BAB III METODOLOGI

A. Waktu dan Tempat Penelitian

Waktu penelitian dimulai sejak tanggal 14 Januari 2023 sampai 14 Juni 2023 yang bertempat di fakultas matematika dan ilmu komputer UNUGHA Cilacap karena penelitian ini masih bersifat uji coba pengembangan skala laboratorium.

B. Alat dan Bahan

Pada penelitian ini dibutuhkan beberapa alat dan bahan untuk membuat metode *load balancing* pada sistem informasi desa berbasis *cloud*. Daftar alat dan bahan yang digunakan dalam melakakukan perancangan dan pembuatan sistem ini adalah sebagai berikut :

Kebutuhan perangkat keras:

Tabel 1 Kebutuhan perangkat keras

No	Perangkat keras	Keterangan
1	*Laptop (core i7 11th gen, RAM 8 Gb)*	Digunakan untuk mencari referensi dan membuat projek penelitian.

Kebutuhan perangkat lunak:

Tabel 2 Kebutuhan perangkat lunak

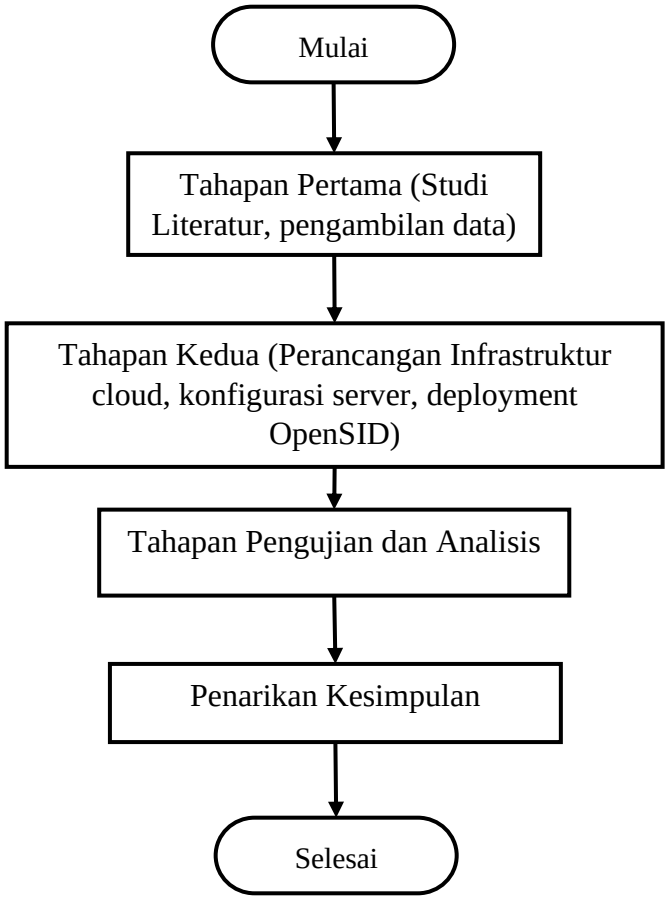
No	Perangkat lunak	Keterangan
1	Ubuntu server 22.04 LTS	Digunakan sebagai sistem operasi *server* di *cloud*
2	OpenSID versi 23.06	Digunakan sebagai *website* sistem informasi desa
3	Mozilla Firefox versi 110.0 64bit	Digunakan sebagai *browser* untuk menjalankan *website* sistem informasi desa
4	Google Chrome Versi 113.0.5672.127 (Build Resmi) (64 bit)	Digunakan sebagai *browser* untuk membuat infrastruktur *cloud* dan untuk menjalankan *website* sistem informasi desa
5	Apache2 versi 2.4.52	Digunakan sebagai *web server*
6	PHP versi 7.4	Digunakan sebagai bahasa pemrograman untuk menjalankan *website* sistem informasi desa

No	Perangkat lunak	Keterangan
7	MariaDB versi 10.6.12	Digunakan sebagai *database server*
8	HAProxy versi 2.4.22	Digunakan sebagai *load balancer* pada *server*
9	Apache JMeter versi 5.5	Digunakan sebagai penguji kinerja *web server* Apache*2*
10	Windows PowerShell versi 5.1	Digunakan sebagai *software* untuk remote *server cloud*

