BAB 3
METODE PENELITIAN

Penelitian ini menggunakan metode pendekatan kuantitatif yang bersifat objektif dengan jenis data numerik dan statistik. Kemudian metode lainnya yang digunakan pada penelitian ini adalah metode eksperimen dengan melakukan investigasi terhadap hubungan sebab akibat melalui uji coba yang dilakukan oleh peneliti.

3.1 PERANGKAT YANG DIGUNAKAN

3.1.1 PERANGKAT KERAS (*HARDWARE*)

Perangkat keras yang akan digunakan pada penelitian ini menggunakan 1 laptop sebagai *server* serta 1 laptop sebagai *client* dengan spesifikasi sebagaimana terdapat pada tabel 3.1.

Tabel 3.1 Spesifikasi Perangkat Keras

Server	OS	Proxmox VE 6.1
	Processor	Intel Core i5
	RAM	16 GB
	SSD	500 GB
Client	OS	Ubuntu 18.04 *Desktop*
	Processor	Intel inside
	RAM	4 GB
	HDD	500 GB

3.1.2 PERANGKAT LUNAK (SOFTWARE)

3.1.2.1 PERANGKAT *VIRTUAL*

Pada penelitian ini terdapat 4 perangkat *virtual* yang dibangun pada *Proxmox VE* yaitu : 1 *load balancer*, 2 *web server* dan 1 *database server*. Spesifikasi perangkat *virtual* tercantum pada tabel 3.2.

Tabel 3.2 Spesifikasi Perangkat *Virtual*

LOAD BALANCER	OS	Ubuntu *Server* 18.04
	RAM	4GB
	Harddisk	30GB
	Alamat IP	192.168.100.252
	CPU	1 Core
WEB SERVER 1	OS	Ubuntu *Server* 18.04
	RAM	4GB
	Harddisk	30GB
	Alamat IP	192.168.100.254
	CPU	2 Core
WEB SERVER 2	OS	Ubuntu *Server* 18.04
	RAM	2GB
	Harddisk	30GB
	Alamat IP	192.168.100.253
	CPU	1 Core
DATABASE SERVER	OS	Ubuntu *Server* 18.04
	RAM	4GB
	Harddisk	100GB
	Alamat IP	192.168.100.251
	CPU	1 Core

3.1.2.2 *SOFTWARE TOOL* DAN APLIKASI

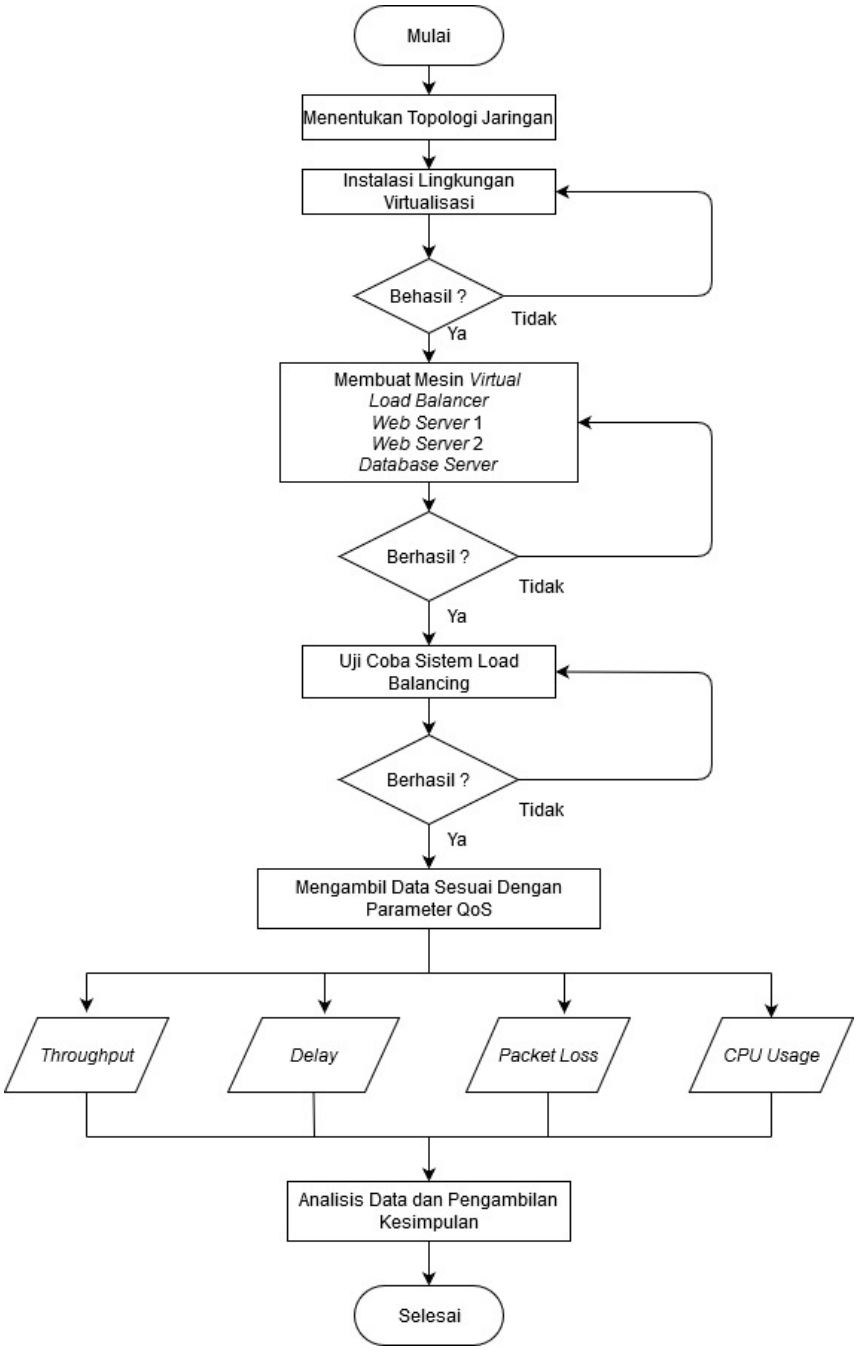
Perangkat lunak sebagai *tool* dan aplikasi yang digunakan pada penelitian ini ditunjukkan pada tabel 3.3.

