PEMANFAATAN GOOGLE CLOUD DAN TEKNIK LOAD BALANCING UNTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB

Disusun Oleh:

N a m a : Ivan Firmansyah

NIM : 16523036

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2021

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PEMANFAATAN GOOGLE CLOUD DAN TEKNIK LOAD BALANCING UNTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB

TUGAS AKHIR

Disusun Oleh:

N a m a : Ivan Firmansyah

NIM : 16523036

Yogyakarta, 17 Desember 2020

Pembimbing,

(Fietyata Yudha, S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

PEMANFAATAN GOOGLE CLOUD DAN TEKNIK LOAD BALANCING UNTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 18 Januari 2021

Tim Penguji

Fietyata Yudha, S.Kom., M.Kom. _____

Anggota 1

Ari Sujarwo, S.Kom., M.I.T. _____

Anggota 2

Fayruz Rahma, S.T., M.Eng. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T.,

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Ivan Firmansyah

NIM : 16523036

Tugas akhir dengan judul:

PEMANFAATAN GOOGLE CLOUD DAN TEKNIK LOAD BALANCING UNTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 18 Januari 2021

(Ivan Firmansyah)

HALAMAN PERSEMBAHAN

Alhamdulillahi rabbil ‘alamin, atas berkat rahmat Allah SWT dan ridho-Nya dapat menyelesaikan tugas akhir yang sekaligus merupakan tanggung jawab dalam proses perkuliahan ini. Tugas akhir ini saya persembahkan kepada.

1. Bapak Satiawan dan Ibu Sri Liestiyani yang senantiasa telah mendidik dan memberikan motivasi kepada saya sehingga dapat menyesaikan perkuliahan ini. Saya tahu apa yang saya berikan selama ini belum cukup untuk membalas segala usaha dan kebaikan untuk saya. Tugas akhir ini saya persembahkan untuk Bapak dan Ibu sebagai wujud cinta dan terima kasih saya kepada kalian.
2. Keluarga saya yang selalu memberikan dukungan dan motivasi untuk menyelesaikan proses perkuliahan
3. Teman-teman dari cumengkling Hayyu, Bama, Lantang dan Erpe yang selalu memberikan dukungan dan motivasi terutama lantang yang membantu dalam berbagai aspek
4. Bapak Fietyata Yudha, S. Kom., M. Kom selaku dosen pembimbing, terima kasih untuk semua nasihat, bimbingan, ilmu, dan semangat yang luar biasa yang selama ini telah diberikan kepada saya.
5. Teman-teman saya, terima kasih untuk dukungan yang luar biasa dan selalu percaya bahwa saya bisa menyelesaikan tugas akhir sesuai dengan target saya

HALAMAN MOTO

Bangkit melawan atau mati tertindas

(Unknown)

Karena sesungguhnya sesudah kesulitan itu ada kemudahan

(Al Qur'an Surat Al-Insyirah Ayat 5-6)

Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya.

(QS. Al-Baqarah: 286)

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh

Alhamdulillah, segala puji dan syukur saya panjatkan atas ke hadirat Allah SWT, yang telah telah memberikan ridho dan rahmat-Nya. Penulis dapat menyelesaikan laporan tugas akhir ini yang berjudul "PEMANFAATAN GOOGLE *CLOUD* DAN DAN TEKNIK *LOAD BALANCING* UKTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB" yang merupakan bagian dari ibadah saya kepada Allah SWT. Laporan ini disusun untuk memenuhi tugas akhir sebagai syarat untuk menyelesaikan pendidikan pada jenjang Strata 1 (S1) pada Jurusan Informatika Universitas Islam Indonesia. Penulis sadar laporan ini tidak akan dapat selesai dengan waktu yang cepat tanpa dukungan serta motivasi dari beberapa pihak. Oleh karena itu penulis tidak lupa menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. ALLAH SWT, atas limpahan rahmat dan hidayah-Nya yang selalu ada di setiap langkah dalam memberikan kekuatan, kemampuan, dan menjaga semangat saya untuk dapat menyelesaikan Tugas Akhir ini dengan lancar.
2. Kedua orang tua. Bapak Satiawan dan Ibu Sri Liestiyani yang selalu memberikan cinta kasih serta motivasi agar saya dapat menyesaikan studi
3. Tante yanti, Tante Irma, Om Edwin, Grandys dan Farhan yang selalu memberi semangat pada saat saya mulai Lelah.
4. Bapak Prof. Dr. Fathul Wahid, S.T., M.Sc., Ph.D., selaku Rektor Universitas Islam Indonesia.
5. Bapak Prof. Dr. Ir. Hari Purnomo, M.T., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
6. Bapak Hendrik, S.T,M.Eng, Selaku Ketua Jurusan Informatika Universitas Islam Indonesia
7. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., Selaku Ketua Program Studi Informatika Program Sarjana Fakultas Teknologi Industri Universitas Islam Indonesia
8. Bapak Fietyata Yudha, S.Kom., M.Kom selaku dosen pembimbing
9. Bapak dan Ibu dosen Jurusan Informatika, yang telah memberikan ilmu yang bermanfaat kepada penulis, semoga bapak dan ibu selalu diberikan kesehatan serta lindungan dari Allah SWT
10. Teman-teman saya terutama untuk Cumengkling (Nur Jati Lantang, Dzikra Rafik Putra, Elbama Giofandra, dan Hayyu Ilham) terutama lantang yang telah membantu saya dalam berbagai aspek thanks tanggg

11. Teman-teman dari north java tech Affan, Dyaz, Dimas, Bagas.
12. Mbak etik, Avin, Assyifa, Dinda, Nazla yang selalu mendengarkan keluhan penulis selama mengerjakan tugas akhir
13. Bu ning e-sport Darmon, Marler, Rizaldi dkk terima kasih atas waktunya selama ini.
14. Tim hall fmipa Zaky, Raka, Ade yang selalu bersemangat pada saat mengerjakan tugas akhir
15. Teman-teman Hexadecima angkatan 2016, dan kakak angkatan yang tidak bisa saya sebutkan satu per satu.
16. Semua pihak yang tidak bisa saya sebutkan satu per satu, terima kasih atas semua bentuk dukungannya

Yogyakarta, 18 Januari 2021

(Ivan Firmansyah)

SARI

Meningkatnya kebutuhan *user* terhadap internet menyebabkan penyedia saling berlomba-lomba untuk meningkatkan layanan agar user dapat memperoleh informasi dengan cepat dan akurat. Maka dari itu web *server* dengan performa yang tinggi sangat dibutuhkan untuk dapat mencukupi kebutuhan *user*.

Dari permasalahan yang telah dijelaskan maka penulis memiliki gagasan untuk membuat sebuah konfigurasi pada *server* menggunakan *load balancing*. *Load Balancing* adalah sebuah teknik yang digunakan untuk menyeimbangkan beban *server* pada saat mendapatkan trafik yang tinggi sehingga *serve*r tidak mengalami *down*. Dengan dibuatnya konfigurasi pendukung untuk membantu kinerja *server*, diharapkan masalah yang sering terjadi dalam memproses permintaan hingga penyajian informasi kepada *client* dapat diperkecil.

Penelitian ini dimulai dengan menentukan kebutuhan perangkat lunak, merancang arsitektur jaringan, membuat web *server* yang akan di *load balancing*, membuat konfigurasi *load balancing*, menentukan variabel pengujian, pengambilan data dan analisis data yang telah didapatkan dari pengujian.

Setelah dilakukan pengujian terhadap *server* dengan menggunakan *apache benchmark* sebanyak 15 kali didapatkan hasil bahwa *server* mampu melakukan *load balancing* ketika mendapatkan trafik yang tinggi. Hal tersebut tentunya sangat bermanfaat untuk mengurangi terjadinya *down* pada sistem.

Kata Kunci: *Load Balancing*, Trafik, *Server*, Web

GLOSARIUM

Apache Benchmark	Sebuah alat yang digunakan untuk mengukur performa pada sever pada saat melayani *request* dari komputer *client*
Cache	Proses yang digunakan oleh *browser* dan aplikasi untuk menyimpan informasi
Cloud Storage	Penyimpanan *virtual* yang dapat menjangkau banyak perangkat penyimpanan fisik di lokasi yang berbeda
Cloud DNS	Sebuah sistem yang mengubah *IP Address* ke dalam bentuk URL web
Group Instance	Kumpulan *instance virtual machine* yang dapat mengelola beberapa mesin *virtual* sebagai entitas tunggal.
Image *Virtual* Machine	Sebuah *virtual machine* yang digunakan untuk menyimpan konfigurasi yang telah dibuat agar tidak perlu melakukan konfigurasi ulang ketika akan membuat *virtual machine* baru
Load Balancing	Sebuah metode yang digunakan untuk menyeimbangkan beban pada trafik agar tidak terjadi *overload*.
Trafik	Banyaknya *visitor* pada suatu web
Template Instance	Sebuah cara untuk menyimpan konfigurasi *instance* VM sehingga dapat menggunakannya nanti untuk membuat VM atau *group* VM
Server	Layanan tertentu yang disediakan oleh sistem komputer.
SQL-*Server*	Sistem manajemen *database* relasional (RDBMS) yang dirancang untuk aplikasi dengan arsitektur *client*/*server*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penggunaan teknologi informasi dan komunikasi saat ini tidak akan lengkap tanpa menggunakan web *server* dengan performa yang tinggi. Untuk meningkatkan angka penggunaan laman web, perlu adanya kemampuan untuk mengakses konten web yang akurat, lebih cepat, berkelanjutan dan menarik. Web *server* adalah *server* yang bertanggung jawab untuk menerima permintaan HTTP dari *client* web dan menyediakan informasi berupa laman web yang berisi konten statis seperti teks, gambar, dan lain sebagainya dan dinamis yang berupa konten naskah. Oleh karena itu, web *server* menerima permintaan *client*, memprosesnya, dan mengirimkan tanggapan. Selain itu, efisiensi dan efektivitas *server* web menentukan daya tarik dengan pada organisasi dan pengembang web (Jader et al., 2019). Hal itu karena semakin cepat dan tepat web *server* dalam menanggapi permintaan *client* akan membuat *client* lebih percaya dan akan menggunakan layanan web *server* tersebut. Di sisi lain, kinerja *server* web dapat dipengaruhi oleh beban yang mengarah kepada web *server*. Maka dari itu, untuk menghindari terjadinya kelebihan beban pada web *server* dilakukan pengendalian *file*. Oleh karena itu, mendesain dan mengusulkan sistem yang efisien yang dapat menangani beban besar penting bagi peneliti. Selanjutnya penyebab utama dari kelebihan beban adalah peningkatan *client* yang sangat besar dan terus-menerus melakukan permintaan ke layanan yang disediakan oleh *server*. Hal ini menyebabkan, *server* gagal untuk memberikan layanan dengan tepat dan cepat (Kunda et al., 2010).

Server akan mengalami kelebihan beban tentu saja hal tersebut akan memengaruhi ketertarikan pengguna, menurunkan pendapatan dan hilangnya reputasi. Solusinya adalah membuat kumpulan *server*, yang banyak digunakan di organisasi untuk memastikan daya tanggap permintaan oleh kumpulan *server*. Selain itu dapat menggunakan kumpulan *server* untuk mendapatkan sumber daya yang lebih baik dan persisten (Jader et al., 2019).

Berdasarkan data yang dipublikasikan oleh (Statista, 2020) pengguna media sosial semakin meningkat setiap tahun. Pada tahun 2020 jumlah pengguna media sosial telah mencapai 3.6 miliar pengguna. Data tersebut dapat dilihat pada Gambar 1.1 terlihat kenaikan pengguna media sosial dari rentang waktu 2017-2025. Kenaikan tersebut menyebabkan banyak *provider* berlomba-lomba untuk meningkatkan pelayanan agar *user* tidak mengalami kendala saat

mengakses web. Salah satu teknik yang digunakan untuk mengatasi lonjakan tersebut adalah *load balancing*.

Gambar 1.1 Jumlah Pengguna Sosial media rentang waktu 2017-2025

Sumber: (Statista, 2020)

Load balancing merupakan suatu teknik yang digunakan untuk menyebarkan trafik pada minimal dua jalur, sehingga trafik dapat lebih dioptimalkan, *delay* minimal, serta pada *server* tidak akan terjadi *overload* (Henanda & Agistira, 2020). Pada sebuah instansi atau perusahaan yang setidaknya memiliki minimal dua sambungan internet, teknik *Load balancing* dapat diterapkan. Teknik ini perlu digunakan pada saat *server* memiliki jumlah pengguna yang telah melebihi kapasitas maksimal dari *server* itu sendiri, karena teknik ini mampu mengurangi beban kerja setiap *server* sehingga tidak ada *server* yang memiliki kelebihan beban (Agung Nugroho, Widhi Yahya, 2017).

Untuk menerapkan *load balancing* akan menggunakan Google Cloud Platform alasannya adalah Google *Cloud* Platform memiliki saldo *trial* yang banyak dibandingkan dengan *provider* lain. Untuk perbandingannya jika menggunakan Google *Cloud* Platform akan mendapatkan saldo *trial* sebesar 300$ dan masa aktif selama 90 hari. Maka dari itu penelitian ini akan menggunakan metode "PEMANFAATAN GOOGLE *CLOUD* DAN DAN TEKNIK *LOAD BALANCING* UKTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB".

1.2 Rumusan Masalah

Berdasarkan permasalahan yang telah diuraikan pada latar belakang rumusan masalah yang akan dihadapi dalam pembuatan tugas akhir adalah:

Bagaimana melakukan *Load balancing* CPU menggunakan Google Cloud Platform untuk menyeimbangkan beban *server*?

1.3 Batasan Masalah

Agar penelitian ini lebih terarah dibutuhkan Batasan masalah agar penelitian ini tidak melebar dari pokok permasalahan. Berikut Batasan masalah pada penelitian ini:

1. Menggunakan Google Cloud Platform untuk membuat arsitektur jaringan
2. Hanya berfokus pada peningkatan performa web
3. Tidak membahas segi keamanan terhadap jaringan.
4. Berfokus pada konfigurasi Google *Cloud* untuk meningkatkan performa pada sebuah web

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan di atas, maka tujuan dari penelitian ini adalah sebagai berikut.

Melakukan *Load balancing* CPU menggunakan Google Cloud Platform untuk menyeimbangkan beban *server*.

1.5 Manfaat Penelitian

Dengan adanya penelitian ini diharapkan dapat memberikan manfaat antara lain:

1. Meningkatkan performa web agar tidak terjadi *down* saat mendapat trafik yang besar.
2. Menerapkan teknologi *cloud computing* dan *load balancing* untuk menghemat pengeluaran.

