BAB III

METODOLOGI PENELITIAN

3.1 SUBJEK DAN OBJEK PENELITIAN

Dalam sebuah penelitian dibutuhkan sebuah subjek penelitian, subjek penelitian pada penelitian ini adalah *single board computer* Raspberry Pi 4 Model B sebagai *computer cluster* yang dapat menjalankan aplikasi berbasis arsitektur *microservices* yang dibangun menggunakan Bahasa pemrograman Python. Selain dibutuhkannya subjek pada sebuah penelitian, dibutuhkan juga objek pada penelitian ini. Objek penelitian pada penelitian ini adalah performa aplikasi berbasis *microservices* yang berjalan pada *container orchestration* Kubernetes menggunakan *computer cluster* Raspberry Pi 4.

3.2 ALAT DAN BAHAN PENELITIAN

3.2.1 Alat Penelitian

Dibutuhkan alat pada penelitian ini untuk dapat melakukan pengujian performa aplikasi berbasis *microservices.* Alat tersebut terdiri dari perangkat keras (*hardware)* yang berfungsi untuk membangun infrastruktur sistem untuk dapat menjalankan perangkat lunak (*software)*. Perangkat lunak disini merupakan Alat dan Bahan untuk membangun *computer cluster* pada arsitektur sistem dan untuk melakukan pengujian performa. Spesifikasi perangkat lunak dan perangkat keras dijelaskan pada *point-point* dibawah.

1. Perangkat Keras (*hardware)*

Tabel 3. 1 Perangkat Keras

No	*Hardware*	Jumlah	Kegunaan
1.	Tplink Switch 8 Port 10/100Mbps - TL-SF1008D	1	Digunakan untuk menghubungkan dan menyebarkan koneksi jaringan antar perangkat melalui kabel RJ45.
2.	RJ45 CAT 6 Ethernet Cable	5	Digunakan sebagai penghubung koneksi antar perangkat.
3.	*Computer cluster* Raspberry Pi 4 Model B 8 dan 4 GB RAM	3	Komponen utama penelitian yang berguna untuk membangun *cluster* Kubernetes.
4.	Memory Card Samsung EVO PLUS 64 GB	3	Berguna untuk menyimpan data dan menjalankan sistem operasi.
5.	Lenovo ThinkPad E14 Gen 2 (Intel) Core i7-1165G7 16GB 512 SSD	1	Digunakan untuk proses pembangunan *computer cluster* Kubernetes dan digunakan untuk proses pengujian performa aplikasi.