Tabel 3.3 *Tools* dan Aplikasi

No	*Software*	Versi	Fungsi
1	Proxmox VE	6.1	*Virtual Environtment*
2	*Haproxy*	1.8.8	*Load balancing*
3	*Apache*	2.4.29	*Web server*
4	MariaDB	10.1.47	*Database server*
5	*Apache Benchmark*	2.6	Tool Pengujian L*oad Balancing Web Server*
6	*Wireshark*	3.2.2	*Network Analyzer*
7	Sysstat	11.6.1	*Resource Utilization*

3.2 ALUR PENELITIAN

Pada penelitian ini dilakukan melalui beberapa tahapan seperti pada diagram alur yang ditunjukkan pada gambar 3.1. Tahapan – tahapan tersebut dimulai dari membuat sistem hingga menganalisis data dan menarik kesimpulan.



Gambar 3.1 Diagram Alur Penelitian

Gambar 3.1 menunjukkan diagram alur perancangan sistem dalam penelitian ini. Tahap pertama menentukan topologi jaringan yang akan digunakan sebagai dasar dari sebuah arsitektur sistem jaringan *load balancing.* Sistem load

balancing pada penelitian ini diimplementasikan pada teknologi virtualisasi, yaitu Proxmox VE. Proxmox VE di-*install* sebagai *hypervisor type* 1 yang digunakan sebagai tempat atau wadah untuk membangun *server virtual*. Selanjutnya membuat mesin *virtual* untuk masing – masing *server*, seperti 1 buah *load balancing*, 2 buah *web server*, dan 1 buah *database server*. Kemudian dilakukan konfigurasi pada masing – masing *server* sesuai dengan fungsinya. Pada *server load balancing* di-*install* HAProxy, *server web* di-*install* Apache2 dengan menambahkan *Moodle* sebagai tampilan layanan *web server*, dan *server database* di-*install* MariaDB untuk penyimpanan data. Pada penelitian ini, sistem load balancing menggunakan metode penjadwalan weighted round robin. Selanjutnya dilakukan uji coba terhadap sistem *load balancing* yang berhasil dibangun.