1.6 Metode Penelitian

a. Identifikasi Kebutuhan

Pada tahap ini dimulai dengan melakukan pencarian informasi tentang sistem yang akan dibuat agar nantinya sistem tersebut sesuai dengan apa yang diinginkan. Pengumpulan data dalam tahap ini menggunakan penelitian terdahulu dan studi

literatur. Pencarian informasi berupa bagaimana sistem itu nantinya berjalan, apa saja fitur yang dibutuhkan dan keluaran dan masukan dari aplikasi tersebut.

b. Perancangan Sistem

Setelah melakukan pencarian informasi, tahap selanjutnya adalah melakukan perancangan desain sistem yang akan dibuat berdasarkan informasi yang didapat pada saat melakukan analisis kebutuhan. Nantinya, sistem yang akan dibuat harus sesuai dengan analisis kebutuhan. Proses ini berfokus pada: pembuatan sistem, topologi jaringan.

c. Implementasi Sistem

Setelah melakukan perancangan sistem langkah selanjutnya adalah melakukan implementasi. Tahapan ini rancangan desain yang telah dibuat akan diubah menjadi menjadi sistem. Tahapan inilah yang merupakan tahapan secara nyata dalam pembuatan suatu sistem

d. Pengujian Sistem

Setelah melakukan analisis kebutuhan, perancangan arsitektur jaringan, dan implementasi maka sistem yang telah selesai dibuat. Sesuatu yang dibuat harus diuji coba. Demikian juga dengan sistem, semua fungsi sistem harus diujicobakan, agar sistem bebas dari error, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya

1.7 Sistematika Laporan

Bab I Pendahuluan

Berisi latar belakang mengenai permasalahan yang menjadi dasar penelitian, pentingnya menerapkan *Load balancing*dan Google *Cloud* CDN, dan solusi yang ditawarkan terhadap masalah yang ada. Berdasarkan latar belakang yang ada, kemudian disusun rumusan masalah sebagai dasar perencanaan penyelesaian masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

Bab II Landasan teori

Berisi uraian literatur-literatur yang sesuai dengan topik penelitian sebagai dasar untuk melakukan penelitian yang dilakukan dan beberapa penelitian sejenis yang telah dilakukan. Teori-teori yang diuraikan dalam bab ini menggunakan jurnal, buku, dan artikel sebagai bahan

referensi dengan topik yang meli*p*uti implementasi *load balancing* pada web *server*, implementasi *load balancing* pada google *cloud*, pemanfaatan CDN untuk meningkatkan performa web.

Bab III Metodologi Penelitian

Berisi berbagai tahapan dan kebutuhan penelitian sebagai acuan untuk mencapai solusi atas permasalahan dalam penelitian ini. Bab ini terdiri dari pengumpulan data, perancangan sistem dan analisis pengujian sistem.

Bab IV Hasil dan Pembahasan

Berisi hasil dan pembahasan dari setiap proses dalam sistem, pengujian kinerja sistem, kelebihan, dan kekurangan sistem. Pengujian dari kinerja sistem menggunakan uji validasi.

Bab V Kesimpulan dan Saran

Berisi kesimpulan mengenai hasil penelitian telah yang sesuai dengan tujuan penelitian atau belum serta saran yang mendukung agar penelitian ini dapat dilanjutkan oleh para peneliti lain dengan mengembangkan keterbatasan dan kekurangan yang terdapat pada penelitian ini.

BAB II
LANDASAN TEORI

2.1 *Load balancing*

Load balancing merupakan sebuah teknik yang berguna untuk menyeimbangkan beban pada *server* untuk meminimalisir terjadinya *overload* pada saat web sedang diakses secara bersamaan secara besar-besaran. *Load balancing* adalah teknik yang digunakan untuk mendistribusikan beban trafik pada minimal dua jalur koneksi secara seimbang, agar trafik dapat berjalan optimal, memaksimalkan throughput, memperkecil waktu tanggap dan menghindari overload pada salah satu jalur koneksi (Henanda & Agistira, 2020). *Load balancing* biasanya digunakan pada saat *server* mempunyai *user* yang melebihi kapasitas dari *server* tersebut. *Load balancing* dapat dijelaskan secara umum sebagai teknik yang digunakan untuk mendistribusikan beban kerja *server* secara merata pada minimal dua komputer, *network links*, CPU, *hard drive* atau sumber daya lain, untuk mendapatkan pemanfaatan sumber daya yang optimal, memaksimalkan *throughput*, meminimalkan waktu respon dan menghindari overload (Septandy & Fauzi, 2019). *Load balancing* tidak akan menambah besarnya *bandwidth* pada sebuah *server* melainkan hanya membagi trafik agak dapat berjalan secara optimal. Dengan menggunakan beberapa bagian dari *load balancing* hal tersebut berefek pada meningkatnya kemampuan melalui *redudancy*. Dengan memiliki beberapa link maka optimalisasi pemakaian sumber daya *throughput* dan *response time* dapat semakin meningkat karena memiliki lebih dari satu link yang dapat saling mendukung pada saat jaringan *down* dan menjadi cepat pada saat jaringan normal (Gene, 2018). Beberapa keuntungan menggunakan *load balancing* adalah:

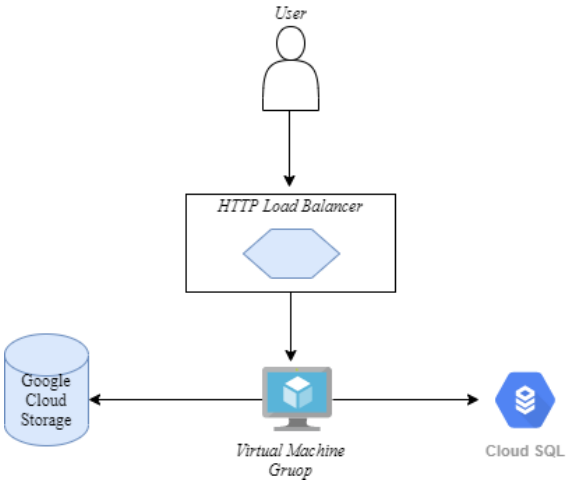
a. *Scalability* adalah salah satu aspek yang *Load balancing* mampu mengatasi terjadinya lonjakan lalu lintas, mungkin *server* sulit menangani lalu lintas yang berlebih, dan situs web mungkin down. Dengan menggunakan *load balancing*, lalu lintas dapat tersebar di beberapa *server*, dan peningkatan lalu lintas dapat ditangani dengan cara yang jauh lebih mudah.
b. *Redudancy* adalah dengan *load balancing* dampak dari kegagalan perangkat keras pada waktu operasional situs secara keseluruhan dapat dibatasi secara signifikan. Dengan menerapkan teknik *load balancing*, dapat mencapai *redundancy*. Artinya ketika lalu lintas situs web dikirim ke dua atau lebih *server* web, dan satu *server* gagal, maka *load balancer* akan secara otomatis memindahkan lalu lintas ke *server* lain yang berfungsi.

c. *Increase Performance* dan *Reduced Downtime* adalah dapat melakukan penjadwalan pemeliharaan dan *Reduced Downtime* yang direncanakan di luar jam kerja seperti pagi hari atau di akhir pekan. Dapat mematikan *server* apa pun untuk pemeliharaan dan lalu lintas saluran ke sumber daya yang lain tanpa mengganggu pekerjaan di lokasi mana pun. Dengan cara ini, dapat mengurangi waktu henti, mempertahankan waktu kerja, dan meningkatkan kinerja.

d. *Efficiently Managed Failures* adalah teknik *load balancing* membantu mendeteksi kegagalan sejak dini dan mengelolanya secara efisien, memastikan bahwa kegagalan apa pun tidak memengaruhi *server* atau beban kerja. Dengan menggunakan beberapa pusat data yang tersebar di sejumlah penyedia *cloud,* dapat melewati kegagalan yang terdeteksi dengan mendistribusikan kembali sumber daya ke area lain yang tidak terpengaruh, sehingga menyebabkan gangguan minimal.

e. *Increase Flexibility* adalah administrator TI dapat menikmati fleksibilitas luar biasa dalam menangani lalu lintas situs web menggunakan beberapa *server* dengan beban seimbang. Mereka dapat melakukan beberapa tugas pemeliharaan di *server* tanpa memengaruhi waktu aktif situs. Dapat tercapai dengan mengarahkan semua lalu lintas ke satu *server* dan menempatkan penyeimbang beban dalam mode aktif/pasif. Jika memiliki fleksibilitas untuk memiliki sistem pemeliharaan bertahap, di mana setidaknya satu *server* selalu tersedia untuk mengambil beban kerja sementara yang lain menjalani pemeliharaan. Ini memastikan bahwa pengguna situs tidak mengalami pemadaman apapun setiap saat.

f. *Security* adalah termasuk semua trafik yang melewati *load balancer*, aturan keamanan dapat diterapkan dengan mudah. Untuk *server* yang menggunakan jaringan pribadi, dari luar sistem alamat *IP* tidak dapat langsung diakses.



Gambar 2.1 Arsitektur *Load Balancing*

Gambar 2.1 adalah arsitektur *Load balancing* gambar tersebut menunjukkan bagaimana proses *load balancing* terjadi. *Load balancer* berperan untuk merutekan permintaan *client* di semua *server* untuk memenuhi permintaan tersebut dengan cara yang memaksimalkan kecepatan dan pemanfaatan kapasitas dan memastikan bahwa tidak ada *server* yang bekerja secara berlebihan, yang dapat menurunkan kinerja. Jika satu *server* mati, *load balancer* mengalihkan lalu lintas ke *server* online yang tersisa. Saat *server* baru ditambahkan ke grup *server*, penyeimbang beban secara otomatis mulai mengirim permintaan ke sana. Google Cloud Storage adalah layanan untuk menyimpan objek di Google Cloud. Objek adalah bagian data yang tidak dapat diubah yang terdiri dari *file* dalam format apa pun. Setelah membuat proyek, dapat membuat keranjang *Cloud Storage*, mengupload objek ke keranjang, dan men*download* objek dari keranjang. Google *Cloud Storage* dapat memberikan izin untuk membuat data dapat diakses oleh anggota yang telah ditentukan. *Cloud SQL* adalah layanan database terkelola sepenuhnya yang membantu untuk menyiapkan, memelihara, mengelola, dan mengatur database relasional di Google Cloud Platform. *Virtual Machine* adalah sistem komputer emulasi yang dibuat menggunakan perangkat lunak. *Virtual machine* menggunakan sumber daya seperti CPU, RAM, dan penyimpanan disk, tetapi dipisahkan dari perangkat lunak lain di komputer. Hal tersebut membuat *virtual machine* dapat dengan mudah dibuat, dimodifikasi, atau dihancurkan tanpa mempengaruhi komputer asli.

2.2 **Content *Network* Delivery**

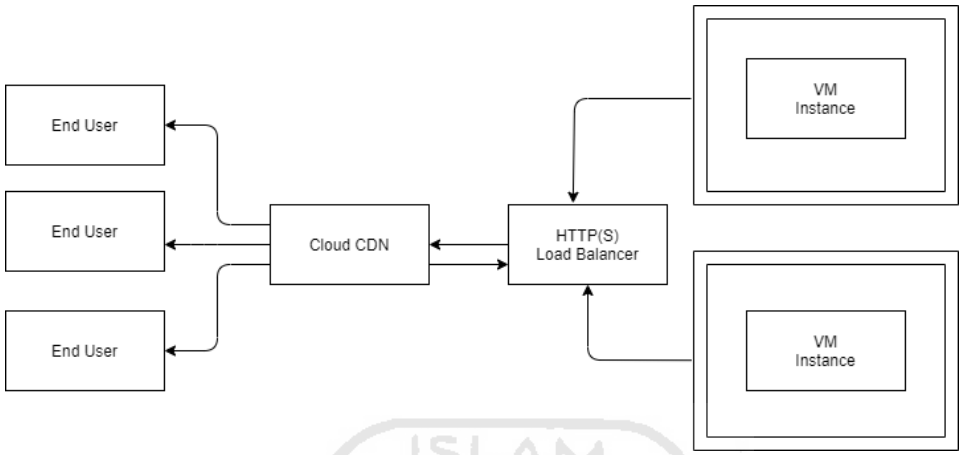
Content Delivery *Network* adalah sejumlah *server* yang diletakkan di berbagai lokasi untuk menyebarkan konten (seperti css, gambar, dan video). Konten web dikirimkan ke pengguna akhir berdasarkan lokasi pengguna. Layanan CDN mampu mempercepat pengiriman konten web (Stocker et al., 2017). CDN juga meningkatkan lalu lintas, yang mengarahkan situs

web ke jangkauan global. Pengguna mendapatkan konten yang lebih cepat untuk dikirim oleh *server* CDN terdekat. CDN juga memberikan keamanan dan perlindungan dari arus besar di lalu lintas.

CDN bekerja dengan cara *request* ditangani oleh *server* jaringan yang paling dekat dengan perangkat yang membuat kueri. Dengan menyimpan data ke *cache* dan menyebarkan banyak *request* untuk informasi yang sama melalui jaringan dar*ip*ada satu *server* web, beban lalu lintas menjadi jauh lebih seimbang. Cara ini meminimalisir masalah seperti kecepatan halaman, kerusakan *browser*, dan gangguan layanan. CDN memiliki beberapa kelebihan yaitu:

a. Keandalan dan waktu respons mendapat dorongan besar: Situs web memiliki kinerja yang tinggi sama dengan meningkatkan penjualan. Masalah latensi dan kecepatan cenderung melumpuhkan bisnis web dan menyebabkan kerusakan. CDN yang andal memastikan bahwa kecepatan optimal dan transaksi *online* dilakukan dengan mulus.
b. CDN mampu memperluas jangkauan sampai skala global: CDN memberikan solusi melalui akselerasi *cloud*. Jangkauan global ini akan menghilangkan masalah latensi yang mengganggu transaksi *online* jarak jauh dan menyebabkan waktu muat yang lambat.
c. Penggunaan CDN bisa menghemat uang: Menggunakan CDN dapat menghemat untuk bisnis; Dar*ip*ada membuat infrastruktur dan layanan di seluruh dunia, CDN dapat menghilangkan kebutuhan untuk membayar *hosting* asing yang mahal, dengan demikian, CDN menghemat banyak uang untuk keperluan bisnis.
d. *Availability*: Karena aset didistribusikan di banyak wilayah, CDN memiliki mekanisme sensor ketersediaan *server* otomatis dengan pengalihan pengguna instan. Maka dari itu situs web CDN memiliki *availability* 100 persen, bahkan selama pemadaman listrik besar-besaran, masalah perangkat keras, atau masalah jaringan.
e. Mengurangi beban *server*: Pada dasarnya, konten tersebar di beberapa *server* alih-alih memindahkannya ke satu *server* besar.
f. Dapat menangani peningkatan *user* yang signifikan: Menempatkan *server* secara strategis dalam CDN dapat menghasilkan kapasitas tulang punggung jaringan yang tinggi, yang setara dengan peningkatan yang signifikan dalam jumlah pengguna yang mengakses jaringan pada waktu tertentu.
g. Perlindungan DDos: Selain menimbulkan kerugian ekonomi yang sangat besar, serangan DDoS juga dapat berdampak serius pada reputasi dan citra perusahaan atau organisasi yang menjadi korban. Setiap kali pelanggan mengetik nomor kartu kredit mereka untuk melakukan pembelian secara *online*, mereka menaruh kepercayaan pada bisnis itu. Solusi

cloud dirancang untuk menghentikan serangan sebelum mencapai pusat data. CDN akan mengambil lalu lintas dan menjaga situs web tetap aktif dan berjalan. Hal itu berarti tidak perlu khawatir tentang serangan DDoS yang memengaruhi pusat data, menjaga situs web bisnis tetap aman dan sehat.



Gambar 2.2 Arsitektur CDN

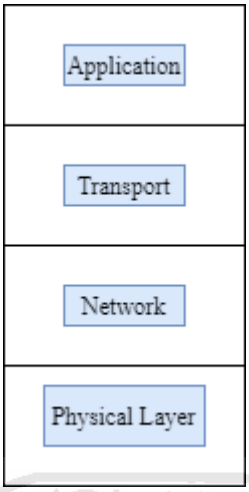
Sumber: (Google, 2021a)

Gambar 2.2 adalah Arsitektur CDN di *Cloud* CDN, gambar berikut menggambarkan bagaimana respons dari *server* yang berjalan pada *instance Virtual Machine* berjalan melalui Load Balancer HTTP (S) sebelum dikirim oleh *Cloud* CDN.

