Load Balancing Berbasis Agent Berdasarkan Penggunaan Sumber Daya Sistem

Hirianinda Malsegianty S [1], Catur Wirawan Wijiutomo, S.T., M.T.[2], Aji Gautama Putrada, S.T., M.T.[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]hirianinda.malsegianty@students.telkomuniversity.ac.id, [2]caturwijiutomo@telkomuniversity.ac.id,
[3]ajigps@telkomuniversity.ac.id

Abstrak

Load balancing berbasis agent adalah salah satu metode yang sangat efektif, dengan menggunakan agent yang mengobservasi resource server dibandingkan perhitungan traffic terkecil, sehingga dapat menjaga performansi server. Kemudian software agent merupakan program yang bertindak atas kehendak user dimana dalam paper ini agen berperan sebagai autoscaling yang telah berisi script yang akan melakukan aksi tertentu pada AWS. Auto scaling yang telah diberikan alarm memantau utilitas cpu semua server ketika mendapat request dari client. Jika nanti grafik utilitas cpu dari server tersebut melebihi threshold maka server tersebutakan diganti oleh load balancer dan memindahkan workload sisa ke server lain yang berada dalam grup auto scaling.

Kata kunci :agen, workload, load balancer, utilitas cpu, cloudwatch, AWS, auto scaling

Abstract

Load balancing based on agents is one of effective method that use agent to observe server resource rather than least traffic calculation, therefore server's performance can be maintained. Software agent is a program that worksin the name of user where in this paper multiple agents will work as an auto scaling that already filled with scripts to perform certain action in AWS. Auto scaling with alarms monitor CPU utility when received requests from client. If CPU utilization's graph from that server exceed the threshold then it will be replaced and load balancer will move the rest of the workload to another server within the auto scaling group.

Keywords: agent, workload, load balancer, CPU utility, cloudwatch, AWS, auto scaling

1. Pendahuluan

1.1. Latar Belakang

Pada era globalisasi dimana semua hal mengalami perkembangan yang sangat pesat termasuk dalam bidang teknologi khususnya internet. Dalam internet siapapun dapat melakukan apapun dan dapat diakses secara bebas, dengan kemajuan internet cara tradisional mulai ditinggalkan, rata-rata semua data dipindahkan ke internet sehingga masyarakat dapat dengan mudah mengakses data tersebut hanya dengan browsing di browser favorit. Browsing dilakukan dengan pengiriman paket-paket jaringan ke server di internet yang nantinya akan dikembalikan ke client sebagai hasil atas permintaan client. Biasanya sebuah aka ada sebuah server yang menangani request dari banyak client sekaligus, dimana server ini suatu saat akan menjadi overloaded oleh tugas-tugas yang diberikan client.

Untuk mengatasi masalah tersebut diterapkan load balancing yang mendistribusikan load yang diterima server ke server lainnya jikalau server tersebut telah overload atau mencapai threshold yang telah ditentukan sebelumnya. Pada umumnya load balancing menggunakan traffic yang terjadi pada sebuah server sebagai tolak ukur bahwa server tersebut overload namun pada paper ini akan disimulasikan load balancing yang menggunakan resource server yaitu utilitas CPU sebagai parameter penentuan overload.

Load balancing ini akan dilakukan dengan bantuan agent yang bekerja sebagai load balancer. Agent sendiri merupakan konsep yang banyak mendapatkan perhatian selama 20 tahun terakhir khususnya dalam computer sains konsep agent dikembangkan menjadi software agent yang merupakan program computer yang bertindak untuk user atau program lain yang berhubungan. Agent adalah jenis software agen yang mempunyai sifat learning dan kemampuan untuk berpindah tempat dari computer 1 ke lainnya yang terhubung pada jaringan, terus hingga berhenti pada computer tujuan[1]. Disini agent akan bertugas mengumpulkan data dan mengamati resource server saat bekerja dimana jika dalam interval waktu tertentu dan dalam pengecekan tertentu utilitas CPU melebihi threshold maka agent akan membangkitkan alarm, yang mengakibatkan server tersebut dinilai unhealthy oleh load balancer sehingga sisa workload berpindah ke server lain dalam grup yang sama dan server tersebut akan dimuat ulang.

Server – server ini akan dibuat pada AWS (Amazon Web Service) sebuah website yang menyediakan banyak layanan berbasis cloud dimana salah satunya adalah EC2 untuk membuat virtual machine. Dengan melihat dari sisi resource server dibandingkan traffic dapat menjaga performansi server.

Setelah melakukan percobaan load balancing dengan parameter CPU Usage menggunakan bantuan agent, nantinya hasil percobaan ini akan dibandingkan dengan sebuah instance EC2 yang tidak menggunakan load balancing.

1.2. Perumusan Masalah

a. Bagaimana perbandingan instance yang menggunakan load balancer dan instance biasa dalam memproses request client?
b. Bagaimana cara load balancer memilih instance?
c. Bagaimana perbandingan rata –rata penggunaan resource tanpa dan ketika menggunakan load balancing berbasis agent?

1.3. Batasan Masalah

a. Agent merupakan alarm yang dibangkitan dalam auto scaling
Agent pada paper ini berperan sebagai alarm dalam auto scaling yang akan bekerja jika threshold tercapai.
b. Sistem yang ada dalam AWS
Sistem yang digunakan merupakan system bawaan AWS, mulai dari server, load balancer, dan matrik mengikuti apa yang telah tersedia dari aws.
c. Sumber daya yang ada di EC2 yang menjadi parameter load balancing
Sumber daya yang dimaksud adalah utilitas CPU instance di EC2.
d. Tipe load balancing Classic Elastic Load Balancing
e. Pemilihan instance yang berada dalam 1 subnet.
Pemilihan server selanjutnya merupakan server yang berada pada subnet yang sama.
f. Amazon Machine Image yang digunakan merupakan Amazon Linux AMI 2018.03.0(HVM), SSD Volume Type.

1.4. Tujuan

a. Simulasi yang dilakukan pada paper ini bertujuan untuk :
b. Membuktikan apakah instance yang menggunakan load balancer dapat lebih baik daripada instance biasa dalam memproses request client.
c. Mengetahui cara load balancer membagi workload server dalam memilih instance lain yang paling sesuai dengan bantuan agent.
d. Melihat perbandingan rata –rata penggunaan resource server tanpa load balance dan ketika menggunakan load balancing berbasis agent.

1.5. Organisasi Tulisan

Setelah pendahuluan, ada bab Studi Terkait membahas tentang teori literatur yang mendukung topik tulisan ini. Kemudian disusul dengan bab yang berisi tentang bagaimana sistem pada tulisan ini dibangun. Selanjutnya evaluasi berisi dua sub-bagian, yaitu Hasil Pengujian dan Analisis Hasil Pengujian.yang dilakukan dimana hasilnya menjawab topik yang dibahas.

2. Studi Terkait

2.1. Studi Literatur

a. Load Balancing

Beberapa penelitian terkait dengan load balancing diantaranya adalah sbb:

- Berdasarkan paper [2], disebutkan Load Balancing dilakukan pada multi agent spasial yang menggunakan algoritma slope-based.
- Pada [3] dilakukan percobaan load balancing bersifat hybrid untuk MAS, dimana semua host memiliki coordinator load balancing sendiri yang memonitor load agen dan traffic message.
- Pada [4], disebutkan bahwa menggunakan load balancing dynamic berbasis mobile agen yang pada sisi server agen akan berpindah mengambil nilai workload jika melebihi rata-rata job sever maka akan dipindahkan ke server lain , cpu usage pada simulasi ini lebih kecil dibandingkan skema load balancing berpusat pada server.
- Pada [5] dilakukan skema load bancing pada simulasi terdistribusi berdasarkan system multi agent dimana agent-agent mengecek sumber daya perangkat keras, jika ditemukan node yang overload maka akan dilakukan migarsi untuk meminimasi waktu simulasi.

