 2, No. 2, 99-107

Pembuatan Infrastruktur Sistem Manajemen Keselamatan Pelayaran Berbasis Google Cloud dan Kubernetes di Balai Teknologi Keselamatan Pelayaran

Development of the Infrastructure of Sistem Manajemen Keselamatan Pelayaran Based on Google Cloud and Kubernetes at Balai Teknologi Keselamatan Pelayaran

Yosef Satrio Aji[1)], Rinta Kridalukmana[2)], R. Rizal Isnanto[3)]

Program Studi Teknik Komputer, Fakultas Teknik, Universitas Diponegoro
Jl. Prof. Soedarto, SH, Kampus Undip Tembalang, Semarang, Indonesia 50275

How to cite: Y. S. Aji, R. Kridalukmana, and R. R. Isnanto, " Pembuatan Infrastruktur Sistem Manajemen Keselamatan Pelayaran Berbasis Google Cloud dan Kubernetes di Balai Teknologi Keselamatan Pelayaran" *Jurnal Teknik Komputer*, vol. 2, no. 2, pp. 99-107, 2023. doi: 10.14710/jtk.v2i2.38347 [Online].

Abstract *– **This final project aims to assist the Balai Teknologi Keselamatan Pelayaran in enhancing the efficiency and effectiveness of its Sistem Manajemen Keselamatan Pelayaran by utilizing a Google Cloud and Kubernetes-based infrastructure. The infrastructure developed consists of several components such as servers, databases, and load balancers. Analysis of the system's requirements and design, utilizing cloud compitong and Kubernetes concepts as a platform for application management and resource allocation scalability. The infrastructure was then implemented gradually and tested to ensure the system's availability, security, and performance.***

The result of this final project is an online-accessible, distributed Sistem Manajemen Keselamatan Pelayaran infrastructure at Balai Teknologi Keselamatan Pelayaran. This infrastructure is capable of improving the speed and accuracy of data processing, reducing operational and maintenance costs, as well as enhancing the quality of service and shipping safety.

Keywords – *Infrastructure;* Sistem Manajemen Keselamatan Pelayaran*, Google Cloud; Kubernetes; cloud computing*

Abstrak – Tugas akhir ini bertujuan untuk membantu Balai Teknologi Keselamatan Pelayaran dalam meningkatkan efisiensi dan efektivitas Sistem Manajemen Keselamatan Pelayaran dengan memanfaatkan infrastruktur berbasis Google Cloud dan Kubernetes. Infrastruktur yang dibangun terdiri dari beberapa komponen seperti server, basis data, dan penyeimbang beban. Analisis kebutuhan dan desain sistem dengan menggunakan konsep komputasi awan dan Kubernetes sebagai platform manajemen aplikasi dan pengaturan sumber daya yang scalable. Kemudian dilakukan implementasi infrastruktur secara bertahap dan diuji coba untuk memastikan ketersediaan, keamanan, dan performa sistem.

Hasil dari tugas akhir ini adalah sebuah infrastruktur sistem manajemen keselamatan pelayaran yang dapat diakses secara online dan terdistribusi di Balai Teknologi Keselamatan Pelayaran. Infrastruktur ini mampu meningkatkan kecepatan dan ketepatan dalam pengolahan data, mengurangi biaya operasional dan pemeliharaan, serta meningkatkan kualitas pelayanan dan keamanan pelayaran.

Kata Kunci – *Infrastruktur; Sistem Manajemen Keselamatan Pelayaran; Google Cloud; Kubernetes; Komputasi awan*

I. PENDAHULUAN

Saat ini, penggunaan komputasi awan (*cloud computing*) dan teknologi kontainerisasi semakin populer karena memungkinkan pengembang aplikasi untuk menyediakan aplikasi secara cepat, aman, dan skalabel. Google Cloud adalah salah satu platform komputasi awan yang menyediakan berbagai layanan seperti penyimpanan data, jaringan, keamanan, dan sebagainya. Dengan menggunakan Google Cloud, pengembang aplikasi dapat menyediakan layanan aplikasi dengan mudah dan cepat, serta dapat menghemat biaya dalam pengelolaan infrastruktur server [1].

Kubernetes, di sisi lain, adalah platform sumber terbuka untuk otomatisasi *deployment*, *scaling*, dan manajemen aplikasi kontainer. Kubernetes memungkinkan pengembang aplikasi untuk menempatkan aplikasi mereka ke dalam kontainer dan

[*)] Penulis Korespondensi (Y. S. Aji)
Email: yosefsatrio26@students.undip.ac.id

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

mengelola aplikasi tersebut dengan lebih efisien dan efektif. Dengan Kubernetes, pengembang aplikasi dapat memastikan ketersediaan aplikasi yang optimal, memastikan waktu aktif (*uptime*) yang tinggi, dan menghindari waktu henti (*downtime*) yang berdampak negatif pada pengalaman pengguna dengan mengunakan metode menyebarkan server di beberapa pusat data (*data center*) [2].