2.3 TCP/*IP*& OSI Layer

Lapisan TCP/*IP* atau disebut juga Transmission Control Protocol/Internet Protocol layer adalah model komunikasi data yang dikembangkan oleh US Department of Defense (DoD), yang menggambarkan komunikasi data yang terjadi antara perangkat jaringan dan antar jaringan (Mundra & Taeib, 2015). Protokol menentukan jenis, waktu, urutan dan pemeriksaan kesalahan yang digunakan dalam jaringan. Transmission Control Protocol/Internet Protocol (TCP/*IP*) adalah protokol yang digunakan untuk mengirim data antar komputer di jaringan. Protokol ini merupakan protokol yang digunakan untuk mengakses jaringan internet sehingga dapat untuk melakukan komunikasi secara global (Mundra & Taeib, 2015). Dalam konsep komunikasi data jaringan komputer terdapat mekanisme data dari komputer sumber ke komputer target. Masalah dalam proses pengirimannya tidak semudah yang dibayangkan. Alasan pertama adalah karena komputer target jauh dari komputer sumber, sehingga paket data yang dikirim dapat hilang atau rusak selama transmisi (Alotaibi et al., 2017). Alasan lain mungkin karena komputer target sedang mengirim atau menunggu data dari komputer sumber lain. Tentunya paket data yang dikirim diharapkan sampai dengan tepat tanpa mengalami

kerusakan, untuk menyesuaikan mekanisme komunikasi data maka perlu dilakukan penyesuaian proses pengiriman data yang disebut dengan protokol. Protokol adalah perangkat lunak yang disertakan dengan sistem operasi.



Gambar 2.3 TCP/IP Model

Sumber: (Mundra & Taeib, 2015)

Gambar 2.3 merupakan susunan *layer* TCP/IP yang dimulai dengan *aplication, transport, network, physical layer*. *Aplication* merupakan lapisan tertinggi sedangkan *physical layer* merupakan lapisan terendah dalam TCP/IP model.

Protokol TCP/*IP* memiliki 4 *layer* yaitu (Mundra & Taeib, 2015):

a. *Physical Layer (Network Interface Layer*) merupakan lapisan paling bawah yang menggambarkan media transmisi jaringan, metode pensinyalan, sinkronisasi bit, arsitektur jaringan, topologi jaringan dan pengkabelan. Hal lain yang terjadi pada layer 1 yakni mendefinisikan *network interface card* (NIC) agar bisa berinteraksi dengan media kabel atau radio.
b. *Network* Layer (Internet Layer) *Network* Acces Layer memiliki fungsi mirip dengan lapisan data link di OSI. Lapisan ini mengatur distribusi *data frame* pada media fisik yang digunakan dengan andal. Lapisan ini biasanya menyediakan layanan untuk mendeteksi dan mengoreksi data yang dikirimkan. *Network* Layer mempunyai peran penting yaitu:
    1. *Address*ing adalah memasangkan tiap-tiap paket data beserta alamat internet atau biasa disebut dengan *internet protocol address (IP address)* sehingga jaringan TCP/IP tidak tergantung pada jenis media, sistem operasi dan komputer yang digunakan.
    2. *Routing* adalah proses untuk meneruskan paket-paket jaringan dari satu jaringan ke jaringan lainnya melalui sebuah antar-jaringan. *Routing* juga dapat merujuk kepada

sebuah metode penggabungan beberapa jaringan sehingga paket-paket data dapat dihantarkan dari satu jaringan ke jaringan selanjutnya.

3. Transport Layer adalah lapisan yang bertanggung jawab atas aliran data antara dua *host* (*client* dan *server*). Lapisan ini menyediakan koneksi dari ujung ke ujung secara efisien, lapisan ini mampu mengirimkan data secara berurutan dan menghindari duplikasi:
    a) User Diagram Protocol (UDP): Sebaliknya, UDP menggunakan mekanisme sederhana pada lapisan bawah untuk mengirimkan data, dan protokol lapisan atas untuk memastikan datanya berhasil dikirim ke tingkat yang diperlukan. UDP adalah protokol sederhana, dan bertanggung jawab untuk mengirim paket (*datagram*) tanpa memperhatikan kehandalan, yang mana ditangani pada lapisan aplikasi. Jika UDP kehilangan *file byte* data hal tersebut tidak akan berpengaruh signifikan, dan lapisan aplikasi akan bertanggung jawab untuk mendeteksi kehilangan data dan dikirim ulang saat aplikasi lapisan memilih untuk menggunakan UDP.

       UDP memiliki karakteristik antara lain:
        1) Protokol yang ringan untuk saling bertukar pesan.
        2) UDP tidak mempunyai mekanisme *flow-control*, seperti yang dimiliki oleh TCP
        3) *Unreliable* (tidak andal). Pesan-pesan UDP akan dikirimkan sebagai datagram yang tidak mempunyai nomor urut atau pesan *acknowledgement*. Protokol lapisan aplikasi yang berjalan di atas lapisan UDP memiliki tugas untuk memulihkan pesan-pesan yang hilang selama pengiriman pesan.

    b) Transmission Control Protocol (TCP): Fungsi dari TCP untuk memastikan bahwa data dapat mencapai tujuan dengan sepenuhnya. Pada TCP data akan dibagi meinjadi ukuran yang sesuai untuk diteruskan ke lapisan berikutnya dan kemudian penerima akan menerima pesan dikirim untuk memastikan bahwa paket telah dikirim.

       TCP memiliki karakteristik antara lain:
        1) *Reliable*: Nantinya data akan dikirimkan kepada penerima secara berurutan seperti ketika dikirim. Mekanisme ini membuat pengiriman data lebih andal dan efisien.

2) *Connection Oriented*: Sebelum data dapat ditransmisikan antara dua *host*, terdapat dua proses yang sedang berjalan pada lapisan aplikasi harus melakukan negOSIasi untuk membuat sesi koneksi terlebih dahulu. Koneksi TCP ditutup dengan menggunakan proses terminasi koneksi TCP (TCP *connection termination*).
3) Full-Duplex: Untuk setiap *host* yang berada di TCP, koneksi akan terbentuk di antara dua *host* terdiri dari dua buah jalur, yaitu jalur keluar dan jalur masuk. Dengan menggunakan teknologi lapisan yang berada di bawahnya hal tersebut dilakukan guna mendukung *full-duplex*, maka data pun dapat secara bersamaan diterima dan dikirim. Header TCP berisi nomor urut (TCP *sequence number*) dari data yang ditransmisikan dan sebuah *acknowledgement* dari data yang masuk
4) Memiliki Layanan *Flow-Control* untuk mencegah agar pengiriman data tidak terlalu banyak pada satu waktu, yang akhirnya membuat akan membuat perlambatan pada jaringan *internetwork IP*, TCP mengimplementasikan layanan *flow control* yang dimiliki oleh pihak pengirim yang secara terus menerus memantau dan membatasi jumlah data yang dikirimkan pada satu waktu. Hal ini dilakukan untuk melakukan antisipasi agar pihak penerima tidak memperoleh data yang tidak dapat disangganya (*buffer*).

1. Application Layer adalah sebuah lapisan ini yang berkaitan dengan interaksi manusia dan bagaimana aplikasi perangkat lunak diimplementasikan. Lapisan aplikasi terdiri dari *interface* dan protokol komunikasi yang dapat diterapkan dalam komunikasi proses-ke-proses. Lapisan aplikasi berkaitan dengan penyediaan layanan jaringan ke aplikasi. Pada lapisan ini setiap jalur aplikasi dan sesi dapat dibedakan dengan penggunaan spesifik soket dan nomor port. Lapisan aplikasi mencakup semua protokol tingkat yang lebih tinggi seperti:
    a) HTTP adalah sebuah protokol yang memungkinkan koneksi antara *server* web dan *client* dan juga mendistribusikan informasi di protokol jaringan lapisan aplikasi (*application layer*) yang dikembangkan untuk membantu proses transfer antar komputer. Protokol ini berguna untuk mentransfer

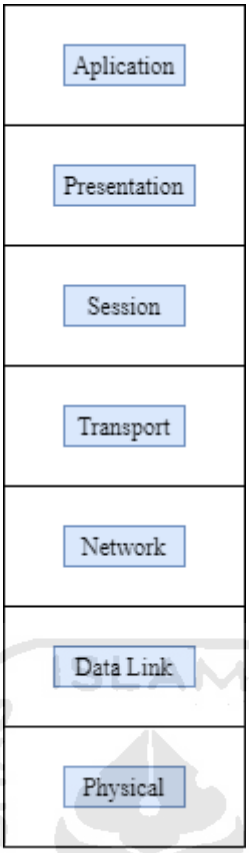
informasi seperti dokumen, *file*, gambar, dan video antar komputer. Protokol ini menggunakan port 80. Pada sisi *server*, contoh penggunaan HTTP adalah Apache Web *Server* dan Informasi Internet *Server* (IIS), sedangkan pada sisi *client* penerapan HTTP berada pada Firefox, Internet Explorer, Mozilla dan Google Chrome adalah yang paling umum.

b) SMTP adalah SMTP adalah standar pengiriman elektronik mail (E-mail) melalui jaringan TCP/*IP*: SMTP memproses pesan dengan menggunakan port 25

c) DNS adalah: Penerjemah dari yang awalnya adalah IP menjadi sebuah URL hal ini memudahkan pengguna untuk dapat mengakses web tanpa harus menghafal seluruh alamat IP web yang akan diakses.

d) FTP adalah Protokol yang digunakan untuk mengirim dan menerima *file* besar dari *server* jarak jauh.

OSI (*Open-Source Interconnection*) dan Model Referensi OSI adalah sekumpulan protokol yang mengontrol berbagai bagian jaringan komputer. Semua tahapan di jaringan komputer dasar dapat dikompilasi menggunakan model OSI. OSI adalah model referensi jaringan komputer yang dapat digunakan untuk desain jaringan komputer, spesifikasi operasi, dan pemecahan masalah koneksi jaringan yang bermasalah (Alotaibi et al., 2017). Banyak protokol yang terkait dengan lapisan jaringan berada pada tahap model lapisan OSI. OSI model dikenalkan oleh International Organization for Standardization (ISO) pada tahun 1984, dan kembali direvisi pada tahun 1994. OSI model diciptakan untuk mengatur standar pengembangan dalam sistem pertukaran data. OSI model menjelaskan 7 lapisan dari lapisan koneksi fisik pertama hingga lapisan aplikasi ketujuh. Setiap lapisan memiliki fungsi yang berbeda dari lapisan atas, misalnya lapisan kedua menyediakan layanan untuk lapisan ketiga.



Gambar 2.4 OSI Models

Sumber: (Ravali, 2013)

Gambar 2.4 merupakan layer OSI *models* lapisan paling atas adalah *aplication* setelah itu *presentation, session, transport, network, data link dan physical layer* setiap bagian dari lapisan tersebut memiliki fungsi masing-masing.

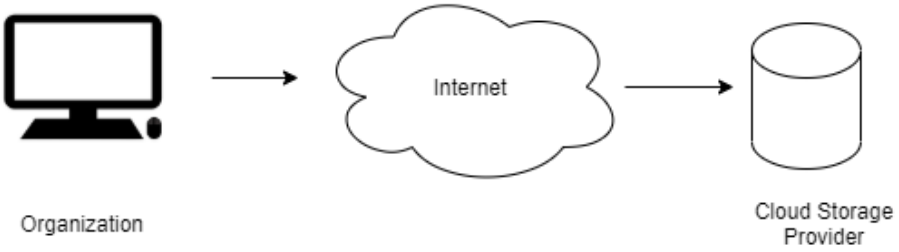
OSI Layer membagi fungsi *network* menjadi 7 lapisan yaitu (Meyer & Zobrist, 2002):

a. Physical Layer: Layer ini berada di dasar model jaringan data dan memiliki fungsi untuk menginterpretasi media transmisi jaringan, metode pensinyalan, sinkronisasi bit, arsitektur jaringan, topologi jaringan dan pengkabelan dan berkaitan dengan data mentah dalam bentuk sinyal-sinyal listrik.
b. Data Link Layer: Lapisan ini berfungsi untuk melakukan pengecekan apabila terdapat kesalahan dalam penyaluran transmisi ke data. Untuk mencegah terjadinya kesalahan tersebut data pada *layer* ini akan dibungkus menjadi *frame*. Data Link Layer dibagi menjadi 2 sublayer yakni MAC dan LLC. Media Acces Control (MAC) adalah sebuah lapisan yang berfungsi untuk mengontrol sebuah perangkat jaringan untuk memperoleh koneksi ke medium. Sedangkan Logical Link Control (LLC) berfungsi untuk melakukan koreksi terhadap kesalahan pada saat penyaluran transmisi bit.

c. *Network* Layer: Pada layer 3 lebih berfokus pada membantu untuk melakukan mendefinisikan *ip address*. Hal tersebut membuat semua komputer yang berada dalam satu jaringan dapat saling berkomunikasi antara satu dengan yang lainnya. Selain melakukan pendefinisian *ip address, network layer* juga berfungsi untuk melakukan proses *routing* dan membuat *header* untuk setiap paket data yang ada.
d. Transport Layer: Memiliki fungsi untuk membagi data ke dalam paket-paket data beserta membagikan nomor urut paket-paket kemudian dapat diurutkan kembali pada saat telah diterima
e. Session Layer: Layer ini mengendalikan komunikasi antar aplikasi. Layer ini membentuk, mengendalikan dan menghentikan koneksi komunikasi (*session*)
f. Presentation Layer: Layer ini menyediakan layanan ke *application layer* di mana *service* tersebut adalah memastikan bahwa informasi terjaga selama proses transmisi. Selain itu pada layer ini terjadi juga proses *compression and encryption* data.
g. Application Layer: Pada layer ini *user* atau *operating system* memiliki akses ke *service* jaringan. Layer ini bertugas untuk menayangkan konten dan data agar dapat dilihat dan juga ditayangkan kepada *user*. Lapisan ini terhubung pada software dengan cara melakukan identifikasi sumber daya yang digunakan untuk komunikasi, menentukan ketersediaan jaringan dan melakukan distribusi mengenai informasi *service*.

2.4 *Cloud Storage*

Cloud Storage adalah penyimpanan *virtual* yang dapat menjangkau banyak perangkat penyimpanan fisik di lokasi yang berbeda (Prasad et al., 2013). Saat menggunakan *cloud* storage, beberapa *file* mungkin ada di *server* fisik di Los Angeles sementara *file* lainnya ada di *server* fisik di California. Sebagian besar data yang diakses melalui internet tidak disimpan di komputer pribadi adalah bagian dari *Cloud*. Misalnya, jika menggunakan layanan email berbasis internet seperti Gmail, Yahoo, atau Live, bisa mengakses email di mana pun selama memiliki koneksi internet. Ini karena datanya disimpan di *server* yang dimiliki oleh masing-masing penyedia email, bukan mesin lokal.



Gambar 2.5 Cloud Storage

Gambar 2.5 adalah gambaran umum dari sebuah *cloud* storage nantinya *cloud* storage ini akan memperingan kerja *server* utama karena *database* dari *server* utama diletakkan pada *cloud*. Maka dari itu ketika *user* ingin mengakses web tidak langsung diarahkan ke *server* utama melainkan diarahkan terlebih dahulu ke *cloud storage* dimana penyimpanan *database* berada.

2.5 NGINX

Nginx adalah *software open-source* yang memiliki kinerja tinggi sebagai *server* HTTP dan *reverse proxy*. Nginx dengan cepat memberikan konten statis dengan penggunaan efisien sumber daya sistem. Hal ini dapat menyebarkan dinamis HTTP konten di jaringan menggunakan *Fast* CGI *handler* untuk *script*, dan dapat berfungsi sebagai perangkat lunak yang sangat mampu penyeimbang beban (Yusuf et al., 2013). Nginx dibangun secara *modular* dan dengan demikian mampu mendukung berbagai fitur seperti *Load Balancing* dan *Reverse Proxying*, *Virtual hosts* berbasis nama dan *IP*, *Fast* CGI, akses langsung ke *cache*, *SSL*, *Flash Video Streaming* dan sejumlah fitur-fitur standar lainnya. Nginx dapat dijalankan dan tersedia untuk *platform* Unix, Linux, varian dari BSD, MacOS X, Solaris, dan Microsoft Windows.