C. Prosedur Penelitian

1. Diagram Alir

   Dalam penelitian ini dilakukan beberapa tahapan mulai dari tahapan pertama, tahapan kedua, tahapan pengujian dan analisis, penarikan kesimpulan. Dimana tahapan pertama yaitu pemilihan topik penelitian, studi literatur, dan pengambilan data. Tahapan kedua yaitu perancangan perangkat lunak. Selanjutnya tahapan pengujian dan analisis hasil data pengujian, penarikan kesimpulan dan selesai. Berikut beberapa tahapan yang dapat dilihat pada diagram alir dibawah ini:



Gambar 1 Diagram alir penelitian

2. Tahapan Pertama

Pada tahapan ini, melakukan beberapa langkah yaitu :

a. Memilih Topik Penelitian

Dalam pemilihan topik, peneliti memilih topik penelitian metode *load balancing* sistem informasi desa berbasis *cloud*.

b. Studi Literatur

Pada tahap ini, peneliti melakukan identifikasi tentang kebutuhan apa saja yang akan digunakan dalam penelitian. Pengumpulan data dilakukan melalui studi literatur tersebut dan selanjutnya menganalisa data-data yang telah dikumpulkan.

c. Pengambilan Data

Pengambilan data ini dilakukan untuk mengisi data ataupun konten pada *website* desa. Teknik yang digunakan dalam pengambilan data yaitu teknik pengambilan sampel acak berdasar area atau wilayah (*Cluster Random Sampling*), dimana nantinya peneliti akan mengambil beberapa sampel data dari Desa Planjan yang akan dimasukan ke *website* desa. Selain itu, juga akan dimasukan beberapa konten artikel ke *website* desa.

3. Tahapan Kedua

Tahapan kedua yaitu perancangan perangkat lunak. Pada perancangan perangkat lunak, peneliti melakukan perancangan metode *load balancing* yang akan diimplementasikan di teknologi *cloud* yaitu di AWS. Ada beberapa tahapan untuk melakukan perancangan perangkat lunak yaitu:

a. Perancangan Infrastruktur *Cloud*

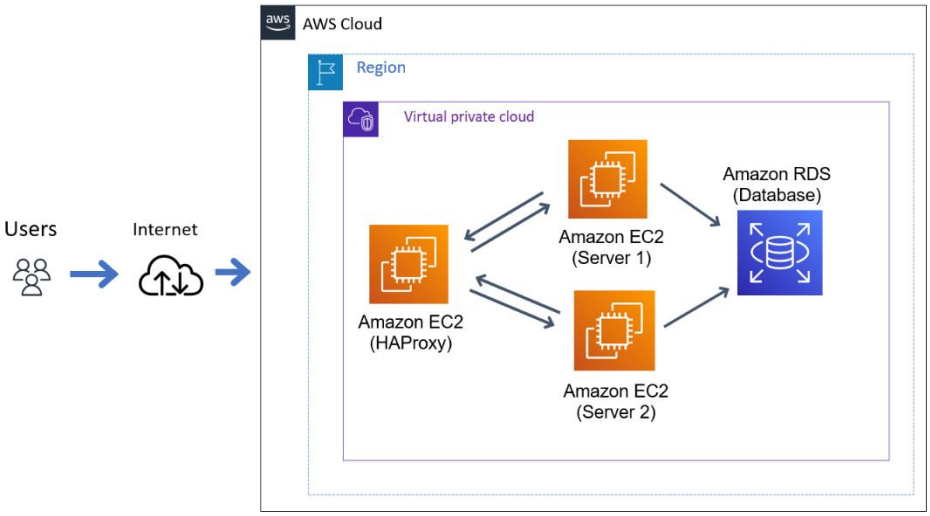
Berikut adalah arsitektur *load balancing* pada AWS dengan 2 *node server* dan 1 *server load balancer*:

1) Buat dua *instance EC2 (node server)* dengan konfigurasi yang diperlukan dan terhubung dengan jaringan *Virtual Private Cloud (VPC)* dan *security group* yang sama, tujuannya adalah agar semua server berada pada jaringan yang sama.
2) Konfigurasi setiap *node server* dengan kebutuhan untuk deployment OpenSID seperti menginstall Apache2, PHP, dan MariaDB, termasuk konfigurasi NFS (Network File System) *server* agar semua *server* bisa saling sinkron.
3) Buat satu *instance EC2 (server load balancer)* dan hubungkan dengan jaringan *VPC* dan *security group* yang sama.
4) Buat satu layanan Amazon *RDS* sebagai *database* dari OpenSID.
5) Hubungkan semua *server* pada *database* yang sama yaitu di *RDS*.