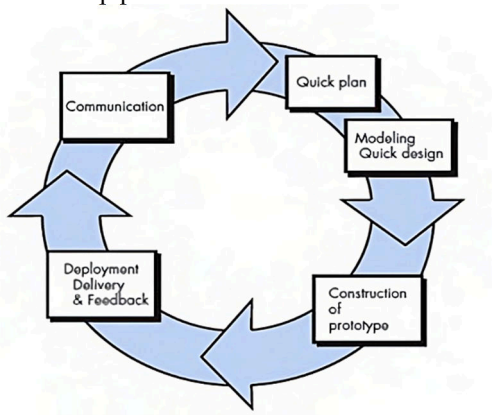
Balai Teknologi Keselamatan Pelayaran (BTKP) adalah sebuah lembaga yang bertanggung jawab melaksanakan penilaian, pengujian, rancang bangun, pembuatan alat-alat dan bahan-bahan keselamatan pelayaran serta penyiapan standarisasi dan sertifikat alat-alat dan bahan-bahan keselamatan pelayaran serta survei dan pemberitaan keselamatan pelayaran [3]. Dalam melakukan tugasnya BTKP memerlukan sistem informasi yang dapat membantu mempercepat proses pengajuan pengujian peralatan keselamatan pelayaran, serta pengarsipan dokumen yang diperlukan dalam pengujian sehingga dapat mempermudah audit. Sistem informasi yang ada harus dapat menyimpan dokumen dalam jumlah besar dan dapat menangani jumlah pengguna yang banyak sekaligus memperkecil waktu waktu henti.

Pemanfaatan Google Cloud dan Kubernetes memungkinkan sebuah sistem informasi memiliki waktu henti yang rendah dan mampu menangani jumlah pengguna yang tinggi. Dengan memanfaatkan *cloud* juga dapat membuat sistem informasi menyimpan data dalam jumlah yang besar dan dalam waktu yang sangat lama.

II. METODE PENELITIAN

Pada penelitian yang digunakan pada tugas akhir ini yaitu metode *Prototyping*. Berikut ini merupakan tahapan dalam menyelesaikan penelitian berdasarkan metode *Prototyping* yang dapat dilihat pada Gambar 1.

Metode *Prototyping* memiliki 5 tahapan pengembangan yaitu pengumpulan kebutuhan, perancangan cepat, membangun prototipe, evaluasi prototipe, dan perbaikan prototipe. Pengumpulan kebutuhan dilakukan analisis kebutuhan sistem yang didefinisikan secara rinci. Perancangan cepat adalah tahapan pembuatan sebuah desain yang sederhana yang dapat memberikan gambaran secara singkat tentang sistem yang akan dibuat atau dikembangkan. Selanjutnya, membangun prototipe adalah tahapan pembangunan prototipe yang sebenarnya akan mulai dikerjakan. Setelah membangun prototipe dilakukanlah evaluasi prototipe yang merupakan tahapan evaluasi dan penilaian dari prototipe yang telah dibangun. Tahapan terakhir adalah memperbaiki prototipe jika dari hasil evaluasi tidak memuaskan. Jika terdapat perbaikan maka proses pengembangan sistem akan dimulai lagi dari awal, pengembangan sistem akan terus berulang hingga pengembangan sistem selesai dan masuk ke tahap pemeliharaan.



Gambar 1. Metode *Prototyping*

A. Pengumpulan Kebutuhan

Kebutuhan fungsional merupakan kebutuhan yang diperlukan supaya sistem dapat berjalan dengan baik. Kebutuhan fungsional dibagi menjadi dua bagian utama, kebutuhan fungsional aplikasi dan kebutuhan fungsional infrastruktur.

1. Kebutuhan fungsional aplikasi

Kebutuhan fungsional aplikasi dibutuhkan supaya aplikasi dapat dijalankan di infrastruktur yang akan dibuat. Kebutuhan fungsional aplikasi dapat dilihat pada Tabel 1.

Tabel 1. Kebutuhan fungsional aplikasi

No	Kebutuhan Fungsional	Penjelasan
1	Aplikasi bersifat tanpa keadaan (*stateless*)	Aplikasi yang akan dijalankan pada infrastruktur bersifat tanpa keadaan.
2	Aplikasi dibangun menggunakan Docker	Aplikasi dibangun menggunakan Docker sehingga aplikasi dapat dijalankan tanpa tergantung pada suatu arsitektur tertentu.
3	Aplikasi menggunakan port yang dapat diakses publik	Aplikasi berjalan pada port tertentu, dimana port tersebut dapat diakses pengguna di luar infrastruktur yang ada.
4	Aplikasi menggunakan basis data MySQL	Aplikasi menggunakan basis data CloudSQL for MySQL.
5	Aplikasi menggunakan penyimpanan berbasis Cloud Storage	Penyimpanan disimpan dalam Cloud Storage untuk menyimpan file seperti dokumen, gambar, video, dll.

2. Kebutuhan fungsional infrastruktur

Kebutuhan fungsionalitas infrastruktur berfungsi untuk menunjang aplikasi dan server yang memenuhi

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

nilai ketersediaan, skalabilitas, dan reliabilitas yang tinggi. Kebutuhan fungsional infrastruktur dapat dilihat pada Tabel 2.