2.6 Wordpress

WordPress adalah sebuah perangkat lunak blog yang ditulis dalam bahasa pemrograman PHP dan mendukung sistem basis data MySQL. WordPress adalah penerus resmi dari cafelog yang dikembangkan oleh Michel Valdrighi. Nama WordPress diusulkan oleh Christine Selleck, teman dari ketua developer, Matthew Charles Mullenweg. Rilis terbaru WordPress adalah versi 2.7 (Muliantara, 2009). WordPress didistribusikan dengan lisensi GNU *General Public License* selain sebagai blog wordpress juga sebagai *Content Management System* (CMS) karena kemampuannya untuk dimodifikasi sesuai kebutuhan.

2.7 Apache Benchmark

Apache benchmark adalah alat untuk mengukur *server Apache Hypertext Transfer Protocol* (HTTP). Ini dirancang untuk memberi kesan tentang bagaimana kinerja instalasi Apache saat ini. Ini secara khusus menunjukkan kepada pengembang berapa banyak

permintaan per detik yang dapat dilayani oleh instalasi Apache. Beberapa fitur yang dimiliki oleh apache benchmark antara lain: *open source, simple command line, platform independent, load and performance test.* Apache benchmark dapat digunakan untuk menguji performa dari web *server* dengan berbagai batasan pengujian seperti *transfer rate* dan *request per second* (Chandra, 2019).

2.8 *Horizontal Scale Up*

Horizontal scale up adalah sebuah metode untuk meningkatkan kapabilitas dari *multiple server* di mana masing-masing dari *server* tersebut tidak memiliki banyak perubahan dari segi spesifikasi. *Horizontal scale up* berarti penskalaan dengan menambahkan lebih banyak mesin ke kumpulan sumber daya. *Horizontal scale up* digunakan untuk membagi beban ke beberapa *server*, dengan menambahkan *server* tambahan untuk meningkatkan kapasitas sesuai kebutuhan. Meskipun kecepatan atau kapasitas keseluruhan dari satu mesin mungkin tidak tinggi, setiap mesin menangani sebagian dari keseluruhan beban kerja, berpotensi memberikan efisiensi yang lebih baik daripada satu *server* berkapasitas tinggi berkecepatan tinggi. Memperluas kapasitas penerapan hanya memerlukan penambahan *server* tambahan sesuai kebutuhan, yang bisa menjadi biaya keseluruhan yang lebih rendah daripada perangkat keras kelas atas untuk satu mesin. Imbalannya adalah peningkatan kompleksitas dalam infrastruktur dan pemeliharaan untuk penerapan. (Bondi, 2000)

2.9 *Image Virtual Machine*

Image virtual machine adalah *resource compute engine* yang menyimpan semua konfigurasi, metadata, izin, dan data dari satu atau beberapa disk yang diperlukan untuk membuat *instance virtual machine* (VM). Image *virtual* machine dapat diguakan untuk pemeliharaan sistem, seperti pembuatan *instance*, pencadangan dan pemulihan, dan kloning *instance*. (Google, 2021d)

2.10 *Group Instance*

Tahapan ini pembuatan *group instance* bertujuan untuk membuat *virtual machine* yang dapat dikelola sebagai satu entitas. *Compute Engine* menawarkan dua jenis *group instance virtual machine*, terkelola dan tidak terkelola: *Manage instance group* (MIG) memungkinkan untuk menjalankan aplikasi pada beberapa *virtual machine* identik. *Group instance* tidak terkelola memungkinkan memuat keseimbangan di seluruh armada *virtual machine* yang dikelola sendiri. *Group instance* tidak terkelola memungkinkan untuk membuat keseimbangan di seluruh armada *virtual machine* yang dikelola sendiri. *Group instance* tidak terkelola memungkinkan untuk membuat keseimbangan di seluruh armada VM yang kelola sendiri.

Group instance tidak terkelola memungkinkan memuat keseimbangan di seluruh armada *virtual machine* yang dikelola sendiri. (Google, 2021b)

2.11 *Template Instance*

Template instance adalah sumber daya yang digunakan untuk membuat *instance virtual machine* dan *manage instance group. Template instance* digunakan untuk menentukan jenis *machine*, *image boot disk* atau *image container*, label, dan properti *instance* lainnya. Selain itu *template instance* dapat digunakan untuk membuat *manage instance group* atau membuat *virtual machine* individual. *Template instance* adalah cara yang mudah untuk menyimpan konfigurasi *instance virtual machine* sehingga dapat menggunakannya nanti untuk membuat *virtual machine* atau *group virtual machine*. (Google, 2021c)

2.12 Penelitian Terkait

Penelitian pada Tabel 2.1 yang dilakukan oleh (Supramana, Prismana, 2016) apache yang digunakan sebagai *load balancer* mampu mengatasi *request* kepada *server* tanpa mengalami kendala berlebih. Pada penelitian yang dilakukan oleh (Gupta & Garg, 2014) CDN yang diterapkan mampu mengoptimalkan konten dengan mereplikasi konten pada *server* pengganti. Hal ini dapat mengurangi konsumsi *bandwidth* dan meningkatkan latensi yang dirasakan pengguna.

Pada penelitian yang telah disebutkan di atas penentuan jalur berdasarkan *request*, sedangkan pada penelitain ini jalur ditentukan berdasarkan CPU (ketika CPU penuh, maka akan membuat CPU baru). Selain itu, pada penelitian-penelitian sebelumnya CDN digunakan untuk mempercepat pengiriman konten. Pada penelitian ini CDN berfungsi sebagai *front-end* atau digunakan hanya untuk menampilkan konten.

Tabel 2.1 Penelitian Terkait

No	Judul	Penulis	Pencapaian	Saran
1	IMPLEMENTASI *LOAD BALANCING* PADA WEB *SERVER* DENGAN	Supramana, I Gusti Lanang Putra Eka Prismana	Setelah menerapkan apache sebagai *load balancer* sistem mampu mendapatkan hasil bahwa apache *load balancer* berdasarkan method by *request*	Saran untuk pengembangan dan penelitian selanjutnya adalah perlu dilakukan

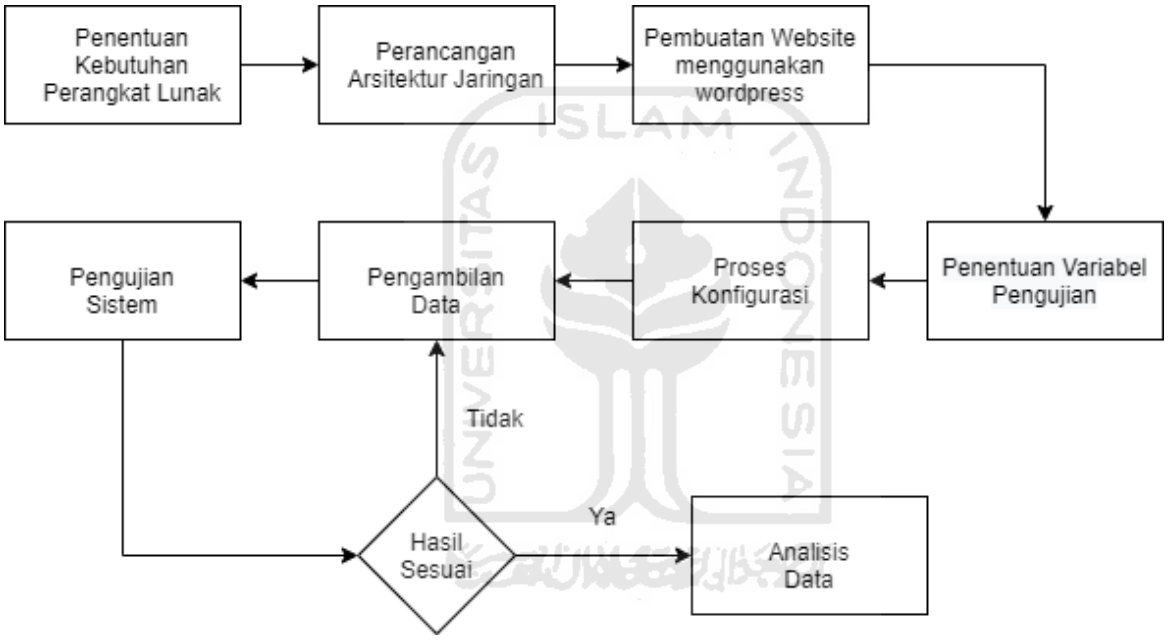
No	Judul	Penulis	Pencapaian	Saran
	MENGGUNAKAN APACHE		akan melanjutkan setiap *request* dari *client* ke beberapa web *server* secara berurutan dan bergantian sehingga setiap web *server* mendapatkan *request* yang sama tanpa salah satu web *server* yang memiliki beban yang berlebih. Sedangkan pada *apache load balancer* berdasarkan *method by business* apabila salah satu web *server* memiliki waktu respon yang rendah maka *apache load balancer* akan mengarahkan *request* dari *client* ke web *server* yang memiliki waktu respon yang tinggi.	penambahan sebuah sinkronisasi antara web *server* agar apabila salah satu web *server* terjadi *update* data ataupun *file* maka web *server* yang lain akan terupdate secara otomatis.
2	CONTENT DELIVERY *NETWORK* APPROACH TO IMPROVE WEB PERFORMANCE: A REVIEW	Meenakshi Gupta, Atul Garg	Sistem ini mampu membantu dalam mengurangi konsumsi *bandwidth* dan meningkatkan latensi yang dirasakan pengguna. Analisis ini membantu mendapatkan pengetahuan tentang jaringan pengiriman konten. Hal ini menunjukkan bahwa desain dan implementasi jaringan pengiriman konten yang dapat diskalakan, andal, dan efisien memerlukan fokus pada beberapa aspek teknis seperti konten apa yang harus direplikasi dan di mana, *server* mana yang	Saran untuk ke depannya kami mengusulkan *framework* baru untuk mendistribusikan konten yang efisien melalui *server* pengganti.

No	Judul	Penulis	Pencapaian	Saran
			sesuai tempat permintaan harus diarahkan.	

BAB III
METODOLOGI PENELITIAN

3.1 Metodologi

Pada tugas akhir ini peneliti menggunakan metode seperti pada Gambar 3.1 yang dimulai dengan melakukan menentukan kebutuhan perangkat lunak, merancang arsitektur jaringan, melakukan konfigurasi, menentukan variabel pengujian, melakukan pengambilan data dan melakukan analisis data yang telah didapatkan dari pengujian yang dilakukan.



Gambar 3.1 Metodologi Penelitian

Gambar 3.1 adalah metode perancangan sistem pertama yang akan dilakukan adalah menentukan kebutuhan perangkat lunak yang akan digunakan dalam penelitian ini seperti Load Balancer, *Virtual Machine*, *Image Virtual Machine*, *Template Instance*, *Group Instance*, *Cloud DNS, Cloud Storage, SQL Server*. Setelah melakukan analisis kebutuhan perangkat lunak langkah selanjutnya adalah melakukan perancangan arsitektur jaringan. Perancangan arsitektur jaringan ini adalah proses untuk merancang desain arsitektur jaringan yang telah ditentukan sebelumnya. Setelah melakukan perancangan arsitektur jaringan selanjutnya membuat web menggunakan wordpress yang nantinya akan di *load balancing.* Setelah membuat web selanjutnya adalah menentukan variabel pengujian, variabel pengujian disini terdiri dari trafik dan CPU *usage*. Langkah selanjutnya adalah melakukan pengambilan data dan pengujian

sistem. Sistem akan diuji menggunakan *apache benchmark*. *Apache benchmark* digunakan untuk menguji apakah *load balancer* sudah mampu untuk mengatasi jika terjadi lonjakan trafik yang didapatkan oleh web sedangkan pengujian digunakan untuk melihat apakah CDN telah berjalan dengan baik dan mampu untuk mendistribusikan konten secara maksimal tanpa mengalami perlambatan.

3.2 Identifikasi Kebutuhan

Tahap ini dimulai dengan melakukan pencarian informasi antara lain spesifikasi dari perangkat lunak yang akan digunakan dan pemodelan sistem saat berjalan. Informasi ini berguna untuk menunjang pengembangan sistem agar sesuai dengan apa yang diinginkan tentang sistem yang akan dibuat agar nantinya sistem tersebut berjalan sesuai dengan apa yang diinginkan. Pengumpulan data dalam tahap ini menggunakan penelitian terdahulu dan kajian literatur. Pencarian informasi berupa bagaimana aplikasi itu nantinya berjalan, apa saja fitur yang dibutuhkan dan keluaran dan masukan dari sistem tersebut. Nantinya sistem ini akan menggunakan Google Cloud Platform untuk menerapkan CDN dan *load balancing*.

Pada tahap ini penulis mengumpulkan informasi tentang sistem yang akan dirancang dalam tugas akhir ini yaitu optimasi performa web menggunakan *google cloud CDN* dan *load balancing*. Informasi yang dikumpulkan mengenai sistem yang akan dibuat adalah informasi perangkat lunak yang sesuai dengan sistem yang akan digunakan pada perancangan sistem. Pengumpulan informasi tentang sistem ini dilakukan guna mendapat gambaran tentang kebutuhan dari sistem agar dapat berjalan dan berfungsi sesuai dengan yang diinginkan.

3.2.1 Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak merupakan kumpulan perangkat lunak yang dibutuhkan untuk pengembangan sistem agar dapat berjalan sesuai dengan yang diinginkan. Analisis kebutuhan perangkat lunak pada sistem ini adalah sebagai berikut:

a. *Virtual Machine*
b. *Image Virtual Machine*
c. *Template Instance*
d. *Group Instance*
e. *Cloud DNS*
f. *Load Balancer*
g. *Cloud Storage*
h. *SQL Server*

3.2.2 Analisis Kebutuhan Input

Pada tahap ini dilakukan analisis terhadap kebutuhan input apa saja yang dibutuhkan dalam pembuatan sistem. Input yang dibutuhkan dalam pengembangan sistem ini adalah sebagai berikut:

a. *Cloud* Storage yang meliputi semua media seperti *image*, video dll agar memperingan pekerjaan *server* utama.
b. *Cloud* SQL nantinya *database*-nya akan ditempatkan di *cloud* sql karena jika diletakkan di *server* utama akan memakan banyak *resource* dan memperberat kerja *server* utama
c. Pada *server* hanya ada *source* karena semua *storage* dan *database* diletakkan di tempat yang terpisah dengan tujuan untuk memperingan kerja sever

3.2.3 Analisis Kebutuhan Output

Pada tahap ini dilakukan analisis kebutuhan output informasi apa saja yang akan diberikan kepada pengguna. Output informasi yang diberikan pada pengguna dalam sistem ini adalah sebagai berikut:

CPU *Usage:* Pada penelitian ini menggunakan CPU telah ditentukan pada bagian *backend virtual machine*. Pemilihan *threshold* 40% dilakukan karena spesifikasi mesin *virtual* yang digunakan mempunyai spesifikasi minimum maka dari itu dipilih 40% sebagai *threshold* penggunaan CPU. Hal tersebut dilakukan jika menggunakan *threshold* melebihi 40% maka akan sulit mencapai batas yang telah ditentukan mengingat spefisikasi mesin *virtual* menggunakan ram 2gb dan 1cpu. Jika penggunaan backend telah melebihi dari 40% maka akan memperbanyak *virtual machine* secara otomatis. Hal ini karena pada saat melakukan konfigurasi pada fitur *instance group* dan *load balancing* yang telah disediakan oleh Google Cloud Platform telah diatur jika penggunaan CPU melebihi 40% maka akan memperbanyak *virtual machine* secara otomatis.