Alasan peneliti memakai dua *node server* adalah karena untuk melakukan *load balancing* dengan HAProxy minimal ada dua *server*.

Dengan menggunakan lebih dari satu *server*, *load balancer* dapat membagi beban kerja (*workload*) antara *server-server* tersebut. Hal ini dapat meningkatkan kinerja dan keandalan sistem secara keseluruhan, karena apabila satu *server* mengalami masalah atau terjadi *overload*, maka *load balancer* dapat mengarahkan *traffic* ke *server* yang lain yang masih beroperasi dengan normal. Kemudian alasan kenapa menggunakan *Virtual Private Cloud (VPC)* dan *security group* yang sama adalah agar semua *server* saling terhubung dalam satu jaringan yang sama dan bisa saling berkomunikasi.

Dengan arsitektur ini, setiap permintaan dari klien akan ditangani oleh *server load balancer*, yang akan memproses permintaan dan membagi beban kerja di antara *node server* yang ada. Ini akan meningkatkan kinerja dan ketahanan aplikasi *web* dengan menyeimbangkan *traffic* di antara *node server.* Berikut merupakan gambaran arsitektur *cloud* yang digunakan.



Gambar 2 Arsitektur *cloud*

b. Konfigurasi *Node Server*

Di tahap ini, peneliti akan menginstall beberapa *software* pendukung pada *node server* yang dibutuhkan untuk melakukan *deployment website* OpenSID. Beberapa *software* pendukung yang dibutuhkan yaitu: Apache*2 versi 2.4.52* yang berfungsi sebagai *web server*, kemudian MariaDB *versi 10.6.12* yang berfungsi sebagai *database server*, dan terakhir yaitu PHP *versi 7.4* yang berfungsi sebagai bahasa pemrograman *web* untuk menjalankan *file script* PHP pada *website* OpenSID. Setelah semua *software* pendukung tepasang, langkah selanjutnya yaitu mengonfigurasi NFS *host* dan *client*, dimana yang akan berperan sebagai NFS *host* adalah *server* dari *HAProxy* dan NFS *client* adalah kedua *node server*. Setelah itu mengunduh kode sumber OpenSID dari repositori

GitHub menggunakan Git dengan *git clone* ke alamat (*https://github.com/*OpenSID/OpenSID), dan *copy file* OpenSID ke direktori yang terhubung dengan NFS *server* agar semua *file* bisa saling sinkron.

c. Konfigurasi HAProxy

Pada tahap ini, yang dilakukan adalah menginstall HAProxy pada *server load balancer* kemudian melakukan konfigurasi pada HAProxy. Untuk *setting*an HAProxy, langkah pertama yaitu buka *file* konfigurasi HAProxy yang ada di */etc/haproxy/haproxy.cfg* kemudian konfigurasikan parameter dasar seperti global, *defaults, frontend,* dan *backend* seperti contoh berikut:

```
global
    log /dev/log    local0
    log /dev/log    local1 notice
    chroot /var/lib/haproxy
    user haproxy
    group haproxy
    daemon

defaults
    log     global
    mode    http
    option  httplog
    option  dontlognull
    option forwardfor
    timeout connect 5000
    timeout client  50000
    timeout server  50000

frontend myapp
    bind *:80
    default_backend myapp_servers

backend myapp_servers
    balance roundrobin
    server node1 10.0.0.1:80 check
    server node2 10.0.0.2:80 check
```

Gambar 3 Contoh konfigurasi HAProxy

Penjelasan konfigurasi:

1) global: Konfigurasi global untuk HAProxy.
2) *defaults*: Konfigurasi *default* untuk HAProxy, termasuk waktu *timeout* koneksi.
3) *frontend myapp*: Menerima permintaan *HTTP* pada *port* 80.
4) *default_backend myapp_servers*: Mengarahkan permintaan ke *backend* dengan nama *myapp_servers*.
5) *backend myapp_servers*: Konfigurasi *backend* untuk aplikasi *myapp*.
6) *balance* round robin: *Load balancing* menggunakan algoritma *round-robin*.