Tabel 2. Kebutuhan fungsional infrastruktur

No	Kebutuhan Fungsional	Penjelasan
1	Aplikasi dapat melakukan pembaruan otomatis	Ketika aplikasi memerlukan pembaruan ke versi terbaru, server dapat memperbarui aplikasi secara otomatis tanpa adanya waktu henti
2	Server memiliki fitur perbaikan otomatis	Jika terjadi kerusakan baik terjadi pada perangkat lunak maupun pada perangkat keras ` dapat melakukan perbaikan dengan sendirinya
3	Basis data memiliki fitur pencadangan otomatis	Basis data dapat melakukan pencadangan otomatis dan dapat dipulihkan ketika dubutuhkan.
4	Basis data dan Cloud Storage memiliki fitur penambahan kapasitas otomatis	Basis data dan Cloud Storage dapat menambahkan kapasistas, sehingga tidak akan pernah kehabisan kapasitas penyimpanan
5	Infrastruktur tersedia di beberapa zona	Infrastruktur tersedia di beberapa zona untuk menghindari aplikasi tidak dapat diakses jika terjadi bencana

B. Perancangan Arsitektur Sistem

Arsitektur sistem yang digunakan pada sistem manajemen keselamatan pelayaran menggunakan infrastruktur tiga tingkat yang akan diimplementasikan di Google Cloud Platform dapat dilihat pada Gambar 2.

Pada arsitektur sistem tiga tingkat terdapat tiga lapisan yang memisahkan antara perangkat pengguna, aplikasi, dan basis data. Tiga lapisan tersebut yaitu:

1. Lapisan klien

Lapisan klien adalah perangkat yang digunakan oleh pengguna, misalnya komputer atau HP.

2. Lapisan aplikasi

Lapisan aplikasi adalah lapisan yang berisi semua logika bisnis seperti validasi data, perhitungan, penyisipan data, enskripsi, dan lainnya yang bertindak sebagai perantara antara lapisan klien dengan lapisan data.

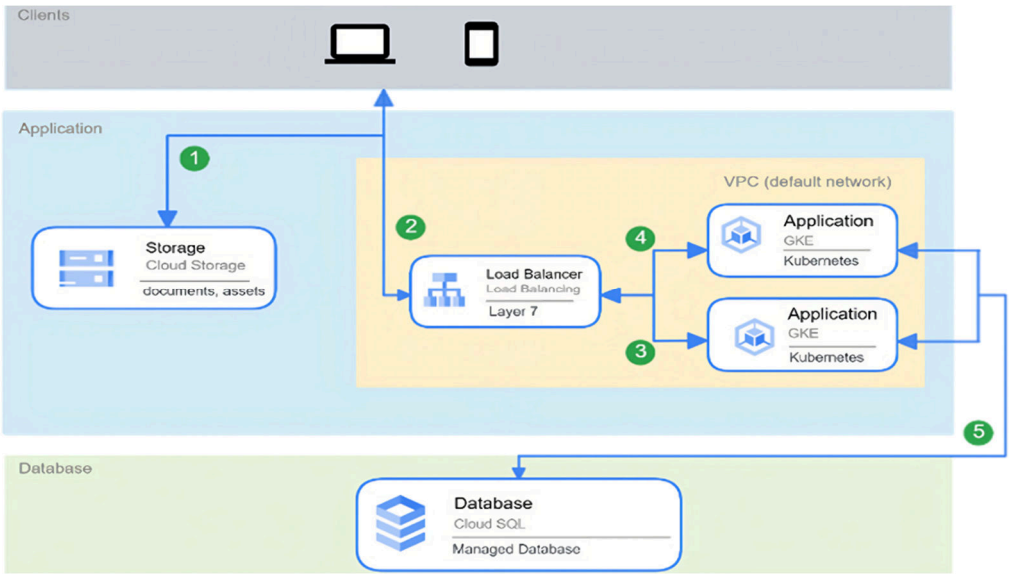
3. Lapisan data

Lapisan data merupakan lapisan yang berisi basis data. Lapisan ini berisi metode untuk terhubung dengan basis data dan memasukkan, mengubah, menghapus, mendapatkan data dari basis data berdasarkan data yang dikirimkan dari lapisan aplikasi. Media transmisi antar komponen pada infrastruktur dijelaskan dalam bentuk tabel yang dapat dilihat pada Tabel 3.

Tabel 3. Media transmisi

Kode	Komponen Pengirim	Komponen Penerima	Media Transmisi	Metode Transmisi Data
1	Cloud Storage	Klien	Internet	HTTPS
2	Load Balancer	Klien	Internet	HTTPS
3	GKE	Load Balancer	Private Network	HTTP
4	GKE	Load Balancer	Private Network	HTTP
5	Database	GKE	Private Network	Kueri SQL

Metode transmisi data data HTTPS digunakan ketika media transmisi adalah internet untuk menjaga keaman data dengan lebih dahulu melakukakan enkripsi yang dilengkapi dengan sertifikat SSL. Sedangkan jika media transmisi adalah koneksi privat (*private network*), metode transmisi data yang digunakan adalah HTTP saja tanpa enkripsi yang akan mempercepat komunikasi antar server karena tidak adanya enkripsi terlebih dahulu.



