BAB 3
METODE PENELITIAN

3.1 PERANGKAT YANG DIGUNAKAN

3.1.1 PERANGKAT KERAS (*HARDWARE)*

Perangkat keras yang akan digunakan pada penelitian ini menggunakan satu pc sebagai server dan satu pc sebagai *client* dengan spesifikasi sebagaimana terdapat pada tabel 3.1.

Tabel 3.1 Spesifikasi Perangkat Keras

Server	OS	Ubuntu 22.04 LTS
	Processor	Intel Core i7-7700
	RAM	8 GB
	Harddisk	100 GB (SSD)
Client	OS	Ubuntu 22.04. LTS
	Processor	Intel Core i7-7700
	RAM	8 GB
	Harddisk	100 GB (SSD)

3.1.2 PERANGKAT LUNAK (*SOFTWARE*)

Pada penelitian ini terdapat perangkat lunak yang digunakan yaitu perangkat virtual dan perangkat lunak *tool* dan aplikasi.

3.1.2.1 PERANGKAT VIRTUAL

Pada penelitian ini terdapat 3 perangkat *virtual* yang dibangun pada *Ubuntu server* yaitu: 1 *load balancer* dan 2 *node docker* swarm.
Spesifikasi perangkat *virtual* tercantum dalam tabel 3.2.

Tabel 3.2 Spesifikasi Perangkat *Virtual*

LOAD BALANCER	OS	Debian
	vCPU	2 *Core*
	RAM	2 GB
	Harddisk	20 GB

	Alamat IP	10.212.20.253
NODE MANAGER	OS	Ubuntu Server
	vCPU	1 *Core*
	RAM	1 GB
	Harddisk	20 GB
	Alamat IP	10.212.20.151
NODE WORKER	OS	Ubuntu Server
	vCPU	1 *Core*
	RAM	1 GB
	Harddisk	20 GB
	Alamat IP	10.212.20.152

3.1.2.2 SOFTWARE *TOOL* DAN APLIKASI

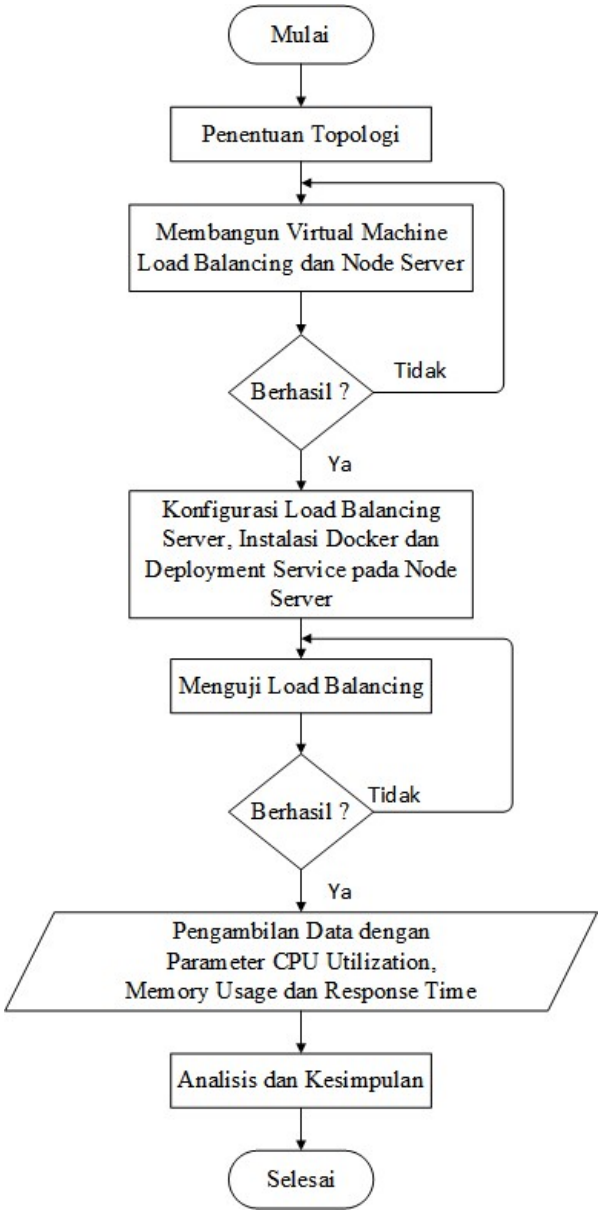
Perangkat lunak sebagai *tool* dan aplikasi yang digunakan pada penelitian ini ditunjukkan pada tabel 3.3.

Tabel 3.3 *Tool* dan Aplikasi

No	Software	Versi	Fungsi
1	Virtualbox	6.1	Virtualisasi
2	Zevenet	5.12.2	*Load Balancer*
3	*Wordprerss*	6.0.2	*Web Server*
4	*H2Load Benchmark*	1.48.0	*Tool* Pengujian *Response Time Load Balancer*
5	*Docker Swarm*	20.10.16	Pengelolaan Kluster *Server*

3.2 ALUR PENELITIAN

Penelitian ini dilakukan dengan melalui beberapa tahapan seperti pada diagram alur yang ditunjukkan pada gambar 3.1.