2. Perangkat Lunak (*software)*

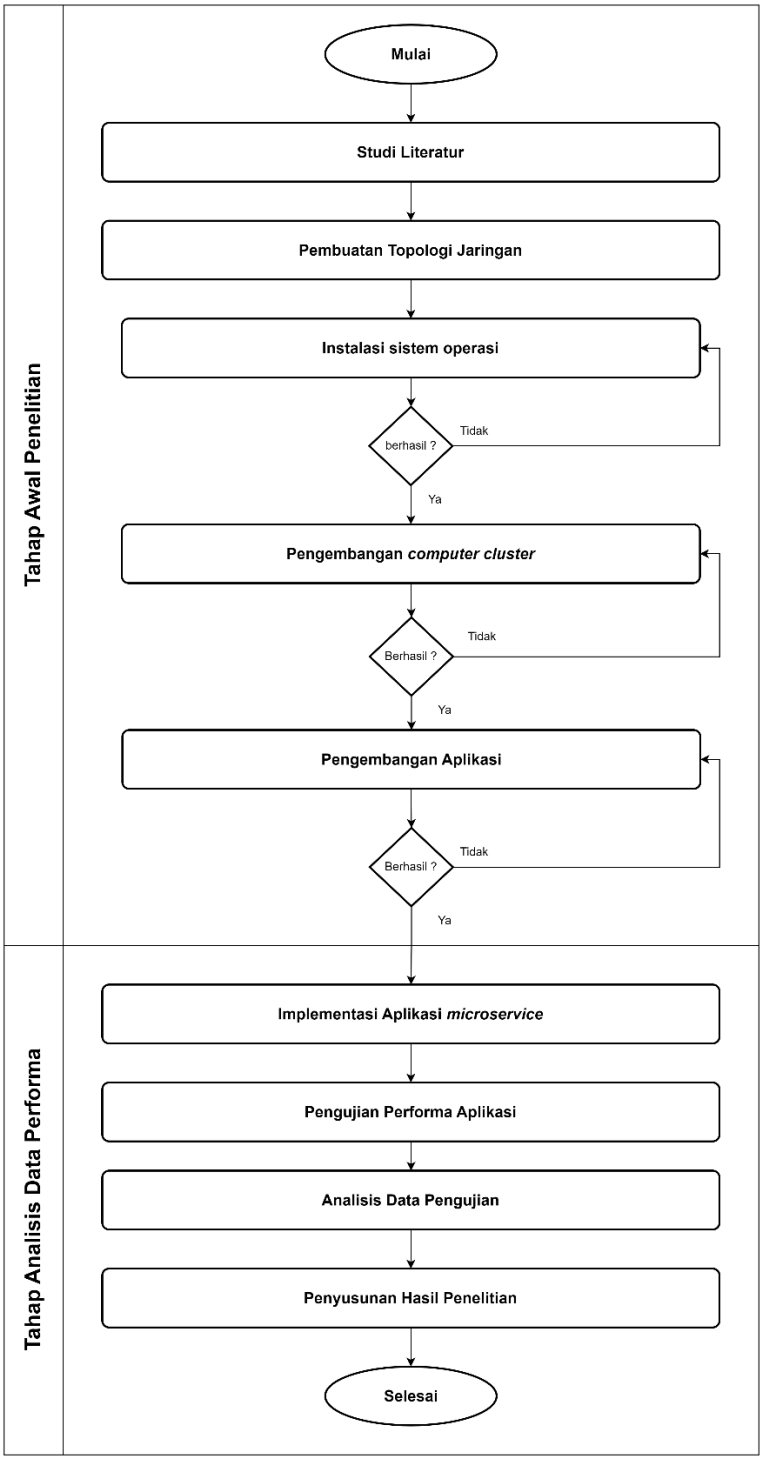
Tabel 3. 2 Perangkat Lunak

No	*Software*	Versi	Kegunaan
1.	Ubuntu *Server*	22.04.2 LTS	Sistem operasi yang digunakan pada *computer cluster* Raspberry Pi untuk menyebarkan aplikasi *microservices*.
2.	Kubernetes (K3s)	v1.22.3+k3s1	Digunakan untuk membangun *computer cluster* dan digunakan sebagai *container orchestration* untuk menjalankan aplikasi.
3.	Containerd	1.5.7-k3s2	Sebagai *container runtime* pada Kubernetes yang berguna untuk menjalankan aplikasi pada *container*.
4.	Ansible	2.9.6	Digunakan untuk menginstal K3s Kubernetes *cluster* pada ketiga Raspberry Pi.
5.	Aplikasi Python Django	4.17	Digunakan sebagai objek pengujian aplikasi berbasis arsitektur *microservices.*
6.	MobaXterm	V22.1 Build 4888	Terminal pada *windows* yang digunakan untuk konfigurasi *server*.
7.	Apache JMeter	5.5	Digunakan sebagai alat pengujian performa aplikasi.
8.	Prometheus	v9.4.3	Digunakan sebagai alat *exporter* data monitoring ke Grafana.
9.	Grafana	2.43.0	Digunakan sebagai alat *exporter* data monitoring ke dalam bentuk file CSV.

3.2.2 Bahan Penelitian

Pada penelitian ini dibutuhkan beberapa bahan atau sumber informasi yang dapat digunakan sebagai dasar dalam menyusun penelitian. Beberapa bahan yang dibutuhkan pada penelitian ini yaitu laporan hasil penelitian sebelumnya tentang aplikasi berbasis *microservices* dan Kubernetes, artikel ilmiah tentang arsitektur *microservices* dan Kubernetes, Hasil uji coba aplikasi dengan menggunakan *computer cluster* Raspberry Pi 4.

3.3 DIAGRAM ALIR PENELITIAN



Gambar 3. 1 Diagram alir penelitian

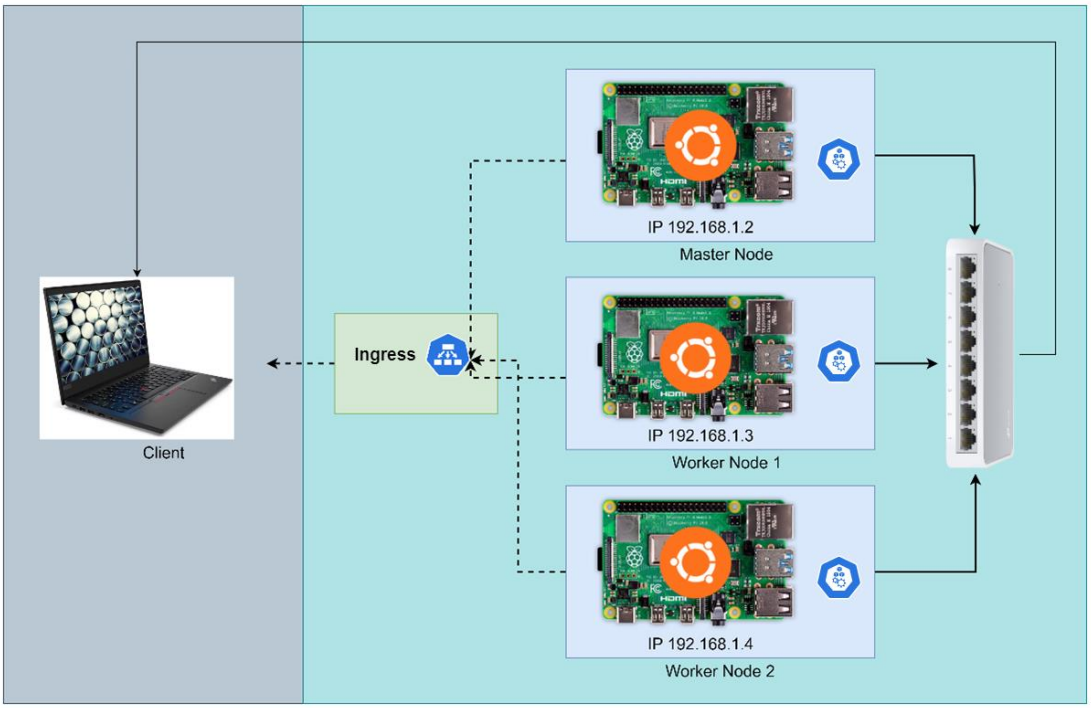
3.3.1 Studi Literatur

Studi literatur merupakan kumpulan sumber literatur yang telah dibaca dan dianalisis, yang tersusun secara sistematis sesuai dengan topik dan konteks penelitian. Studi literatur memberikan gambaran umum tentang kajian yang telah dilakukan sebelumnya terkait dengan topik yang relevan dengan penelitian saat ini, serta mengidentifikasi kekurangan yang ada dalam penelitian sebelumnya yang dapat menjadi landasan pada penelitian ini. Metode studi literatur ini sangat membantu dalam mengumpulkan dan menganalisis informasi yang relevan, sehingga dapat memberikan landasan yang kuat untuk penelitian selanjutnya.

3.3.2 Pembuatan Topologi Jaringan

Pada penelitian ini dibutuhkan topologi jaringan mengenai infrastruktur *computer cluster* menggunakan Raspberry Pi 4 Model B yang menggambarkan bagaimana aplikasi berbasis arsitektur *microservices* berjalan pada *computer cluster* yang dikonfigurasi menggunakan Kubernetes. Topologi jaringan ini mencakup koneksi antar Raspberry Pi dalam *cluster*, dengan konfigurasi menggunakan *static* IP *address* pada Raspberry Pi untuk memudahkan proses pengaksesan kepada Raspberry Pi demi memudahkannya proses pengujian aplikasi berbasis *microservices* pada *computer cluster*.