Gambar 2. Arsitektur sistem

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

C. Perancangan Prototipe

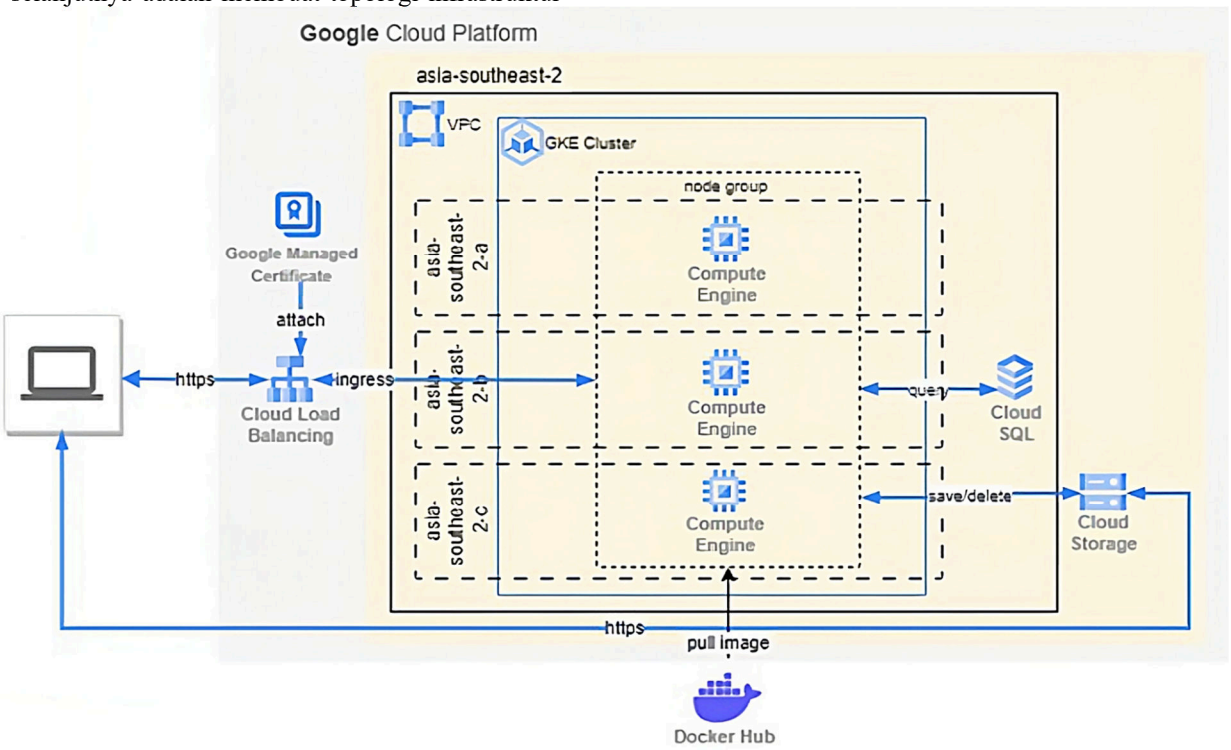
Pada tahapan perancangan prototipe langkah-langkah yang dilakukan yaitu:

1. Pemilihan Layanan Google Cloud, layanan Google Cloud yang dipilih untuk memenuhi kebutuhan sistem yang telah dianalisa pada tahapan sebelumnya dapat dilihat pada **Tabel 4**.

Tabel 4. Layanan GCP yang digunakan

No	Nama layanan	Spesifikasi yang memenuhi kebutuhan sistem
1	Google Kubernetes Engine	1. Dapat menggunakan *deployment* menggunakan Docker Image 2. Memiliki *autoscaling* jika memiliki pengguna yang banyak 3. Memiliki *autohealing* apabila aplikasi/ *node* mengalami kerusakan 4. Dapat berjalan di beberapa zona untuk menjaga ketersediaan aplikasi
2	Google Cloud SQL	1. Mendukung semua basis data bertipe SQL 2. Memiliki fitur pencadangan otomatis
3	Google Cloud Storage	1. Untuk memenuhi penyimpanan dari aplikasi 2. Tidak ada Batasan pemakaian kapasitas 3. Layanan memiliki SLA ketersediaan 99.5% 4. Penyimpanan dapat berjalan di beberapa zona

2. Perancangan Topologi Infrastruktur, Setelah menentukan layanan yang akan digunakan tahapan selanjutnya adalah membuat topologi infrastruktur berdasarkan arsitektur sistem yang telah dibuat sebelumnya. Pada perancangan topologi ini, setiap layanan akan disusun menjadi sebuah infrastruktur server yang akan dibuat. Topologi infrastruktur yang akan dibuat dapat dilihat pada Gambar 3.
3. Perancangan *High Availability* Server, metode untuk membuat server selalu dapat menangani permintaan dari pengguna walaupun terdapat kerusakan. Pada Google Kubernetes Engine, *autohealing* bekerja pada *pod* dan juga pada *node.* Metode yang digunakan dalam penelitian ini, yaitu:
   a. *Autohealing* adalah metode yang akan melakukan perbaikan secara otomatis apabila terjadi kerusakan atau sistem tidak merespon. Ketika Compute Engine tidak merespon Health Check maka Kubernetes Engine akan membuat Compute Engine baru secara otomatis dan mematikan yang sebelumnya.
   b. Penyeimbang beban adalah metode untuk mendistribusikan beban antar sumber daya sehingga tidak hanya satu sistem saja yang terbebani. Tujuannya adalah membagi beban permintaan pengguna sehingga sistem menjadi lebih cepat. Penyeimbang beban juga memiliki *health check* yang berfungsi supaya permintaan pengguna tidak dialihkan kepada sistem yang tidak merespon.