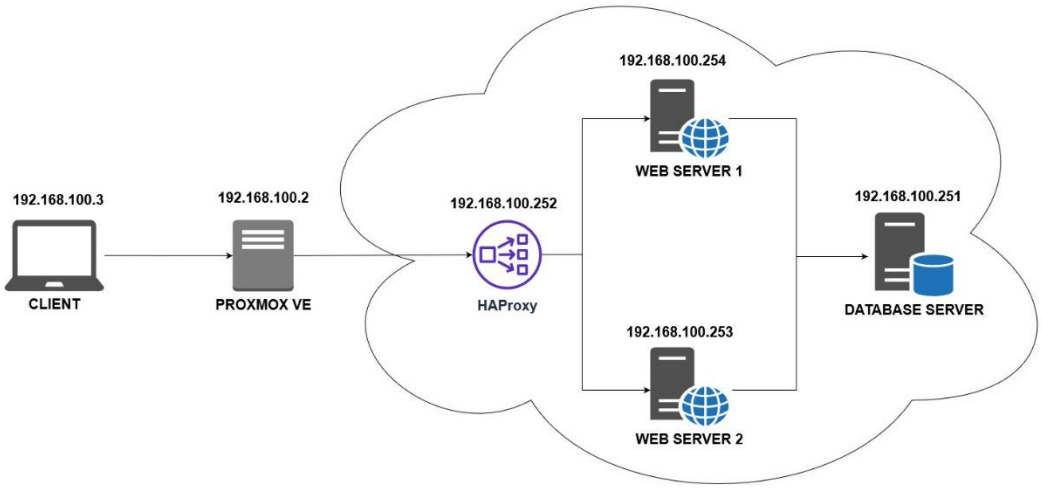
Apache Benchmark digunakan sebagai stress tool untuk memberikan beban trafik ke sistem load balancing dan aplikasi Wireshark digunakan untuk capture paket data selama proses transmisi data berlangsung. Pengukuran kualitas jaringan dilakukan dengan mengirim trafik HTTP *request* pada beberapa skenario pembagian beban *weighted round robin*, yaitu WRR 2:1, 3:1, 4:1, dan 5:1. Algoritma WRR 2:1, 3:1, 4:1, dan 5:1 adalah pembagian bobot beban server yang diberikan pada masing – masing server. Algoritma WRR 2:1 artinya *load balancer* akan memberikan beban 2 *request* ke *web server* 1 dan 1 *request* ke *web server* 2, selanjutnya skenario pembagian beban 3:1 artinya *load balancer* akan memberikan beban 3 *request* ke *web server* 1 dan 1 *request* ke *web server* 2, skenario pembagian beban 4:1 artinya *load balancer* akan memberikan beban 3 *request* ke *web server* 1 dan 1 kali *request* ke *web server* 2, dan skenario pembagian beban 5:1 artinya *load balancer* memberikan beban 5 kali request ke *web server* 1 dan 1 kali *request* ke *web server* 2.

Proses analisis dilakukan dengan mengukur nilai parameter *Quality of Service* (QoS), seperti *throughput, delay, packet loss*, dan CPU *usage*. Selanjutnya melakukan analisa terhadap data – data yang berhasil diperoleh dari pengujian untuk mengetahui performa pada *load balancing* menggunakan algoritma *weighted round robin* yang dijalankan pada lingkungan virtualisasi *hypervisor* tipe 1 yaitu Proxmox VE. Pengambilan kesimpulan dilakukan dengan memperhatikan dari

tujuan penelitian ini agar mendapatkan hasil yang sesuai. Pemberian saran dilakukan agar penelitian dengan tema yang sama dapat dikembangkan kembali.

3.3 TOPOLOGI JARINGAN

Topologi jaringan yang digunakan pada penelitian ini ditunjukkan pada gambar 3.2 yaitu topologi jaringan yang terdiri dari 1 buah *server* untuk membangun lingkungan virtualisasi dan 1 buah *client* yang digunakan untuk mengirim serta menerima data dari atau ke *server*. Membagi beban trafik ke *web server*, 2 buah *server web* untuk menampilkan layanan Moodle dan 1 buah *server database* untuk tempat penyimpanan data dari *web server*.