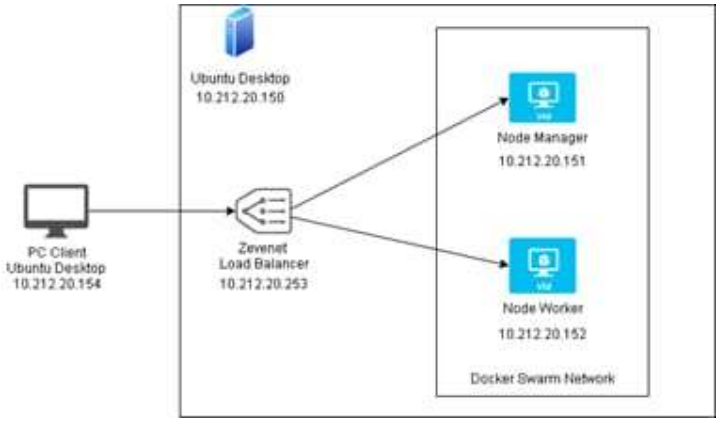
Gambar 3.1 Diagram Alur Penelitian

Gambar 3.1 menunjukkan diagram alur perancangan sistem dalam penelitian ini. Pertama-tama melakukan penentuan topologi yang akan digunakan sebagai dasar sebuah arsitektur jaringan *load balancing*. Selanjutnya melakukan instalasi *load balancing* dengan *iso image zevenet ce 12.0.2* dan dua buah *node server* dengan *iso image ubuntu server* menggunakan *virtual machine*. Jika berhasil, Selanjutnya melakukan konfigurasi *load balancing server* dan melakukan instalasi *docker* pada *node server* kemudian melakukan *deployment service* berupa

nginx. Jika konfigurasi *load balancing server* dan instalasi *docker* pada *node server* dan *deployment service* sudah berhasil langkah berikutnya adalah melakukan uji *load balancing* baik algoritma *round robin* maupun *least connection* dengan cara akses alamat IP *load balancer* melalui *pc client*. Langkah selanjutnya yaitu melakukan pengambilan data pada setiap parameter yang diuji. Untuk itu, penggunaan *software tools h2load* diperlukan dalam penelitian ini. Parameter yang diuji pada penelitian ini adalah *CPU utilization, memory usage,* dan *response time*. Parameter tersebut dapat menggambarkan tingkat kecepatan dan kehandalan web server dalam menangani *request* yang masuk. Setelah mendapatkan data dari parameter tersebut dilanjutkan dengan melakukan analisis terhadap data-data yang sudah diperoleh untuk mengetahui unjuk kerja *load balancing web server* berbasis *container docker swarm* dan kemudian bisa ditarik kesimpulan.

3.3 TOPOLOGI JARINGAN

Topologi jaringan yang digunakan pada penelitian ini terdiri dari 1 *client* yang digunakan untuk melakukan simulasi pengujian data, 1 *server* yang digunakan untuk membangun infrastruktur sistem *load balancing* menggunakan *zevenet CE*. Nantinya akan dibangun 3 *virtual machine* yaitu 1 *load balancer* yang berfungsi untuk mengatur beban trafik yang akan diberikan ke *server* dan 2 *web server* sebagai *user interface* layanan. Uraian tersebut dapat dilihat dalam topologi jaringan pada gambar 3.2.



Gambar 3.2 Topologi Jaringan

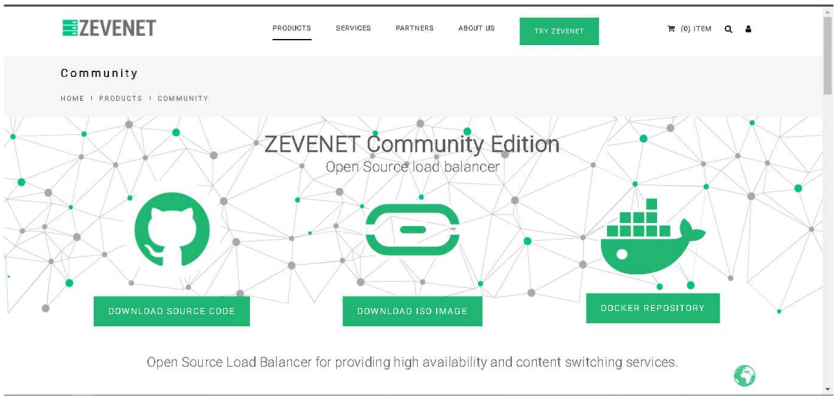
3.4 KONFIGURASI VIRTUAL MACHINE

3.4.1 INSTALASI LOAD BALANCING SERVER

Instalasi *load balancing server* dilakukan menggunakan *virtualbox* pada PC *server*. *Load balancing server* ini menggunakan *iso image zevenet CE 5.12.2* yang diunduh melalui *website zevenet*. Pertama-tama lakukan pembuatan *virtual machine load balancing server* dengan *virtualbox* kemudian sesuaikan spesifikasi *load balancing server* sesuai dengan spesifikasi perangkat virtual pada tabel 3.2 dengan cara berikut untuk menyesuaikan jumlah *vCPU*:

