Implementasi *Automation Deployment* pada *Google Cloud Compute VM* menggunakan Terraform

Debi Gustian[1], Yuli Fitrisia[2], *Sugeng Purwantoro E.S.G.S[3], Wenda Novayani[4]
[1,2,3,4] Politeknik Caltex Riau, Jl. Umbansari No. 1 Rumbai, Pekanbaru, Indonesia
*Email: debi22trk@mahasiswa.pcr.ac.id[1], uli@pcr.ac.id[2], *sugeng@pcr.ac.id[3], wenda@pcr.ac.id[4]*

Abstract – *This study presents an implementation of automated deployment using Terraform on Google Cloud Compute VM. The aim is to streamline the deployment process and increase efficiency in the deployment of applications. The study involves setting up a cloud environment on Google Cloud, configuring the Terraform code to deploy the necessary resources, and automating the deployment process. The results of the study indicate that using Terraform for deployment automation on Google Cloud Compute VM significantly reduces the deployment time and effort. The implementation of automation deployment also provides benefits such as improved consistency, increased productivity, and reduced errors in the deployment process. From the results of the tests that have been carried out, the author can create 4 VM instances (servers) in Google Cloud at one time with the configured code, the number of VMs can be set as much as needed with specifications that can be set as needed, can also delete all VM that has been created at one time.*

Keywords - *terraform, cloud computing, google cloud.*

Intisari – Penelitian ini menyajikan implementasi penerapan otomatis menggunakan Terraform pada Google Cloud Compute VM. Tujuannya adalah untuk merampingkan proses penerapan dan meningkatkan efisiensi dalam penerapan. Penelitian ini melibatkan penyiapan lingkungan cloud di Google Cloud, mengonfigurasi kode Terraform untuk menerapkan sumber daya yang diperlukan, dan mengotomatiskan proses penerapan. Hasil penelitian menunjukkan bahwa menggunakan Terraform untuk otomatisasi penerapan di Google Cloud Compute VM secara signifikan mengurangi waktu dan upaya penerapan. Implementasi penerapan otomatisasi juga memberikan manfaat seperti peningkatan konsistensi, peningkatan produktivitas, dan pengurangan kesalahan dalam proses penerapan. Dari hasil pengujian yang telah dilakukan penulis dapat membuat 4 *VM instance* (*server*) yang ada di *google cloud* dalam satu waktu dengan *code* yang telah dikonfigurasikan, untuk jumlah *VM* dapat diatur sebanyak yang dibutuhkan dengan speksifikasi yang bisa di atur sesuai kebutuhan, juga dapat menghapus semua *VM* yang telah dibuat dalam satu waktu.

Kata Kunci - *terraform, cloud computing, google cloud.*

I. PENDAHULUAN

Pesatnya perkembangan teknologi komputer di negara-negara maju, membuat para penelitinya semakin "haus" akan tenaga komputasi yang dapat menjawab tantangan dan permasalahan yang mereka hadapi. Walaupun sudah memiliki super komputer dengan kapasitas dan kecepatan kerja yang sangat tinggi, apa yang sudah ada ini dirasa tetap kurang, karena mereka berusaha memecahkan permasalahan yang lebih besar lagi, lalu munculah teknologi *Cloud Computing*.

Cloud Computing mengacu pada aplikasi dan layanan yang berjalan di jaringan terdistribusi yang menggunakan sumber daya virtual dan diakses oleh protokol Internet umum dan standar jaringan. Seorang pengguna layanan *Cloud Computing* (*cloud* services) seperti Google *Cloud*. Kemudian ingin membuat sebuah *server*. Untuk melakukan hal ini tentunya harus melalui beberapa tahap, seperti mengisi sejumlah formulir, memilih sistem operasi,

mengatur kapasitas, dan lainnya. Semua langkah tersebut memang tidak terlalu menjadi beban apabila hanya membuat satu atau dua *server*. Akan tetapi, akan menjadi sangat merepotkan apabila perlu membuat banyak *server* sekaligus, dan dalam durasi waktu yang singkat, untuk itu diperlukan automation agar mempermudah dalam membuat banyak *server*. Salah satu tools yang dapat digunakan untuk melakukan otomatisasi tersebut adalah Terraform. Terraform adalah alat/aplikasi *open-source* yang memungkinkan mendefinisikan infrastruktur sebagai kode menggunakan bahasa pemrograman deklaratif yang sederhana, dan untuk menerapkan serta mengelolanya infrastruktur di berbagai penyedia cloud (termasuk Amazon Web Services, Azure, Google Cloud, DigitalOcean, dan banyak lainnya) menggunakan beberapa perintah.

Namun, pertanyaan mulai muncul terhadap kualitas nya pada *instance* web *server* yang dibuat secara otomatis akankah sesuai standar, hal ini menjadi pertanyaan para penyedia layanan web *server*. Pada parameter yang dilihat pada penelitian ini yaitu perbandingan terhadap waktu yang dibutuhkan dalam pembuatan *instance* web *server* dari kedua metode tersebut dan perbandingan kualitas web *server* dengan melakukan pengujian beban *request* terhadap web *server* dan performa web *server*. untuk itu penulis akan melakukan pengujian terhadap performa dan beban *requests* pada sistem menggunakan *tools Quality of Service* dan JMeter

Berdasarkan dari permasalahan yang telah diuraikan di atas, maka penulis berkeinginan untuk mengembangkan penelitian tentang *Cloud Computing* yang dikolaborasikan dengan penggunaan Terraform dengan judul penelitian "Implementasi *Automation Deployment* pada *Google Cloud Compute VM* menggunakan Terraform".