Gambar 3. Topologi infrastruktur SIMAKESPEL

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

c. *Multiple zones* merupakan metode penggunaan server di beberapa *data center* atau zona. Metode ini berfungsi untuk menjaga sistem tetap dapat berjalan jika terdapat bencana di salah zona seperti kebakaran, sehingga aplikasi masih dapat digunakan dengan lancar. Metode *multiple zones* juga dapat digabungkan dengan penyeimbang beban.
d. *Autoscaling* berfungsi untuk menambahkan atau mengurangi sumber daya jika sistem mengalami beban yang lebih tinggi atau lebih kecil dari batasan yang telah ditentukan sebelumnya.
e. *Rolling update* merupakan salah satu metode dari tiga metode ketika ingin melakukan pembaruan aplikasi yang sedang berjalan tanpa adanya waktu henti.

D. Implementasi Prototipe

Tahapan implementasi prototipe adalah tahapan untuk membuat infrastruktur berdasarkan perancangan yang telah dilakukan sebelumnya. Tahapan implementasi sistem ini akan menggunakan Terraform sebagai alat *Infrastructure as Code* (IaC) yang akan membantu memperkecil kesalahan ketika pembuatan infrastruktur. IaC adalah metode mendefinisikan infrastruktur sebagai kode terstruktur sehingga pembuatan infrastruktur akan menjadi lebih efesien. Dalam tahapan ini Google Kubernetes Engine, Cloud SQL, Cloud Storage, *firewall*, VPC, dan *subnet* akan didefinisikan sebagai kode.

Pada tahapan implementasi prototipe juga dilakukan beberapa perubahan terhadap aplikasi supaya menjadi tanpa keadaan. Perubahan yang dilakukan meliputi:

1. Memindahkan sesi (*session*) dari penyimpanan lokal server ke *cookie*, pemindahan sesi ini sangatlah penting supaya pengguna tidak perlu melakukan otentikasi lagi ketika penyeimbang beban mengalihkan proses ke *pod* yang berbeda.
2. Mengubah penyimpanan aplikasi ke Google Cloud Storage. Penyimpanan yang dimaksud adalah konten yang terdapat di aplikasi, baik konten statik seperti gambar, video, dan skrip, maupun konten yang diunggah oleh pengguna seperti dokumen yang diperlukan.
3. Menggunakan basis data eksternal, basis data yang sebelumnya terdapat di *localhost* dipindahkan ke Cloud SQL supaya setiap *pod* memiliki isi dari basis data yang sama.

E. Evaluasi Prototipe

Setelah tahapan implementasi prototipe, selanjutnya adalah proses evaluasi prototipe. Pada tahap ini dilakukan tes terhadap aplikasi apakah aplikasi dapat berjalan dengan baik dan apabila tidak berjalan dengan baik akan dilanjutkan ke tahap selanjutnya yaitu perbaikan prototipe.

Pada tahapan ini juga dilakukan tes terhadap *autohealing* dan *rolling deployment* dapat berjalan. Tes *autohealing* dilakukan dengan cara menghapus *pod* yang sedang berjalan, apabila ketika *health check* gagal melakukan tugasnya maka Kubernetes akan secara otomatis membuat *pod* baru. Selanjutnya, tes *rolling deployment* dilakukan dengan cara merilis pembaruan aplikasi. Ketika *rolling update* berjalan maka ketika proses pembaruan Kubernetes akan membuat *pod* baru.

F. Perbaikan Prototipe

Perbaikan prototipe adalah tahapan apabila prototipe yang sebelumnya telah dievaluasi mengalami kegagalan. Pada tahapan ini juga akan ditentukan apakah prototipe hanya diperlukan perbaikan saja atau diperlukan pengulangan langkah pembuatan yang dimulai dari awal yaitu pengumpulan kebutuhan.

III. HASIL DAN PEMBAHASAN

A. Hasil Penelitian

1. Google Kubernetes Engine

Google Kubernetes Engine merupakan layanan Kubernetes dari Google Cloud Platform akan menjadi tempat aplikasi dijalankan. Aplikasi yang dijalankan di dalam Kubernetes harus dikonteinerisasi. Google Kubernetes Engine memiliki *dashboard* yang mempermudah dalam memonitor Kubernetes, seperti detail dari Kubernetes seperti nama, lokasi, versi, dan alamat IP dari Kubernetes Cluster.

Nodes merupakan VM Instance yang berjalan di dalam Kubernetes Cluster. Kubernetes Cluster simakespel menggunakan tiga *nodes* yang terdapat pada *zone* yang berbeda-beda.

2. Penyeimbang Beban

Penyeimbang beban dapat meratakan beban permintaan dari pengguna ke beberapa server. Supaya dapat melakukan tugasnya, penyeimbang beban memerlukan *health check* untuk memeriksa apakah server sedang merespon atau tidak, jika server tidak merespon maka akan dialihkan. Supaya pengguna dapat dialihkan ke server lain maka aplikasi harus dibuat *stateless*.

