BAB 1
PENDAHULUAN

1.1 Latar Belakang

Pada era perkembangan teknologi digitalisasi saat ini, keberadaan *website* sangat dibutuhkan oleh pengguna *internet* guna memenuhi kebutuhan dalam mencari sebuah informasi serta membatasi pertemuan secara fisik antar individu dalam rangka memutus rantai penyebaran virus COVID-19. *Website* merupakan kumpulan halaman yang menampilkan informasi data teks, data gambar diam atau gerak, data animasi, suara, video dan atau gabungan dari semuanya, baik bersifat statis maupun dinamis yang membentuk suatu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman (*hyperlink*).[1] Singkatnya, *website* terdiri dari beberapa halaman yang saling terhubung untuk menyalurkan informasi melalui jaringan *internet*.

Untuk menjalankan sebuah *website* diperlukan *server* yang berjalan selama 24 jam demi melayani permintaan dari pengguna. *Web server* adalah layanan *server* yang berfungsi menerima permintaan *HTTP* atau *HTTPS* dari klien dengan menggunakan *web browser* dan mengirimkan kembali hasilnya dalam bentuk halaman-halaman *web* yang umumnya berbentuk dokumen html dan format dokumen *web* lainnya.[2] Sebuah *web server* biasanya berjalan diatas Sistem Operasi berbasis Linux pada sebuah perangkat keras yaitu *server* dan ditempatkan pada lingkungan institusi atau perusahaan yang membutuhkan layanan berbasis *web.*

Permasalahan yang sering dialami ketika banyak pengunjung mengakses ke sebuah *website* adalah *server* yang berperan sebagai *web server* tidak mampu menangani permintaan sehingga layanan tidak dapat diproses. Beberapa penyebab *server failure* adalah putusnya sumber daya listrik, kegagalan perangkat keras, sistem operasi yang *crash* dan kegagalan jaringan.[3] Salah satu cara untuk mengatasi kegagalan adalah melakukan mekanisme *load balancing. Load balancing* adalah proses distribusi beban terhadap sebuah servis yang ada pada sekumpulan *server* atau perangkat jaringan ketika ada permintaan dari pengguna.[4]

Teknologi saat ini yang cukup banyak digunakan untuk menjalankan sebuah layanan *(service)* di dalam *server* yaitu menggunakan *Container*. *Container* merupakan perangkat lunak virtualisasi jenis *Operating system* level *virtualization* berbasis LXC (Linux *Container*) yang memungkinkan pengembang aplikasi membuat, menguji dan menjalankan aplikasi dalam sebuah lingkungan yang berbeda, terisolasi dan fleksibel. Virtualisasi mempunyai tujuan akhir membuat IT infrastuktur memiliki optimalisasi dan *portability*.[5] Sehingga dapat memudahkan para pengembang dan sistem *administrator* aplikasi berbasis *server*-klien (Aplikasi *Web*) dalam proses pengembangan, uji coba dan pemaketan kode program dan *hosting*. *Container* dikeluarkan oleh beberapa perusahaan seperti, Amazon Elastic *Container* Service (Amazon ECS), Red Hat OpenShift *Container* Platform, dan salah satunya dari Docker.

Docker sangat ringan dan mempunyai mekanisme yang lebih maju jika dibandingkan dengan perangkat lunak virtualisasi berbasis *hypervisor*. Indikasinya adalah adanya efektivitas lebih pada Docker dalam hal penggunaan sumber daya mesin *host*, karena dalam proses pembangunannya Docker akan menjalankan sebuah *Container* menggunakan *base image* dengan metode *file system as a layer*, yang berarti Docker hanya akan menyalin lapisan perubahannya saja untuk dijalankan sebagai duplikasi *container* yang berbeda dengan *base image* yang sama. Berbeda dengan virtualisasi tradisional di mana kita harus menyalin seluruh *container* beserta *kernel* dan *library* yang berada di dalamnya, hal ini mengakibatkan adanya penghematan penyimpanan dan memori pada Docker. Kelebihan yang lain adalah Docker merupakan perangkat lunak yang berlisensi *open source*. Karena itu, IT industri dengan stabil telah mulai mengembangkan Docker yang terinspirasi dari pembuatan *container*isasi.[5]