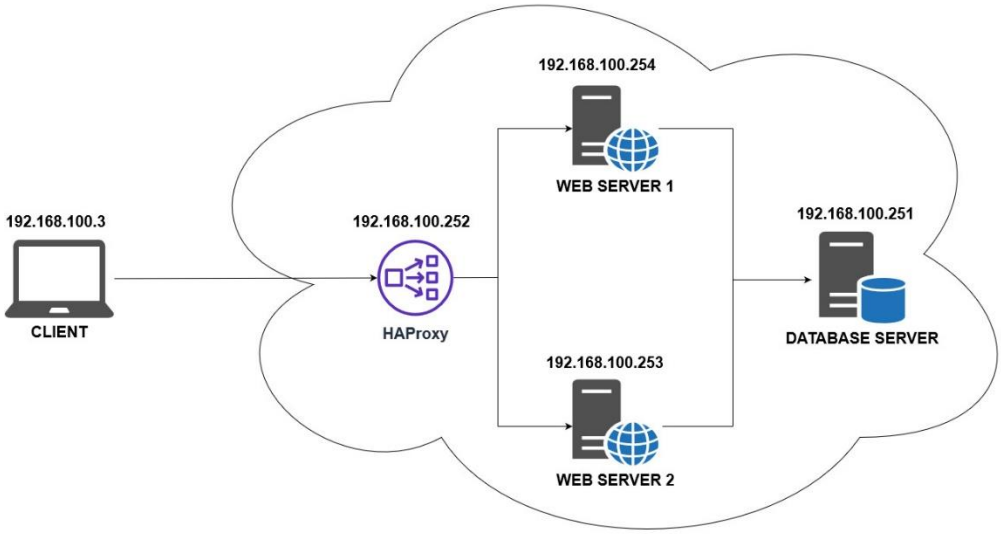
Gambar 3.2 Topologi Jaringan

3.4 SKENARIO PENGUJIAN

3.4.1 MEMBUAT SKENARIO JARINGAN

Pada gambar 3.3 yaitu topologi *logic* (topologi pengujian) yang terdiri dari 1 buah *client*, 4 buah *virtual machine* yang di-*install* pada lingkungan virtualisasi Proxmox VE. Pengujian dilakukan dengan memberikan beban request ke *server load balancer* yang nantinya trafik akan dibagi dan diteruskan ke *web server* berdasarkan algoritma penjadwalan *weighted round robin*. Seluruh *virtual machine* yang berhasil dibangun kemudian dikonfigurasi sesuai dengan fungsi masing – masing *server*. Dimana *server load balancer* akan di-*install* HAProxy yang berfungsi sebagai *frontend* untuk membagi beban trafik dan memetakan jalur trafik

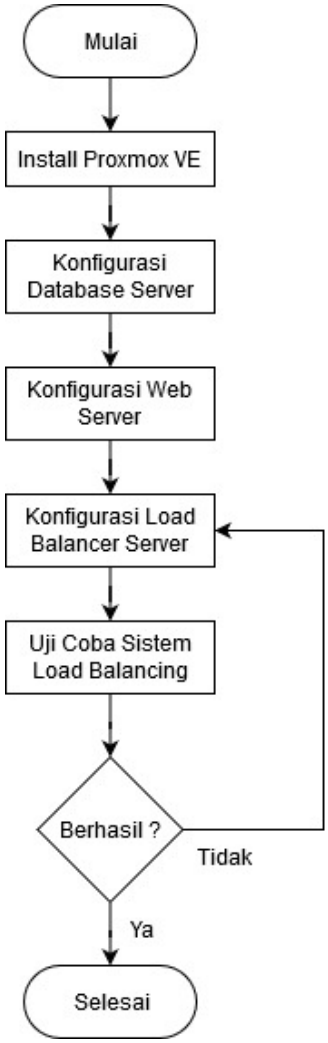
data yang diberikan oleh *client*, *server web* akan di-*install* Apache2 yang berfungsi sebagai *backend* untuk memberikan *service* berupa layanan *website* Moodle, dan *server database* akan di-*install* MariaDB yang berfungsi sebagai tempat penyimpanan data. Satu komputer *client* dengan OS Ubuntu *Desktop* 18.04 dipersiapkan untuk melakukan *capture* paket menggunakan aplikasi Wireshark terhadap jaringan yang telah diuji menggunakan Apache Benchmark.



Gambar 3.3 Topologi Jaringan *Logic* (Topologi Pengujian)