Diatara paper diatas ada yang menyinggung CPU usage tetapi hanya sebatas analisis perbandingan saja, dimana pada paper ini dijadikan parameter load balancing itu sendiri.

b. Agent

Kemudian Software agent yang merupakan program computer yang bertindak untuk user atau program lain yang berhubungan. Sesuai dengan batasan masalah, maka agent yang dimaksud merupakan cloudwatch dan load balencer. Agent ini merupkan contoh dari Remote Agent yang menghasilkan rencana untuk mencapai sebuah goal yang telah ditetapkan sebelumnya dan memonitor eksekusi dari rencana tersebut[8].

2.2. Teori Penelitian

2.2.1. Aws Secara Umum

Amazon Web Services (AWS) adalah platform berbasis cloud yang aman dan menyediakan sekumpulan servis infrastruktur seperti kemampuan komputasi, penyimpanan basis data, pengiriman konten dan fungsionalitas lainnya untuk membantu bisnis tumbuh. AWS merupakan layanan dari amazon yang bekerja sesuai permintaan dan telah banyak digunakan mulai dari perusahaan besar hingga mahasiswa yang dapat diakses secara gratis dengan program AWS Educate.[7]

2.2.2. Layanan AWS

AWS menyediakan lebih dari 50 servis yang dapat digunakan dan diatur sesuai keinginan untuk membangun sebuah infrastruktur. [7]

2.2.3. EC2

Amazon Elastic Compute Cloud adalah layanan yang menyediakan kapasitas komputasi terukur pada AWS cloud. Dengan menggunakan EC2 menghapus kebutuhan investasi untuk perangkat keras, sehingga dapat mengembangkan dan mendistribusikan aplikasi lebih cepat. EC2 dapat digunakan untuk meluncurkan server virtual sebanyak yang diinginkan, konfigurasi keamanan dan jaringan, juga mengelola penyimpanan. EC2 memungkinkan peningkatan atau penurunan dalam menangani perubahan kebutuhan dan mengurangi estimasi traffic. [7]

2.2.4. Key Pairs

AWS menggunakan kriptografi berupa sebuah kunci publik untuk mengamankan informasi login untuk instance yang dimiliki, kunci ini dinamakan key pairs dimana kunci public ini diberikan dan pengguna menyimpannya pada tempat yang aman. uses public-key cryptography to secure the login information for your instance. Pengguna menentukan nama key pair saat menjalankan instance kemudian menyediakan kunci pribadi untuk mendapatkan sandi administrator untuk instance Linux agar dapat login menggunakan network tool. [7]

2.2.5. Load Balancer

Elastic Load Balancing mendistribusikan lalu lintas jaringan ke lebih dari satu instance EC2. Load balancer meningkatkan ketersediaan aplikasi, dimana pengecekan kesehatan dapat dikonfigurasi untuk memonitor kesehatan resource computer agar load balancer hanya mengirim perminyaan ke computer yang sehat[7]

2.2.6. Security Group

Security group atau kelompok keamanan bertindak sebagai firewall untuk instance yang berkaitan, mengontrol lalu lintas inbound dan outbound pada level instance. Pengguna harus menambahkan security group yang memungkinkan untuk terhubung dengan instance yang dimiliki dari alamat IP menggunakan network tool. Pengguna juga dapat menambahkan aturan yang mengizinkan akses HTTP dari mana saja dan SSH. [7]

2.2.7. Auto Scaling

AWS Auto Scaling memonitor aplikasi dan secara otomatis menyesuaikan kapasitas instance untuk menjaganya tetap stabil. Dengan menggunakan auto scaling, instance dapat ditambahkan atau dikeluarkan dari grup jika kondisi yang membangkitkan alarm terpenuhi. [7]

2.2.8. Cloudwatch

Amazon CloudWatch sebuah layanan manajemen yang melakukan observasi.Cloudwatch menyediakan data yang dapat digunakan untuk memonitor aplikasi atau server, mengoptimasi utilisasi.Cloudwatch mengumpulkan data onservasi dan operasional dalam bentuk metric, log, dan event yang menunjukkan resource, aplikasi dan servis yang sedang berjalan pada AWS. Pada cloudwatch bisa dipasang alarm yang dapat memicu load balancer bekerja jika utilitas cpu mencapai threshold. [7]

2.2.9. Deskripsi healthy dan unhealthy

Pada Elastic Load Balancing instance healthy dan unhealthy akan dicek secara otomatis. Untuk menentukan apakah sebuah instance healty atau tidaknya dapat dikonfigurasi pada bagaian pengecekan kesehatan untuk load balancer, jika sebuah instance dinilai unhealthy maka workload akan dipindahkan ke instance yang healthy. [7]

2.3. Matrik Penelitian

2.3.1. Threshold

Untuk thresholdnya pada paper ini digunakan 70%. Performance computer sangat bergantung dari workload yang diberikan dan threshold yang digunakan. Utilitas CPU biasanya dikatakan ideal maksimum ketika berada di level ~70% karena diatas itu CPU biasanya menjadi lamban denan banyak thread dan workload.

Berikut merupakan contoh persentasi utilitas CPU berdasarkan paper Zhou dan Ferarri[6].

type	CPU utilization	average load index
light (L)	30-45%	0.3-0.7
moderate (M)	60-70%	1.0-1.8
heavy (H)	70-85%	1.8-3.0