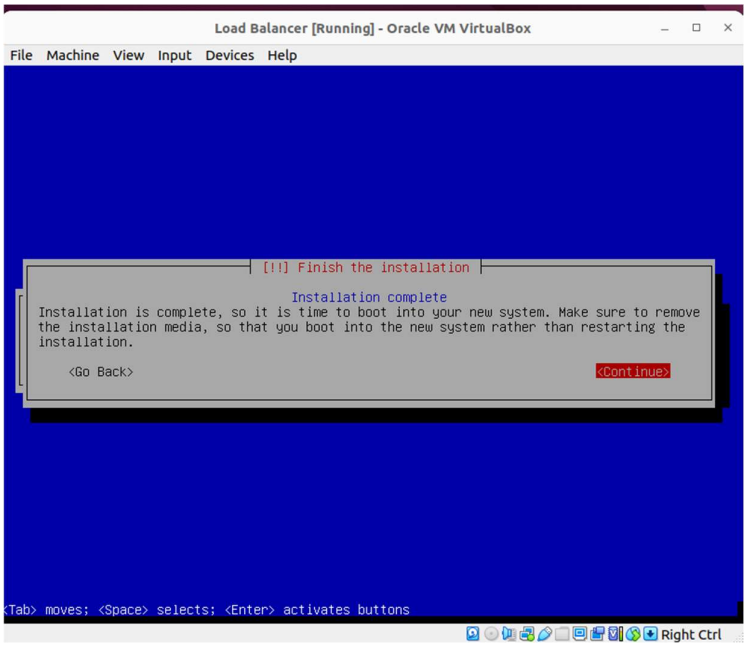
```
Klik tab setting → system →base memory → processor
```

Selanjutnya agar *virtual machine load balancing server* dapat digunakan, maka perlu dipasang (instalasi) *ISO image zevenet-ce* yang dapat diunduh melalui situs https://zevenet.com/products/community Tampilan laman situs tersebut dapat diilustrasikan pada gambar 3.3.



Gambar 3. 3 Tampilan situs zevenet

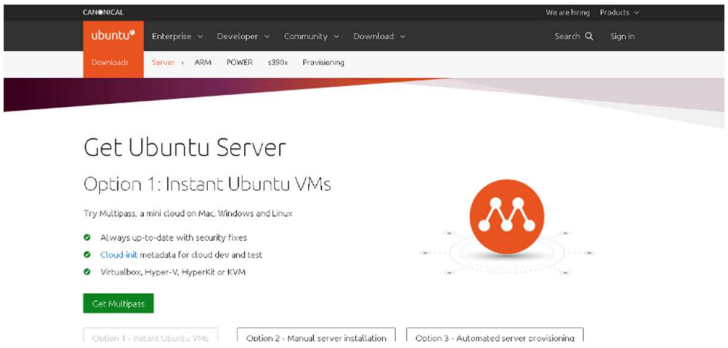
Dari tampilan situs pada gambar 3.3 tersebut, setelah *zevenet ce* berhasil dipasang maka akan muncul tampilan seperti pada gambar 3.4.



Gambar 3.4 Instalasi *Zevenet CE* telah selesai

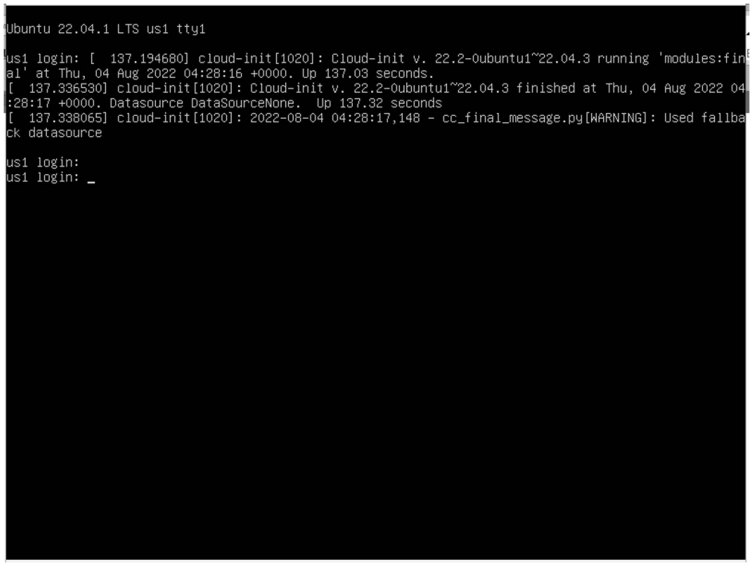
3.4.2 INSTALASI NODE SERVER

Perlu dilakukan pembuatan *virtual machine node server* dengan *virtualbox* yang mana spesifikasi *node manager* dan *node worker* disesuaikan dengan spesifikasi perangkat virtual pada tabel 3.2. Selanjutnya perlu dilakukan konfigugrasi *virtual machine manager* dan *worker* agar dapat digunakan. *Iso image ubuntu server* 22.04 dibutuhkan untuk menjalankan *node manager* dan *node worker*. *Iso image ubuntu server 22.04* dapat diunduh di situs https://ubuntu.com/download/server. Tampilan situs tersebut dapat diilustrasikan pada gambar 3.5



Gambar 3. 5 Tampilan situs ubuntu

Setelah *virtual machine node manager* dan *node worker* selesai dikonfigurasi, maka *virtual machine kedua node* tersebut dapat dijalankan dan menghasilkan tampilan seperti pada gambar 3.6.