Untuk menjalankan *web server* berbasis *container* salah satunya menggunakan *Public Cloud.* Penggunaan *Public Cloud* sudah mulai populer digunakan untuk meminimalisir biaya pemeliharaan perangkat. Secara garis besar, *Public Cloud* merupakan tipe komputasi awan yang didedikasikan untuk pengguna secara umum dan luas. *Cloud provider* sebagai penyedia jasa *public cloud* akan melayani fasilitas tersebut dari sebuah data center terpusat, yang digunakan secara

bersama-sama (*resource sharing*) oleh banyak pengguna.[6] *Public Cloud* yang biasa digunakan saat ini adalah keluaran dari Google yaitu Google Cloud Platform. Google Cloud Platform memungkinkan anda membuat, menguji, dan menerapkan aplikasi di infrastruktur google yang amat besar dan dapat diandalkan. Cloud platform menawarkan layanan komputansi, penyimpanan, dan aplikasi untuk solusi *backend*, seluler dan *web* anda. Dengan Google Cloud Platform, anda memasuki dunia jaringan layanan terkelola yang berdedikasi untuk menawarkan kinerja jaringan lokal yang dioptimalkan dan nyaris tak pernah nonaktif.[7]

Fitur Google Cloud Platform yang dapat digunakan untuk melakukan *load balancing* terhadap *Web Server* diatas *Container* bernama *HTTP Load Balancing*. Selain itu, metode *load balancing* yang dapat digunakan diatas Google Platform yaitu NGINX. HTTP *Load Balancing* Google Cloud Platform didasarkan pada teknologi yang dikembangkan Google untuk aplikasi kami. Ada dua jenis penyeimbang beban, Penyeimbang Beban Jaringan (L3) dan Penyeimbang Beban HTTP (L7). HTTP *Load Balancer* bersifat global sehingga IP yang sama dapat digunakan di mana saja di dunia, tetapi masih mendukung skalabilitas yang sangat tinggi tanpa pemanasan.[8] Maka dapat diartikan bahwa *HTTP Load Balancing* dan NGINX memiliki persamaan yaitu sebagai *Load Balancer*, namun NGINX dapat digunakan tidak hanya di Google Cloud Platform, karena NGINX adalah *software web server* yang *open source*.[9] Sehingga HTTP *Load Balancing* dan NGINX merupakan solusi yang ideal untuk meminimalisir terjadinya *website* yang tidak dapat diakses karena pembagian beban tidak merata ke beberapa *Web Server*.

Untuk membuktikan seberapa handal antara *HTTP Load Balancing* dan NGINX dapat membagi beban antar *Web Server*, maka dilakukan *stress test* menggunakan *Apache JMeter* serta *Server Monitor* dari *smartphone*. Apache JMeter adalah aplikasi *open source* berbasis Java yang dapat dipergunakan untuk *performance test*. Bagi seorang QA Engineer JMeter bisa digunakan untuk melakukan *load/stress testing Web Application*, FTP *Application* dan *Database server test*.[10] Kesimpulannya, Apache JMeter digunakan untuk mengetahui seberapa cepat respon *website* agar dapat memproses sebuah *request* dan *response* ke pengguna.

Berdasarkan penelitian yang telah dilakukan oleh Muhammad Dedy dan Imam Riadi dengan menerapkan optimalisasi *load balancing* dalam *web* UAD kampus 3 yang belum menerapkannya. Mereka membuat 1 ISP berbeda dari sumber yang sudah ada (PT. Telkom Indonesia), lalu menerapkan *load balancing* tanpa mengubah jaringan internal *web* UAD kampus 3 Yogyakarta tersebut. Hasilnya adalah pembagian *bandwidth* yang seimbang antara yang utama dan cadangan, lalu saat satu sumber koneksi mati maka *backup* akan berjalan otomatis dengan mengambil koneksi dari sumber yang kedua.[11] Sedangkan penelitian Muhammad Aldi Aditya Putra dengan pengujian setiap *web server* aplikasi di dalam *container* dengan menggunakan algoritma *round robin* dan *least connections* menghasilkan nilai *respons time* pada Docker dengan *request* 7000 data pada *round bin* 2,128 *second* dan pada *least connections* 2,127 *second*. Dari semua nilai *throughput* yang dihasilkan, pengujian dengan menggunakan *system load balancing* menghasilkan respon yang jauh lebih cepat.[12]

Dua contoh diatas menggambarkan betapa efisien dan efektif suatu *web* yang menggunakan *load balancing*, antara lain; penyeimbang *bandwidth*, penyalur *backup* ke *server* cadangan, mempercepat *respons time* pada saat membuka suatu *web*, serta meminimalisir adanya *downtime* pada *web server* yang diakses oleh pengguna. Namun, dalam kedua jurnal tersebut belum terdapat penelitian konkret tentang perbandingan apabila mengukur performa HTTP *Load Balancing* serta NGINX *Load Balancer* pada *web server* menggunakan Docker *Container* yang dijalankan dalam *Public Cloud* dari Google. Inilah yang membuat penulis tertarik untuk memaparkan secara lebih mendalam.