Topologi jaringan menggambarkan arsitektur dengan Raspberry Pi 4 Model B sebanyak 3 buah dengan IP *address* masing-masing yaitu 192.168.1.2 pada *master node*, kemudian pada kedua *worker node* mempunyai IP address 192.168.1.3 pada *worker node 1* dan 192.168.1.4 pada *worker node 2*. Selanjutnya untuk Laptop yang nantinya akan dilakukan proses pengujian dapat mengakses *service* arsitektur *computer cluster melalui* SSH dengan *port 22*. Semua *device* pada arsitektur saling terhubung melalui *switch* dengan cara menghubungkan kabel RJ45 pada *switch* dan pada tiap *device*. Berikut merupakan Gambar 3. 2 Topologi Jaringan Arsitektur Cluster mengenai topologi jaringan arsitektur *computer cluster* pada penelitian.



Gambar 3. 2 Topologi Jaringan Arsitektur *Cluster*

3.3.3 Instalasi Sistem Operasi

Instalasi sistem diperlukan untuk dapat menerapkan *computer cluster* menggunakan kubernetes pada Raspberry Pi 4. Sistem operasi Ubuntu sangat optimal untuk dapat menjalankan Kubernetes pada Raspberry Pi 4, maka dari itu dilakukan instalasi sistem operasi Ubuntu pada *memory card* menggunakan *software* Balena Etcher pada Windows 11 demi menerapkan sistem operasi pada *memory card.* Ubuntu juga merupakan sistem operasi berbasis linux yang paling banyak digunakan dan mudah untuk digunakan. Instalasi sistem operasi dilakukan pada tiga buah *memory card* yaitu sejumlah *node* pada *cluster*. *Memory card* yang telah siap digunakan setelah proses instalasi dapat dipasangkan pada tiap-tiap Raspberry Pi 4 untuk selanjutnya dilakukan proses pengembangan *computer cluster* menggunakan Kubernetes. Dapat dilihat pada Gambar 3. 3 berikut ini hasil dari instalasi sistem operasi Ubuntu *server*.

```
root@bismillahta:~# uname -r
5.15.0-1025-raspi
root@bismillahta:~# cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.2 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

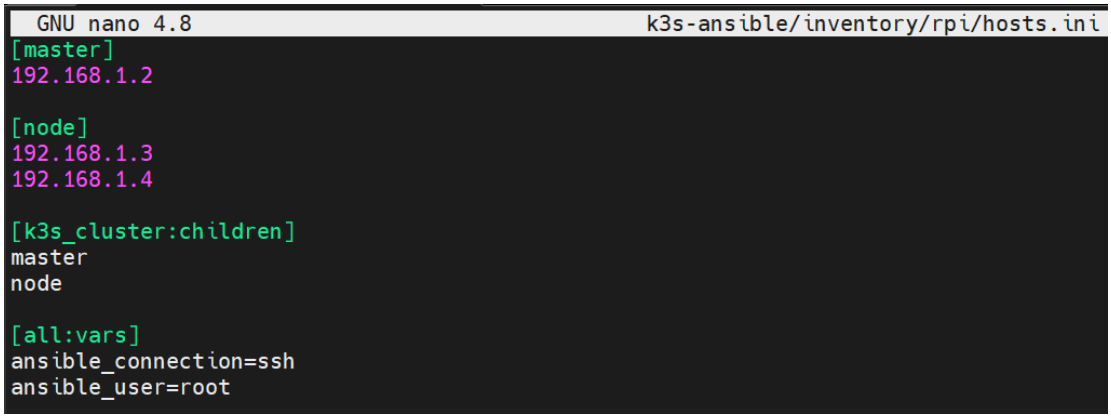
Gambar 3. 3 Sistem operasi Ubuntu *server*

3.3.4 Pengembangan Computer Cluster

Pada pengembangan *computer cluster* dilakukan konfigurasi Raspberry Pi sebagai *node cluster* menggunakan Kubernetes untuk mengelola *container* yang nantinya akan digunakan untuk menjalankan aplikasi. Pengembangan *computer cluster* didasarkan pada topologi jaringan yang sebelumnya telah dibuat, dilakukan konfigurasi pada sebanyak satu Raspberry Pi menggunakan Kubernetes dengan inisiasi sebagai *master node* atau *control-plane*, kemudian dilakukan konfigurasi sebanyak dua Raspberry Pi menggunakan Kubernetes dengan inisiasi sebagai *worker node*. *Computer cluster* dengan dua *worker node* dan satu *master node* pada *cluster* optimal untuk dapat menjalankan aplikasi pada *container* yang berjalan menggunakan Kubernetes. Pada penelitian ini digunakan K3s untuk menginisiasi *cluster computer* Kubernetes pada Raspberry Pi.

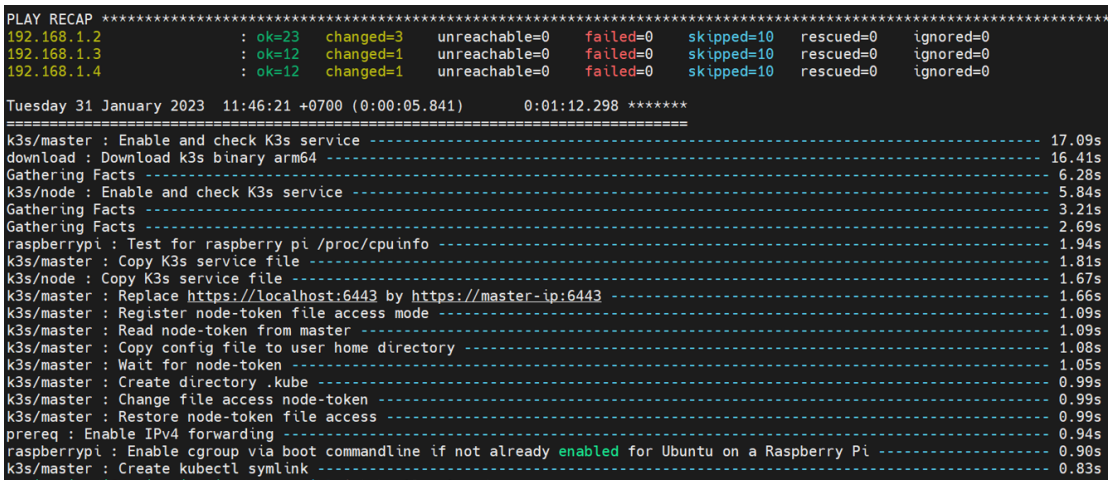