II. SIGNIFIKANSI STUDI

A. Tinjauan Pustaka

Pada penelitian pertama, sebelumnya telah dilakukan oleh [1] yang berjudul "Implementasi *Web Server* menggunakan *Infrastructure As Code* Terraform Berbasis *Cloud Computing*" menyatakan bahwa Kemajuan teknologi otomatisasi menjadikan suatu kemudahan untuk pembangunan infrastruktur web server, tanpa harus melakukan perintah pembuatan secara berulang ulang, hanya dengan serangkaian kode yang telah didefinisikan menggunakan Infrastructure as code Terraform maka infrastruktur web server berhasil dibangun. Namun, pertanyaan mulai muncul terhadap kualitas nya pada instance web server yang dibuat secara otomatis akankah sesuai standar, hal ini menjadi pertanyaan para penyedia layanan web server. Pada parameter yang dilihat pada penelitian ini yaitu perbandingan terhadap waktu yang dibutuhkan dalam pembuatan instance web server dari kedua metode tersebut dan perbandingan kualitas web server dengan melakukan pengujian beban request terhadap web server.

Penelitian kedua, telah dilakukan oleh [2] yang berjudul "Implementation of Infrastructure as Code using Terraform and Ansible for Multi-Tier Application on Google Cloud Platform" Penelitian ini bertujuan untuk mengimplementasikan Infrastructure as Code (IaC) menggunakan Terraform dan Ansible untuk aplikasi multi-tier pada Google Cloud Platform (GCP). Terraform digunakan untuk memprovisioning infrastruktur sementara Ansible digunakan untuk mengkonfigurasi instance.

Sementara penelitian ketiga, dilakukan oleh [3] yang berjudul "Infrastructure as Code (IaC) Implementation using Terraform on Google Cloud Platform" Tujuan dari penelitian ini adalah untuk mengimplementasikan Infrastructure as Code (IaC) menggunakan Terraform pada Google Cloud Platform (GCP). Metodologi penelitian yang digunakan dalam studi ini adalah metode penelitian eksperimental. Metode penelitian eksperimental digunakan untuk membuktikan atau menolak suatu hipotesis.

Pada penelitian keempat, dilakukan oleh [4] yang berjudul "Infrastructure as Code (IaC) Implementation on Google Cloud Platform using Terraform". Bertujuan untuk

mengimplementasikan Infrastructure as Code (IaC) pada Google Cloud Platform (GCP) menggunakan Terraform. IaC memungkinkan para pengguna untuk memprovisioning dan mengelola infrastruktur secara otomatis dengan menggunakan kode. Penerapan IaC pada GCP menggunakan Terraform dapat mempermudah para pengguna dalam mengelola infrastruktur mereka. Dalam penelitian ini, Terraform digunakan untuk memprovisioning berbagai jenis sumber daya pada GCP seperti Virtual Private Cloud (VPC), firewall, dan instance.

B. Landasan Teori

1. Cloud Computing

Cloud Computing adalah sistem komputerisasi berbasis jaringan/internet, di mana suatu sumber daya, *software*, informasi dan aplikasi disediakan untuk digunakan oleh komputer lain yang membutuhkan. Model *billing* dari layanan ini umumnya mirip dengan modem layanan publik. Ketersediaan sesuai kebutuhan, mudah untuk di kontrol, dinamik dan skalabilitas yang hampir tanpa limit adalah beberapa atribut penting dari *Cloud Computing*. Menurut NIST (National Institute of Standards and Technology), terdapat 5 karakteristik sehingga sistem tersebut disebut *Cloud Computing*, yaitu *Resource Pooling, Broadband Network Access, Measure Service, Rapid Elasticity dan Self Service*. [5]

2. Terraform

Terraform merupakan sebuah sarana untuk membuat, merubah, dan menggabungkan infrastruktur dengan aman dan efisien menggunakan kode. Konsep ini dikenal juga dengan istilah "Infrastructure as Code". Container Terraform mendukung banyak *cloud* service besar, seperti IaaS (contoh AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g., Heroku), or SaaS services (e.g., Atlas, DNSimple, *Cloud*Flare). Sedangkan untuk Terraform sendiri bisa dijalankan di hampir semua OS, mulai dari macOS, Windows, hingga sejumlah distro Linux yang telah didukung. [6]

3. QoS (Quality of Service)

Quality of service (QoS) (Bahasa Indonesia : kualitas layanan) mengacu pada teknologi apa pun yang mengelola lalu lintas data untuk mengurangi packet *loss* (kehilangan paket), *latency*, dan *jitter* pada jaringan. QoS mengontrol dan mengelola sumber daya jaringan dengan menetapkan prioritas untuk tipe data tertentu pada jaringan. [7]

Parameter Quality of Service terdiri dari:

1) Throughput yaitu kecepatan (*rate*) *transfer* dataefektif, yang diukur dalam bps (*bit per second*).