3.4.2 UJI COBA SISTEM LOAD BALANCING PADA PROXMOX VE

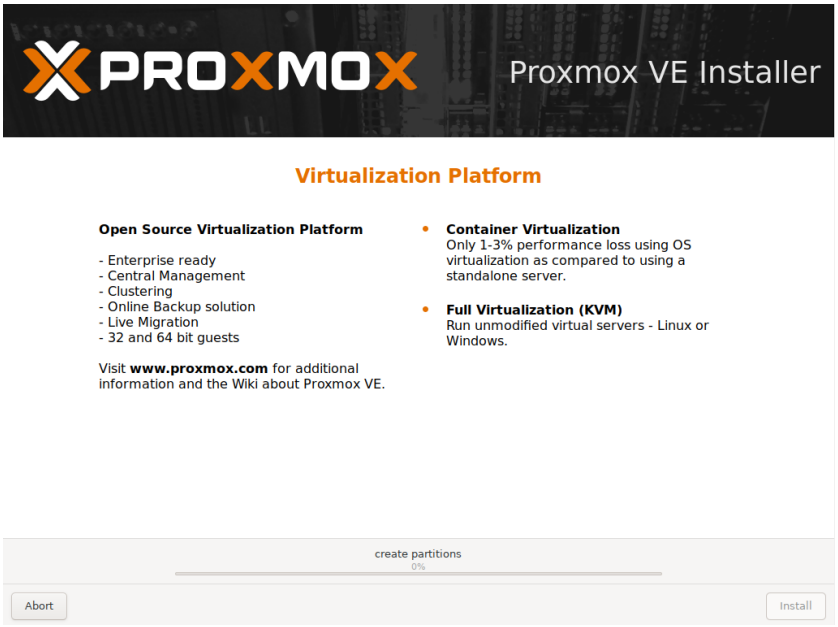
Uji coba sistem *load balancing* dilakukan untuk memastikan pembagian beban trafik dari *server load balancer* ke sisi *web server* berjalan dengan baik pada lingkungan virtualisasi Proxmox VE. Sebelum melakukan uji coba sistem *load balancing* pada Proxmox VE, peneliti terlebih dahulu melakukan konfigurasi pada masing – masing *server*. Konfigurasi meliputi *install* Proxmox VE sebagai lingkungan *server virtual*, konfigurasi *web server* dengan meng-*install* software Apache2, konfigurasi *database server* dengan meng-*install* MariaDB dan menghubungkannya ke *web server*, dan konfigurasi *load balancer* dengan meng-*install* HAProxy dan menjadikan *web server* sebagai *target* untuk distribusi trafiknya. Adapun beberapa tahapan instalasi sistem *load balancing* ditunjukkan pada gambar 3.4.



Gambar 3.4 Diagram Alur Konfigurasi Sistem *Load Balancing*

3.4.2.1 KONFIGURASI PROXMOX VE

Pada gambar 3.5 adalah proses instalasi Proxmox VE, sama halnya seperti meng-*install* sistem operasi pada umumnya. Proxmox VE di-*install* sebagai *hypervisor type* 1 atau berjalan langsung diatas perangkat fisik (*hardware*) tanpa memerlukan sistem operasi tambahan. Proxmox VE difungsikan untuk wadah atau tempat lingkungan virtualisasi, sehingga dapat membangun mesin – mesin *server* dalam bentuk *virtual* (*virtual machine*).



Gambar 3.5 *Install* Proxmox VE

3.4.2.2 KONFIGURASI *WEB SERVER*

Pada gambar 3.6 adalah perintah untuk meng-*install* Apache2 sebagai *web server* pada sistem operasi Ubuntu 18.04. *Web server* yang digunakan pada penelitian ini memerlukan 2 buah *virtual machine* dengan sistem operasi Ubuntu *server* 18.04. Pada kedua *web server* ini memiliki spesifikasi yang berbeda. Dimana pada *web server* 1 memiliki spesifikasi lebih besar yaitu CPU 2 *core* dan 4 GB RAM, sedangkan pada web server 2 memiliki spesifikasi yang lebih kecil yaitu CPU 1 *core* dan 2 GB RAM. Konfigurasi *web server* dilakukan pada setiap mesin *virtual* yang telah dibuat menggunakan beberapa perintah.

```
$ sudo apt update
$ sudo apt install -y apache2
```

Gambar 3.6 Perintah *Install Web Server* Apache2

Untuk mengubah tampilan antarmuka *web server* agar lebih menarik, peneliti mengubah tampilan layanan *website default* Apache2 menjadi layanan *Learning Management System* (LMS), yaitu Moodle. Dengan menambahkan folder Moodle pada directory /var/www/. Hal ini bertujuan agar *server* terlihat lebih menarik dan *web server* tidak terlihat kosong. Sehingga diberikan tampilan moodle agar proses pengujian terlihat lebih menarik dan *real*.