Tahapan-tahapan untuk dapat mengembangan *computer cluster* Kubernetes menggunakan Raspberry Pi dapat dimulai dengan konfigurasi jaringan antar Raspberry Pi agar dapat terhubung internet dan dapat diakses oleh perangkat lain, kemudian instalasi dan konfigurasi Kubernetes mengikuti dokumentasi instalasi yang tersedia, terakhir dapat dilakukan proses inisiasi *control-plane* atau *master node* pada Raspberry Pi dan inisiasi *worker node* pada Raspberry Pi lainnya. Untuk memastikan Raspberry Pi berhasil diinisiasi menjadi *node-node* pada *cluster*, dapat dilihat daftar *node* yang

berhasil diinisiasi pada *master node* atau *control-plane*. Sebelum dilakukan konfigurasi *cluster*, dilakukan konfigurasi IP *address* dan *variable* pada *Inventory* pada Ansible yang dapat dilihat pada Gambar 3. 4 berikut ini.



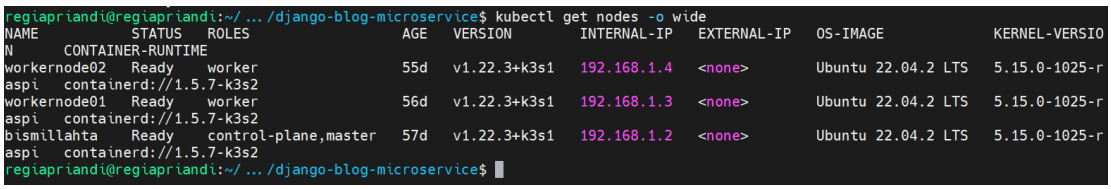
Gambar 3. 4 Konfigurasi *Inventory* Ansible

Berikut merupakan inisiasi *master node* dan *worker node* menggunakan beberapa perintah Ansible pada komputer *host* berbasis sistem operasi Ubuntu yaitu ansible-playbook untuk dapat mengeksekusi berkas konfigurasi *inventory* yang menyimpan data mengenai alamat IP dan host masing-masing *node* seperti pada Gambar 3. *4*. Dapat dilihat pada Gambar 3. *5* telah berhasil dilakukan konfigurasi Kubernetes *cluster* menggunakan K3s yang merupakan alat untuk inisiasi Kubernetes *cluster* dengan keunggulan dapat berjalan sangat ringan. Selanjutnya telah berhasil dilakukan konfigurasi menggunakan alat Ansible pada ketiga Raspberry Pi dengan perintah “ansible-playbook site.yml -i inventory/rpi/hosts.ini”.



Gambar 3. 5 Konfigurasi Kubernetes *cluster* menggunakan Ansible

Setelah inisiasi *master node* dan *worker node* berhasil dilakukan, *computer cluster* yang diinisiasi menggunakan Ansible sudah dapat digunakan untuk dapat melakukan *deployment* aplikasi berbasis arsitektur *microservices.* untuk dapat melihat status mengenai *node-node* yang telah berhasil dibangun pada *cluster* dapat dilihat menggunakan perintah kubernetes seperti pada Gambar 3. 6 berikut.

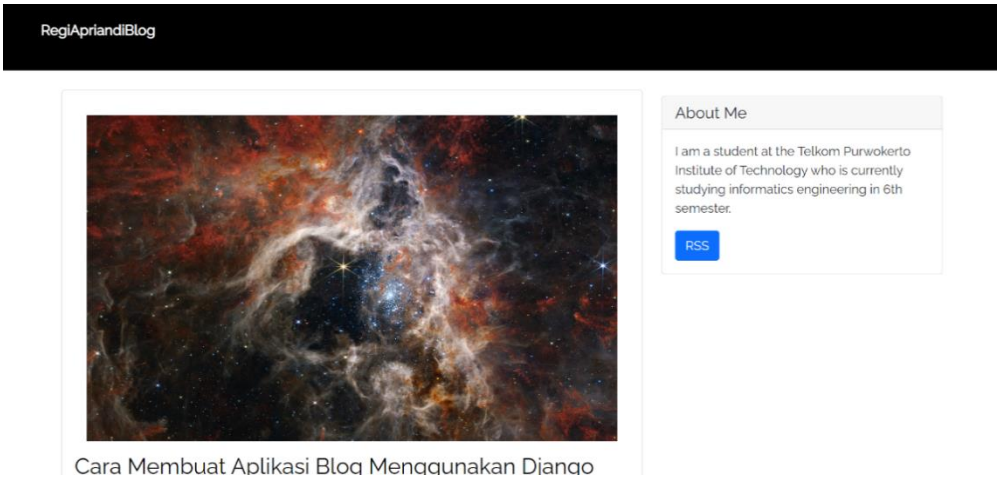


Gambar 3. 6 Status *Cluster Node*

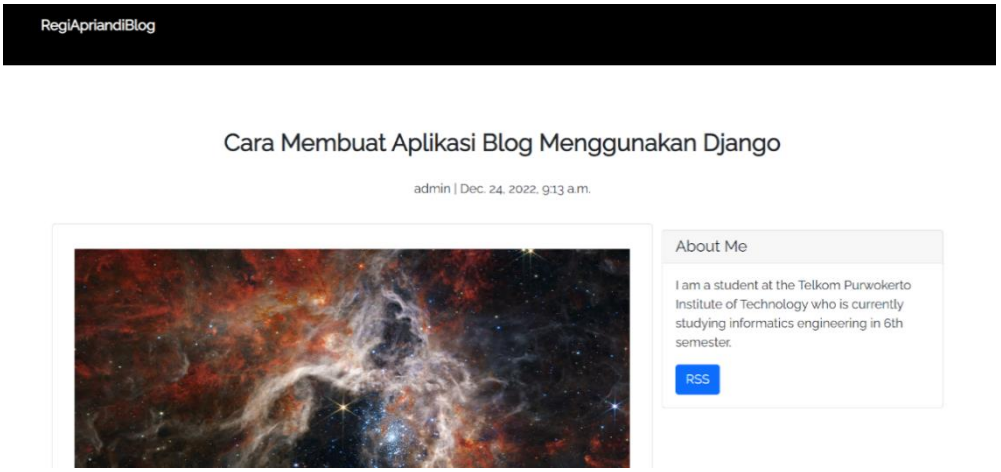
3.3.5 Pengembangan Aplikasi

Pengembangan aplikasi diperlukan untuk dapat melakukan pengujian performa aplikasi pada penelitian ini. Aplikasi yang dikembangkan untuk diimplementasikan menggunakan arsitektur *microservices* adalah aplikasi berbasis web yang dibangun menggunakan *framework* Django dan Bahasa pemrograman Python. Berikut merupakan tampilan mengenai aplikasi blog yang telah dibangun pada Gambar 3. 7

dan Gambar 3. 8. Selanjutnya akan diimplementasikan menggunakan arsitektur *microservices* pada Kubernetes bersama dengan *database* PostgreSQL.



Gambar 3. 7 Tampilan Aplikasi Daftar Konten Blog



Gambar 3. 8 Tampilan Aplikasi Detail Blog

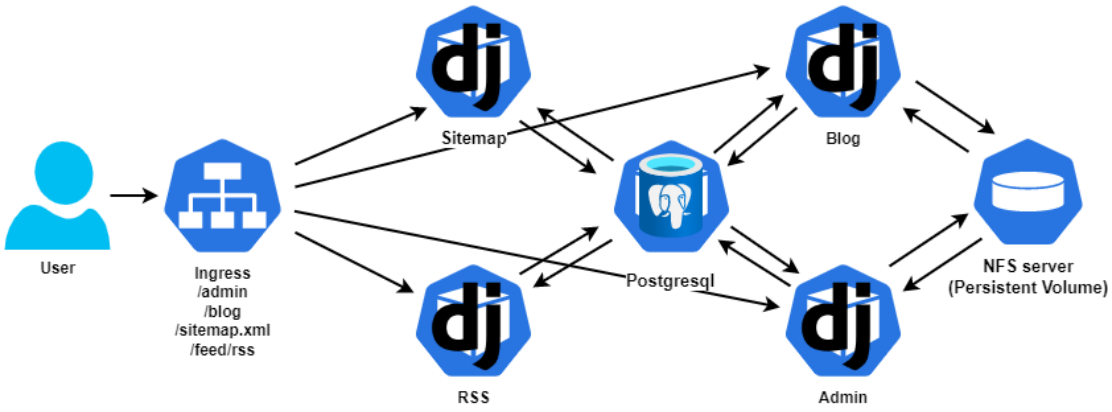
Selain pengembangan aplikasi Blog, dilakukan juga pengembangan aplikasi RSS dan Sitemap pada Django yang berfungsi untuk mempermudah pengguna dalam mengakses konten di dalam blog dan membantu mesin pencari seperti Google untuk mengindeks konten blog dengan lebih baik. Kemudian untuk aplikasi Django admin