TABEL I
THROUGHPUT

Kategori Throughput	Throughput (bps)	Indeks
Sangat Bagus	100	**4**
Bagus	75	**3**
Sedang	50	**2**
Jelek	<25	**1**

Persamaan perhitungan *troughput:*

$$Throughput = \frac{paket\ data\ diterima}{lama\ pengantaran} \quad (1)$$

2) Packet Loss, merupakan suatu parameter yang menggambarkan suatu kondisi yang menunjukkan jumlah total paket yang hilang dapat terjadi karena collision dan congestion pada jaringan.

TABEL II
PACKET LOSS

Kategori Degredasi	Paket loss (%)	Indeks
Sangat Bagus	0 %	**4**
Bagus	3 %	**3**
Sedang	15 %	**2**
Jelek	25 %	**1**

Persamaan perhitungan paket loss:

$$Paket\ loss = \frac{(paket\ \ dikirim - paket\ diterima)}{paket\ data\ yang\ dikirim}\ x\ 100\% \quad (2)$$

3) *Delay (Latency),* merupakan waktu yang dibutuhkan data untuk menempuh jarak dari asal ke tujuan. Delay dapat dipengaruhi oleh jarak, media fisik, congesti atau juga waktu proses yang lama.

TABEL III
DELAY

Kategori Latensi	Besar Delay(ms)	Indeks
Sangat Bagus	<150 ms	**4**
Bagus	150 ms s/d 300 ms	**3**
Sedang	300 ms s/d 450 ms	**2**
Jelek	> 450 ms	**1**

Persamaan perhitungan *Delay* (*Latency):*

$$Rata\ rata\ Delay = \frac{Total\ Delay}{Total\ paket\ diterima} \quad (3)$$

4) *Jitter atau Variasi Kedatangan Paket, Jitter* diakibatkan oleh variasi-variasi dalam panjang antrian, dalam waktu pengolahan data, dan juga dalam waktu penghimpunan ulang paket-paket diakhir perjalanan jitter.

TABEL IV
JITTER

Kategori Jitter	Jitter(ms)	Indeks
Sangat Bagus	0 ms	**4**
Bagus	0 ms s/d 75 ms	**3**
Sedang	75 ms s/d 125 ms	**2**
Jelek	125 ms s/d 225 ms	**1**

Persamaan perhitungan *Jitter:*

$$Jitter = \frac{Total\ variasi\ delay}{Total\ paket\ diterima} \quad (4)$$

$$Total\ variasi\ delay = Delay - (Rata - rata\ Delay)$$

4. JMeter

Apache JMeter adalah sebuah perangkat lunak pengujian beban atau performa aplikasi yang dapat digunakan untuk menguji kinerja berbagai jenis aplikasi web. Berdasarkan pengujian menggunakan JMeter, kategori web *server* yang dikatakan bagus adalah web *server* yang mampu menangani beban pengguna (*user load*) dengan baik tanpa terlalu mempengaruhi waktu respon (*response time*) dan *throughput*. Web *server* yang memiliki performa bagus harus mampu menghasilkan respon dengan cepat dan stabil dalam kondisi beban yang tinggi [8].

5. Otomatisasi

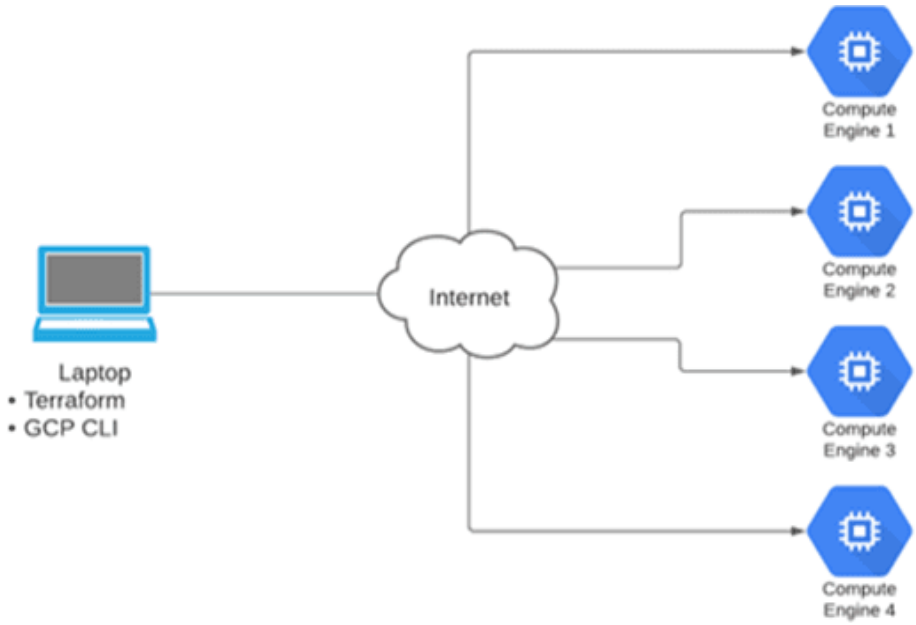
Otomatisasi adalah suatu teknologi yang menggabungkan aplikasi ilmu mekanika, elektronika dan sistem berbasis komputer melalui proses atau prosedur yang biasanya disusun menurut program instruksi serta dikombinasikan dengan pengendalian otomatis untuk meyakinkan apakah semua instruksi itu sudah dilaksanakan seluruhnya dengan benar sehingga produktivitas, efisiensi dan fleksibilitas meningkat. [9]

C. Metode Penelitian

Metode penelitian yang digunakan yaitu pengujian verfikasi pada google *cloud* dan pengujian system terraform.