3.4.2.3 KONFIGURASI *DATABASE SERVER*

Pada gambar 3.7 adalah perintah untuk meng-*install* MariaDB sebagai *database server* pada sistem operasi Ubuntu 18.04. *Database server* pada penelitian ini menggunakan 1 buah *virtual machine* yang terpisah dengan *web server*. Sehingga diaharapkan antara *server database* dan *server web* bisa bekerja lebih optimal. *Database* yang digunakan pada penelitian ini adalah MariaDB yang dimana *database* ini salah satu bagian dari database MySQL. *Database* MariaDB dapat menyimpan *file content* dan beberapa data – data yang dimana menjadi keunggulan sebuah *database* ini dikarenakan dapat menyimpan *file content* dan data hanya dalam satu *database server*. Sehingga data yang dimasukkan oleh *user* melalui *web server* akan diteruskan dan disimpan kedalam *server database*. Konfigurasi dilakukan dengan menginstall *mariadb-server* pada *server database*.

```
$ sudo apt install -y mariadb
$ sudo systemctl restart mariadb
```

Gambar 3.7 Perintah *Install Database Server* MariaDB

Selanjutnya pada gambar 3.8 perintah untuk membuat *database* Moodle dan membuat *user* untuk masing – masing *web server* agar dapat terkoneksi dengan *database server*. Hal ini bertujuan agar *web server* dapat meneruskan *file content* yang di-*upload* oleh *user* ke *server database*.

```
$ sudo mysql root -p

CREATE DATABASE moodle DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
CREATE USER 'web1'@'192.168.100.254' IDENTIFIED BY 'rahasia';
CREATE USER 'web2'@'192.168.100.253' IDENTIFIED BY 'rahasia';
GRANT ALL PRIVILEGES ON moodle.* TO 'web1'@'192.168.100.254';
GRANT ALL PRIVILEGES ON moodle.* TO 'web2'@'192.168.100.253'
FLUSH PRIVILEGES;
```

Gambar 3.8 Perintah Untuk Menambahkan *Database* dan *User*

Karena *server database* dan *server web* berbeda perangkat, sehingga perlu mengubah alamat ip *bind-address* pada *server database*. Ubah alamat ip *bind-address* dari *localhost* (127.0.0.1) menjadi alamat ip *server database* (192.168.100.251) pada *file 50-server.cnf* di direktori */etc/mysql/mariadb.conf.d/*.

3.4.2.4 KONFIGURASI DAN UJI COBA SISTEM *LOAD BALANCING*

Pada gambar 3.9 adalah perintah untuk meng-install HAProxy sebagai *load balancer* pada sistem operasi Ubuntu 18.04. Konfigurasi sistem *load balancing* menggunakan *software* HAProxy yang dibangun pada sistem operasi Ubuntu *server* 18.04. *Load balancing* HAProxy ini menggunakan algoritma *weighted round robin* untuk mengatur pembagian beban kepada dua *web server* yang memiliki spesifikasi yang berbeda. Dengan penerapan algoritma *weighted round robin* diharapkan dapat menyeimbangkan pembagian beban trafik terhadap kedua *web server* yang memiliki spesifikasi yang berbeda. Sebelum menambahkan algoritma pada *server load balancing*, terlebih dahulu meng-*install software* HAProxy terlebih dahulu. Karena *software* HAProxy inilah yang akan menyediakan *service load balancer*.

```
$ sudo apt install -y haproxy
```

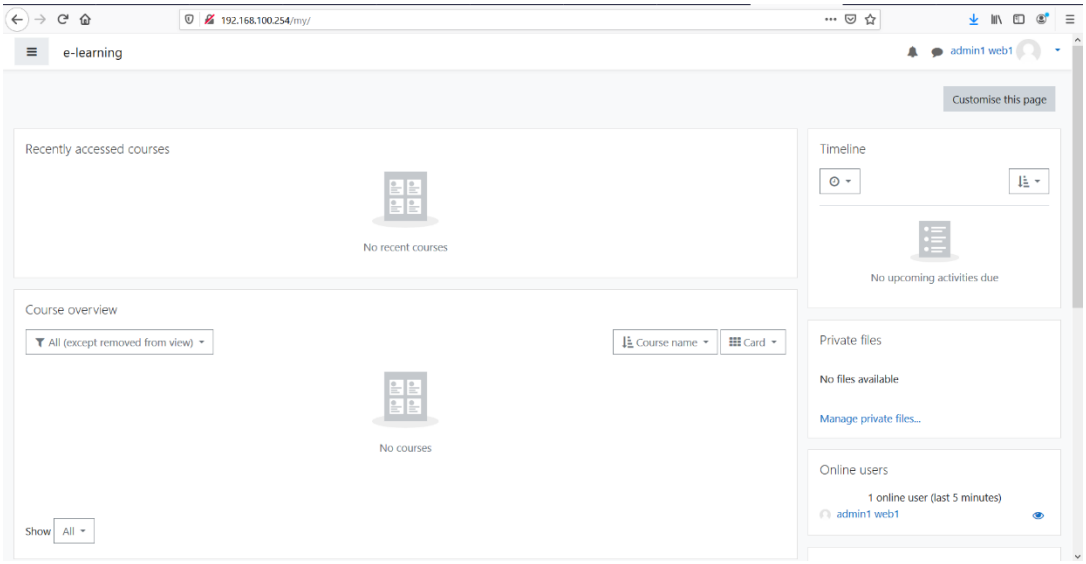
Gambar 3.9 Perintah *Install Load Balancer* HAProxy