3.3 Perancangan Sistem

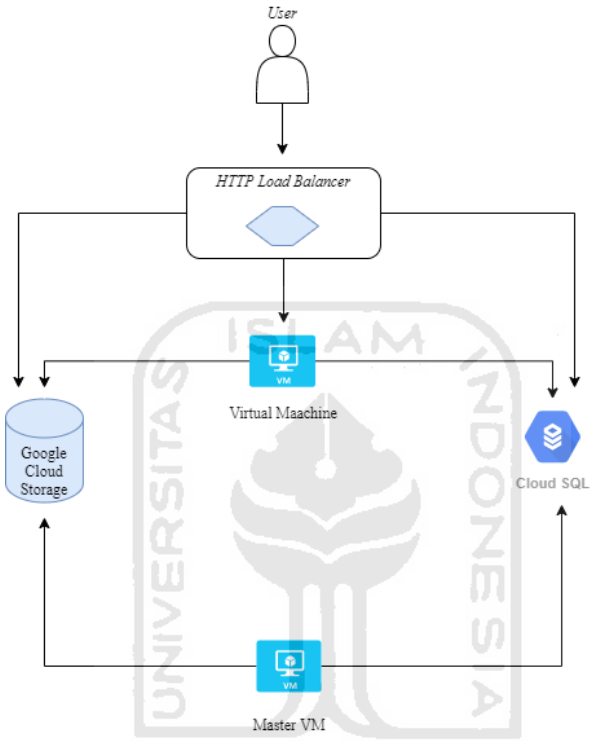
Setelah melakukan pencarian informasi, tahap selanjutnya adalah melakukan perancangan sistem yang akan dibuat berdasarkan informasi yang didapat pada saat melakukan analisis kebutuhan. Nantinya, sistem yang akan dibuat harus sesuai dengan analisis kebutuhan.

Dalam pengembangan dan perancangan optimasi performa web menggunakan *google cloud* CDN dan *load balancing* ini membutuhkan beberapa tahap perancangan. Perancangan

yang dilakukan dalam pembangunan sistem ini antara lain adalah perancangan arsitektur jaringan.

3.3.1 Perancangan Desain Arsitektur Jaringan

Perancangan desain arsitektur jaringan adalah tahap perancangan sistem yang akan dikembangkan dan implementasinya pada jaringan yang digunakan.



Gambar 3.2 Perancangan Arsitektur Jaringan

Gambar 3.2 untuk proses awal ketika *user* akan melakukan akses kepada web maka akan diarahkan terlebih dahulu ke HTTPS *load balancer* yang telah dibuat. Di sini HTTPS *load balancer* berfungsi sebagai frontend untuk menampilkan data. Setelah melalui HTTPS *load balancer* maka akan diarahkan ke *virtual machine*. Kemudian setelah melewati *virtual machine* selanjutnya akan melewati *sql server* pada *sql server* berisi data-data berupa directory dari *virtual machine*. Nantinya *virtual machine* tersebut akan mengambil konten yang akan diakses oleh *user* dari *google cloud storage*. *Sql server* digunakan untuk menyimpang database dari web yang dibuat sedangkan google cloud storage menyimpan *image, file*, video, *assets* dll. *Master virtual machine* berperan sebagai *server development* sehingga lebih memudahkan ketika akan membuat mesin *virtual* baru karena tidak udah melakukan konfigurasi dari awal karena sudah mempunyai *server development*.

3.3.2 Perancangan Web *Server*

Tahap ini dimulai dengan melakukan *install* web *server* yang akan digunakan yaitu Nginx. Web *server* digunakan untuk menyimpan semua data seperti HTML dokumen, gambar, *file CSS stylesheets*, dan file JavaScript. Dari sisi software, fungsi web *server* adalah sebagai pusat kontrol untuk memproses permintaan yang diterima dari browser.

Semua yang berhubungan dengan web biasanya juga berhubungan dengan web *server*, karena tugas web *server* adalah mengatur semua komunikasi yang terjadi antara *browser* dengan *server* untuk memproses sebuah web.

3.3.3 Perancangan Web menggunakan Wordpress

Tahap ini adalah perancangan web menggunakan wordpress, wordpress dipilih karena lebih fleksibel dibandingkan dengan blogger. Pada wordpress dapat melakukan konfigurasi secara manual sehingga memudahkan untuk membuat web sesuai dengan yang diinginkan. Selain pembuatan website tahapan ini juga melakukan penyambungan domain dengan wordpress sehingga akan terbentuk web dengan domain wartaholic.web.id. Penggunaan domain dilakukan agar mempermudah untuk mengakses web tanpa perlu memasukkan *ip* dari web yang akan diakses.

3.4 Pengujian Sistem

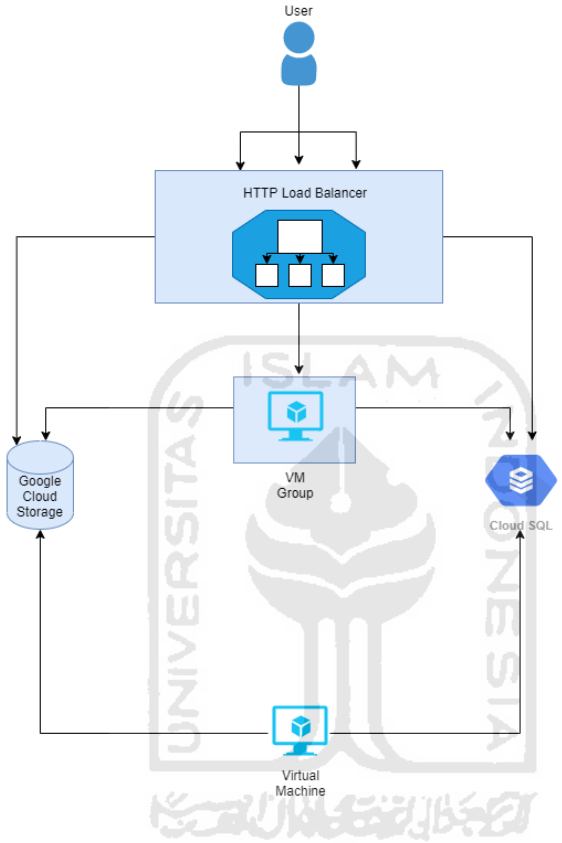
Tahapan ini bisa dikatakan final dalam pembuatan sebuah sistem. Nantinya sistem akan diuji menggunakan *apache benchmark*. *Apache benchmark* digunakan karena merupakan salah satu metode yang paling efektif digunakan untuk melakukan pengujian terhadap web *server*. Apache Bench (AB) alat standar untuk menguji performa arsitektur web *server* secara konkret. Apache Benchmark akan mengirimkan jumlah total permintaan HTTP ke *server* web yang sedang diuji sesuai dengan *command* ab -n 1000 -c 100 https://wartaholic.web.id/ yang dimasukkan. Jumlah angka pada variabel dapat disesuaikan dengan keinginan *developer* yang akan melakukan testing.

```
ab -n 1000 -c 100 https://wartaholic.web.id/
```

Gambar 3.3 Command Apache Benchmark

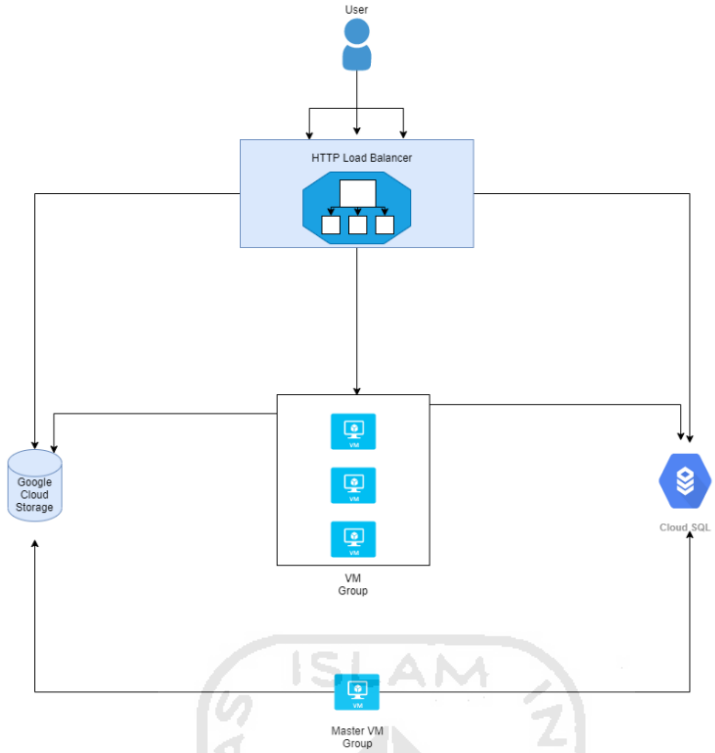
Gambar 3.3 *adalah penggunaan command apache benchmark* untuk melakukan pengujian terhadap sistem yang dibuat. Perintah -n menunjukkan berapa kali user mengakses url tersebut, sedangkan perintah -c menunjukkan jumlah user mengakses URL secara bersamaan, kemudian diikuti dengan URL dari aplikasi yang ingin diuji. Dari *output* di atas menunjukkan performa aplikasi web yang kita miliki, misal waktu yang dibutuhkan untuk melayani permintaan, besaran data yang ditransfer, time per *request*, dan lain sebagainya.

Hal yang harus diperhatikan dalam pengujian adalah berapa hasil dari *Requests* per second karena ini menunjukkan seberapa banyak pengunjung yang dilayani dalam satu detik, perlu diingat bahwa satu pengunjung dapat membuka beberapa koneksi/permintaan konten sekaligus. Angka pada *Percentage of the requests served within a certain time* (ms), ini menandakan seluruh permintaan koneksi berhasil dilakukan dalam waktu berapa milidetik



Gambar 3.4 *User Testing*

Gambar 3.4 adalah skenario pengujian pertama yang dilakukan terlihat bahwa *virtual machine* tidak bertambah karena trafik yang masuk masih mampu ditampung oleh *server* maka dari itu tidak terjadi penambahan *virtual machine*. Terjadi 500 *request* dan *user* yang masuk sebanyak 100 *user*. Dengan *command* ab -n 500 -c 100 https://wartaholic.web.id/. Jumlah tersebut masih mampu untuk ditampung oleh *server* sehingga tidak perlu melakukan *load balancing.*



Gambar 3.5 *User Testing Overload*

Gambar 3.5 adalah skenario kedua yang dilakukan pada tahap pengujian pada Tabel 3.1 terlihat pada skenario ke-15 bahwa web akan diakses sebanyak 300.000 dan *user* yang masuk sebanyak 3.000. Hal tersebut menyebabkan *server* tidak mampu untuk menampung *request* yang masuk sehingga terjadi proses *load balancing*. *Virtual Machine* akan memperbanyak secara otomatis jika pemakaian CPU sudah melebihi ambang batas yang ditentukan.

Pada penelitian ini akan terdapat beberapa skenario seperti yang terlihat pada Tabel 3.1

Pengujian akan dilakukan manual dengan menggunakan *virtual private server (VPS)* yang telah disiapkan. Dengan menggunakan command *apache benchmark* yang variabelnya telah ditentukan yaitu CPU dan trafik dan juga skenario yang telah dibangun maka dari itu terbentuk 15 pengujian untuk melihat apakah *server* dapat menampung traffic yang masuk atau tidak. Jika tidak maka *server* akan melakukan *load balancing* dengan memperbanyak *virtual machine* secara otomatis

Tabel 3.1 Skenario Pengujian

Pengujian ke-	Skenario	Hasil yang diharapkan
1	web akan diakses sebanyak 500 *request* dan *user* yang masuk berjumlah 100 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
2	web akan diakses sebanyak 700 *request user* dan yang masuk berjumlah 200 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
3	web akan diakses sebanyak 900 *request user* dan yang masuk berjumlah 300 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
4	web akan diakses sebanyak 1000 *request* dan *user* yang masuk berjumlah 400 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
5	web akan diakses sebanyak 5000 *request* dan *user* yang masuk berjumlah 500 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
6	web akan diakses sebanyak 7000 *request* dan *user* yang masuk berjumlah 600 u*ser*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
7	web akan diakses sebanyak 9000 *request* dan *user* yang masuk berjumlah 700 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
8	web akan diakses sebanyak 50000 *request* dan *user* yang masuk berjumlah 500 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
9	web akan diakses sebanyak 60000 *request* dan *user* yang masuk berjumlah 600 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
10	web akan diakses sebanyak 700000 *request* dan *user* yang masuk sebanyak 700 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
11	web akan diakses sebanyak 80000 *request* dan *user* yang masuk berjumlah 800 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
12	web akan diakses sebanyak 90000 *request* dan *user* yang masuk berjumlah 900 *user*	Sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing*
13	web akan diakses sebanyak 100000 *request* dan *user* yang masuk berjumlah 1000 *user*	Sistem mampu untuk menampung trafik yang masuk dan diharapkan melakukan *load balancing*
14	web akan diakses sebanyak 200000 *request* dan *user* yang masuk berjumlah 2000 *user*	Sistem tidak mampu untuk menampung trafik yang masuk dan diharapkan melakukan *load balancing*

Pengujian ke-	Skenario	Hasil yang diharapkan
15	web akan diakses sebanyak 300000 *request* dan *user* yang masuk berjumlah 3000 *user*	Sistem tidak mampu untuk menampung trafik yang masuk dan diharapkan melakukan *load balancing*

Tabel 3.1 adalah skenario pengujian yang akan dilakukan untuk melakukan pengujian nantinya web akan diakses dengan trafik yang sudah ditentukan untuk melihat apakah web masih mampu untuk menampung trafik tersebut atau tidak. Jika tidak mampu untuk menampung trafik yang diberikan diharapkan sistem akan memperbanyak mesin *virtual* secara otomatis. Pada bagian *load balancing* sudah diatur jika penggunaan CPU melebihi dari 40% maka secara otomatis sistem akan memperbanyak *virtual machine*.

BAB IV
KONFIGURASI, IMPLEMENTASI DAN PENGUJIAN

4.1 Pembuatan NGINX

Tahap ini adalah langkah untuk meng*install* nginx sebagai web *server*.

```
Sudo apt update
```

Gambar 4.1 Update Package

Gambar 4.1 untuk mengunduh informasi paket dari semua sumber yang dikonfigurasi.

```
Sudo apt upgrade
```

Gambar 4.2 Upgrade Package

Gambar 4.2 untuk memperbarui dari internet, untuk menginstal pembaharuan yang tersedia dari semua paket yang saat ini diinstal pada sistem dari sumber yang dikonfigurasi melalui file *sources list*.

```
Sudo apt install nginx
```

Gambar 4.3 Install Nginx

Gambar 4.3 untuk meng*install* web *server* pada *operating system* ubuntu yang telah dibuat.

```
Nginx -v
```

Gambar 4.4 Cek Nginx

Gambar 4.4 untuk melakukan pemeriksaan nginx apakah telah berjalan dengan baik atau tidak.

4.2 Pembuatan web dengan wordpress

```
sudo apt install software-properties-common
sudo add-apt-repository ppa: ondrej/php
sudo apt update
sudo apt install php7.3-fpm php7.3-common php7.3-mysql php7.3-xml php7.3-xmlrpc php7.3-curl php7.3-gd php7.3-imagick php7.3-cli php7.3-dev php7.3-imap php7.3-mbstring php7.3-opcache php7.3-soap php7.3-zip unzip -y
```

Gambar 4.5 Install PHP

Gambar 4.5 digunakan untuk melakukan installasi php pada wordpress yang akan dibuat.