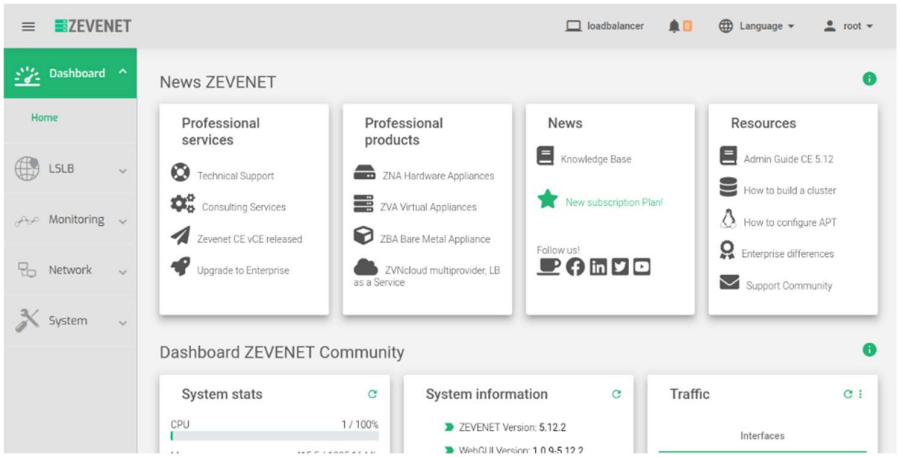
Gambar 3. 6 Tampilan Ubuntu Server pertama masuk

3.5 KONFIGURASI PERANGKAT

3.5.1 KONFIGURASI PC *SERVER*

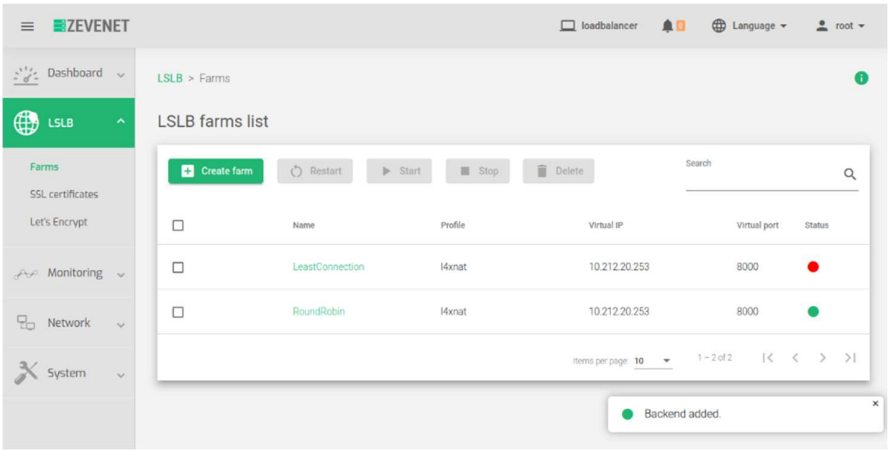
1. Konfigurasi *Load Balancer*

Konfigurasi *load balancer* dilakukan memalui *graphical user interface* (GUI) dengan akses alamat ip *load balancer* yaitu https://10.212.20.253:444 melalui *web browser*. Tampian *graphical user interface* (GUI) *dashboard zevenet* dapat diilustrasikan pada gambar 3.7.



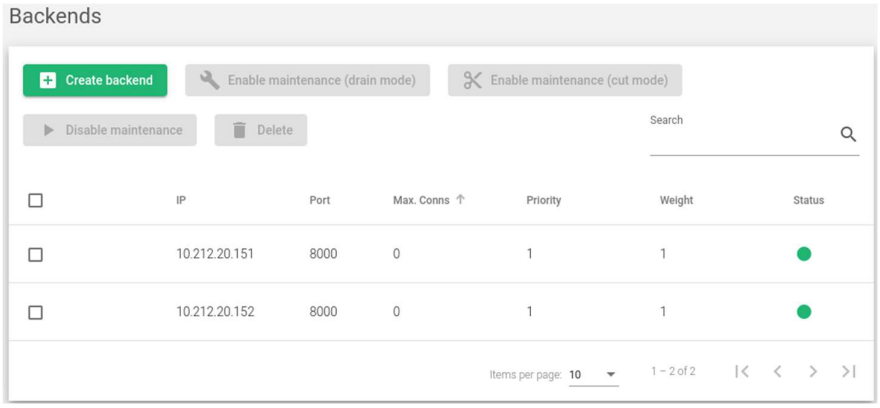
Gambar 3.7 Tampilan *Dashboard Zevenet CE*

Selanjutnya, dilakukan konfigurasi *farms* pada menu LSLB. Disini, *farms round robin* dan *least connection* ditambahkan. Daftar *farms* yang telah ditambahkan ditampilkan pada gambar 3.8.