Selanjtunya pada gambar 3.10 menambahkan beberapa perintah pada *file haproxy.cfg* yang terletak pada direktori */etc/haproxy/* untuk mengatur algoritma *load balancing* HAProxy. Dengan menambahkan alamat ip *target* yaitu alamat ip dari dua *web server* dan ditambahkan perintah *weight* untuk mengatur pembagian bobot (beban trafik) pada masing – masing *web server*. *Server web* yang memiliki spesifkasi lebih besar akan diberikan beban lebih banyak dan *server web* yang memiliki spesifikasi lebih kecil akan diberikan beban lebih sedikit.

```
mode http
balance round robin
server web1 192.168.100.254:80 check weight 2
server web2 192.168.100.253:80 check weight 1
```

Gambar 3.10 Perintah Untuk Menambahkan Algoritma *Load Balancing*

Uji coba dilakukan dengan mengakses alamat ip *load balancer* melalui *browser* pada pc *client*. Pada saat mengakses alamat ip *load balancer* secara otomatis akan terhubung dengan laman *web server* yang sudah ditambahkan sebagai *target* pada *server load balancer*, seperti pada gambar 3.11.



Gambar 3.11 Tampilan *Web Server* Moodle

Jika berhasil maka *server load balancer* akan langsung meneruskan *request* yang diberikan *user* ke *server web*. Oleh karena itu pada gambar 3.4 secara otomatis mengakses alamat ip *server* yang menjadi *target* dari *server load balancer*. Selanjutnya melakukan uji coba sistem *load balancing* dengan memberikan 30 *request* dengan mengakses kembali alamat ip *server load balancer*. Pengujian sederhana dilakukan untuk mengetahui pembagian trafik *load balancer* ke masing – masing *web server* menggunakan algoritma *weight round robin*. Hasilnya ditunjukkan pada Tabel 3.4 hingga Tabel 3.7.

Tabel 3.4 Hasil Uji Akses *Load Balancing* Algoritma WRR 2:1

Pengujian	***Web Server***		**Pengujian**	***Web Server***		**Pengujian**	***Web Server***	
Request	1	2	*Request*	1	2	*Request*	1	2
1			11			21		
2			12			22		
3			13			23		
4			14			24		
5			15			25		
6			16			26		
7			17			27		
8			18			28		
9			19			29		
10			20			30		

Pada uji coba mengakses *load balancing* menggunakan algoritma *weighted round robin* 2:1, didapatkan pembagian trafik pada pengujian 10 *request* pertama, yaitu 7 *request* diberikan ke *web server* 1 dan 3 *request* diberikan ke *web server* 2. Kemudian pada pengujian 10 *request* kedua mendapatkan jumlah beban yang sama seperti pengujian di 10 *request* pertama, yaitu 7 *request* diberikan ke *web server* 1 dan 3 *request* diberikan ke *web server* 2. Selanjutnya pada pengujian 10 *request* ketiga terjadi perubahan beban trafik, dimana *web server* 1 mendapatkan beban lebih ringan sebesar 6 *request* dan *web server* 2 mendapatkan beban lebih berat sebesar 4 *request*. Dengan hasil pengujian sederhana ini, *load balancing* menggunakan algoritma *weighted round robin* 2:1 memiliki 3 variasi pembagian beban.