yang berfungsi untuk menambahkan, menghapus, dan mengedit data blog pada aplikasi Django dikembangan secara terpisah. Maka pada pengembangan aplikasi Blog, RSS, Sitemap, dan Admin dilakukan secara terpisah yang nantinya akan dilakukan penyebaran secara independen yang saling terhubung satu sama lain.

3.3.6 Implementasi Aplikasi Microservices

Sebelum dilakukan implementasi aplikasi berbasis arsitektur *microservices*, dilakukan pengembangan aplikasi berbasis web menggunakan Bahasa pemrograman Python dan *framework* Django. Di mana pada pengembangan aplikasi tersebut telah dilakukan implementasi penggunaan teknologi *framework* Django yang merupakan teknologi *framework* dengan mengedepankan fleksibilitas. Selain itu pada implementasi arsitektur *microservices* digunakan teknologi *database* yaitu PostgreSQL yang berfungsi sebagai penyimpanan data blog pada aplikasi.

Kemudian pada arsitektur *microservices*, aplikasi Django dan *database* PostgreSQL berjalan dengan sendirinya yang mana dapat berjalan secara independen dengan beban infrastruktur yang kecil pada setiap aplikasinya, namun dapat saling terhubung satu sama lain secara lancar dan cepat. Untuk memudahkan implementasi menggunakan arsitektur *microservices* dilakukan desain arsitektur *microservices* seperti pada Gambar 3. 9 berikut.



Gambar 3. 9 Arsitektur aplikasi berbasis *microservices*

Pada proses *deployment* aplikasi pada Kubernetes, dilakukan replikasi *pod* pada setiap aplikasi untuk memastikan aplikasi tetap berjalan dengan lancar Ketika terjadinya kegagalan salah satu *pod* aplikasi pada Kubernetes. Pada aplikasi Blog dilakukan replikasi *pod* sebanyak 12 replikasi untuk memastikan aplikasi berjalan ketika terjadi kegagalan pada salah satu *pod* dan supaya aplikasi dapat secara cepat dan lancar dari kemungkinan banyak pengguna menggunakan aplikasi Blog secara bersamaan. Kemudian untuk aplikasi Sitemap, RSS, dan Admin dilakukan replikasi sebanyak masing-masing 3 replikasi sesuai kebutuhan pengguna dan memastikan aplikasi tetap berjalan Ketika terjadi salah satu kegagalan pada *pod*.

Arsitektur aplikasi berbasis *microservices* yang sebelumnya telah dibuat dapat diimplementasikan pada Kubernetes dengan menggunakan konfigurasi untuk melakukan *deployment* pada keempat aplikasi dan *database* PostgreSQL. Konfigurasi untuk *deployment database* PostgreSQL dibuat untuk dapat membangun *pod* Kubernetes pada *cluster*. Selain itu *persistent volume* digunakan untuk dapat membuat penyimpanan secara permanen pada *cluster*. Untuk dapat menerapkan konfigurasi aplikasi dan *database* pada *cluster*, dilakukan perintah *apply* menggunakan perintah Kubernetes yaitu kubectl pada terminal *cluster* terhadap file konfigurasi yang sebelumnya telah dibuat. Berikut merupakan Gambar 3. 10 penerapan konfigurasi untuk NFS *server* pada kubernetes menggunakan helm *repository manager*.