7) *server node1* 10.0.0.1:80 *check*: Menambahkan *node* 1 ke *backend* dengan alamat *IP* 10.0.0.1 dan *port* 80, dan menandai *server* tersebut sebagai "*check*" agar HAProxy melakukan pengecekan pada *server* sebelum mengirimkan permintaan.
8) *server node2* 10.0.0.2:80 *check*: Menambahkan *node* 2 ke *backend* dengan alamat *IP* 10.0.0.2 dan *port* 80, dan menandai *server* tersebut sebagai "*check*" agar HAProxy melakukan pengecekan pada *server* sebelum mengirimkan permintaan.

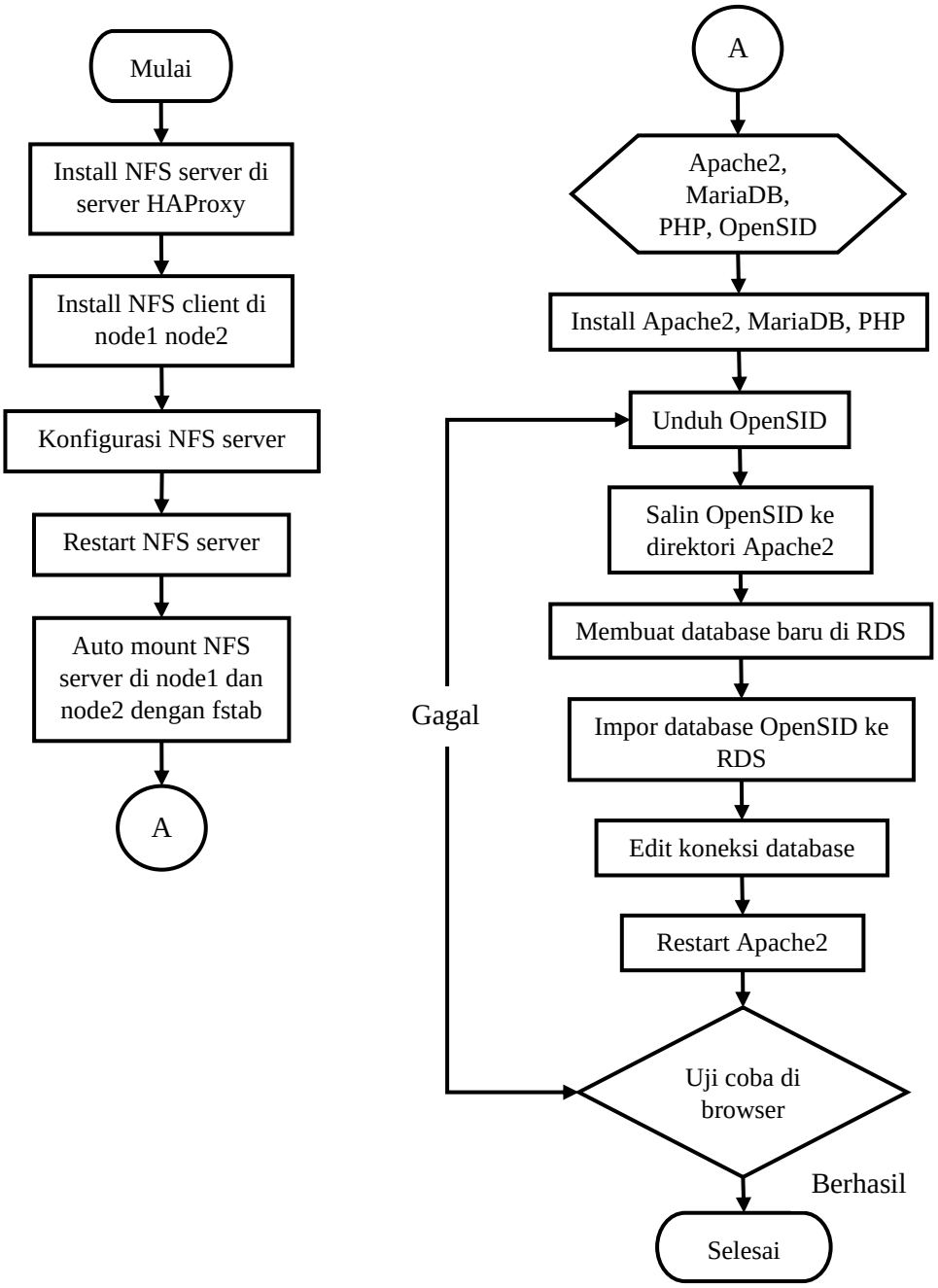
d. *Deployment* OpenSID

Langkah pertama adalah menyiapkan NFS *host* dan *client* untuk ketiga *server* dimana *server* HAProxy akan menjadi *host* sedangkan *node1 node2* akan menjadi *client*, tujuannya adalah agar nanti ketika OpenSID berhasil diinstall maka ketiga *server* memiliki *file* yang sama dan saling sinkron. Langkah selanjutnya yang harus dilakukan yaitu mengunduh kode sumber OpenSID dari repositori GitHub menggunakan Git melalui alamat (*https://www.github.com/*OpenSID/OpenSID*)*. Setelah terunduh, pindahkan atau salin *folder* OpenSID ke dalam direktori Apache*2* yang terletak di */var/www/html*. Selanjutnya membuat *database* baru pada Amazon *RDS* menggunakan MariaDB dan buatlah *user* dengan hak akses ke *database* tersebut. Kemudian *impor file database* dari OpenSID ke dalam *database* baru di *RDS* yang telah dibuat. Langkah selanjutnya yaitu mengedit koneksi *database* pada *file config.php* di dalam *folder* OpenSID dengan informasi *database* dan *user* yang telah dibuat sebelumnya. Contohnya:

```php
<?php
$servername = "localhost"; //nama server database
$username = "username"; //nama pengguna database
$password = "password"; //password pengguna database
$dbname = "nama_database"; //nama database
```

Gambar 4 Contoh *script* PHP koneksi ke *database*

Setelah selesai mengedit kemudian simpan *file* tersebut lalu *restart service* Apache*2* untuk memastikan bahwa semua perubahan konfigurasi sudah diterapkan. Kemudian buka *browser* dan akses halaman *web* OpenSID dengan alamat *IP* atau nama *domain* dari *node server EC2*. Berikut merupakan *flowchart* proses *deployment* OpenSID pada *server EC2* dibawah ini:



Gambar 5 *Flowchart deployment* OpenSID

e. Menentukan Algoritma HAProxy