Table 1 Presentasi Utilitas CPU

3. Sistem yang Dibangun

3.1. Materi Pengujian

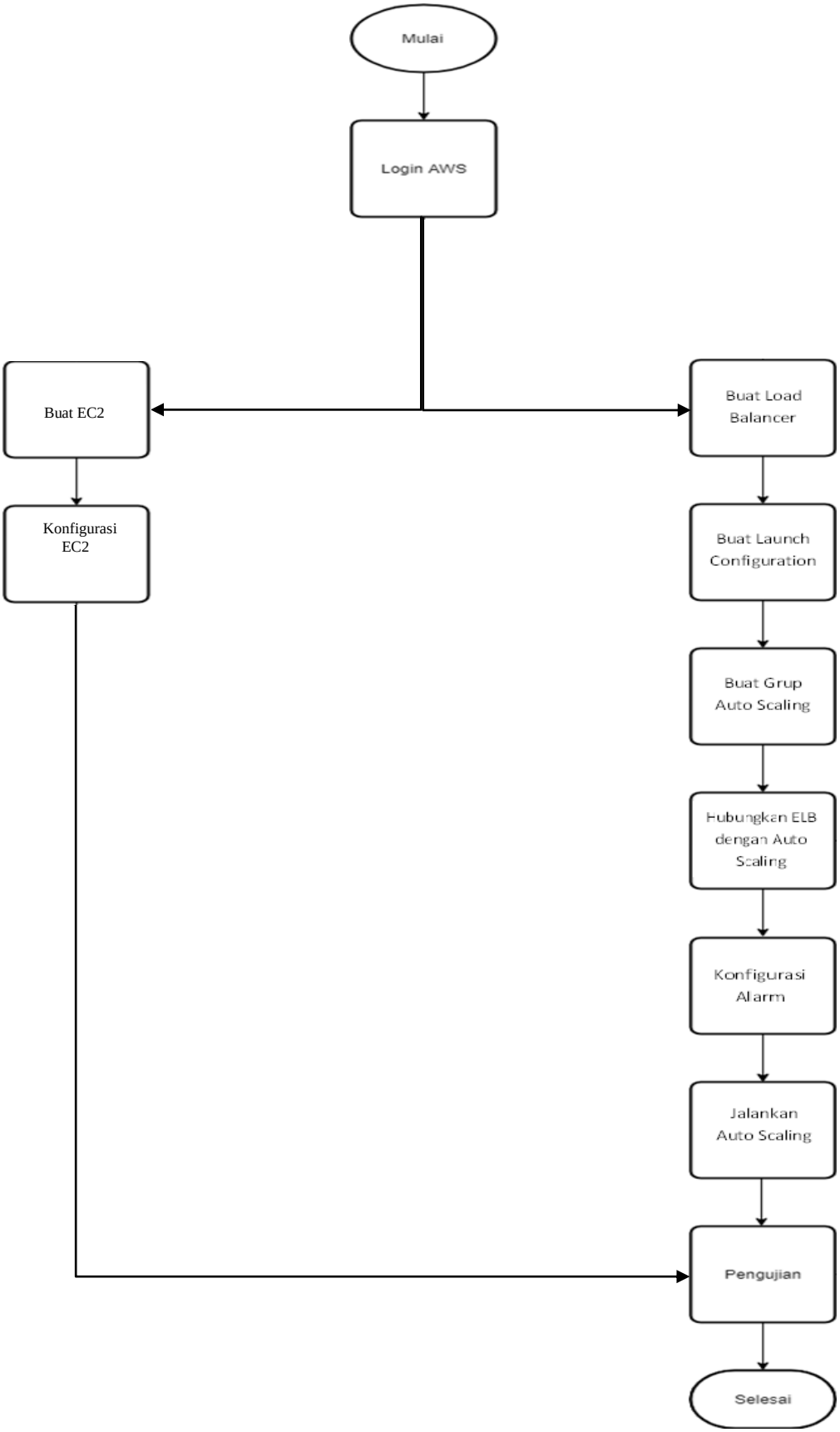
- Sistem pada paper ini dibangun pada AWS, dimana terdapat 2 instace yang aktif dan maksimum hanya 4 instance yang terdapat pada grup auto scaling yang akan digunakan untuk melakukan load balancing.
- Pengujian dilakukan dengan memasang alarm yang akan memeriksa sebanyak 2 titik utilitas CPU melebihi threshold dalam interval waktu 5 menit dengan mengirimkan HTTP Request sebanyak 1000 thread diulang 1000 kali dengan waktu tunggu 100 detik per thread.

- Treshold Utilisasi CPU yang digunakan adalah ketika mencapai 70 % maka alarm akan aktif dan ini akan dinilai sebagai tidak sehat oleh load balancer sehingga akan dilakukan perpindahan sisa workload ke server lain yang berada pada grup auto scaling.

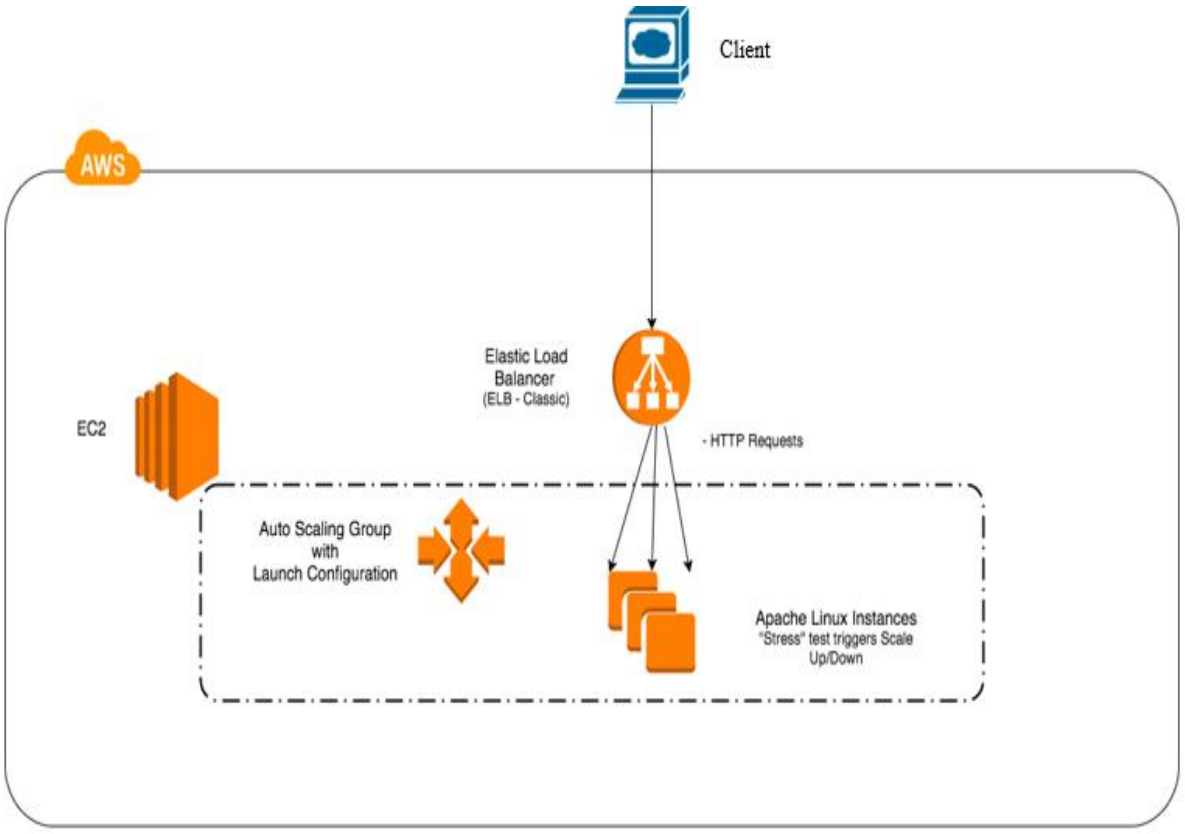
- Spesifikasi Instance yang akan digunakan pada penelitian
  - AMI :Amazon Linux AMI 2018.03.0(HVM), SSD Volume Type.
  - Type : t2 micro
  - vCPU : 1
  - Memory : 1GB
  - Instance Storage : EBS only
  - Network Performance : Low to Moderate

- Pada pengujian ini, akan digunakan bantuan sebuah aplikasi bernama JMeter untuk mengirimkan perintah HTTP berbentuk GET kepada IP adresss instance dan DNS address Load Balancing sebanyak 1000 threads , rump up period 100 detik, dan loop count 1000 kali..