Setelah analisa terhadap permasalahan diatas, maka penulis mengambil judul penelitian yaitu **"IMPLEMENTASI DAN PENGUKURAN PERFORMANSI *LOAD BALANCING WEB SERVER* MENGGUNAKAN *CONTAINER*"**.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang ada, maka dapat diperoleh beberapa identifikasi masalah, yaitu sebagai berikut:

1. Bagaimana *website* tetap dapat diakses oleh pengguna apabila terjadi gangguan?

2. Bagaimana sistem Docker *Container* dapat berjalan dalam *Public Cloud*?
3. Bagaimana konsep *Public Cloud* dari Google *Cloud Platform* dapat berjalan?
4. Bagaimana mekanisme *Load Balancing* dapat membagi beban pada sebuah *Web Server*?
5. Bagaimana sistem HTTP *Load Balancing* dan NGINX *Load Balancer* dapat berjalan?
6. Bagaimana pengukuran performansi terhadap *Load Balancing* menggunakan *HTTP Load Balancing* dengan NGINX *Load Balancer?*

1.3 Tujuan Tugas Akhir

Berdasarkan identifikasi masalah yang ada, maka diperoleh beberapa tujuan, yaitu sebagai berikut:

1. Merancang virtualisasi infrastruktur di dalam *Public Cloud* dari Google untuk mengukur kecepatan *response time* dalam metode *load balancing* terhadap *web server* yang dimana *web server* tersebut menggunakan *container.*
2. Mengetahui mengenai kinerja *HTTP Load Balancing* dari Google dapat membagi beban ke beberapa *Web Server* dengan simulasi *stress test.*
3. Mengetahui mengenai kinerja *NGINX Load Balancer* dapat membagi beban ke beberapa *Web Server* dengan simulasi *stress test.*
4. Mengetahui komparasi performansi *load balancing web server* diantara *HTTP Load Balancing* dengan *NGINX Load Balancer*.

1.4 Manfaat Tugas Akhir

Adapun manfaat yang di dapat yaitu sebagai berikut :

1. Mengetahui teknologi virtual infrastruktur yang mulai banyak digunakan berbasis *Public Cloud* dari Google.
2. Mengetahui manfaat dari metode HTTP *load balancing* guna pembagian beban antar *web server.*
3. Mengetahui kecepatan *response time* pada saat *HTTP Load Balancing* berjalan dengan simulasi *stress test.*

4. Mengetahui kecepatan *response time* pada saat *NGINX Load Balancer* berjalan dengan simulasi *stress test.*
5. Mengetahui *Load Balancer* yang terbaik dalam menghadapi kondisi *stress test* atau beban yang berat dari permintaan pengguna.