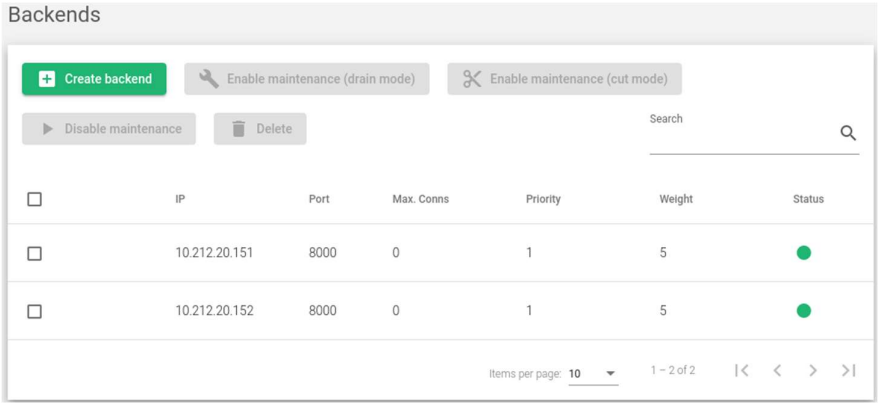


Gambar 3.8 Tampilan Daftar *farms* yang telah dibuat

Selanjutnya, perlu ditambahkan *backends* pada masing-masing *farms*. *Backend* yang didaftarkan adalah alamat ip *manager* yaitu 10.212.20.151 dan alamat ip *worker* yaitu 10.212.20.152. *backend* yang telah ditambahkan dapat dilihat pada gambar 3.9 untuk algoritma *round robin* dan 3.10 untuk algoritma *least connection*.



Gambar 3. 9 Daftar *Backend* Algoritma *Round Robin*



Gambar 3. 10 Daftar *Backend* Algoritma Least Connection

2. Konfigurasi *node manager* dan *node worker* (*node server)*

Setelah ubuntu server 22.04 berhasil terinstall langkah selanjutnya adalah dilakukan instalasi *docker* melalui *terminal* dengan perintah sebagai berikut:

```
#sudo apt-get update
#sudo apt-get install \
 ca-certificates \
 curl \
 gnupg \
 lsb-release
#sudo mkdir -p /etc/apt/keyrings
#sudo curl -fsSL https://download.docker.com/linus/ubuntu/gps | sudo
gpg --dearmor -o /etc/apt/keyrings/docker.gpg
#sudo echo \
“deb        [arch=$(dpkg        -print-architectur)        signed-
by=/etc/apt/keyrings/docker.gpg]                    https://download
.docker.com/linux/ubuntu \
$(lsb_release       -cs)       stable”       |       sudo       tee
/etc/apt/sources.list.d/docker.list > /dev/null
#sudo apt-get update
#sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
compose-plugin
```

D*ocker* yang telah berhasil diinstall perlu dilakukan verifikasi seperti pada gambar 3.11.

```
root@skripsmanager1:/home/manager1# docker --version
Docker version 20.10.17, build 100c701
```

Gambar 3. 11 Versi Docker

Selanjutnya dibangun sebuah *docker swarm cluster* dengan cara *node server* dijadikan *node manager* dan *node worker* dengan perintah:

```
#sudo docker swarm init
```

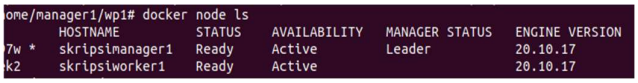
Konfigurasi demikian dilakukan pada *node manager*. Sedangkan *node worker* dilakukan konfigurasi dengan perintah sebagai berikut:

```
#sudo docker swarm join –token SWMTKN-1-
3n9r7bn9jp3i9h6bp4ejrukfkjme692zc43ivt78da24p11d72-
eif5l2boex6tjqwqqc6p6x0oi 10.212.20.151:2377
```

sehingga terbentuk sebuah *docker swarm cluster* dengan satu *node manager* dan satu *node worker*. Kedua *node* telah terdaftar dalam *cluster* dengan melihat kolom *manager status* dengan perintah:

```
#sudo docker node ls
```

maka akan muncul keterangan seperti pada gambar 3.12.



Gambar 3. 12 Informasi Node dalam satu kluster

3. *Deployment service*

Deployment service dilakukan pada *node manager* dengan perintah berikut:

```
# mkdir wordpress
# cd wordpress/
```

Perintah diatas digunakan untuk membuat folder *wordpress* dan berpindah ke dalam folder *wordpress* tersebut. Selanjutnya dibuat file *docker compose* di dalam folder *wordpress* dengan perintah berikut:

```
# nano wordpress.yaml
```

Isi file *docker compose* tersebut adalah sebagai berikut:

```
version: "3.9"

services:
  db:
    image: mysql:latest
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
```

```
    restart: always
    environment:
      WORDPRESS_DB_HOST: db
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress

volumes:
  db_data: {}
  wordpress_data: {}
```

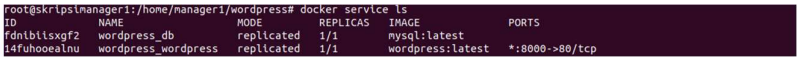
Selanjutnya dilakukan *deployment service* dengan perintah sebagai berikut:

```
#sudo docker stack deploy -c wordpress.yaml wordpress
```

cek *service* yang berjalan dengan cara:

```
#sudo docker service ls
```

maka akan muncul daftar service yang sedang berjalan seperti pada gambar 3.13.