Frontend dapat melakukan pengalihan sesuai dengan *paths* tertentu, pada sistem saat ini semua *paths* akan dialihkan ke satu *backend* saja yaitu Kubernetes Cluster yang digunakan pada SIMAKESPEL. *Backend* adalah tempat tujuan dari *frontend*.

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

3. Cloud SQL

Cloud SQL merupakan basis data SQL yang dikelola oleh GCP, dimana pencadangan, penambahan kapasitas penyimpanan, pembaruan sistem, dan *autohealing* akan dikelola oleh GCP. Cloud SQL mendukung basis data MySQL, PostgreSQL, dan SQL Server yang merupakan basis data yang paling sering digunakan oleh perusahaan.

Konfigurasi yang digunakan menggunakan MySQL versi 5.7 sedangkan untuk koneksinya menggunakan koneksi privat yang hanya dapat diakses oleh VPC yang sama, sedangkan IP yang diperbolehkan diset menjadi semua dikarenakan IP yang digunakan oleh *nodes* pada Kubernetes Cluster akan dibuat oleh GCP. Sedangkan nama dari Cloud SQL akan memiliki angka acak. Sedangkan tempat dan spesifikasi yang akan digunakan oleh Cloud SQL akan ditentukan oleh variabel yang akan ditentukan ketika kode Terraform dijalankan. Pencadangan akan dijalankan setiap jam 00:00 setiap harinya, dikarenakan pencadangan akan membebani sistem. Oleh karena itu, pencadangan sebaiknya dilakukan ketika aplikasi sedang tidak digunakan oleh banyak pengguna.

4. Google Cloud Storage

Google Cloud Storage akan digunakan sebagai penyimpanan SIMAKESPEL, baik konten statis seperti file CSS, JS, gambar, dan video maupun konten dinamis yang diunggah oleh pengguna seperti dokumen yang diperlukan dalam penerbitan sertifikasi dan dibuat oleh aplikasi seperti sertifikat. Sistem akan memiliki dua *bucket* yang digunakan untuk memisahkan konten statis dan konten dinamis supaya menambahkan keamanan ekstra pada file.

5. Firewall

Firewall digunakan untuk mengatur lalu lintas data menggunakan *port-port* tertentu. Pada infrastruktur yang dibuat dalam penelitian ini terdapat dua *firewall* yang digunakan, yaitu *firewall* internal dan *firewall* eksternal.

Pada konfigurasi *firewall* internal, hanya rentang IP tertentu saja yang diperbolehkan. *Firewall* internal hanya digunakan untuk komunikasi antar komponen infrastruktur menggunakan koneksi *private*.

B. Pengujian Sistem

Pengujian ini dilakukan untuk mengetahui apakah sistem yang dibuat berjalan sesuai dengan analisis kebutuhan dan perancangan saat pembuatan.

1. Pengujian fungsional
   a. Pengujian *autohealing*, pengujian ini bertujuan untuk mendapatkan berapa rata-rata waktu yang dibutuhkan saat *autohealing* berjalan. Pengujian dilakukan dengan dua skenario, kegagalan *pod* dan kegagalan *node*. Skenario dijalankan sebanyak lima kali. Waktu yang dibutuhkan dari pengujian skenario kegagalan *node* seperti dapat dilihat pada Tabel 5.

Tabel 5. Pengujian kegagalan *node*

No	Waktu *autohealing* (s)
1	255
2	253
3	302
4	255
5	263
Rata-rata	265,6

Skenario dijalankan sebanyak lima kali. Waktu yang dibutuhkan dari pengujian skenario kegagalan *pod*, dapat dilihat pada Tabel 6.

Tabel 6. Pengujian kegagalan *pod*

No	Waktu *autohealing* (s)
1	3
2	2
3	3
4	5
5	3
Rata-rata	3,2

   b. Pengujian *autoscaling*, pengujian ini bertujuan untuk mendapatkan berapa rata-rata waktu yang dibutuhkan saat *autoscaling* berjalan. Proses *autoscaling* dibagi menjadi dua yaitu *scale up* dan *scale down*. *Scale up* berarti menambahkan sumber daya tambahan kepada sistem, sedangkan *scale down* berarti mengurangi sumber daya tambahan kepada sistem. Waktu *scale up* dapat dilihat pada Tabel 7.

Tabel 7. Waktu *scale up*

No	Waktu *scale up* (s)
1	60
2	35
3	60
4	30
5	45
Rata-rata	46

Untuk pengujian *scale down* memiliki waktu yang sama yaitu 120 detik dikarenakan pada konfigurasi telah diatur untuk menurunkan sumber daya setelah 120 detik. Waktu *scale down* dapat dilihat pada Tabel 8.

Tabel 8. Waktu *scale down*

No	Waktu *scale down* (s)
1	120
2	120
3	120
4	120
5	120
Rata-rata	120

   c. Pengujian *rolling update*, pengujian ini bertujuan untuk mendapatkan berapa rata-rata waktu yang dibutuhkan saat *rolling update* berjalan. Proses *rolling update* dilakukan ketika aplikasi

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

mengalami pembaruan. Skenario yang dijalankan untuk pengujian *rolling update* adalah dengan melakukan pembaruan pada aplikasi. Pembaruan dilakukan dengan cara mengganti *pod* yang sedang berjalan dengan *pod* baru yang berisi dengan aplikasi yang baru.