1. Arsitektur Sistem/Jaringan

Rancangan arsitektur sistem ini menjelaskan, yaitu ada sebuah laptop yang terhubung ke jaringan internet untuk melakukan konfigurasi terhadap terraform melalui google *cloud Platform*, Terraform adalah *Platform* yang digunakan untuk melakukan otomatisasi, selanjutnya konfigurasi akan dirancang untuk membuat 4 VM compute engine. Sistem Operasi pada *Virtual Machine* menggunakan Ubuntu. Untuk dapat melakukan otomatisasi VM compute google *cloud* diperlukan *software* SDK google *cloud* sebagai gcp cli nya yang berguna untuk melakukan konfigurasi terhadap terraform yang ada di google *cloud*



Gambar 1 Arsitektur Sistem

2. Kebutuhan Perangkat

Berikut speksifikasi mesin virtual yang digunakan yaitu:

TABEL VI
SPESIFIKASI MESIN VIRTUAL

	Spesifikasi
Processor	64 Bit Intel Core i5
Storage	30 GB
RAM	2 GB
Operating System	Ubuntu 20.04
Software	Terminal

3. Metode Pengujian

a. *Pengujian System Terraform*

Pengujian ini dilakukan untuk melihat bagaimana kemampuan dari modul terraform saat menangani permintaan dari *user* saat semua VM telah muncul yang berada di *website google cloud* dan saat salah satu VM di hapus. Langah awal user mengakses *website google cloud* untuk melihat ke empat VM yang telah dibuat, setelahnya hapus salah satu dari VM tersebut dan dilihat apakah modul terraform akan mengembalikan VM yang telah dihapus.

b. Pengujian Scale Up

Pengujian Scale up pada instance VM Google Cloud merupakan proses untuk memastikan bahwa infrastruktur dapat bertahan dan berjalan dengan baik ketika jumlah instance VM ditambah. Pada pengujian ini, dilakukan beberapa langkah seperti menentukan jumlah instance VM yang diperlukan, melakukan pengujian performa, memantau ketersediaan sumber daya, memeriksa ketersediaan dan keamanan, dan melakukan penyesuaian jika terdapat masalah. Dalam pengujian ini, tujuan utama adalah untuk memastikan bahwa performa instance VM tetap optimal dan ketersediaan sumber daya mencukupi ketika jumlah instance VM ditingkatkan.

c. Pengujian Scale Down

Pengujian Scale down pada instance VM Google Cloud merupakan proses untuk memastikan bahwa infrastruktur dapat beradaptasi dengan jumlah instance VM yang berkurang. Pada pengujian ini, tujuan utama adalah untuk memastikan bahwa performa dan ketersediaan sumber daya tetap optimal ketika jumlah instance VM dikurangi. Beberapa langkah yang dapat dilakukan dalam pengujian Scale down antara lain memantau penggunaan sumber daya, mengevaluasi ketersediaan dan keamanan.

d. Analisis pengujian Quality of Service (QoS)

Pengujian *Quality of Service* (QoS) pada *instance* vm google *cloud* dapat dilakukan dengan menggunakan *tools* Wireshark untuk memantau lalu lintas jaringan yang melewati *instance* vm tersebut. Dalam pengujian QoS menggunakan wireshark, dapat memeriksa berbagai parameter QoS seperti packet *loss rate*, *latency*, dan *jitter*. Hasil analisis QoS tersebut dapat digunakan untuk memastikan bahwa kualitas layanan jaringan yang diberikan memenuhi standar yang diinginkan.

e. Analisis pengujian JMeter

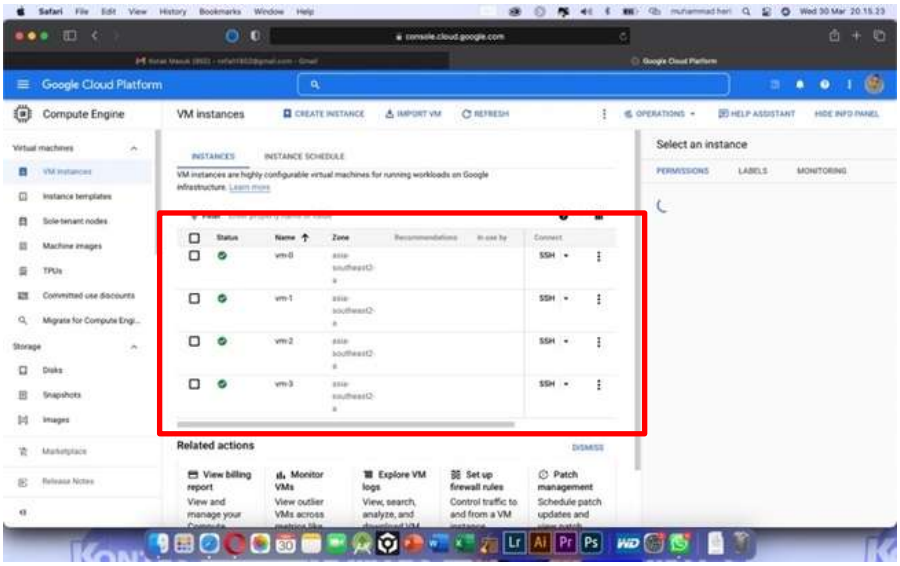
Pengujian JMeter pada web server instance VM Google Cloud dapat memberikan informasi yang sangat berguna dalam mengevaluasi performa dan keandalan server. Berikut adalah beberapa analisis pengujian JMeter pada web server instance VM *Google Cloud:*