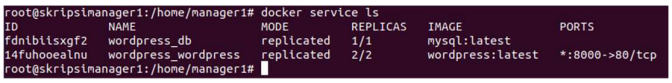


Gambar 3.13 Informasi service yang sedang berjalan

Selanjutnya, dilakukan *scale up container wordpress* agar terdapat masing-masing satu *container* pada *node manager* dan *node worker* dengan perintah berikut:

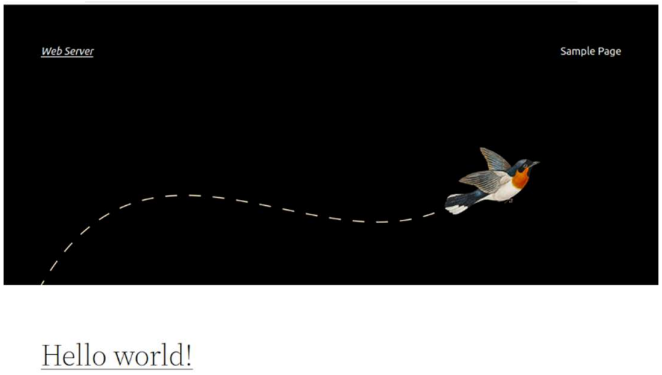
```
#sudo docker service scale wordpress_wordpress=2
```

Kemudian dilakukan kembali pengecekan *service* yang sedang berjalan, maka akan muncul daftar *service* yang baru seperti pada gambar 3.14.

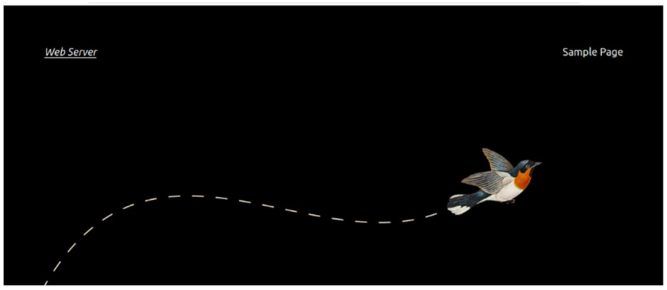


Gambar 3.14 Informasi Service yang sedang berjalan setelah dilakukan scale up

Selanjutnya, dilakukan uji akses *web server* melalui *web browser pc client* dengan alamat ip *node manager* yaitu http://10.212.20.151:8000 dan ip *node worker* yaitu http://10.212.20.152:8000. Maka akan ditampilkan halaman *web* seperti pada gambar 3.15 untuk *node manager* dan 3.16 untuk *node worker*.



Gambar 3. 15 Tampilan Halaman Web *Node Manager*



Gambar 3. 16 Tampilan Web *Node Worker*

3.5.2 KONFIGURASI PC *CLIENT*

Pertama-tama lakukan konfigurasi alamat IP pada sisi pc *client* dengan menggunakan perintah sebagai berikut:

```
#ip addr add [10.212.20.154] dev enp0s3
```

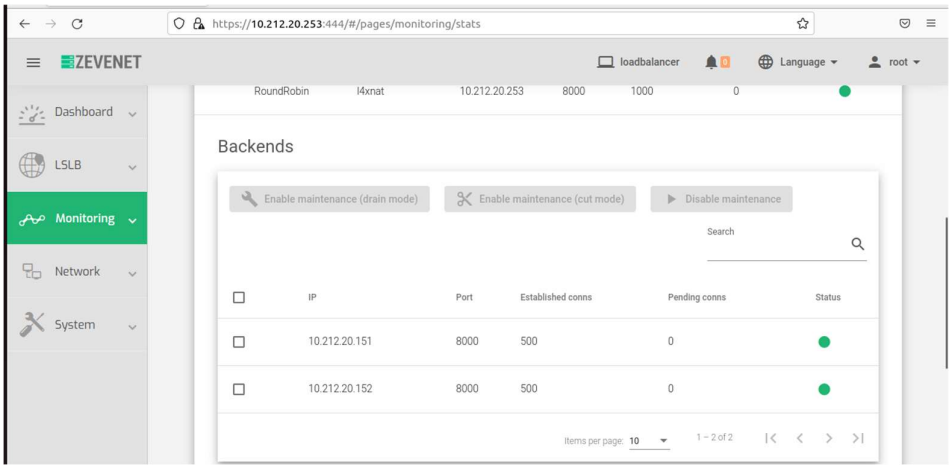
Selanjutnya, lakukan instalasi perangkat lunak penguji *h2load* benchmark pada pc *client* melalui *terminal* dengan perintah sebagai berikut:

```
#sudo apt-get install nghttp2
```