2. Pengujian beban

Pengujian beban adalah pengujian yang dilakukan untuk memastikan suatu sistem dapat berjalan dengan baik dan dengan waktu respon sesuai yang diinginkan. Pada pengujian aplikasi SIMAKESPEL digunakan perangkat lunak k6.io yang merupakan pengujian beban berbasis Javascript. Pengujian beban akan dilakukan terhadap beberapa fitur pada aplikasi SIMAKESPEL dengan jumlah pengguna virtual yaitu 500 pengguna. Fitur yang diujikan dan metodenya dapat dilihat pada Tabel 9.

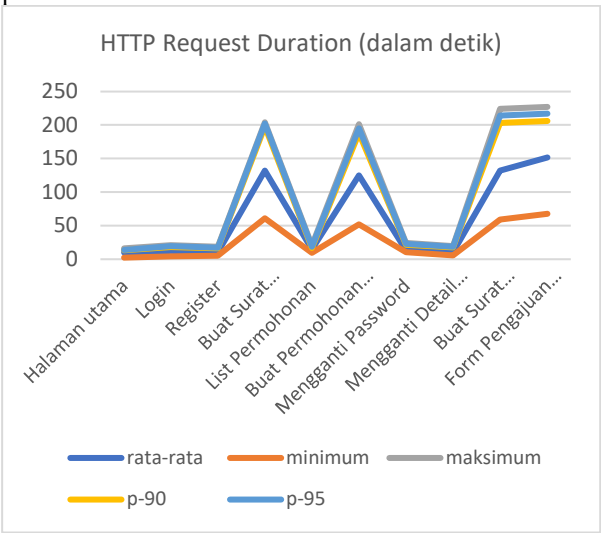
Tabel 9. List fitur yang akan diujikan

Fitur	Metode
Halaman utama	GET
Login	POST
Register	POST
Buat Surat Persetujuan Kewenangan Baru	POST
List Permohonan	GET

Tabel 9. List fitur yang akan diujikan

Fitur	Metode
Buat Permohonan Pemeriksaan Tahunan	POST
Mengganti Password	POST
Mengganti Detail Profil	POST
Buat Surat Persetujuan Kewenangan Perpanjangan	POST
Form Pengajuan Permohonan Sertifikasi Pengujian Pertama	POST

Berdasarkan pengujian beban yang dilakukan terhadap aplikasi SIMAKESPEL dengan jumlah pengguna *virtual* sebanyak 500, didapatkan data seperti yang digambarkan dalam bentuk grafik ditunjukkan pada Gambar 4.

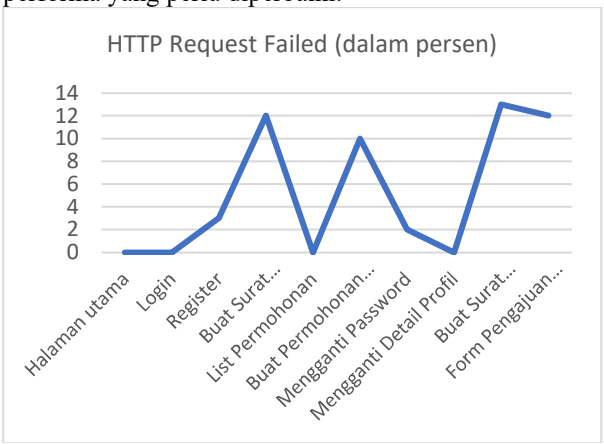


Gambar 4. Hasil pengujian beban HTTP *Request Duration*

Pada Gambar 4 menunjukkan waktu respon untuk metode GET yang rendah dengan rata-rata **Halaman Utama** terbuka secara sempurna dalam waktu 10.11 detik dan **Halaman List Pemohon** rata-rata 14.52 detik. Beberapa metode POST juga masih sangat rendah seperti *login* dengan rata-rata 12.1 detik, **Register** 10.21 detik, **Mengganti Password** 13.21 detik dan **Mengganti Detail Profil** 10.22 detik. Tetapi metode POST yang memiliki unggahan berupa dokumen yang harus diunggah memiliki waktu respons yang sangat lambat, yaitu **Buat Surat Persetujuan Kewenangan** selama dengan rata-rata 132 detik, **Buat Permohonan Pemeriksaan Tahunan** 125 detik, **Buat Surat Persetujuan Perpanjangan** 132 detik, dan **Formulir Pengajuan Permohonan Sertifikasi Pengujiian Pertama** 152 detik. Waktu respon yang lambat ini dikarenakan terdapat banyak dokumen yang wajib diunggah ketika mengisi formulir.

Nilai persentil ke-90 menunjukkan bahwa 90% dari pengguna memiliki waktu yang lebih rendah dari nilai persentil ke-90. Sedangkan nilai persentil ke-95 menunjukkan bahwa 95% dari pengguna memiliki waktu lebih rendah daripada nilai persentil ke-95.