1. Kecepatan respon
2. Tingkat kesalahan
3. Waktu *response*
4. Konfigurasi *server*

III. HASIL DAN PEMBAHASAN

A. Menjalankan Konfigurasi

Perintah yang digunakan yaitu terraform apply, Selanjutnya melakukan konfirmasi agar VM dapat dibuat oleh terraform dengan menginputkan “yes” untuk konfirmasi persetujuan. Setelah menjalankan konfigurasi, maka *Virtual Machine* akan muncul di *web* google *cloud* yang telah dibuat terraform berjumlah 4 VM.



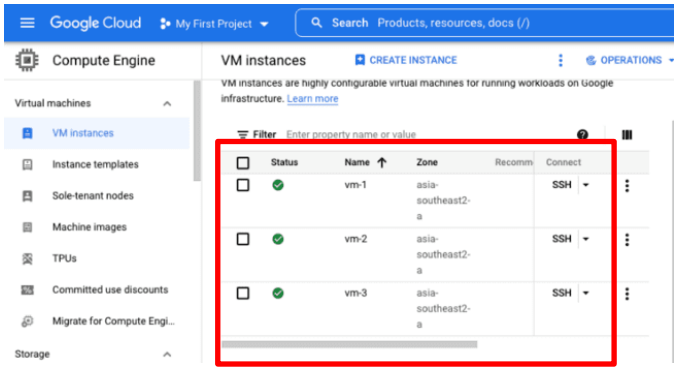
Gambar 2 Hasil konfigurasi

B. Pengujian Sistem

1. Pengujian Sistem Terraform

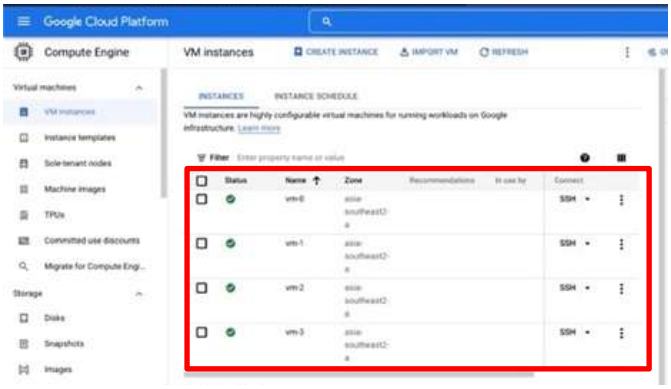
Berikut langkah-langkah pengujian sistem terraform:

1) Untuk tahap pertama pada pengujian ini penulis masuk ke *google cloud* kemudian mencari menu *Compute Engine* kemudian ada beberapa pilihan, klik VM *Intances*. Berikut tampilan halaman VM *Intance* yang sudah menjalankan konfigurasi dan dari 4 VM yang dibuat, hapus salah satunya dan tersisa 3 VM.



Gambar 3. Hapus VM

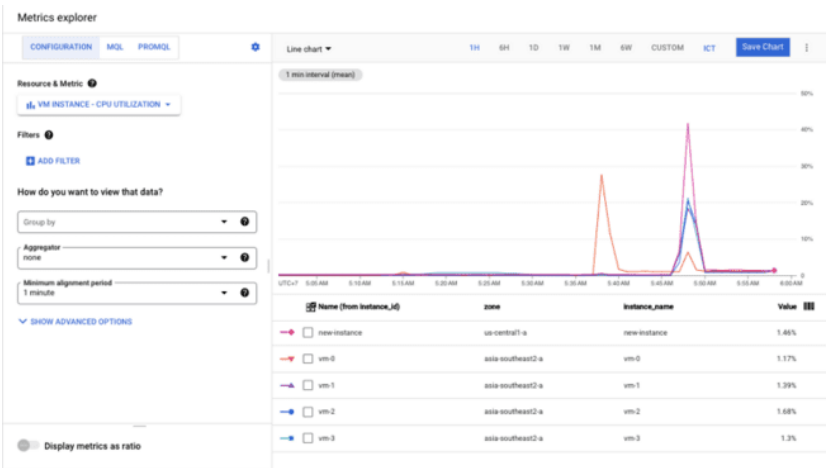
2) Selanjutnya buka terminal untuk menjalankan kembali konfigurasi dengan memasukkan perintah terraform *apply*.
3) Setelah dijalankan, konfigurasi hanya akan membuat VM yang telah dihapus, dan hasil akhirnya tetap akan menyesuaikan dengan yang ada dikonfigurasi yaitu empat VM.
4) Berikut hasil vm yang telah dikembalikan yaitu sebanyak 4 vm.