```
$ helm repo add nfs-subdir-external-provisioner https://kubernetes-
sigs.github.io/nfs-subdir-external-provisioner/

$ helm install nfs-subdir-external-provisioner nfs-subdir-external-
provisioner/nfs-subdir-external-provisioner --set nfs.server=192.168.1.2 --
set nfs.path=/mnt/nfs_share
```

Gambar 3. 10 Konfigurasi NFS *server*

NFS *server* pada Kubernetes yang telah berhasil diinstal nantinya akan digunakan sebagai penyimpanan media pada aplikasi Blog, selanjutnya dapat diterapkan konfigurasi untuk *database* dan aplikasi Blog, RSS, Admin, dan Sitemap pada terminal *cluster* menggunakan perintah Kubernetes seperti pada Gambar 3. 11 berikut ini.

```
$ git clone https://github.com/regiapriandi012/blog-microservices.git
$ cd blog-microservices/
$ kubectl apply -f database/manifests/
$ kubectl apply -f django-blog-admin/manifests/
$ kubectl apply -f django-blog-main/manifests/
$ kubectl apply -f django-blog-rss/manifests/
$ kubectl apply -f django-blog-sitemap/manifests/
```

Gambar 3. 11 Konfigurasi penerapan aplikasi *microservices*

Kemudian dapat dilakukan pemeriksaan status aplikasi yang telah diterapkan pada Kubernetes dengan memastikan status aplikasi berjalan atau *running* yang berarti aplikasi telah berjalan pada sistem dan dapat diakses tanpa adanya masalah. Selain itu *database* PostgreSQL dan NFS *storageclass* dengan status *running* yang berarti dapat berjalan secara lancar. Berikut merupakan Gambar 3. 12 hasil penerapan aplikasi pada Kubernetes.

```
regiapriandi@regiapriandi:~$ kubectl get pods
NAME                                          READY   STATUS    RESTARTS       AGE
nfs-subdir-external-provisioner-6f6b887c85-r5jvw   1/1     Running   16 (30m ago)   6d1h
postgres-deployment-5c96c8cc8d-wgv28          1/1     Running   6 (30m ago)    2d19h
djangoblog-main-app-6959d845cf-5bprv          1/1     Running   0              22m
djangoblog-main-app-6959d845cf-c4xv6          1/1     Running   0              22m
djangoblog-main-app-6959d845cf-qs8lg          1/1     Running   0              22m
djangoblog-main-app-6959d845cf-qfm9f          1/1     Running   0              22m
djangoblog-main-app-6959d845cf-6gtgg          1/1     Running   0              22m
djangoblog-main-app-6959d845cf-6lwmc          1/1     Running   0              22m
djangoblog-rss-app-5dccf4fdfd-jpdk9           1/1     Running   0              21m
djangoblog-admin-app-678644546-4crtt          1/1     Running   0              21m
djangoblog-sitemap-app-b5748d958-gm9xp        1/1     Running   0              21m
djangoblog-main-app-6959d845cf-rk9hx          1/1     Running   0              21m
djangoblog-main-app-6959d845cf-dgch7          1/1     Running   0              21m
djangoblog-main-app-6959d845cf-h8zhj          1/1     Running   0              21m
djangoblog-main-app-6959d845cf-ck2k8          1/1     Running   0              21m
djangoblog-admin-app-678644546-v8nb5          1/1     Running   0              21m
djangoblog-sitemap-app-b5748d958-8rjwm        1/1     Running   0              21m
djangoblog-main-app-6959d845cf-lqbmn          1/1     Running   0              21m
djangoblog-main-app-6959d845cf-97lp5          1/1     Running   0              21m
djangoblog-rss-app-5dccf4fdfd-ctbv9           1/1     Running   0              21m
djangoblog-admin-app-678644546-vnph9          1/1     Running   0              20m
djangoblog-sitemap-app-b5748d958-t5ffl        1/1     Running   0              20m
djangoblog-rss-app-5dccf4fdfd-xxxng           1/1     Running   0              20m
regiapriandi@regiapriandi:~$
```

Gambar 3. 12 Hasil penerapan *deployment* pada Kubernetes

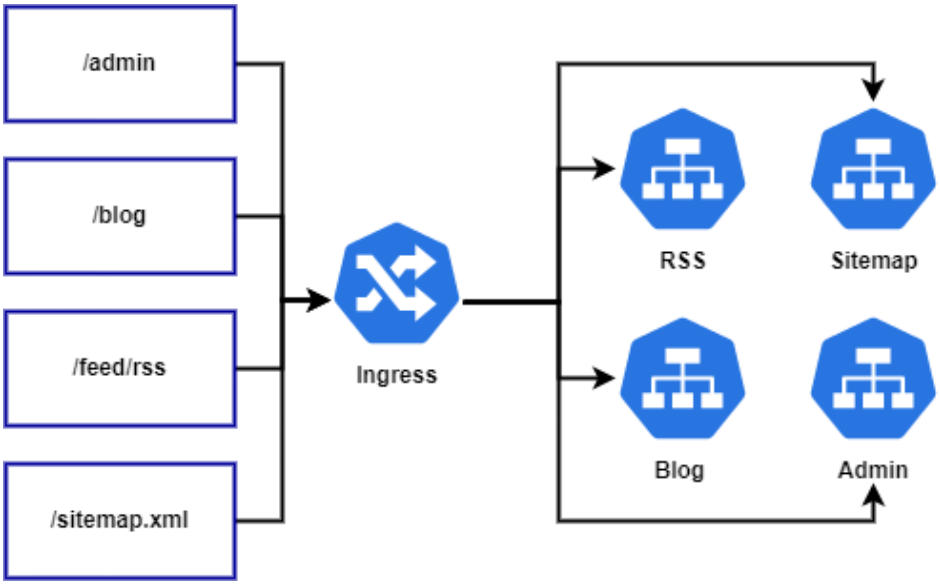
Dilakukan implementasi ingress pada Kubernetes dengan melakukan perintah berikut untuk dapat mengakses layanan aplikasi pada Kubernetes secara mudah melalui *host* yang telah ditentukan. Setelah perintah diterapkan pada terminal, konfigurasi ingress akan menginisiasi *host* http://regiapriandi.com/ yaitu *domain* lokal pada komputer *client* dengan beberapa *service* pada *host* yaitu /admin, /blog, /feed/rss, dan /sitemap.xml yang nantinya akan digunakan untuk mengakses *service* aplikasi berbasis arsitektur *microservices*, dapat dilihat pada Gambar 3. 13 berikut ini.