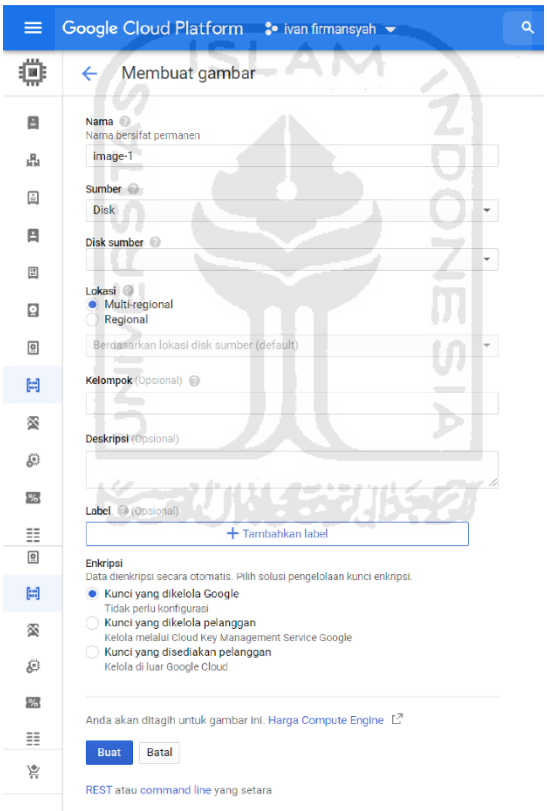
```
sudo nano /etc/php/7.3/fpm/pool.d/www.conf

user = username
group = username
listen. owner = username
listen. group = username
```

Gambar 4.6 konfigurasi *username*

Gambar 4.6 digunakan untuk melakukan konfigurasi pengaturan *username*, *group* dll.

4.3 Pembuatan *Image Virtual Machine*

Pada tahapan ini pembuatan *image* berfungsi untuk menyimpan semua konfigurasi, metadata, izin, dan data dari satu atau beberapa *disk* untuk *instance virtual machine* yang berjalan di *Compute Engine*. *Instance virtual machine* yang digunakan untuk membuat *image machine* disebut sebagai *instance* sumber.

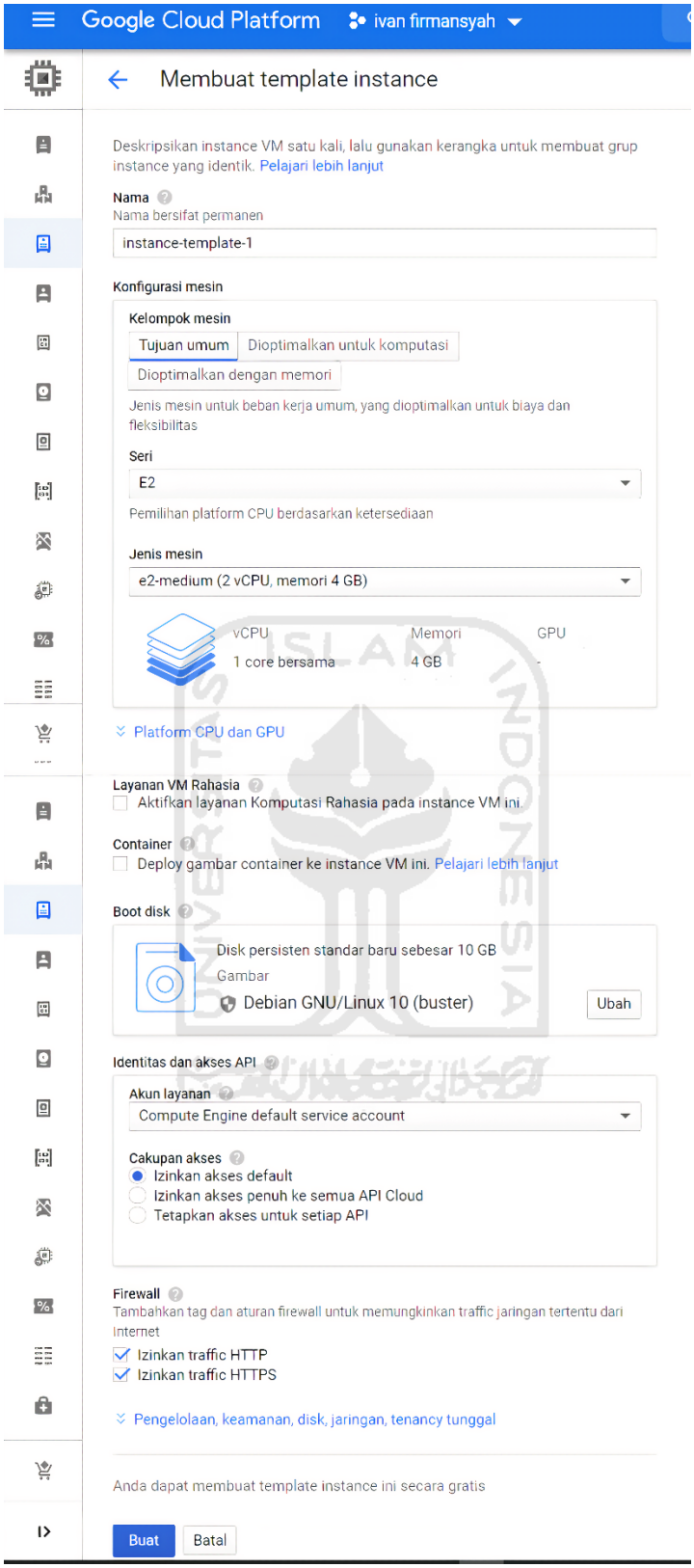


Gambar 4.7 Pembuatan Image *Virtual* Machine

Gambar 4.7 dapat dilihat pembuatan *image virtual machine* terdapat menu nama, sumber, *disk*. Pada menu sumber merupakan menu untuk memilih kita ingin menggunakan sumber yang mana dan menu *source disk* penulis memilih untuk menggunakan disk yang telah dibuat pada menu *template instance*. Karena pada *disk* yang telah dibuat terdapat konfigurasi untuk pembuatan *load balancing*. Untuk lokasi dipilih yang multiregional karena bisa di mana saja sedangkan jika memilih yang regional maka hanya di satu wilayah tertentu.

4.4 **Pembuatan *Template Instance***

Tahap ini pembuatan template instance bertujuan untuk menentukan jenis mesin, gambar *boot disk* atau gambar container, label, dan properti *instance* lainnya. kemudian dapat menggunakan *template instance* untuk membuat *Manage Instance Group* atau membuat *virtual machine* individual. *Template instance* adalah cara yang mudah untuk menyimpan konfigurasi *instance* VM sehingga dapat menggunakannya nanti untuk membuat VM atau *group* VM.

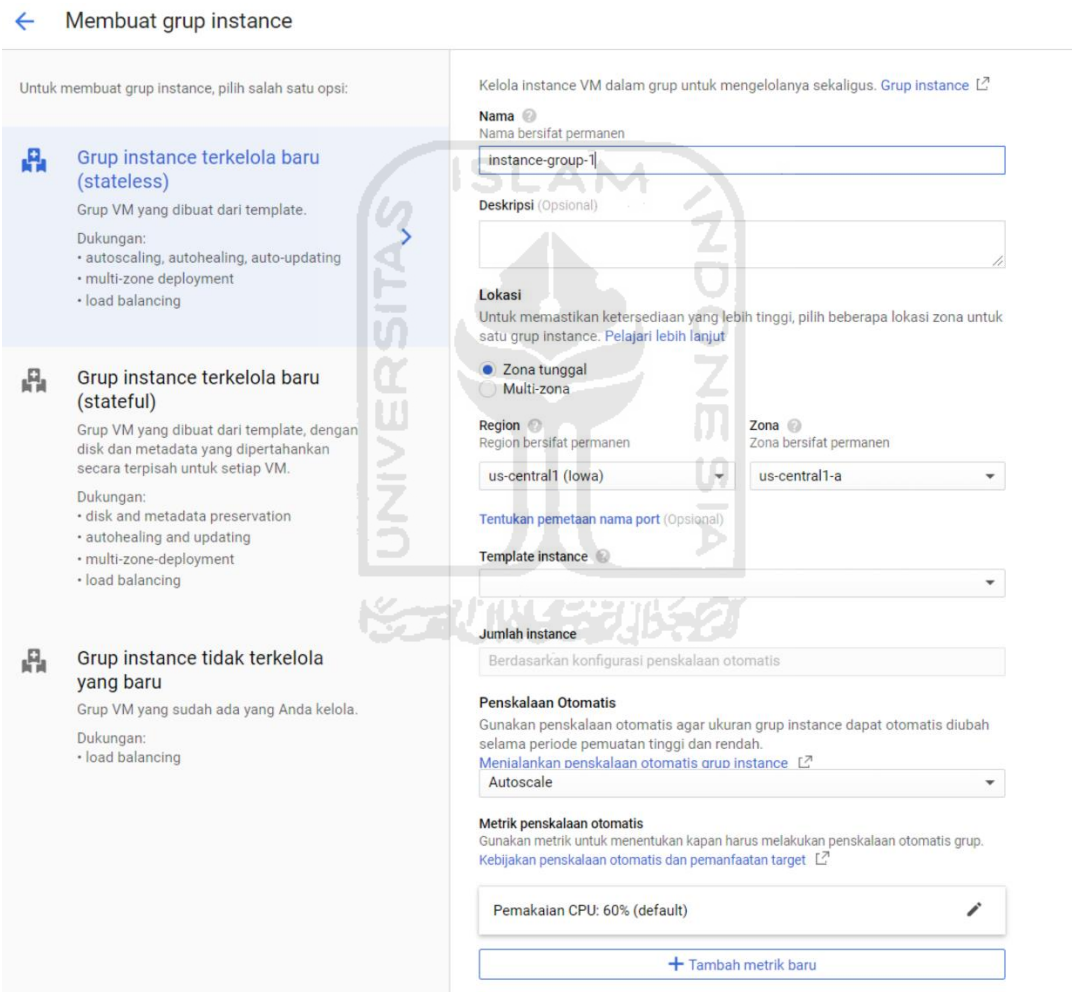


Gambar 4.8 Pembuatan Template Instance

Gambar 4.8 adalah proses pembuatan *Template Instance* terdapat menu nama, konfigurasi mesin, jenis mesin, jenis disk dll. *Template instance* adalah sumber daya yang dapat digunakan

untuk membuat instance *Virtual Machine* (VM) dan *Manage Instance Group* (MIG). *Template instance* adalah sumber daya global yang tidak terikat ke zona atau wilayah. Label yang ditentukan dalam *template instance* diterapkan ke semua *instance* yang dibuat dari *template instance* tersebut. Nantinya dapat memilih disknya disini penulis memilih menggunakan sistem operasi ubuntu. Dapat juga melakukan pengaturan terhadap *firewall* disini penulis memilih untuk mengizinkan trafik HTTP dan HTTPS.

4.5 **Pembuatan *Group* Instance**



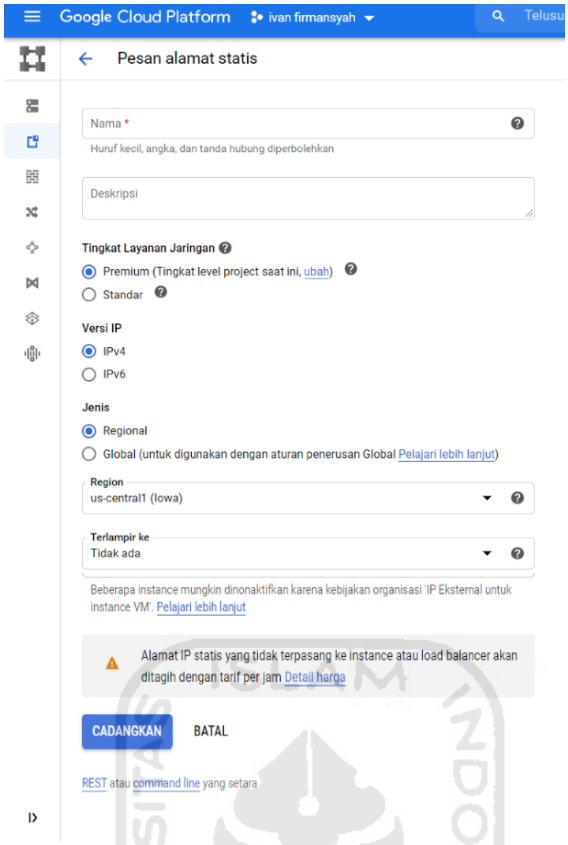
Gambar 4.9 Pembuatan *Group Instance*

Gambar 4.9 adalah proses pembuatan *Group Instance* terdapat beberapa menu antara lain nama, lokasi, *region*, *template instance* dll. Untuk pemilihan zona dipilih multi zona karena jika memilih zona tunggal maka hanya bisa menggunakan zona yang dipilih contohnya jika memilih zona tunggalnya di Jakarta maka *virtual machine* akan berada di Jakarta saja namun jika memilih multi zona *virtual machine* akan berada di beberapa tempat dan tidak hanya

terpatok pada satu tempat saja. Setelah itu akan memilih instance template pemilihan *instance template* berdasarkan pada *instance* yang telah dibuat sebelumnya. Karena pada *instance template* terdapat konfigurasi untuk pembuatan *load balancing*. Sedangkan untuk penggunaan CPUnya akan diatur sebesar 40%. Pemilihan *threshold* 40% dilakukan karena spesifikasi mesin *virtual* yang digunakan mempunyai spesifikasi minimum maka dari itu dipilih 40% sebagai threshold penggunaan CPU. Hal tersebut dilakukan jika menggunakan *threshold* melebihi dari 40% maka akan sulit mencapai batas yang telah ditentukan mengingat spefisikasi mesin *virtual* menggunakan ram 2gb dan 1cpu. Jika penggunaan *backend* telah melebihi dari 40% maka akan memperbanyak secara otomatis. Hal ini karena pada saat melakukan konfigurasi pada fitur *instance group* dan *load balancing* yang telah disediakan oleh Google Cloud Platform telah diatur jika penggunaan CPU melebihi 40% maka akan memperbanyak *virtual machine* secara otomatis.

4.6 Pembuatan *IP* Statis untuk Domain Sekaligus Frontend

Sebelum melakukan konfigurasi pembuatan *load balancing* tahapan yang harus dilakukan adalah membuat *IP* statis.



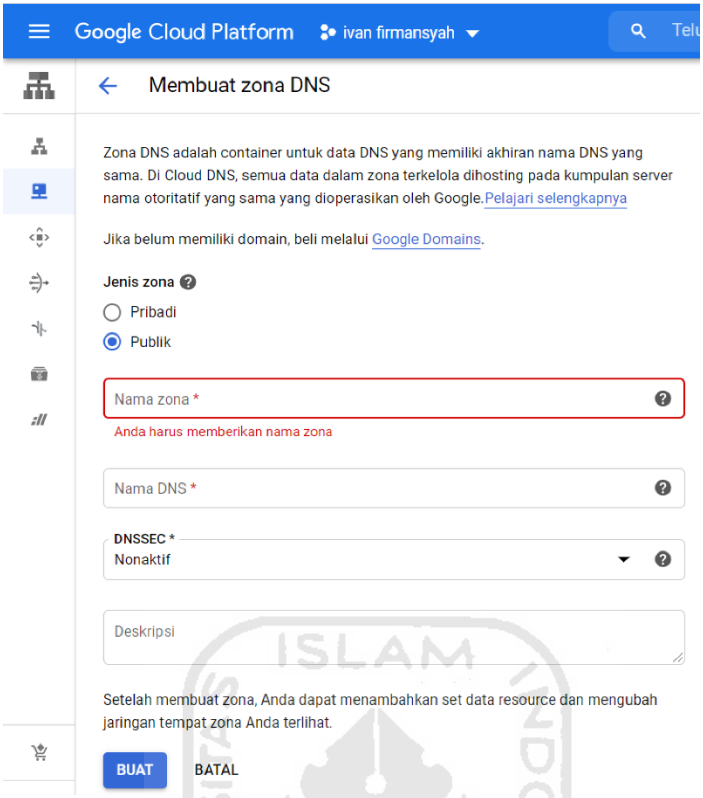
Gambar 4.10 Pembuatan IP statis

Gambar 4.10 terdapat beberapa menu antara lain nama, deskripsi, tingkat layanan, versi *IP* dll. Pada penelitian ini penggunaan *ip* statis berfungsi sebagai perantara dikarenakan *virtual machine* yang memiliki *ip* yang dinamis yang artinya *ip* tersebut akan berubah-ubah setelah *virtual* machine mati. Maka dari itu *IP* statis ini diperlukan agar *user* dapat mengakses web yang kita buat. Pada penelitian ini penulis memilih menggunakan tingkat layanan jaringan yang premium karena tingkat premium memberikan lalu lintas Google Cloud Platform melalui jaringan global Google yang disediakan dengan baik, latensi rendah, dan sangat andal.