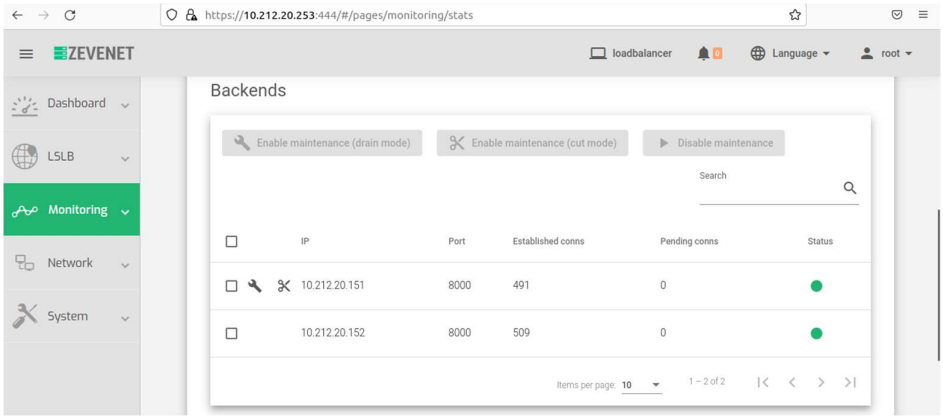
3.6 PENGUJIAN *LOAD BALANCER*

Uji coba dilakukan dengan menggunakan *h2load benchmark* yaitu dengan cara memberikan 5000 koneksi dengan 1000 konkurensi dari PC *client* ke alamat IP *load balancer* dan *port*nya yaitu http://10.212.20.253:8000. Hasil uji coba algoritma *round robin* terdapat pada gambar 3.17. Dari gambar 3.17 terdapat beberapa kolom yaitu kolom IP yang berisi informasi IP *node manager* yaitu 10.212.20.151 dan IP *node worker* yaitu 10.212.20.155, kolom *port* berisi informasi *port* dari *node manager* dan *node worker* yaitu 8000, kolom *establish conns* berisi informasi jumlah koneksi yang sedang dilayani oleh *node manager* dan *node worker* yaitu masing-masing 500 koneksi untuk kedua *node,* kolom *pending conns* berisi informasi tentang banyaknya koneksi yang tertunda pada *node manager* dan *node worker*, dan kolom *status* berisi informasi tentang kondisi *node manager* dan *node worker* yaitu warna hijau yang berarti kedua *node* tersebut aktif. Sedangkan hasil uji coba algoritma *least connection* terdapat pada gambar 3.18. sama dengan gambar 3.17, pada gambar 3.18 terdiri dari beberapa kolom, hanya saja pada kolom *establish conns* terdapat nilai yang berbeda dari gambar 3.17, yaitu 491 koneksi

dilayani oleh *node manager* dan 500 dilayani oleh *node worker*. Dari kedua gambar tersebut dapat diketahui bahwa kedua algoritma telah berjalan sebagaimana seharusnya.



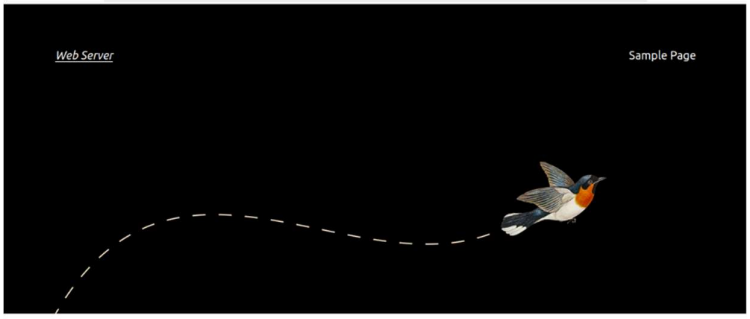
Gambar 3.17 Uji Coba Algoritma *Round Robin*



Gambar 3.18 Uji Coba Algoritma *Least Connection*

3.6.1 PENGUJIAN MELALUI *BROWSER CLIENT*

Uji coba dilakukan dengan mengakses alamat IP *load balancer* yaitu http://10.212.20.253:8000 melalui *web browser client*. Pada saat mengakses alamat ip *load balancer* secara otomatis akan terhubung dengan laman *web server* yang sudah ditambahkan sebagai *backend* pada *server load balancer*. Jika berhasil, *server load balancer* akan meneruskan *request* yang diterima ke *web server* dan *web server* akan merespon *request* tersebut dengan menampilkan halaman *web.* Tampilan *web server* dapat dilihat pada gambar 3.19.



Gambar 3. 19 Tampilan *web server*

3.6.2 PENGUJIAN MELALUI *H2LOAD BENCHMARK*

Pengujian *load balancing* menggunakan *software h2load* b*enchmark* dengan perintah sebagai berikut:

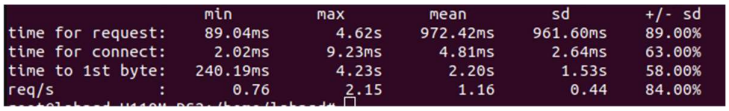
```
H2load -n 500 -c 100 --h1 http://10.212.20.253:8000/
```

Keterangan:

1. -n = Opsi ini digunakan untuk menentukan total *request* yang akan diberikan saat pengujian.
2. -c = Opsi ini digunakan untuk menentukan total *concurrent request* (*request* bersamaan) dalam satu waktu.
3. --h1 = opsi ini digunakan untuk menentukan versi protocol HTTP/1.1

3.6.2.1 HASIL *RESPONSE TIME*

Baris *Time to 1st byte* dan kolom *mean* pada hasil *h2load benchmark* menunjukkan nilai response time yang dapat dilihat pada gambar 3.20.