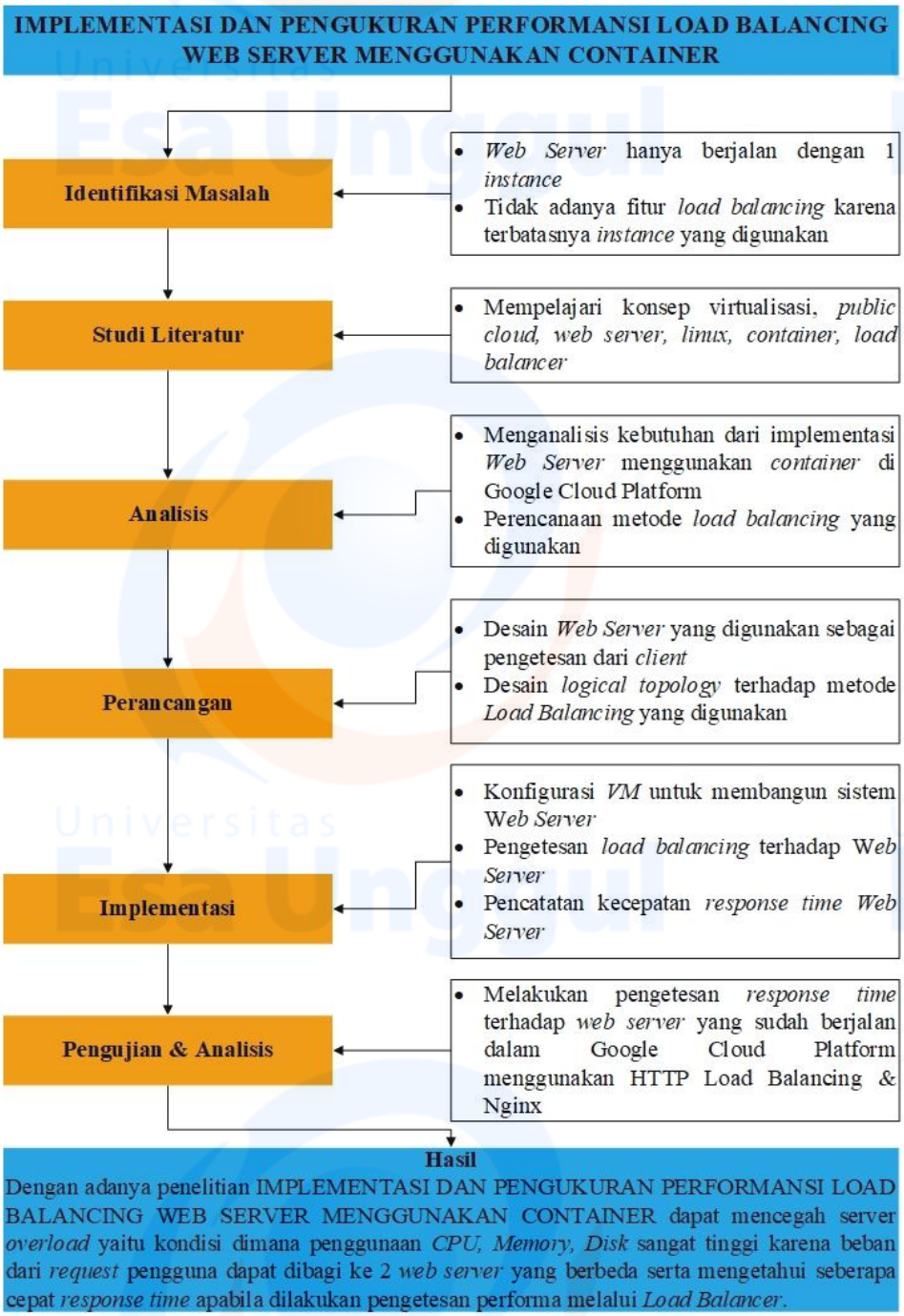
1.5 Lingkup Tugas Akhir

Agar pembahasan menjadi lebih fokus, dalam penelitian ini hanya membahas beberapa ruang lingkup, diantaranya :

1. Mengimplementasikan *Web Server* menggunakan Apache HTTPD berbasis *Container* hanya di Google Cloud Platform.
2. Menggunakan Docker *Container* hanya untuk menjalankan *Web Server.*
3. Menggunakan HTTP *Load Balancing* dan NGINX *Load Balancer* yang sudah tersedia di Google Cloud Platform.
4. Melakukan pengetesan performansi ke *Web Server* menggunakan Apache JMeter dan Server Monitor dari *Client* dan di proses oleh HTTP *Load Balancing* dari Google Cloud Platform.
5. Melakukan pengetesan performansi ke *Web Server* menggunakan Apache JMeter dan Server Monitor dari *Client* dan di proses oleh NGINX *Load Balancer* dari Google Cloud Platform.
6. Melakukan komparasi nilai *response time* rata-rata (*average*) antara HTTP *Load Balancing* dan Nginx *Load Balancer*.

1.6 Kerangka Berpikir

Berikut dibawah ini merupakan hasil rancangan dari kerangka berpikir terhadap penelitian IMPLEMENTASI DAN PENGUKURAN PERFORMANSI LOAD BALANCING *WEB* SERVER MENGGUNAKAN CONTAINER *:*



Gambar 1-1 Kerangka Berpikir

1.7 Sistematika Penulisan Tugas Akhir

Sistematika penulisan dapat dibagi menjadi tiga bagian, yaitu awal, isi, dan akhir. Berikut ada sistematika penulisannya :

BAB I PENDAHULUAN

Pada bab ini dijelaskan mengenai hal yang terdiri dari latar belakang, perumusan masalah, batasan masalah, maksud dan tujuan penelitian, sistematika penulisan penelitian.

BAB II TINJAUNAN PUSTAKA

Pada bab ini berisi tentang teori-teori yang berhubungan dengan Judul Tugas Akhir.

BAB III METODE

Pada bab ini berisi tentang metode penelitian yang penulis gunakan dalam melakukan penelitian pokok permasalahan, metode terdiri dari metode penelitiannya, teknik pengumpulan data dan teknik analisis data.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini berisi hasil desain *logical topology* dan implementasi *web server* menggunakan *container* serta pengetesan performa *response time* melalui *Load Balancer* ke *web server.*

BAB V KESIMPULAN DAN SARAN

Pada bab ini berisi mengenai kesimpulan mengenai hasil analisis pada penelitian dan saran untuk pengembangan sistem lebih lanjut.