Gambar 4 Hasil Vm yang telah dikembalikan

2. *Pengujian Scale up*

Proses pengujian Scale Up pada Terraform dilakukan dengan Memantau kinerja infrastruktur setelah penambahan sumber daya baru, dengan menggunakan layanan monitoring Stackdriver Monitoring.

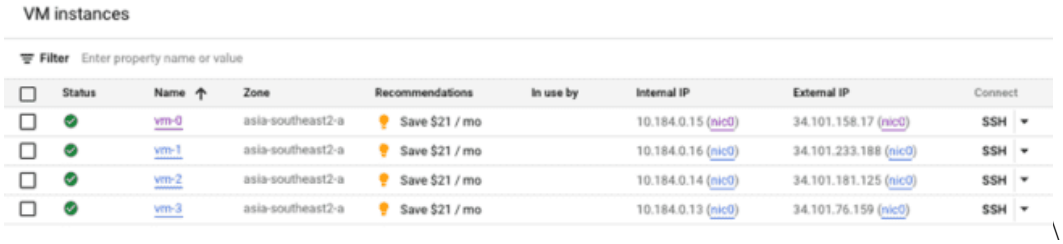


Gambar 5 Monitoring Stackdriver

3. *Pengujian Scale Down*

Untuk melakukan pengujian scale down, berikut langkah-langkahnya:

1) Untuk melakukan pengujian scale down, ubah nilai target_size pada file konfigurasi sebelumnya menjadi nilai yang lebih kecil dari jumlah instance yang sedang berjalan.
2) Jalankan perintah terraform apply untuk mengubah konfigurasi dan mengeksekusi pengujian scale down.



Gambar 6 Sebelum di Scale Down

3) Periksa kembali instance group di konsol Google Cloud untuk memastikan bahwa jumlah instance telah dikurangi sesuai dengan nilai target_size.

Gambar 7 Sesudah di Scale Down

Vm-3 berhasil di destroy secara otomatis menggunakan konfigurasi yang telah dibuat sebelumnya.

4. Pengujian Quality of Service (QoS)

1) Throughput

Untuk mencari *throughput* menggunakan filter wireshark "*tcp*.", selanjutnya klik *button statistics – Capture* File *Propeties*.

menghitung *throughput*:

$$\frac{90085497}{39.845} = 2.260.898{,}4063245 \text{ b x } 8 \text{ (}\textit{kilo byte}\text{ ke }\textit{mega byte}\text{)}$$

= 18.087.187,250596

= 18 M/s

dari kategori *troughput* pada perhitungan ini troughput termasuk kategori Sangat bagus yaitu dengan indeks 4.

2) Paket Loss

Untuk mencari paket loss menggunakan *filter* wireshark "*tcp.analysis.lost_segment*", selanjutnya klik *button statistics – Capture File Propeties*.

menghitung paket *loss*:

$$= \frac{40580 - 40578}{40580} \text{ x } 100\%$$

$$= \frac{2}{40580} \text{ x } 100\%$$

= 0,0049

dari kategori *paket loss* pada perhitungan ini termasuk kategori Sangat bagus yaitu 0,0%.

Untuk menghitung *Delay* dan *Jitter*, simpan file *capture file propeties* ke dalam penyimpanan laptop dengan cara mengklik tombol file – *Export Packet Dissections* – As CSV.

3) Delay (Latency)

Total delay : 39,84450508 s

$$= \frac{39{,}84450508}{40580}$$

= 0,000981875 s x 1000

= 0,981875 ms

dari kategori *paket loss* pada perhitungan ini termasuk kategori Sangat bagus yaitu <150 ms.

4) Jitter

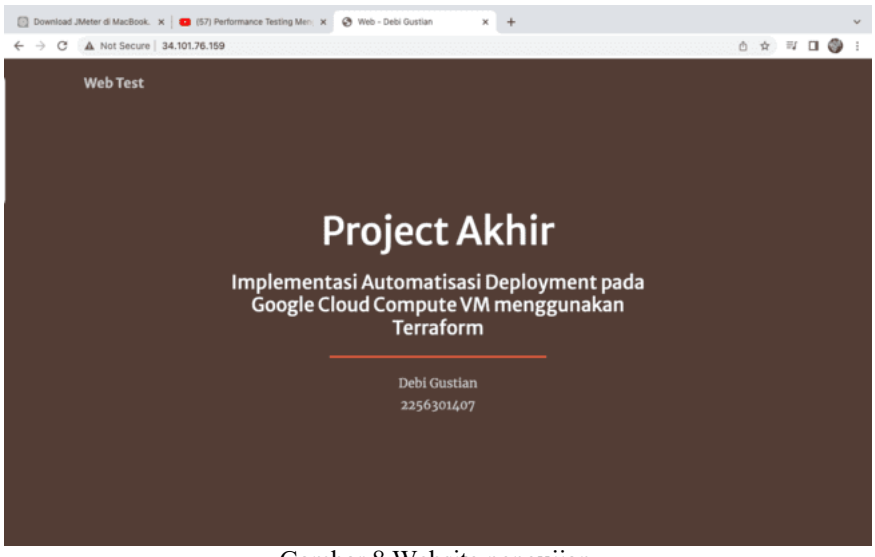
Total Jitter : 0,001770295 s

$= \frac{0{,}001770295}{40580}$
= 4,36248E-08 s x 1000
= 4.362,48 ms
dari kategori *paket loss* pada perhitungan ini termasuk kategori Sangat bagus yaitu <75 ms.