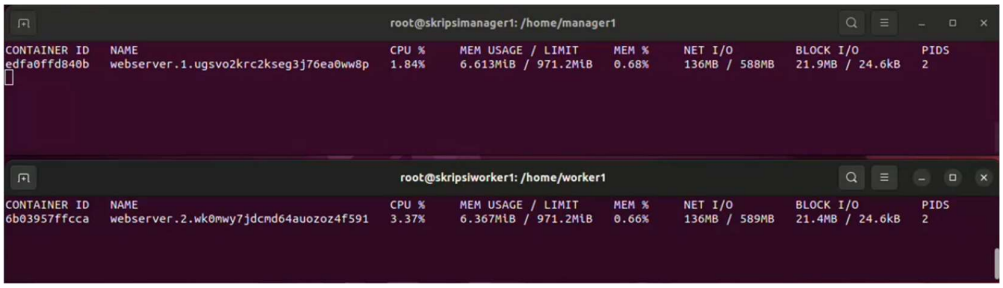


	min	max	mean	sd	+/- sd
time for request:	89.04ms	4.62s	972.42ms	961.60ms	89.00%
time for connect:	2.02ms	9.23ms	4.81ms	2.64ms	63.00%
time to 1st byte:	240.19ms	4.23s	2.20s	1.53s	58.00%
req/s :	0.76	2.15	1.16	0.44	84.00%

Gambar 3.20 Hasil *Response time* di *h2load benchmark*

3.6.2.2 HASIL CPU UTILIZATION DAN MEMORY USAGE

Dengan menggunakan perintah *docker stats* pada masing-masing *node* maka akan diperoleh gambaran seperti pada gambar 3.21.



Gambar 3.21 Hasil *CPU Utilization* dan *Memory Usage*

Pada gambar 3.21, terdapat beberapa kolom yaitu *Container ID* yang berisi informasi tentang ID kontainer, kolom *name* yang berisi tentang nama kontainer, CPU % yang berisi informasi tentang persentase penggunaan *CPU* yang dipakai oleh container, kolom *MEM USAGE / LIMIT* berisi informasi tentang banyaknya *memory* atau *memory usage* yang dipakai oleh kontainer dalam satuan *megabyte* (MB), kolom *MEM* % berisi informasi tentang persentase *memory* yang digunakan oleh kontainer, kolom NET I/O berisi informasi tentang banyaknya data yang dikirim dan diterima oleh kontainer melalui *network interface*nya, kolom BLOCK I/O berisi informasi tentang banyaknya data yang telah dibaca dan ditulis oleh kontainer dari blok perangkat di *host*, kolom PIDs berisi informasi tentang banyaknya proses atau *thread* yang dibuat oleh *container*.

3.7 SKENARIO PENGUJIAN

Seluruh *virtual machine* yang sudah dikonfigurasi melalui *console* diantaranya adalah *load balancing* sebagai pembagi beban *web server*, *web server* yang digunakan untuk menampilkan *service* kepada *client* yang sudah dilakukan *deployment nginx*. *Personal Computer client* berfungsi untuk menjalankan aplikasi *tool h2load benchmark* untuk mengirimkan *request* atau beban trafik ke sistem *load balancing*. Parameter *response time* dapat dilihat pada baris *time to first byte* pada *h2load benchmark*, kemudian untuk parameter *CPU Utilization, Memory Usage* didapatkan pada *vtop* yang berjalan pada masing-masing *node*. Pengujian kali ini bertujuan untuk mengetahui unjuk kerja layanan sistem *load balancing pada* masing-masing algoritma. Peneliti akan melakukan pengujian sebanyak tiga skenario. Pada masing-masing skenario dilakukan pengujian sebanyak 20 kali sehingga didapatkan nilai rata-rata pada parameter *response time* dan nilai tertinggi pada parameter *CPU Utilization* dan *memory* usage. Pengujian pemberian beban

dilakukan dengan *software h2load benchmark* yang kemudian hasil datanya akan dilihat melalui *h2load benchmark* yang *terinstall* pada pc *client* dan perintah *docker stats* yang pada *node manager* dan *node worker*. Jumlah koneksi merupakan total koneksi yang diberikan oleh *client* ke *server* sedangkan konkurensi adalah jumlah koneksi bersamaan yang dilakukan oleh *client* dalam satu waktu tertentu. Jumlah beban koneksi yang diberikan ditunjukkan pada tabel 3.4.

Tabel 3.4 Jumlah Beban Koneksi dan Banyaknya Pengujian

No	Jumlah Koneksi	Konkurensi	Banyaknya Pengujian
1	500	100	20
2	2000	400	20
3	5000	1000	20