Pada Gambar 4 menunjukkan bahwa nilai maksimal, persentil ke-90, dan persentil ke-95 memiliki nilai yang berdekatan. Jika nilai persentil ke-90 atau ke-95 mendekati nilai maksimum, artinya sebagian besar kinerja sistem dalam pengujian tersebut berada di bawah nilai tersebut. Hal ini menunjukkan bahwa sebagian besar waktu respons dari sistem dalam pengujian tersebut cukup cepat dan hanya sebagian kecil pengguna yang mengalami waktu respons yang lambat. Dalam pengujian beban k6, persentil ke-90 atau ke-95 yang mendekati nilai maksimum dapat digunakan sebagai salah satu indikator untuk mengevaluasi performa sistem dalam kondisi beban yang tinggi. Nilai persentil yang tinggi menunjukkan performa sistem yang baik, sedangkan nilai persentil yang rendah dapat menunjukkan adanya masalah performa yang perlu diperbaiki.



Gambar 5. Hasil pengujian beban HTTP *Request Failed*

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023

Pada Gambar 5 menunjukkan ketika metode GET tidak terjadi kegagalan sama sekali, tetapi ketika metode POST terdapat kegagalan yang sangat tinggi bahkan hingga 13% pada fitur **Buat Surat Persetujuan Kewenangan Perpanjangan**. Kegagalan ini disebabkan oleh server yang tidak mampu menangani permintaan yang sangat.

C. Pembahasan

Infrastruktur Sistem Manajemen Keselamatan Pelayaran yang dibuat menggunakan Kubernetes dan Google Cloud ini merupakan sebuah sistem yang mengutamakan ketersediaan dan durabilitas dengan cara mengimplementasikan *autohealing*, *autoscaling*, dan *rolling update*. Sehingga dapat mengoptimalkan kinerja Balai Teknologi Keselamatan Pelayaran dalam melaksanakan fungsinya karena sistem memiliki waktu henti yang rendah dan minim pemiliharaan karena perangkat keras akan dikelola oleh Google Cloud.

Infrastruktur ini terdiri atas Google Kubernetes Engine sebagai Kubernetes yang dikelola oleh Google Cloud, Cloud SQL sebagai basis data dari aplikasi, Cloud Storage sebagai penyimpanan file dari aplikasi serta *firewall*, *subnet*, serta VPC yang berfungsi sebagai konektivitas baik antar server maupun berkomunikasi dengan perangkat dari pengguna. Terdapat juga penyeimbang beban dan *health check* yang berfungsi mendistribuskikan permintaan pengguna serta mengecek kesehatan dari server.

Setelah pembuatan infrastruktur selesai dilakukan pengujian fungsional dan pengujian beban. Pengujian dilakukan untuk memerikasa fungsional sistem dan memastikan sistem dapat berjalan dengan jumlah pengguna yang ditargetkan. Berdasarkan pengujian yang telah dilakukan, sistem dapat berjalan dengan baik sesuai perancangan sistem yang telah dilakukan sebelumnya.

IV. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Pembuatan dan pengembangan infrastruktur menggunakan Terraform dapat mempermudah pengembang karena tidak perlu membuat infrastruktur secara manual yang memungkinkan kesalahan dari manusia.
2. Metode *Prototyping* berhasil diimplementasikan dalam penelitian ini sebagai metode pembuatan infrastruktur.
3. Penggunaan Kubernetes dan penyeimbang beban dapat menurunkan waktu henti sistem yang disebabkan oleh kegagalan sistem baik dari perangkat keras maupun dari perangkat lunak.
4. Penggunaan Cloud Storage sebagai tempat penyimpanan aplikasi dapat memenuhi kebutuhan dari SIMAKESPEL dikarenakan tidak memiliki batas penyimpanan.
5. Berdasarkan pengujian fungsional dan pengujian beban, semua fungsi berjalan dengan semestinya sesuai dengan analisis yang telah dilakukan.

DAFTAR PUSTAKA

[1] N. Ramsari and A. Ginanjar, "Implementasi Infrastruktur Server Berbasis Cloud Computing untuk Web Service Berbasis Teknologi Google Cloud Platform," *Prosiding Seminar Nasional Teknologi Informasi dan Kedirgantaraan,* vol. VII, 2022.

[2] S. R. Dira and M. A. F. Ridha, "Monitoring Kubernetes Cluster Menggunakan Prometheus dan Grafana," *ABEC Indonesia,* pp. 345-351, 2023.

[3] Balai Teknologi Keselamatan Pelayaran, "Tugas dan Fungsi," Balai Teknologi Keselamatan Pelayaran, [Online]. Available: https://hubla.dephub.go.id/btkp/page/tugas-dan-fungsi. [Accessed 5 Maret 2023].

[4] M. A. Nugroho and C. Subiyantoro, "Analisis Cluster Container Pada Kubernetes Dengan Infrastruktur Google Cloud Platform," *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika),* vol. 3, pp. 84-93, 2018.

[5] Y. T. Sumbogo, M. Data and R. A. Siregar, "Implementasi Failover Dan Autoscaling Kontainer Web Server Nginx Pada Docker Menggunakan Kubernetes," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* vol. 2, pp. 6849-6854, 2018.

[6] Budiyanto and Alex, Pengantar Cloud Computing, Komunitas Cloud Computing Indonesia, 2012.

Submitted: 29 March 2023; Accepted 1 April 2023: ;Published: 1 May 2023