3.2. Diagram Perancangan



Gambar 1 Diagram Perancangan Sistem



Gambar 2 Topologi Perancangan Sistem

3.3. Langkah Perancangan Sistem

1. Login AWS
   Buka amazon console dan login ke akun root aws agar dapat menggunakan layanan AWS.
2. Buat Load Balancer
   Buat load balancer terlebih dahulu dimana load balncer yang digunakan merupakan Elastic Load Balancer.
3. Buat Launch Configuration
   Launch configuration berguna untuk menjalankan instance sesuai konfiguarsi kita yang nantinya dimasukkan pada auto scaling
4. Buat Grup Auto Scaling
   .Auto scaling digunakan jika kita ingin mengatur instance ke dalam 1 grup dengan jumlah yang stabil.
5. Hubungkan ELB Auto Scaling
   Pada halaman konfigurasi launch configuration, hubungkan load balancer yang dibuat dengan grup auto scaling.
6. Konfigurasi Alarm
   Atur penambahan dan penguran jumlah instance jika alarm menyala.
7. Buat EC2
   Buat sebuah EC2 biasa untuk dijadikan pembanding dengan instance yang menggunakan ELB.
8. Konfigurasi EC2
   Konfigurasi EC2 tersebut sesuai dengan yang telah dilakukan pada pembuatan instance di launch configuration tetapi dengan script yang berbeda.
9. Lakukan Pengujian
   Pengujian antara instance auto scaling yang di load balance dan instance EC2 biasa. Kemuadian juga membandingkan load test pada kedua jenis instance tersebut.

4. Evaluasi

4.1. Implementasi

Dibawah ini dijelaskan bagaimana system diimplementasikan dan hasilnya dianalisis sehingga menjawab topik dan sesuai dengan tujuan awal.

a. Buat Load Balancer

- Pada dashboard EC2, buatlah sebuah load balancer yang bertipe Classic Load Balancer.

- Atur Health Check dengan memasukkan detail tempat yang akan dicek oleh load balancer dan juga waktu dan tresholdnya. Pengaturannya adalah seperrti dibawah ini yaitu perhitungan waktu responnya adalah 5 detik setiap 30 detik dimana batas kegagalan pengecekan health check adalah sebesar 2 kali dan instance dikatakan sehat jika berhasil melewati health check sebanyak 10 kali.

Health Check	Nilai
Ping Protocol	HTTP
Ping Port	80
Ping Path	/
Request Timeout	5
Interval	30
Unhealthy Threshold	2
Healthy Threshold	10

Table 2

b. Membuat Launch Configuration

- Pilih AMI untuk instance sesuai dengan yang telah didefinisikan diatas, dimana AMI yang dipilih merupakan Amazon Linux AMI 2018.03.0(HVM), SSD Volume Type dengan t2 micro, vCPU 1, Memory 1GB, jenis ruang instance adalah EBS dan performa jaringan low to moderate.

Fitur	Jenis
AMI	Amazon Linux AMI 2018.03.0(HVM)
SSD Volume Type	t2 micro
vCPU	1
Memory	1 GB
Instance Memory Type	EBS
Network Performance	low to moderate

Table 3

c. Membuat Auto Scaling Group

- Auto scaling group dibuat dengan mendefinisikan berapa instamce yang diinginkan , dan subnetnya, dibawah ini terdapat 2 subnet sehingga 2 instance tersebut ditempatkan di masing-masing subnet. Kemudian pada advanced details, tambahkan ELB (elastic load balencer) yang telah dibuat sebelumnya.

- Tambahkan alarm cloudwatch yang akan bekerja jika rata-rata utilisasi CPU sama dengan atau melebihi 5% setidaknya 1 kali dalam 5 menit waktu pengujian. Ketika hal ini terjadi maka 1 instance akan ditambahkan ke dalam auto scaling.
- Dilakukan hal yang sama juga untuk pengurangan instance yang alarmnya bangkit dikarenakan utilitas CPU rata-rata kurang dari 5% sehingga 1 instance akan dikurangi jika melebihi 10%.
- Instance pada auto scaling yang sudah aktif di halaman load balancer. Instance-instance ini akan dicek status kesehatannya jika sudah dianggap tidak sehat oleh load balancer maka instance itu akan diganti oleh instance berikutnya.

d. Membuat instance EC2 biasa.

- Langkah selanjutnya adalah membuat instance EC2 biasa yang akan dijadikan perbandingan. Pilih AMI Linux dan tipe instance menggunakan konfigurasi sebelumnya. Atur detail instance, mirip dengan pembuatan instance di auto scaling, EC2 ini juga diisi dengan script sederhana namun dengan output bertuliskan ‘ini instance biasa’.

```
#!/bin/bash
//perintah ke shell untuk mengeksekusi script
yum update -y
//perintah install update
yum install -y httpd24 php56 mysql55-server php56-mysqlnd
//perintah install paket http, php, mysql server, mysqlnd
usermod -a -G apache ec2-user
//perintah menambahkan user ec2-user ke group apache
chown -R ec2-user:apache /var/www
//perintah mengubah pemilik dan direktori /var/www
chmod 2775 /var/www
//izin untuk mengubah file pada direktori
find /var/www -type d -exec  chmod 2775 {} \;
// mencari direktori /var/www dengan izin 2775
find /var/www -type f -exec chmod 0664 {} \;
// mencari dalam direktori /var/www sebuah file dengan izin 0664
echo Ini instance biasa > /var/www/html/index.html
// memasukkan untuk dimunculkan tulisan “Ini instance biasa” pada file
/var/www/html/index.html
service httpd start
// memulai servis httpd
```

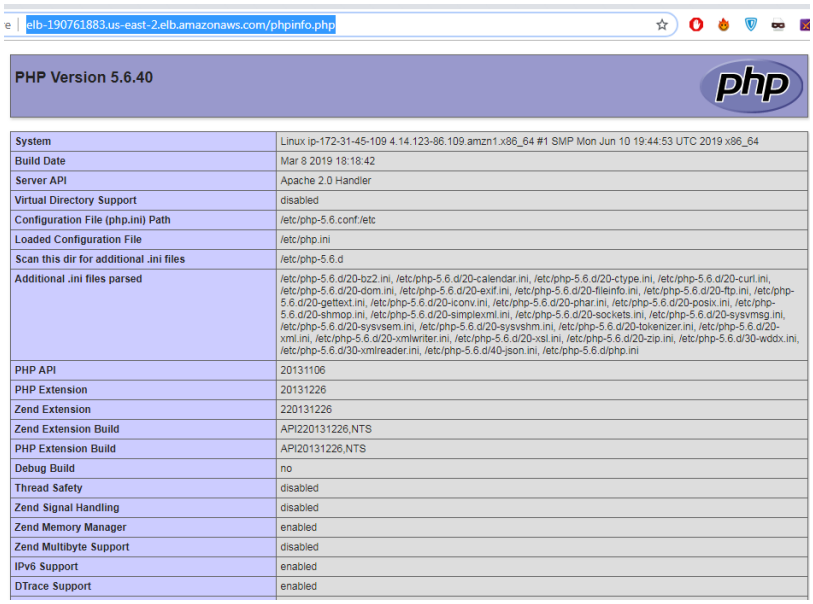
Script 1

- Kemudian untuk security group, ditambahkan SSH & HTTP agar dapat terkoneksi dengan instance dan jalankan instance.

e. Menguji ELB dengan load test menggunakan JMeter.

- Setelah melakukan NSlookup dari DNS ELB didapatkan IP address 18.221.82.141 merupakan interface dari Instance 1 pada ELB dengan IP Public : 18.191.239.146, dan 3.13.75.88 yang merupakan interface dari Instance 2 pada ELB dengan IP Public : 3.17.205.238.