Selain menggunakan tingkat layanan premium penulis juga menggunakan *IP*V4 dan jenis penulis memilih regional. Alamat *IP* regional digunakan oleh *instance* VM dengan satu atau beberapa antarmuka jaringan atau dengan *load balancer* jaringan.

4.7 Pembuatan *Cloud* DNS

Domain Name *Server* adalah Penerjemah dari yang awalnya adalah *ip* menjadi sebuah url hal ini memudahkan pengguna untuk dapat mengakses web tanpa harus menghafal seluruh alamat *ip* web yang akan diakses. Tujuan penggunaan *cloud* dns adalah untuk menstabilkan *latency.*

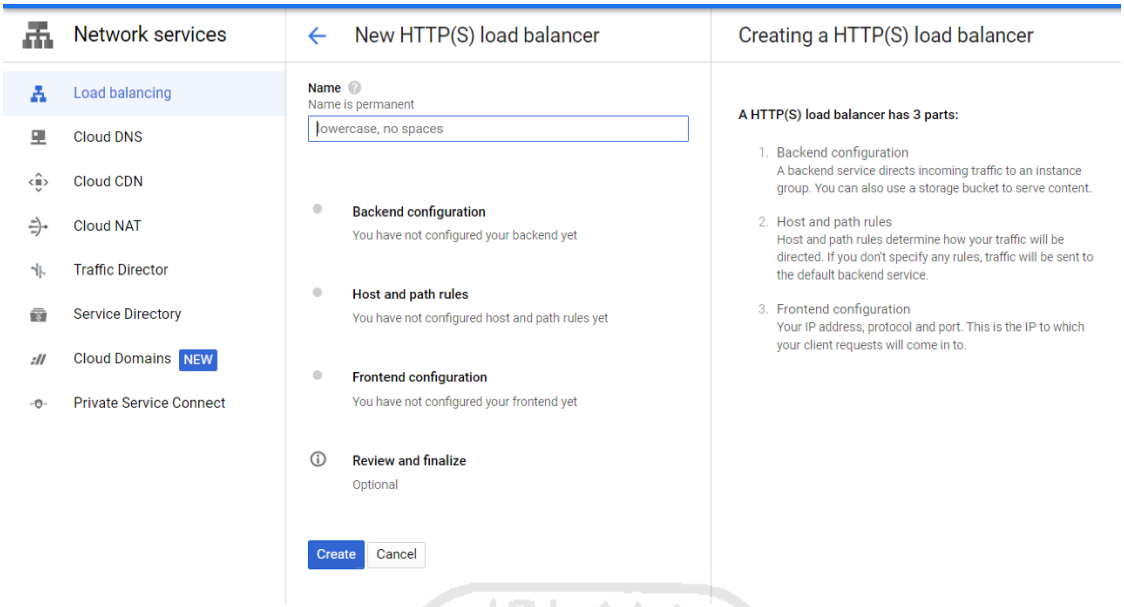


Gambar 4.11 Pembuatan Cloud DNS

Gambar 4.11 terdapat beberapa menu antara lain jenis zona, nama zona dll. Pada jenis zona perbedaan antara zona publik dan private adalah Zona DNS pribadi berisi data DNS yang hanya dapat dilihat secara internal dalam jaringan GCP sedangkan zona publik dapat dilihat di internet.

4.8 **Konfigurasi *Load balancing***

Dengan menggunakan *Cloud Load balancing*, dapat menyajikan konten sedekat mungkin dengan pengguna di sistem yang dapat merespons lebih cepat daripada tidak menggunakan *load balancing*. Google Cloud menawarkan fitur *load balancing* berikut: Satu alamat *IP* yang berfungsi sebagai *frontend*, Penskalaan otomatis dari *backend*, *Load Balancer external* saat pengguna menjangkau aplikasi dari internet, teknik *load balancing* internal saat *client* berada di dalam Google *Cloud*, Teknik *load balancing* region saat aplikasi tersedia di satu *region*.



Gambar 4.12 Konfigurasi *Load Balancer*

Gambar 4.12 terdapat beberapa menu antara lain konfigurasi *backend*, pengaturan *host* dan jalur, konfigurasi *frontend*. Pada konfigurasi *backend* terdapat beberapa menu antara lain yaitu nama, jenis *backend*, *protocol* dll. Setelah melakukan konfigurasi pembuatan *backend* selanjutnya adalah pembuatan *host* dan jalur. Pembuatan *host* meliputi *host*, jalur dan *backend*. Setelah membuat backend dan *host* selanjutnya adalah konfigurasi frontend, terdapat beberapa menu antara lain nama, *protocol*, tingkat layanan jaringan, versi *IP* dan port nantinya *frontend* tersebut yang akan menampilkan data yang diminta oleh *user*.

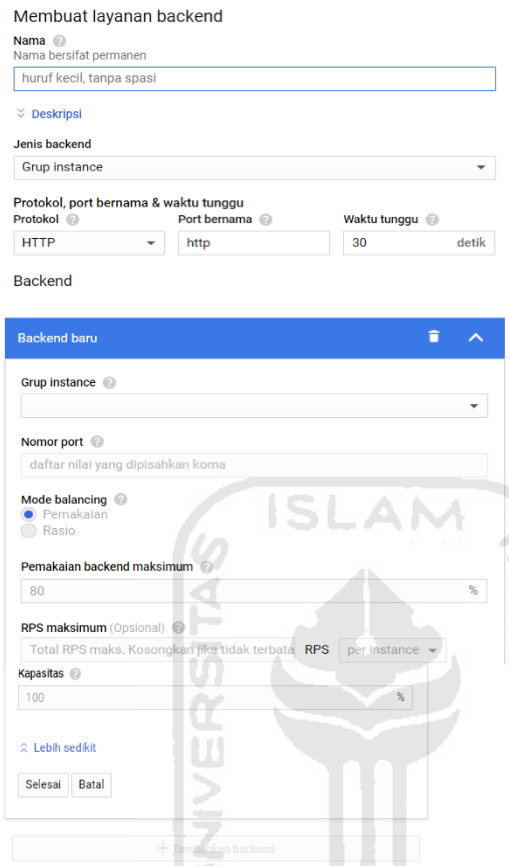
4.8.1 Konfigurasi Pembuatan Backend

Layanan backend memberikan informasi konfigurasi ke *load balancer*. *Load Balancer* HTTP(S) eksternal harus memiliki setidaknya satu layanan *backend* dan dapat memiliki beberapa layanan backend. Load Balancer memakai informasi yang berasal dari layanan backend untuk mengarahkan lalu lintas masuk dari backend pertama atau beberapa backend yang terpasang.

Layanan *backend* berupa *group instance* atau *network endpoint group* (NEG), tetapi bukan gabungan antara keduanya hanya salah satu saja. Saat menambahkan *group instance backend* atau *network endpoint group*, dapat menentukan metode *load balancer*, mendistribusikan permintaan dan kapasitas target.

Load balancing HTTP (S) memiliki fitur *load balancing auto scaler*, yang memungkinkan pengguna melakukan penskalaan otomatis pada *group* instance dalam layanan backend. Pengguna bisa mengaktifkan penghentian koneksi pada layanan *backend* untuk memastikan

terjadinya gangguan yang seminimal mungkin bagi pengguna saat sebuah *instance* yang melayani lalu lintas dihentikan, dihapus secara manual, atau dihapus oleh penskala otomatis.

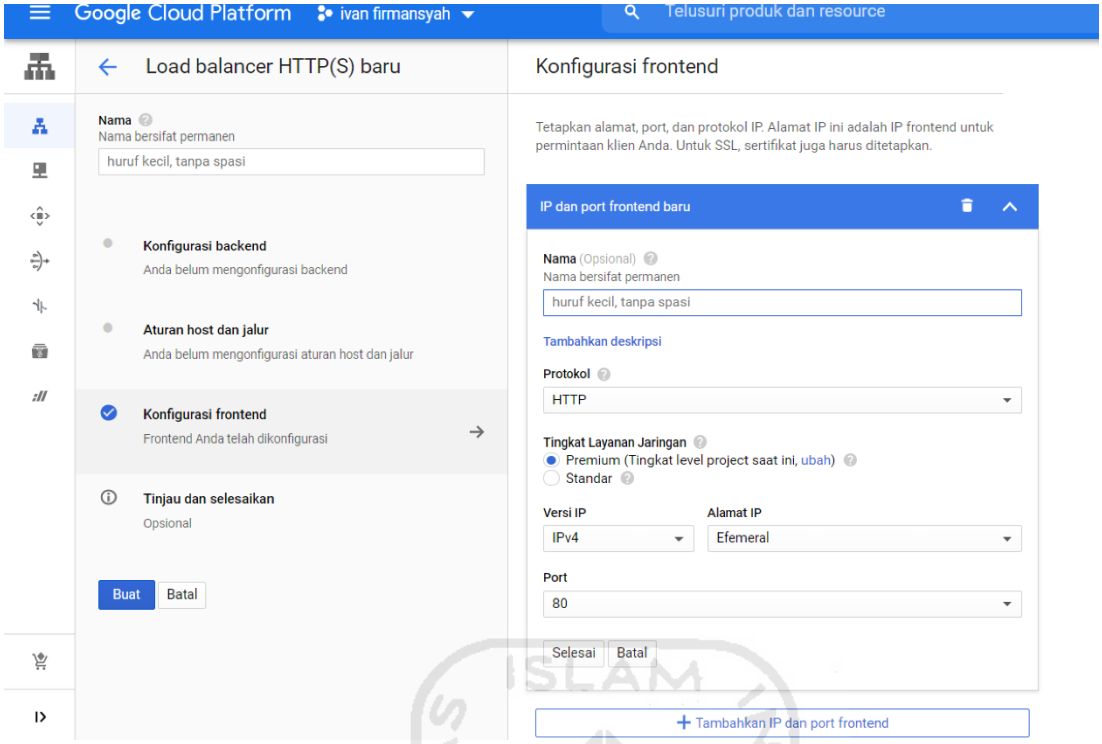


Gambar 4.13 Pembuatan *Backend*

Gambar 4.13 adalah proses pembuatan backend ada beberapa menu antara lain nama, jenis *backend*, protocol dll. Pada menu *group instance* dapat memilih untuk menggunakan *group instance* yang sebelumnya telah dibuat. Selanjutnya dapat mengatur penggunaan *backend*, pemakain *backend* dapat disesuaikan misalkan jika mengatur pemakaian *backend* 40% maka jika pemakaiannya telah melebihi dari 40% maka otomatis akan membuat *backend* baru. Selain melakukan pengaturan pada pemakaian *backend* dapat juga langsung mengaktifkan CDN fungsi CDN di sini adalah untuk mempercepat distribusi konten kepada *user*.

4.8.2 Konfigurasi Pembuatan Frontend

Konfigurasi *frontend* ini berguna untuk pembuatan *ip* statis sebagai jembatan antara *user* dan *server*. Agar user dapat mengakses web dan menerima *content* dari web tanpa mengalami perlambatan. Dikarenakan *virtual machine* memiliki *ip* dinamis maka diperlukan *ip* statis sebagai jembatan antara *user* dan *server*.



Gambar 4.14 Pembuatan *Frontend*

Gambar 4.14 beberapa menu antara lain nama, protokol, tingkat layanan jaringan, versi *IP* dan *port*. Pada penelitian ini penulis menggunakan protokol HTTP dan menggunakan tingkat layanan premium yang berguna untuk memaksimalkan fitur yang dimiliki oleh Google Cloud Platform. Pembuatan *frontend* di sini adalah untuk menampilkan konten yang adal di web, sedangkan untuk *IP* penulis menggunakan *IP*V4 dan port 80. Port 80 adalah jenis port di dalam sebuah jaringan yang berfungsi untuk mengkoneksikan web *server*. Web *server* sendiri merupakan sebuah *server* yang dibangun guna memenuhi kebutuhan sebuah web atau situs tertentu. Untuk dapat menghubungkan koneksi antara *client* dengan web *server*, maka port 80 ini dibutuhkan.

4.9 Pengujian Sistem

Pada pengujian ini akan dilakukan 2 skenario yaitu pada saat sistem mampu menampung trafik yang masuk dan tidak melakukan *load balancing* dan yang kedua adalah pada saat sistem tidak mampu menampung trafik dan melakukan *load balancing*. Dilakukan pengujian sebanyak 15 kali. Pada pengujian pertama dan seterusnya sampai ke-15 masing-masing dilakukan 3 kali pengujian *benchmarking* pada *server* kemudian hasil rata-rata dari *benchmarking* tersebut akan dimasukkan ke *time request* dan *cpu load*.