```
$ cd blog-microservices/
$ kubectl apply -f ingress/manifests/
```

Gambar 3. 13 Konfigurasi ingress

Pada ingress, terdapat konfigurasi mengenai *path* pada host yang akan mengarahkan ke layanan itu sendiri, seperti *path* /blog yang akan mengarahkan ke layanan aplikasi blog, dan *path* /admin akan mengarahkan ke layanan aplikasi admin, begitupun pada *path* yang lainnya mengarahkan ke layanan aplikasi yang dituju. Hasil

dari implementasi ingress untuk dapat memudahkan pengguna dalam mengakses layanan aplikasi pada Kubernetes dapat dilihat pada Gambar 3. 14 berikut.



Gambar 3. 14 Struktur Ingress

Kemudian dapat dilihat pada Gambar 3. 15 berikut ini hasil dari implementasi ingress pada Kubernetes dengan *host* http://regiapriandi.com/ yaitu *domain* lokal pada komputer *client* yang dapat dikonfigurasi pada berkas *hosts* di sistem operasi Windows untuk dapat mengarahkan ke alamat IP dari salah satu *node* yaitu 192.168.1.2, 192.168.1.3, dan 192.168.1.4 dengan protokol 80.

```
regiapriandi@regiapriandi:~/ .../django-blog-microservice$ kubectl get ingr
ess
NAME                    CLASS    HOSTS              ADDRESS
                PORTS   AGE
django-blog-ingress     <none>   regiapriandi.com   192.168.1.2,192.168.1.3,
192.168.1.4   80      6d5h
regiapriandi@regiapriandi:~/ .../django-blog-microservice$
```

Gambar 3. 15 Hasil penerapan ingress Kubernetes

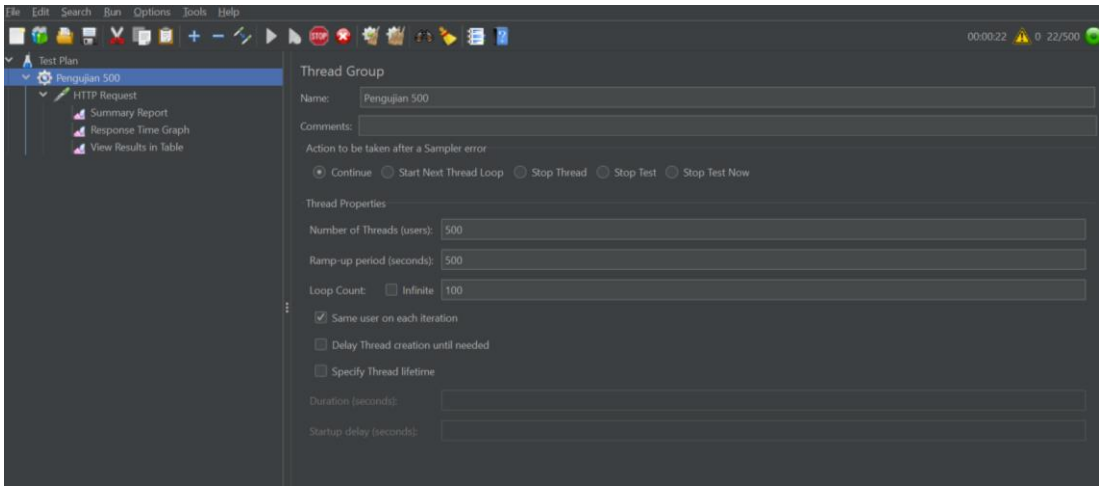
3.3.7 Skenario Pengujian Performa Aplikasi

Skenario pengujian aplikasi menjadi hal yang sangat penting pada penelitian ini, karena dari pengujian yang dilakukan, didapatkan mengenai data hasil pengujian dan kesimpulan mengenai hasil penelitian. Skenario pengujian digunakan untuk dapat menguji performa aplikasi berbasis arsitektur *microservices*. Skenario pengujian performa aplikasi didasarkan pada serangkaian tes yang mencakup beberapa aspek performa seperti *throughput, delay,* dan *packet loss*. Selain itu dilakukan pengujian berdasarkan ketersediaan infrastruktur penunjang aplikasi berbasis arsitektur *microservices* menggunakan Raspberry Pi dan Kubernetes. Berikut merupakan Tabel 3. 3 tahapan skenario pengujian performa aplikasi berbasis arsitektur *microservice*.

Tabel 3. 3 Skenario Pengujian

No	Number of Thread	Ramp-up Period (Second)	Loop Count	Request per detik	Total requests	Total pengetesan
1.	100	100	100	100	10000	10
2.	250	250	100	100	25000	10
3.	500	500	100	100	50000	10
4.	750	750	100	100	75000	10
5.	1000	1000	100	100	100000	10

Proses pengujian dilakukan dengan implementasi *software* pengujian performa aplikasi yaitu Apache JMeter. Pada JMeter dilakukan konfigurasi skenario pengujian seperti yang telah diinisiasi pada tabel di atas, seperti contoh pengujian dengan 100 *thread* dengan *ramp-up period* sebanyak 100 detik, dan jumlah koneksi per detiknya sejumlah 100 koneksi per detik. Dapat dilihat pada Gambar 3. 16 berikut ini yaitu konfigurasi pada Apache JMeter.



Gambar 3. 16 Konfigurasi Apache JMeter

1. Pengujian *throughput*

Proses pengujian *throughput* berfungsi untuk menguji kemampuan sistem aplikasi berbasis arsitektur *microservices*. Pengujian ini didasarkan pada kemampuan aplikasi dalam menangani jumlah *traffic* atau permintaan pada aplikasi dalam satuan waktu tertentu. Tujuan utama dari pengujian ini adalah untuk menentukan batas maksimal sebuah *throughput* yang dapat ditangani oleh aplikasi, serta untuk mengetahui bagaimana aplikasi atau sistem berperilaku dibawah beban yang berbeda.