- Untuk dapat melihat tampilan dan instance apa yang bekerja pada load balancer, maka buka DNS load balancer di tab baru dengan menambahkan /phpinfo.php seperti yang dituliskan pada userdata. untuk melihat perubahan instance yang menghandle request. *http://elb-190761883.us-east-2.elb.amazonaws.com/phpinfo.php*



System	Linux ip-172-31-45-109 4.14.123-86.109.amzn1.x86_64 #1 SMP Mon Jun 10 19:44:53 UTC 2019 x86_64
Build Date	Mar 8 2019 18:18:42
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php-5.6.conf./etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php-5.6.d
Additional .ini files parsed	/etc/php-5.6.d/20-bz2.ini, /etc/php-5.6.d/20-calendar.ini, /etc/php-5.6.d/20-ctype.ini, /etc/php-5.6.d/20-curl.ini, /etc/php-5.6.d/20-dom.ini, /etc/php-5.6.d/20-exif.ini, /etc/php-5.6.d/20-fileinfo.ini, /etc/php-5.6.d/20-ftp.ini, /etc/php-5.6.d/20-gettext.ini, /etc/php-5.6.d/20-iconv.ini, /etc/php-5.6.d/20-phar.ini, /etc/php-5.6.d/20-posix.ini, /etc/php-5.6.d/20-shmop.ini, /etc/php-5.6.d/20-simplexml.ini, /etc/php-5.6.d/20-sockets.ini, /etc/php-5.6.d/20-sysvmsg.ini, /etc/php-5.6.d/20-sysvsem.ini, /etc/php-5.6.d/20-sysvshm.ini, /etc/php-5.6.d/20-tokenizer.ini, /etc/php-5.6.d/20-xml.ini, /etc/php-5.6.d/20-xmlwriter.ini, /etc/php-5.6.d/20-xsl.ini, /etc/php-5.6.d/20-zip.ini, /etc/php-5.6.d/30-wddx.ini, /etc/php-5.6.d/30-xmlreader.ini, /etc/php-5.6.d/40-json.ini, /etc/php-5.6.d/php.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled

Gambar 2 Tampilan hasil DNS ELB

- Pengaturan ELB dalam JMeter sesuai dengan yang ditetapkan sebelumnya yaitu sebanyak 1000 threads dengan rump up period 100 detik, dan loop count 1000 kali, yang berarti bahwa ada 1000 user yang melakukan HTTP request dimana thread akan dimulai setelah 0.1 detik dan seluruh proses ini akan diulang sebanyak 1000 kali.

Konfigurasi	Nilai
Nama	Thread ELB
Banyak Thread	1000
Rump Up Period	100s
Loop Count	1000 kali

Table 4

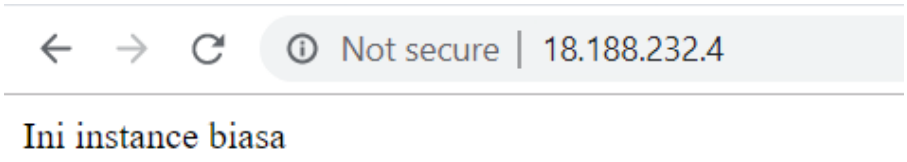
- Pengaturan HTTP Request yang akan dilakukan ke DNS ELB. Metode GET akan dilakukan pada DNS address ELB dengan path phpinfo.php.

Konfigurasi	Nilai
Name	HTTP Request
Protocol	HTTP
Nama Server atau IP	elb-190761883.us-east-2.elb.amazonaws.com
Method	GET
Path	/phpinfo.php

Table 5

f. Menguji instance EC2 biasa.

- Tampilan EC2 ketika telah terhubung, tampilan sederhana ini merupakan hasil script yang telah dimasukkan di awal pembuatan instance.



Gambar 3 Tampilan halaman EC2 biasa

- Sama seperti pada ELB, buat juga load test untuk EC2 menggunakan JMeter, dengan IP publiknya, kemudian dilakukan load test untuk melihat perbandingan performa instance biasa ini terhadap ELB.

4.2 Analisis Hasil Pengujian

Dibawah ini merupakan semua hasil pengujian yang dilakukan pada semua server EC2 Instances yang terdapat pada fasilitas AWS baik yang menggunakan Load Balancer sebagai pengelola proses load balancing. Grafik yang ditampilkan adalah grafik yang dapat menunjukkan adanya perubahan pada penggunaan sumber daya server terhadap proses yang dilakukan oleh load balancer, dan sebagai pembanding juga akan ditampilkan grafik penggunaan sumber daya server tanpa melibatkan proses load balancing. Secara jelasnya dapat dilihat pada semua grafik hasil pengujian di bawah ini :

a. Perbandingan tabel hasil HTTP Request ke DNS

- **Tabel hasil HTTP Request ke DNS ELB menggunakan JMeter**

  Seperti yang dapat dilihat pada table hasil load test ELB dibawah, yang didapatkan merupakan pengiriman HTTP request dimana semuanya berhasil dilakukan dan instance pada ELB tidak mengalami error sedikit pun.

Sample	Start Time	Thread Name	Label	Status	Sent Bytes
1	18:55:24.646	Thread Group 1-4	HTTP Request	Success	156
10	18:55:26.548	Thread Group 1-3	HTTP Request	Success	156
11	18:55:27.744	Thread Group 1-4	HTTP Request	Success	156
12	18:55:27.653	Thread Group 1-2	HTTP Request	Success	156
13	18:55:27.785	Thread Group 1-1	HTTP Request	Success	156
14	18:55:27.785	Thread Group 1-5	HTTP Request	Success	156
15	18:55:27.788	Thread Group 1-3	HTTP Request	Success	156
16	18:55:29.215	Thread Group 1-2	HTTP Request	Success	156
17	18:55:29.216	Thread Group 1-5	HTTP Request	Success	156
18	18:55:29.216	Thread Group 1-1	HTTP Request	Success	156

Table 6 hasil HTTP Request ke DNS ELB

- **Tabel Hasil HTTP Request ke EC2 dengan IP 18.188.232.4.**

  Tabel pertama (Tabel 6) ini menunjukkan HTTP Request ke EC2, namun pada table ke 2 (Tabel 7) dapat terlihat bahwa HTTP Request gagal karena server tidak kuat menerima request yang banyak. Hal ini terjadi karena instance tidak mampu menangani workload yang diterima dan instance tersebut tidak menggunakan load balancer yang dapat menjaganya dari error seperti ini.

Sample	Thread Name	Start Time	Label	Status	Sent Bytes
1	12:31:56.616	Thread EC2 1-1	HTTP Request	Success	116
2	12:31:56.715	Thread EC2 1-2	HTTP Request	Success	116
3	12:31:56.817	Thread EC2 1-3	HTTP Request	Success	116
4	12:31:56.920	Thread EC2 1-4	HTTP Request	Success	116
5	12:31:57.020	Thread EC2 1-5	HTTP Request	Success	116
6	12:31:57.122	Thread EC2 1-6	HTTP Request	Success	116
7	12:31:57.174	Thread EC2 1-1	HTTP Request	Success	116
8	12:31:57.222	Thread EC2 1-7	HTTP Request	Success	116
9	12:31:57.299	Thread EC2 1-2	HTTP Request	Success	116
10	12:31:57.326	Thread EC2 1-8	HTTP Request	Success	116

Table 7 Hasil HTTP Request ke EC2 Biasa yang Sukses

Sample	Thread Name	Start Time	Label	Status	Sent Bytes
85653	12:34:30.398	Thread EC2 1-828	HTTP Request	Warning	0
85654	12:34:30.278	Thread EC2 1-626	HTTP Request	Warning	0
85655	12:34:30.347	Thread EC2 1-463	HTTP Request	Warning	0
85656	12:34:30.346	Thread EC2 1-6	HTTP Request	Warning	0
85657	12:34:29.088	Thread EC2 1-665	HTTP Request	Warning	0
85658	12:34:29.134	Thread EC2 1-713	HTTP Request	Warning	0
85659	12:34:30.504	Thread EC2 1-831	HTTP Request	Warning	0
85660	12:34:30.508	Thread EC2 1-611	HTTP Request	Warning	0
85661	12:34:30.623	Thread EC2 1-75	HTTP Request	Warning	0
85662	12:34:30.281	Thread EC2 1-53	HTTP Request	Warning	0

Table 8 hasil HTTP Request ke IP EC2 berstatus gagal

- Setelah dilakukan percobaan dapat terlihat bahwa instance yang menggunakan load balancing dapat dengan mudah memproses request client yang sangat banyak dan bekerja dengan lancer, dibuktikan dengan semua paket yang dikirimkan sukses. Berbeda dengan instance EC2 biasa tanpa load balancer yang mengalami error karena tidak mampu memproses HTTP request yang dikirimkan mengakibatkan banyak request yang gagal dilakukan.

b. Pembagian Workload ELB

- **Hasil tangkapan Wireshark pada semua IP di ELB**
  Didapatkan 30 sample pengiriman HTTP Request pada DNS ELB.