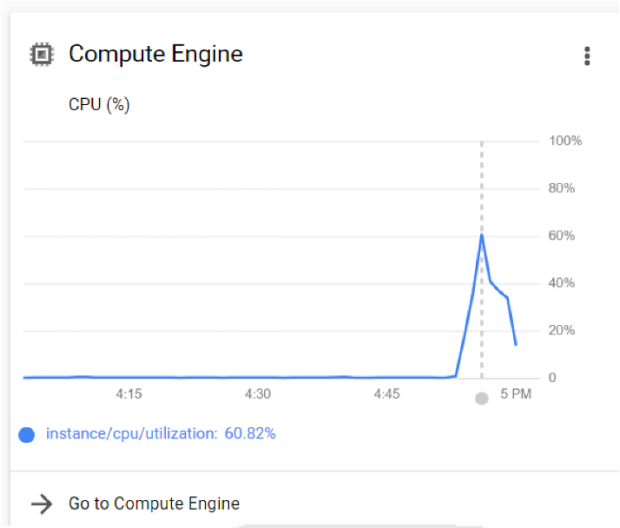
Tabel 4.1 Hasil Pengujian

Pengujian ke-	Skenario	Hasil Pengukuran		
		Time Request	CPU Load	Status
1	web akan diakses sebanyak 500 *request* dan dan *user* yang masuk berjumlah 100	159.681 [ms]	2.00%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
2	web akan diakses sebanyak 700 *request* dan dan *user* yang masuk berjumlah 200	201.327 [ms]	3.71%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
3	web akan diakses sebanyak 900 *request* dan dan *user* yang masuk berjumlah 300 *user*	241.673 [ms]	4.53%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
4	web akan diakses sebanyak 1000 *request* dan *user* yang masuk berjumlah 400 *user*	305.223 [ms]	5.21%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
5	web akan diakses sebanyak 5000 *request* dan *user* yang masuk berjumlah 500	318.488 [ms]	6.25%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
6	web akan diakses sebanyak 7000 *request* dan *user* yang masuk berjumlah 600	323.721 [ms]	6.81%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
7	web akan diakses sebanyak 9000 *request* dan *user* yang masuk berjumlah 700	387.287 [ms]	7.28%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
8	web akan diakses sebanyak 50000 *request* dan *user* yang masuk berjumlah 500	267.685[ms]	7.43%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang

Pengujian ke-	Skenario	Hasil Pengukuran		
		Time Request	CPU Load	Status
9	web akan diakses sebanyak 60000 *request* dan *user* yang masuk berjumlah 600	316.254 [ms]	8.37%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
10	web akan diakses sebanyak 700000 *request* dan *user* yang masuk sebanyak 700 *user*	825.225[ms]	11.34%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
11	web akan diakses sebanyak 80000 *request* dan *user* yang masuk berjumlah 800 *user*	1052.575[ms]	13.17%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
12	web akan diakses sebanyak 90000 *request* dan *user* yang masuk berjumlah 900 *user*	1401.873[ms]	23.40%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
13	web akan diakses sebanyak 100000 *request* dan *user* yang masuk berjumlah 1000 *user*	1601.928[ms]	33.93%	Tidak terjadi *load balancing* karena *server* mampu menampung *request* yang datang
14	web akan diakses sebanyak 200000 *request* dan *user* yang masuk berjumlah 2000 *user*	3834.542[ms]	46.26%	Terjadi *load balancing* karena *server* tidak mampu menampung *request* yang datang
15	web akan diakses sebanyak 300000 *request* dan *user* yang masuk berjumlah 3000 *user*	4314.850[ms]	60.82%	Terjadi *load balancing* karena *server* tidak mampu menampung *request* yang datang

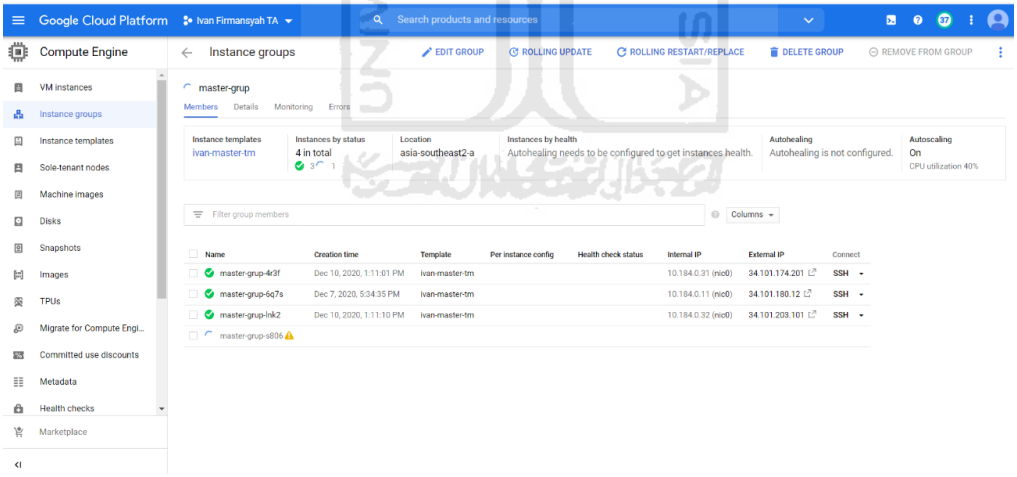
Tabel 4.1 adalah hasil pengujian yang dilakukan oleh penulis sebanyak 15 kali dari 15 kali pengujian diperoleh hasil faktor yang mempengaruhi penggunaan CPU ada 2 aspek antara lain *request* ke web dan *user* yang mengakses web secara bersamaan. Hal tersebut terlihat pada kolom *time request* dan CPU *load* pada tabel tersebut jika web mendapatkan *request* dan *user*

yang mengakses web secara bersamaan dan penggunaan CPU telah melebihi batas yang ditentukan maka akan terjadi *load balancing*.



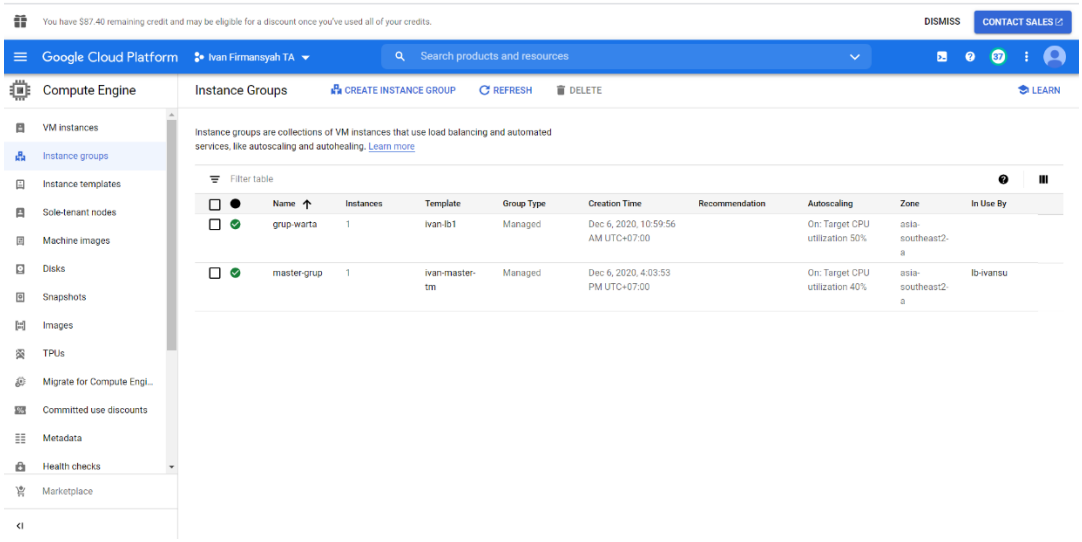
Gambar 4.15 CPU Load

Gambar 4.15 adalah salah satu indikator ketika web mendapatkan trafik yang besar terlihat pada Gambar 4.15 penggunaan CPU mencapai 60.82% tentunya hal tersebut menyebabkan terjadinya *load balancing*.



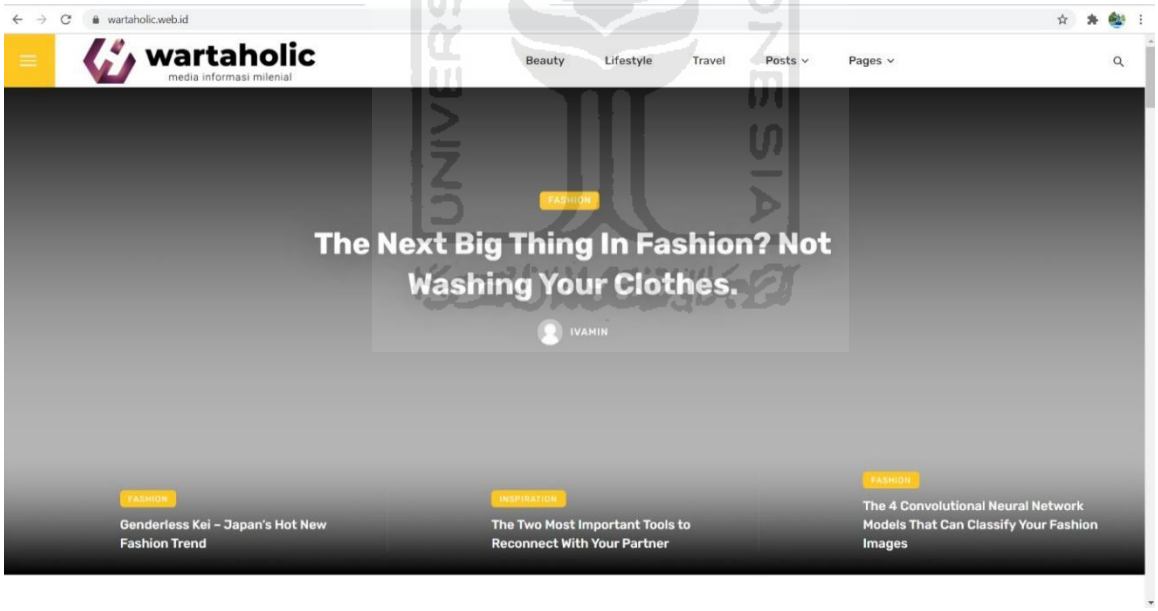
Gambar 4.16 *Virtual* Machine memperbanyak secara otomatis

Gambar 4.16 adalah *virtual machine* yang memperbanyak secara otomatis hal tersebut terjadi karena besarnya trafik yang masuk. terlihat bahwa *virtual machine* yang awalnya hanya 1 akan memperbanyak secara otomatis ketika mendapatkan trafik yang besar. Hal ini dikarenakan penggunaan CPU yang telah melebihi dari 40%. Jika pemakain CPU telah melebihi dari 40% maka akan melakukan pembuatan *virtual machine* secara otomatis.



Gambar 4.17 Jumlah *Virtual Machine* Tetap

Gambar 4.17 terlihat bahwa jumlah *virtual machine* tetap dan tidak bertambah jumlahnya hal ini karena pemakaian CPUnya tidak lebih dari 40% maka dari itu *server* masih mampu untuk menampung trafik yang masuk tanpa harus memperbanyak *virtual machine.*



Gambar 4.18 Testing CDN

Gambar 4.18 adalah testing content *network* delivery (CDN) untuk melakukan pengecekan apakah CDN sudah aktif atau belum aktif maka dilakukan pengujian dengan mengakses web yang telah dibuat. Pada Gambar 4.18 terlihat bahwa web sudah dapat diakses dan dapat menampilkan konten yang berarti CDN yang dikonfigurasi telah aktif. Fungsi CDN di sini adalah menjadi *frontend* dan untuk memastikan bahwa content dapat terkirim dengan cepat dan efektif.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan implementasi dan pengujian menggunakan metode *apache benchmark* maka diperoleh kesimpulan dari penelitian "PEMANFAATAN GOOGLE *CLOUD* DAN DAN TEKNIK *LOAD BALANCING* UKTUK OPTIMALISASI PERFORMA AKSES HALAMAN WEB" sebagai berikut:

Berdasarkan pengujian menggunakan metode *apache benchmark* yang telah dilakukan dapat ditarik kesimpulan bahwa *load balancing* CPU menggunakan Google Cloud Platform telah berhasil dibuat untuk menyeimbangkan beban *server*. Caranya adalah dengan melakukan konfigurasi pada fitur *load balancing* yang tersedia oleh Google Cloud Platform dengan menetapkan *threshold* sebesar 40%. Nilai *threshold* tersebut ditentukan berdasarkan spesifikasi *virtual machine* yang dibuat dalam penelitian ini.

5.2 Saran

Pembuatan sistem optimasi web menggunakan Google Cloud Platform dan *load balancing* masih jauh dari sempurna. Maka dari itu, peneliti memberikan saran untuk penelitian atau pengembangan sistem sejenis agar lebih sempurna, yaitu:

1. Untuk ke depannya mungkin bisa menambah *private network* agar bisa bebas melakukan konfigurasi dan menambah fitur tanpa perlu khawatir akan keamanannya
2. Mengintegrasikan sistem supaya jika ingin membuat *virtual machine* baru tidak usah melakukan konfigurasi dari awal lagi. Karena pada saat ini ketika *virtual machine* induk dihapus maka seluruh konfigurasi yang ada akan hilang

DAFTAR PUSTAKA

Agung Nugroho, Widhi Yahya, K. A. (2017). Analisis Perbandingan Performa Algoritma Round Robin dan Least Connection untuk Load Balancing pada Software Defined Network. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *1*(12), 1568–1577.

Alotaibi, A. M., Fahaad Alrashidi, B., Naz, S., & Parveen, Z. (2017). Security issues in Protocols of TCP/IP Model at Layers Level. *International Journal of Computer Networks and Communications Security*, *5*(5), 96–104. www.ijcncs.org

Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In Proceedings of the 2nd international workshop on Software and performance. *Characteristics of Scalability and Their Impact on Performance. In Proceedings of the 2nd International Workshop on Software and Performance*, 195–203.

Chandra, A. Y. (2019). Analisis Performansi Antara Apache & Nginx Web Server Dalam Menangani Client Request. *Jurnal Sistem Dan Informatika (JSI)*, *14*(1), 48–56. https://doi.org/10.30864/jsi.v14i1.248

Gene, E. R. (2018). Implementasi Load Balancing Dengan Dua Isp Menggunakan Metode (Koneksi Ke-N) Dan Per Connection Classifier (Pcc) Pada Mikrotik. *Skripsi. Universitas Sanata Dharma*, 63.

Google. (2021a). *Cloud CDN overview*. https://cloud.google.com/cdn/docs/overview#:~:text=Cloud CDN (Content Delivery Network,deliver content to your users.

Google. (2021b). *Instance groups*. Instance Groups. https://cloud.google.com/compute/docs/instance-groups

Google. (2021c). *Instance templates*. Instance Templates. https://cloud.google.com/compute/docs/instance-templates

Google. (2021d). *Machine Images*. https://cloud.google.com/compute/docs/machine-images

Gupta, M., & Garg, A. (2014). Content Delivery Network Approach to Improve Web Performance: A Review. *International Journal of Advance Research in Computer Science and Management Studies*, *2*(12), 374–385.

Henanda, Y., & Agistira, D. (2020). Implementasi Load Balancing Menggunakan Nginx

Dengan Metode Generate Hash. *Jurnal Teknologi Informasi*, *1*(2020), 2–6.

Jader, O. H., Zeebaree, S. R. M., & Zebari, R. R. (2019). A state of art survey for web server performance measurement and load balancing mechanisms. *International Journal of Scientific and Technology Research*, *8*(12), 535–543.

Kunda, D., Chihana, S., & Sinyinda, M. (2010). Web Server Performance of Apache and Nginx: A Systematic Literature Review. *Computer Engineering and Intelligent Systems*, *8*(2), 43–52. https://iiste.org/Journals/index.php/CEIS/article/view/35842

Meyer, D., & Zobrist, G. (2002). TCP/IP versus OSI. *IEEE Potentials*, *9*(1), 16–19. https://doi.org/10.1109/45.46812

Muliantara, A. (2009). Penerapan Regular Expression Dalam Melindungi Alamat Email Dari Spam Robot Pada Konten Wordpress. *Jurnal Ilmu Komputer*, *2*(1), 16–24.

Mundra, S., & Taeib, T. El. (2015). *Tcp / Ip Protocol Layering*. *3*(1), 415–417.

Prasad, M. R., Naik, R. L., & Bapuji, V. (2013). Cloud Computing : Research Issues and Implications. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, *2*(2). https://doi.org/10.11591/closer.v2i2.1963

Ravali, P. (2013). A Comparative Evaluation of OSI and TCP/IP Models. *International Journal of Science and Research*, *4*(7), 2319–7064. www.ijsr.net

Septandy, A. N., & Fauzi, F. A. (2019). *Implementasi Load Balancing Menggunakan Apache Load Balancer*. *December*, 0–5.

Statista. (2020). *Number of social network users worldwide from 2017 to 2025*. https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/#:~:text=Social media usage is one,almost 4.41 billion in 2025.

Stocker, V., Smaragdakis, G., Lehr, W., & Bauer, S. (2017). The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem. *Telecommunications Policy*, *41*(10), 1003–1016. https://doi.org/10.1016/j.telpol.2017.02.004

Supramana, Prismana, I. G. L. P. E. (2016). Implementasi Load Balancing Pada Web Server Dengan Menggunakan Apache. *Jurnal Manajemen Informatika*, *5*, 117–125.

Yusuf, E., Riza, T. A., Ariefianto, T., & Elektro, F. (2013). *Implementasi Teknologi Load Balancer Dengan*. 11–16.

ISLAM
UNIVERSITAS
INDONESIA