Tabel 3.5 Hasil Uji Akses *Load Balancing* Algoritma WRR 3:1

Pengujian	***Web Server***		**Pengujian**	***Web Server***		**Pengujian**	***Web Server***	
Request	1	2	*Request*	1	2	*Request*	1	2
1			11			21		
2			12			22		
3			13			23		
4			14			24		
5			15			25		
6			16			26		
7			17			27		
8			18			28		
9			19			29		
10			20			30		

Pada uji coba mengakses *load balancing* menggunakan algoritma *weighted round robin* 3:1, didapatkan pembagian trafik pada pengujian 10 *request* pertama, yaitu 8 *request* diberikan ke *web server* 1 dan 2 *request* diberikan ke *web server* 2. Kemudian pada pengujian 10 *request* kedua terjadi perbedaan beban trafik, dimana *web server* 1 mendapatkan beban lebih ringan sebanyak 7 *request* dan *web server* 2 mendapatkan beban lebih berat sebanyak 3 *request*. Selanjutnya pada pengujian 10 *request* ketiga memiliki jumlah pembagian beban yang sama seperti 10 *request* pertama, yaitu 8 *request* diberikan ke *web server* 1 dan 2 *request* diberikan ke *web server* 2. Dengan hasil pengujian sederhana ini, *load balancing* menggunakan

algoritma *weighted round robin* 3:1 memiliki 2 variasi pembagian beban. Karena pada 10 *request* ketiga memiliki jumlah dan pembagian yang sama seperti 10 *request* pertama.

Tabel 3.6 Hasil Uji Akses *Load Balancing* Algoritma WRR 4:1

Pengujian	***Web Server***		**Pengujian**	***Web Server***		**Pengujian**	***Web Server***	
Request	1	2	*Request*	1	2	*Request*	1	2
1			11			21		
2			12			22		
3			13			23		
4			14			24		
5			15			25		
6			16			26		
7			17			27		
8			18			28		
9			19			29		
10			20			30		

Pada uji coba mengakses *load balancing* menggunakan algoritma *weighted round robin* 4:1, didapatkan pada pengujian 10 *request* pertama, kedua, dan ketiga memiliki pembagian beban yang sama, yaitu 8 *request* diberikan ke *web server* 1 dan 2 *request* diberikan ke *web server* 2. Dengan hasil pengujian sederhana ini, *load balancing* menggunakan algoritma *weighted round robin* 4:1 tidak memiliki variasi pembagian beban atau bersifat konstan pada setiap pembagian 10 *request* nya.

Tabel 3.7 Hasil Uji Akses *Load Balancing* Algoritma WRR 5:1

Pengujian	***Web Server***		**Pengujian**	***Web Server***		**Pengujian**	***Web Server***	
Request	1	2	*Request*	1	2	*Request*	1	2
1			11			21		
2			12			22		
3			13			23		
4			14			24		
5			15			25		
6			16			26		
7			17			27		
8			18			28		
9			19			29		
10			20			30		

Pada uji coba mengakses *load balancing* menggunakan algoritma *weighted round robin* 5:1, didapatkan pembagian trafik pada pengujian 10 *request* pertama, yaitu 9 *request* diberikan ke *web server* 1 dan 1 *request* diberikan ke *web server* 2. Kemudian pada pengujian 10 *request* kedua mendapatkan jumlah beban yang berbeda, dimana *web server* 1 mendapatkan beban lebih ringan sebesar 8 *request* dan *web server* 2 mendapatkan beban lebih berat sebesar 2 *request*. Selanjutnya pada pengujian 10 *request* ketiga memiliki jumlah pembagian beban trafik yang sama seperti pengujian 10 *request* kedua, yaitu 9 *request* diberikan ke *web server* 1 dan 1 *request* diberikan ke *web server* 2. Dengan hasil pengujian sederhana ini, *load balancing* menggunakan algoritma *weighted round robin* 5:1 memiliki 3 variasi pembagian beban.

3.4.3 PENGUJIAN QOS SISTEM *LOAD BALANCING WEB SERVER* BERDASARKAN JUMLAH BEBAN

Pengujian kali ini bertujuan untuk mengetahui performa kualitas (*Quality Of Service*) layanan sistem *load balancing* pada setiap algoritma. Peneliti akan melakukan pengujian sebanyak 3 skenario, yaitu 5000, 10000, dan 15000. Pada masing – masing skenario diberikan nilai *concurrency* sebanyak 10 koneksi dan dilakukan pengujian sebanyak 30 kali sehingga didapatkan hasil rata – rata pada setiap parameter QoS. Pengujian pemberian beban dilakukan dengan bantuan

software Apache Benchmark yang kemudian hasil datanya di-*capture* pada *wireshark* yang ada pada pc *client*. Jumlah beban yang diberikan ditunjukkan pada tabel 3.8.

Tabel 3.8 Jumlah Koneksi, *Concurrency* dan Banyaknya Pengujian

No	Jumlah Koneksi	*Concurrency*	Banyaknya Pengujian
1	5000	10	30
2	10000	10	30
3	15000	10	30