No.	Time	Source	Destination	Protocol	Length	Info
21	7.449.510	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
29	7.470.476	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
30	7.470.481	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
31	7.470.555	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
32	7.470.616	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
393	9.228.796	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
418	9.285.479	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
423	9.286.955	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
428	9.295.620	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
438	9.345.678	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
872	10.464.157	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
877	10.465.666	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
890	10.513.118	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
891	10.513.503	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
900	10.541.487	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
1283	11.974.717	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
1285	11.974.726	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1286	11.974.728	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1287	11.974.844	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1300	12.038.885	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
1658	13.095.482	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1698	13.145.235	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1699	13.145.461	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
1706	13.158.395	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
1711	13.204.879	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
2077	14.280.034	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1

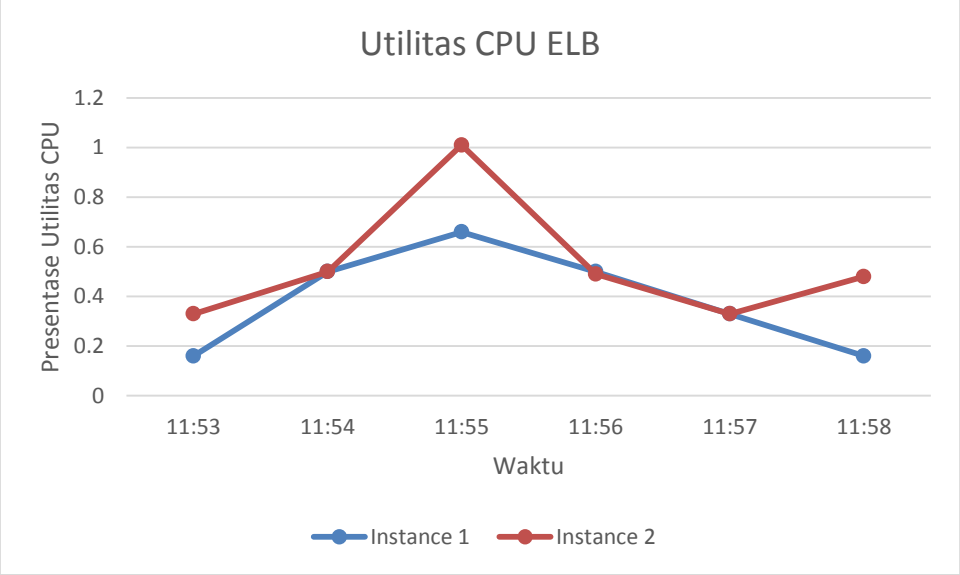
2091	14.349.168	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
2097	14.349.755	192.168.43.12	3.13.75.88	HTTP	210	GET /phpinfo.php HTTP/1.1
2099	14.349.918	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1
2104	14.351.026	192.168.43.12	18.221.82.141	HTTP	210	GET /phpinfo.php HTTP/1.1

Table 9. Hasil Tangkapan Wireshark pada IP dalam ELB

- Yang berwarna biru adalah instance 1 dan oranye merupakan instance 2. Dapat terlihat dari capture wireshark bahwa Instance 1 menerima 12 HTTP Request dan Instance 2 menerima 18 HTTP Request.

c. Grafik Utilitas CPU

- **Grafik CPU Utilization kedua Instance ELB**
  - Dapat terlihat dari grafik perrbandingan 2 instance dibawah dengan jelas bahwa ELB memilih instance 1 terlebih dahulu untuk menerima HTTP Request namun ketika instance 1 tidak mampu, workload dipindahkan ke instance 2, yang menyebabkan instance 2 mempunyai utilitas CPU yang lebih tinggi.



Gambar 4 Grafik CPU Usage Instances EC2 pada ELB

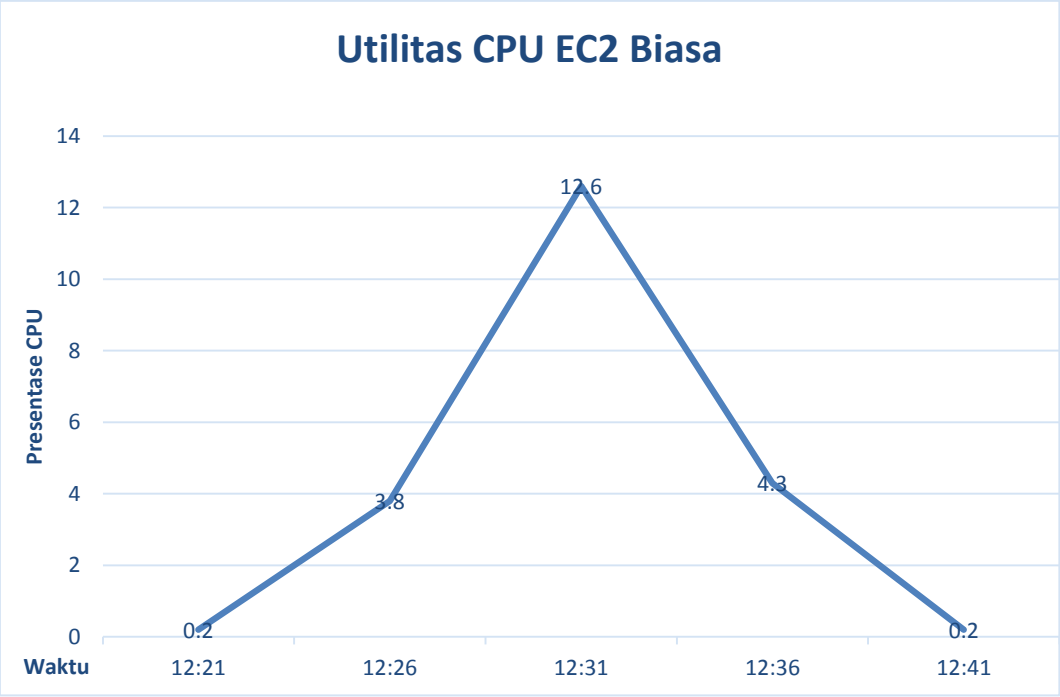
- Kemudian Load Balancer melakukan load balancing atau pembagian workload secara bergantian kepada semua instance yang menggunakannya sehingga hasil grafik yang didapatkan semua instance, mirip antara satu dan lainnya.

- Hasil presentase utilitas CPU yang dihasilkan paling tinggi hanya 1 % yang merupakan presentase yang sangat baik setelah menanggapi semua HTTP request selama 5 menit.