TABEL VII
HASIL PENGUJIAN

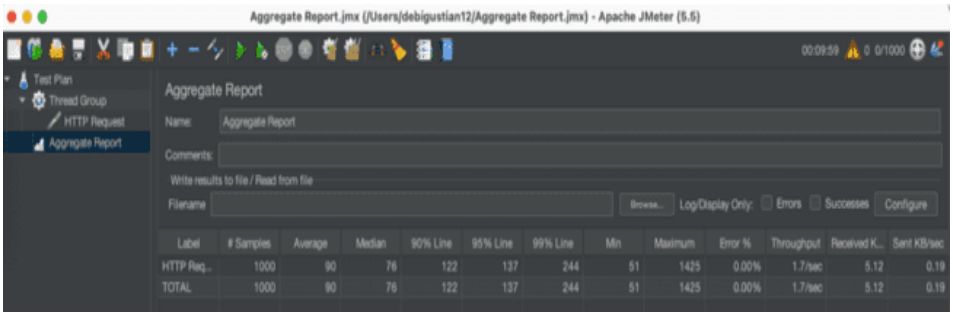
Pengujian Ke -	Throughput	Paket Loss	Delay	Jitter
1	18.087.187,25	0,0049	0,981875	4.362,48
2	18.013.238,68	0,0345	0,834753	3.857,67
3	18.244.352,24	0,102	0,345533	3.565,45
4	18.234.342,58	0,0022	0,653563	3.244,56
5	18.465.536,25	0,0464	0,674364	3.633,63
6	18.536.636,24	0,0053	0,845363	3.942,63
7	18.672.764,78	0,0535	0,874342	3.963,72
8	18.627.742,75	0,0322	0,366536	4.265,62
9	18.632.646,74	0,0653	0,632635	4.322,14
10	18.636.962,63	0,0636	0,735667	4.242,56
Total	**184.151.410,14**	**0,4099**	**6,944631**	**39.400,46**
Rata-rata	**18.415.141,01**	**0,0409**	**0,6944631**	**3.940,05**
Kategori	**Sangat Bagus**	**Sangat bagus**	**Sangat Bagus**	**Bagus**

5. *Pengujian Kinerja (Performance Test)*

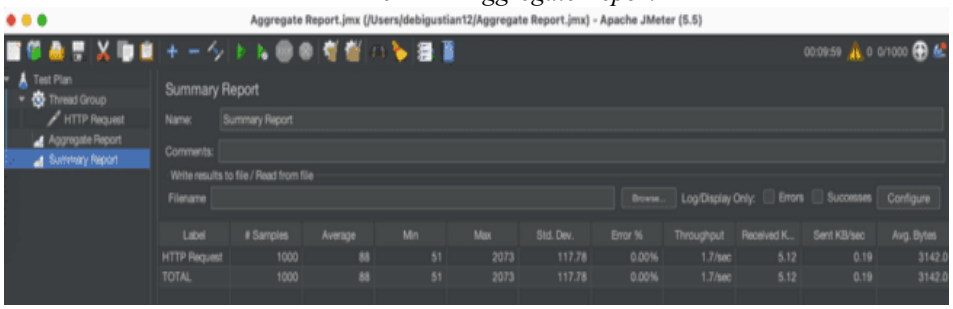


Gambar 8 Website pengujian

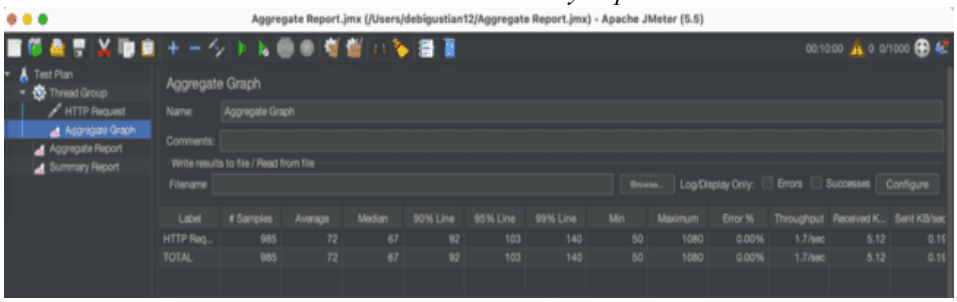
Pengujian performa web *server* menggunakan JMeter, Jalankan pengujian beban dan menunggu selama 10 menit sampai proses selesai, kemudian muncul hasilnya berikut ini:



Gambar 9 Hasil *Aggregate Report*



Gambar 10 Hasil *Summary Report*



Gambar 11 *Hasil Aggregate Graph*

Berikut analisis hasil pengujian menggunakan *listener* pada JMeter:

1. *Aggregate Report*:
   (a) Jumlah pengguna (*threads*): 1000
   (b) Jumlah permintaan sukses: 1000
   (c) Rata-rata waktu respons: 90 ms
   (d) Persentase error: 0%
   (e) Waktu paling lambat: 1525 ms
   (f) Waktu paling cepat: 51 ms
2. *Summary Report*:
   (a) Jumlah pengguna (*threads*): 1000
   (b) Jumlah permintaan sukses: 1000
   (c) Rata-rata waktu *respons*: 88 ms
   (d) Persentase error: 0%
   (e) Throughput: 1.7 *requests*/*second*
   (f) Waktu paling lambat: 2073 ms
   (g) Waktu paling cepat: 51 ms
3. *Graph Result*:
   (a) Grafik menunjukkan waktu respons dari setiap permintaan HTTP dalam bentuk grafik garis.