Pada HAProxy, terdapat tujuh algoritma *balance* yang dapat digunakan untuk mendistribusikan beban secara merata di antara *server-server backend* yang tersedia. Berikut adalah beberapa algoritma *balance* yang dapat digunakan pada HAProxy:

1) *Round robin (default)*: Algoritma ini mendistribusikan beban secara merata ke setiap *server backend* dengan memilih *server* secara bergiliran. Setiap *request* diteruskan ke *server* berikutnya dalam daftar *server backend*.

2) Least connections: Algoritma ini mendistribusikan beban ke *server* dengan jumlah koneksi yang paling sedikit saat itu. Ini berguna untuk menghindari *overload* pada *server backend* yang terlalu sibuk.
3) Source: Algoritma ini mendistribusikan beban berdasarkan alamat *IP* klien yang mengirimkan permintaan. Setiap alamat *IP* akan dikirim ke *server backend* yang sama setiap kali klien membuat permintaan.
4) URI: Algoritma ini mendistribusikan beban berdasarkan *path* permintaan. Misalnya, jika ada dua *server backend* yang menyediakan layanan berbeda (contoh: /*api* dan /*website*), algoritma URI dapat digunakan untuk memastikan permintaan yang masuk ke *backend* yang sesuai.
5) Random: Algoritma ini memilih *server backend* secara acak untuk mendistribusikan beban.
6) RDP-cookie: Algoritma ini mendistribusikan beban berdasarkan *cookie RDP* yang dikirimkan oleh klien. Algoritma ini biasanya digunakan untuk memastikan bahwa koneksi *RDP* berulang ke *server* yang sama.
7) Custom: HAProxy juga memungkinkan pengguna untuk menulis algoritma penyeimbang beban khusus mereka sendiri.

Algoritma yang akan digunakan pada penelitian ini ada 4 yaitu, round robin, least connections, source, dan URI. Alasan menggunakan algoritma tersebut adalah karena dalam kasus untuk *website*, 4 algoritma tersebut cocok digunakan dan bisa dibandingkan agar terpenuhi kebutuhan algoritma apa yang akan digunakan untuk *website* OpenSID. Selain itu, 4 algoritma tersebut memiliki keunggulan dan fitur khususnya masing-masing[23]. Misalnya algoritma source dan URI yang memiliki fitur *session persistence* (ketahanan sesi), round robin yang bisa memastikan setiap *server backend* menerima jumlah koneksi yang seimbang, dan least connections yang bisa mengarahkan lalu lintas ke *server* dengan beban kerja yang paling rendah.