- **Grafik Utilitas CPU EC2**

  Grafiik ini menunjukkan perubahan presentasi utilitas CPU setelah dikirim request HTTP dari JMeter. Walaupun tidak sampai 70% yang merupakan threshold untuk dibangkitkan alarm, EC2 ternyata sudah mengalami error. Dapat terlihat jelas betapa tidak efisiennya instance tanpa menggunakan load balancing.

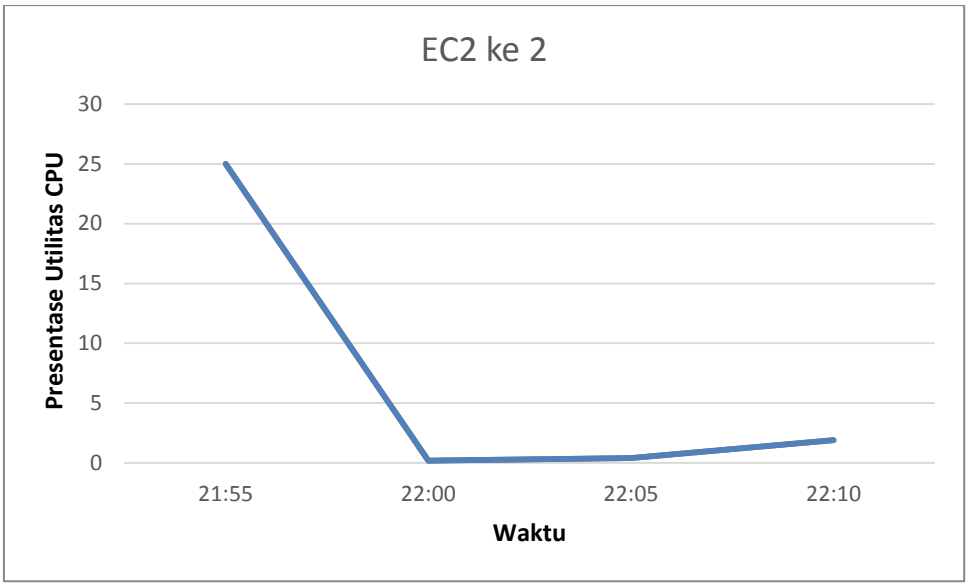
  Setelah utilitas CPU mencapai 12,6%, EC2 mengalami error yang dapat terlihat dengan menurunnya penggunaan CPU 5 menit kemudian ke 4,2% yang menandakan bahwa instance sudah error sehingga CPU pun berhenti bekerja.



Gambar 5 Grafik CPU Usage Instances EC2 biasa

- Penggunaan resource instance yang menggunakan ELB terlihat lebih baik yaitu hanya 1 % dalam pengecekan setiap 5 menit selama 1 jam dibandingkan dengan instance EC2 biasa yang mencapai maksimal 12,6% penggunaan CPU.

- Dilakukan 2 jmeter pada satu instance lagi sebagai pembanding instance EC2 biasa. Seperti yang terlihat dibawah, EC2 ke-2 ini jauh lebih buruk dibandingan EC2 pertama karena menggunakan lebih banyak utilitas CPU hamper 2 kali lipat instance sebelumnya dengan presentase tertinggi yaitu 25% yang langsung jatuh akibat error.



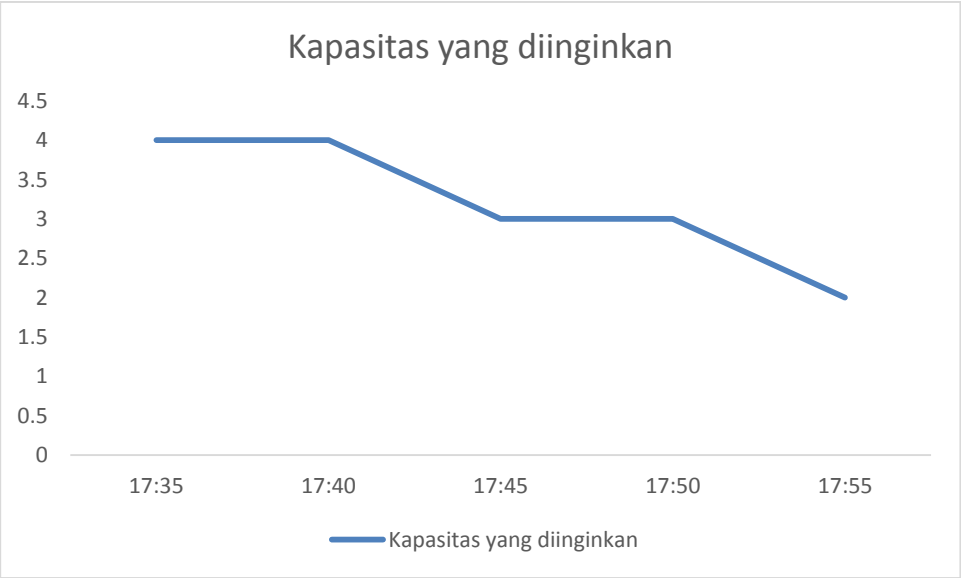
Gambar 6 EC2 ke-2

d. Grafik Instance Keluar Masuk pada Auto Scaling

- Setelah menetapkan threshold menjadi 5% auto scaling dapat terlihat dari grafik dibawah ini yang menunjukkan pergerakan instance yang ditambah dan dikeluarkan setelah alarm bangkit karena threshold terpenuhi.

- **Grafik Kapasitas Yang diinginkan**

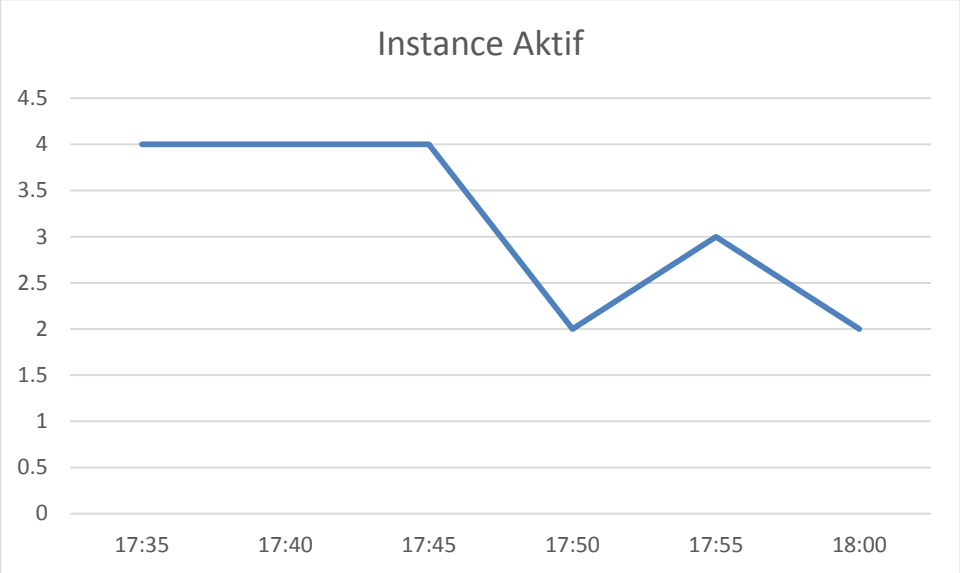
  Kapasitas yang diinginkan telah diset diawal yaitu 2, perubahan yang terjadi pada grafik dikarenan oleh penambahan instance pada auto scaling group namun auto scaling group akan menjadi stabil kembali ke kapasitas awal yang diinginkan setelah dikeluarkannya instance yang memenuhi peraturan scaling.