(b) Dapat dilihat bahwa waktu respons stabil dan kebanyakan di bawah 300 ms, dengan puncak tertinggi hanya mencapai 1500 ms.

Dari analisis di atas, dapat disimpulkan bahwa performa web *server* sangat baik karena:

(a) Jumlah permintaan sukses mencapai 1000 dengan persentase *error* 0%
(b) Rata-rata waktu *respons* hanya sekitar 51 ms dan waktu paling lambat hanya 1525 ms
(c) Throughput mencapai 1.7 *requests*/*second*

Dengan demikian, dapat dikatakan bahwa web *server* mampu menangani beban yang cukup besar dengan waktu *respons* yang cepat dan stabil.

IV. KESIMPULAN

Berdasarkan hasil pengujian dan pembahasan yang sudah dilakuan di atas maka dapat diambil kesimpulan yaitu Implementasi terraform menggunakan *cloud Platform* google *cloud* dan Terraform dapat digunakan untuk mengelola infrastruktur di Google Cloud Platform secara otomatis. Penelitian ini menggunakan Terraform untuk mengkonfigurasi dan mendeploy instance VM pada Google Cloud Platform. Dari hasil penelitian ini, konfigurasi terraform yang telah dibuat dapat dimodifikasi sesuai kebutuhan, baik dari segi jumlah maupun spesifikasi vm yang diinginkan serta kualitas dan performa web server pada instance vm google cloud sangat baik, mampu menangani beban yang cukup besar yaitu sekitar 40580 paket yang dikirimkan dengan waktu respon yang cepat dan stabil.

Sementara beberapa saran yang dapat dikembangkan untuk penelitian selanjutnya yaitu Penelitian dapat dilakukan dengan selain google *cloud* seperti dengan menggunakan amazon dan *microsoft azure*. Penambahan *Platform open-source* kubernetes agar lebih terkelola dengan baik dan dapat dijalankan dimanapun dan Mengimplementasikan fungsi dari *Virtual Machine* yang telah dibangun.

REFERENSI

[1] Sosinsky, B. (2011). Cloud Computing Bible. Published by Wiley Publishing, Inc., Indianapolis, Indiana, ISBN: 978-0-470-90356-8.
[2] Brikman, Y. (2016). Terraform: Up and Running. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472. ISBN: 978-1-491-97703-3.
[3] Amalia, E. I. (2016). Microsoft Punya Kaca Pintar, Bisa Kenali Emosi Pengguna. Retrieved from 06 Juni 2016 12:40 *web*site: https://www.medcom.id/teknologi/news-teknologi/VNnxlaEk-microsoft-punya-kaca-pintar-bisa-kenali-emosi-pengguna
[4] Eko, R. I. (2000). Managemen Sistem Informasi dan Teknologi Informasi.
[5] Ellora, D. (2018). Smart Mirror, Cermin Serbaguna yang Memiliki Fitur Analisis Kulit. Retrieved from 27 December 2018 *web*site: https://journal.sociolla.com/journal/smart-mirror/
[6] Firman, A., Wowor, H. F., Najoan, X., Teknik, J., Fakultas, E., & Unsrat, T. (2016). Sistem Informasi Perpustakaan Online Berbasis *Web*. 5(2).
[7] Gorde, K. S. (2017). Raspberry Pi Powered Magic Mirror. 8824–8827. https://doi.org/10.15662/IJAREEIE.2017.0612009
[8] Jhonsen. (2004). *Web* Designer untuk Pemula. Jakarta: Elex Media Komputindo Kelompok Media, Anggota IKAPI.
[9] Kasiman, P. (2006). Aplikasi *Web* dengan PHP dan MySQL. Yogyakarta: Andi.

[10] Kurniawan, D. E., Fani, S., Informatika, J. T., Batam, P. N., & Barelang, O. P. (2017). Perancangan sistem kamera pengawas berbasis perangkat bergerak menggunakan raspberry pi. III (2).
[11] Kustiyahningsih, Y., & Rosa, D. (2011). Pemrograman Basis Data Berbasis *WEB* MEnggunakan PHP dan Mysql. Yogyakarta: Graha Ilmu.
[12] Nirsal. (2012). Perangkat Lunak Pembetukan Bayangan pada Cermin dan Lensa. 2, 24–33.
[13] Pi, R. (2016). Pengembangan Prototipe Portal Otomatis Dengan Pendeteksian Plat Nomor Kendaraan Berbasis. 5.
[14] Sutabri, T. (2012). Analisis Sistem Informasi. Yogyakarta: Andi.
[15] Sutarman. (2012). pengantar teknologi informasi. Jakarta: PT. Bumi Aksara.
[16] Tedyyana. A, Ratnawati. F, Syam. E, Putra. F.P. (2022). Threat modeling in application security planning citizen service complaints. Indonesian Journal of Electrical Engineering and Computer Science. Vol. 28, No. 2. pp. 1020~1027