4. Tahapan Pengujian dan Analisis

Pada tahapan ini melakukan pengujian performa dan juga menganalisis hasil data pengujian.

a. Pengujian Performa

Di tahap ini akan dilakukan *performance testing* pada *server* HAProxy dengan menggunakan *software* atau aplikasi Apache JMeter. *Performance testing* merupakan proses menentukan kecepatan atau efektivitas, jaringan program komputer, perangkat lunak atau perangkat

keras. *Performance testing* digunakan untuk mengukur kemampuan performa suatu aplikasi sampai suatu batas tertentu. *Performance testing* mengukur atribut kualitas sistem, seperti penggunaan *resource*, *speed*, *scalability*, *stability*, dan *reability*. Dalam *performance testing* tidak hanya fitur dan fungsi dari sistem saja yang harus di *test*, tetapi juga sangat penting untuk memastikan aplikasi dapat diakses dengan mudah oleh *user* (*user friendly*). Kadang suatu aplikasi mengalami masalah yakni seperti *load* yang terlalu lama karena banyak *user* sehingga *user* malas mengaksesnya. Oleh karena itu *performance testing* sangat diperlukan. Ada 2 *performance testing* yang akan dilakukan menggunakan *software* Apache JMeter pada HAProxy, yaitu: *Load test* dan *Soak test*. Untuk penjelasan *load test* dan *soak test* akan dijelaskan pada Tabel 4 *performance testing* dibawah ini.

Tabel 3 *Performance testing*

No	*Performance testing*	Keterangan
1	*Load test*	*Load test* merupakan sebuah pengujian untuk memeriksa kemampuan dari aplikasi dalam melakukan *load* aplikasi atau *website*, gunanya agar dapat mengetahui beban dari aplikasi atau *website* ke *database* atau *server*.
2	*Soak test*	*Soak test* merupakan pengujian yang digunakan untuk mengetahui kemampuan sistem dalam menangani data secara terus-menerus dengan beban dan waktu tertentu.

Semua pengujian ini dilakukan dengan menggunakan algoritma *balance* yang berbeda-beda dari HAProxy. Algoritma *balance* yang akan diuji yaitu round robin, least connections, source, dan URI. Setelah melakukan pengujian maka langkah selanjutnya yaitu menganalisis data agar didapatkan kesimpulan dari hasil pengujian.

b. Analisis Data

Menganalisis data adalah proses mengubah data mentah menjadi informasi yang dapat dipahami dan digunakan untuk membuat keputusan atau menyajikan hasil penelitian. Hasil data pengujian yang dilakukan menggunakan Apache JMeter pada penelitian ini akan ditampilkan dalam bentuk tabel. Apache JMeter juga menyediakan tabel hasil yang dapat menampilkan data seperti rata-rata waktu respon, *throughput*, dan kesalahan selama pengujian. Tabel hasil dapat digunakan untuk melihat

data secara rinci dan membandingkan performa aplikasi pada setiap pengujian.

Selanjutnya data-data tersebut akan dibandingkan dengan algoritma-algoritma yang di uji, yang disajikan dalam bentuk tabel dan grafik untuk melihat algoritma mana yang paling cocok diterapkan di *website* OpenSID. Langkah selanjutnya adalah membuat kesimpulan algoritma mana yang akan diterapkan pada HAProxy setelah didapatkan hasil yang paling baik.

D. Jadwal Penelitian

Tabel 4 Jadwal penelitian

No	Nama Kegiatan	Tahun 2023 Bulan ke-					
		1	2	3	4	5	6
1	Tahap Persiapan						
	a. Studi Literatur						
	b. Membangun Metode Load Balancing						
	c. Merancang Arsitektur Cloud						
	d. Implementasi Load Balancing Pada Arsitektur Cloud						
	e. Seminar Proposal						
2	Tahap Penelitian						
	a. Pengujian Performa						
	b. Pengambilan Data						
	c. Analisis Data						
	d. Evaluasi						
	e. Penulisan Skripsi						
3	Tahap Akhir						
	a. Sidang Skripsi						
	b. Publikasi Ilmiah						