Gambar 7 Kapasitas yang diinginkan

- **Grafik Instance Aktif**

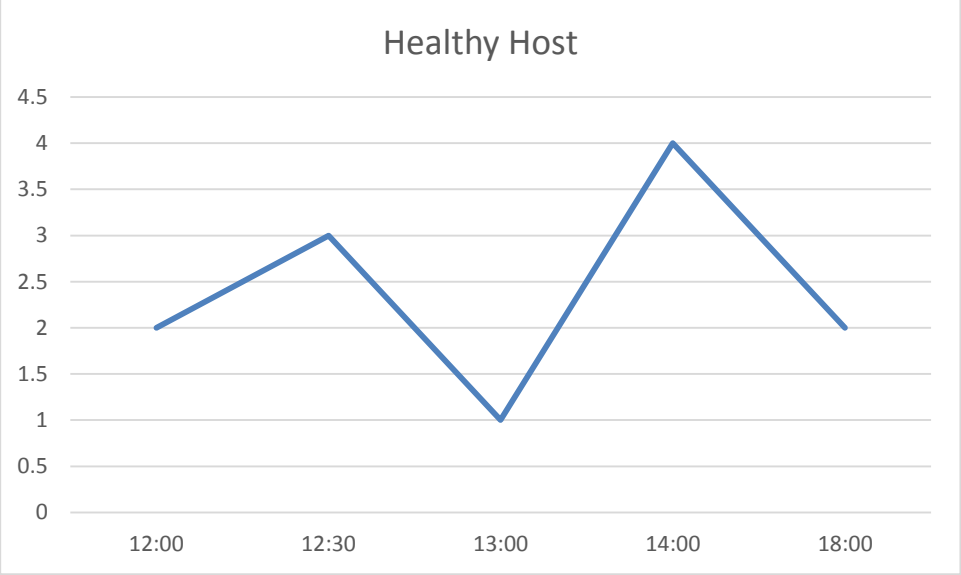
  Dibawah ini menunjukkan perubahan dari jumlah instance yang aktif pada auto scaling group karena peraturan scaling yang menyebabkan instance dapat ditambahkan atau dikeluarkan dari group.



Gambar 8 Grafik Instance Aktif

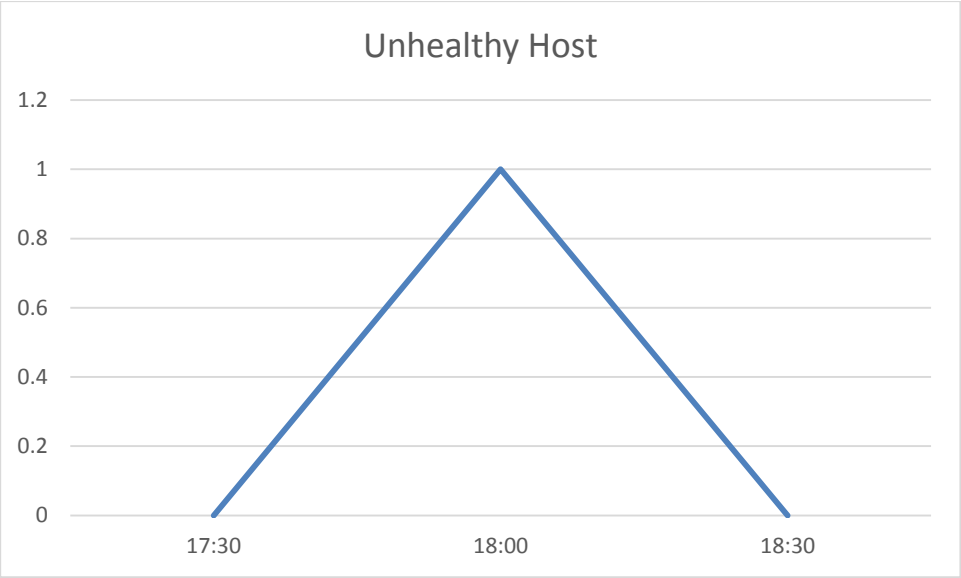
e. Grafik hasil Healthy dan Unhealthy Instances

- Dari grafik dibawah dapat terlihat jelas bahwa instance yang dinilai sehat oleh ELB jumlahnya berubah-ubah sesuai dengan jumlah instance pada auto scaling group.



Gambar 9. Grafik instance Healthy ELB

- Hanya 1 instance yang dinilai tidak sehat oleh ELB pada waktu ke 18:00 yang disebabkan oleh kegagalan instance tersebut merespon sebanyak 2 kali ketikan dicek oleh ELB.



Gambar 10 Grafik instance Unhealthy ELB

5 Kesimpulan

Setelah melakukan pengujian diatas, dapat disimpulkan bahwa instance yang menggunakan load balancing utilitasnya maksimal hanya 1 % dibandingkan dengan yang tanpa load balancer yang mencapai 12,6% dikarenakan load balancer membagi rata workload kepada semua instance yang menggunakannya sehingga overload dapat terhindari dan semua request client berapapun banyaknya dapat diproses oleh instance load balancer.

EC2 biasa tanpa load balancer mengalami error setelah menerima banyak request HTTP dari client, sehingga sesudah mencapai 12,6% penggunaan CPU menurun dikarenakan instance berhenti berjalan, begitu halnya dengan EC2 ke-2 yang mengalami error namun lebih parah dalam penggunaan utilitas CPU. Lain halnya dengan instance dengan ELB yang tetap bekerja dengan baik, setelah diuji.

Auto scaling dapat dilakuka setelah menurunkan threshold dan terlihat dengan jelas sangat membantu menstabilkan kondisi grup instance karena menerapkan peraturan scaling ketika mencapai threshold yang memasukkan atau mengeluarkan sebuah instance telah membangkitkan alarm. Karena terhubung, grafik health check ELB sesuai dengan instance yang berada pada auto scaling group.

Maka dapat disimpulkan, bahwa instance yang menggunakan load balancing akan jauh lebih terlindungi dibandingkan yang tidak.

Daftar Pustaka

[1] H. S. Nwana, "Software agents: An overview," Knowl. Eng. Rev., vol. 11, no. 3, pp. 205–244, 1996.
[2] B. Mistry and M. Fukuda, "Dynamic Load Balancing in Multi-Agent Spatial Simulation," vol. 141, no. 1, pp. 2–7, 2015.
[3] I. Özcan and Ş. Bora, "A hybrid load balancing model for multi-agent systems," INISTA 2011 - 2011 Int. Symp. Innov. Intell. Syst. Appl., pp. 182–187, 2011.
[4] J. Grover and S. Katiyar, "Agent based dynamic load balancing in Cloud Computing," Hum. Comput. Interact. (ICHCI), 2013 Int. Conf., pp. 1–6, 2013.
[5] Y. J. Lee, G. Y. Park, H. K. Song, and H. Y. Youn, "A Load Balancing Scheme for Distributed Simulation Based on Multi-agent System," pp. 613–618, 2012
[6] Ferrari, Zhou, " An Empirical Investigation of Load Indices for Load Balancing Applications", pp.7,1987.
[7] Amazon Web Server, *AWS Documentation*, diakses 10 Juli 2019 <https://docs.aws.amazon.com>
[8] Russell, Stuart J.; Norvig, Peter (2003), 'The State Of The Art' , in Russell, Norvig " *Artificial Intelligence A Modern Approach*" (3rd ed.), pp.47.