Pada penelitian ini digunakan Apache JMeter untuk dapat melakukan pengujian *throughput*, Kemudian skenario pengujian *throughput* didasarkan pada jumlah koneksi yang dikirimkan ke sistem atau aplikasi dan permintaan ke aplikasi setiap detiknya atau *request* per detik. Setelah itu dapat dilakukan analisis hasil pengujian *throughput* yang telah dilakukan. Jumlah koneksi yang dikirimkan pada skenario pengujian adalah 100, 250, 500, 750, dan 1000 jumlah koneksi.

2. Pengujian *delay* (latensi)

Latensi merupakan waktu yang diperlukan untuk mengirim atau menerima informasi melalui jaringan atau sistem. Pengujian latensi ini dilakukan untuk dapat mengukur seberapa cepat jaringan atau sistem untuk dapat mengirim dan menerima

informasi. Pengujian ini sangat berguna untuk dapat mengevaluasi kinerja suatu sistem atau jaringan dan dapat memastikan bahwa jaringan atau sistem dapat beroperasi dengan efisien dan efektif. Jumlah koneksi yang dikirimkan pada skenario pengujian adalah 100, 250, 500, 750, dan 1000 jumlah koneksi.

3. Pengujian *packet loss*

Pengujian *packet loss* dilakukan untuk dapat melihat jumlah *packet* yang hilang pada proses pengiriman *packet.* Pengujian dilakukan pada setiap *microservices* secara terpisah. Jumlah koneksi yang dikirimkan pada skenario pengujian adalah 100, 250, 500, 750, dan 1000 jumlah koneksi. Berikut merupakan standarisasi nilai *packet loss* yang dapat dilihat pada Tabel 3. 4.

Tabel 3. 4 Standarisasi *Packet Loss*

No	Kategori	Besar (%)
1	Sangat Baik	0 – 2
2	Baik	3 – 14
3	Cukup Baik	15 – 25
4	Buruk	> 25

4. Pengujian *availability*

Pengujian *availability* dilakukan untuk dapat mengetahui tingkat ketersediaan sebuah sistem atau layanan untuk dapat melakukan tugasnya sesuai dengan yang diharapkan. Tingkat *availability* tinggi sangat diperlukan untuk dapat menjamin sebuah sistem atau layanan dapat diandalkan dan dapat diakses oleh pengguna saat dibutuhkan. Skenario pengujian tingkat *availability computer cluster* Raspberry Pi 4 menggunakan Kubernetes yaitu dengan melakukan pengujian pemberhentian salah satu *node* pada *cluster* dengan memastikan bahwa apakah sistem atau layanan dapat tetap berjalan ketika terjadi suatu kegagalan pada *node cluster*.

3.3.8 Analisis Data Performa

Setelah dilakukannya pengujian performa aplikasi dengan beberapa skenario pengujian yang sebelumnya telah dirumuskan, dilakukan analisis data hasil performa pengujian aplikasi. Sebelum itu, dipersiapkan data hasil pengujian performa aplikasi yang didapatkan dari hasil pengujian pada Apache JMeter. Selanjutnya dapat dilakukan analisis data hasil pengujian performa aplikasi berbasis *microservices* menggunakan Kubernetes. Dari analisis yang dilakukan, didapatkan hasil analisis untuk dapat menyimpulkan performa sebuah performa aplikasi yang berjalan pada arsitektur *microservices* menggunakan Kubernetes pada *computer cluster* menggunakan Kubernetes.

Kemudian pada analisis data performa penggunaan CPU dan penggunaan RAM pada sistem, dilakukan implementasi Prometheus sebagai *exporter* data CPU dan RAM pada sistem yang kemudian data tersebut digunakan Grafana sebagai visualisasi data. Pengujian performa CPU dilakukan untuk dapat mengetahui nilai penggunaan CPU masing-masing *node* pada *computer cluster* pada saat proses pengujian. Melalui diketahuinya penggunaan CPU pada masing-masing *node*, dapat ditentukan seberapa besar penggunaan CPU pada layanan aplikasi, NFS server, dan *Database* di Kubernetes pada masing-masing *node.*

Selanjutnya pada saat pengujian dilakukan, didapatkan data penggunaan RAM hasil pengujian, penggunaan RAM pada saat pengujian menjadi informasi yang penting untuk dapat mengetahui seberapa besar penggunaan RAM pada saat proses pengujian dilakukan. Setelah dilakukannya pengujian performa aplikasi, dilakukan penyimpanan data performa CPU dan RAM pada saat pengujian pada Grafana ke data dokumen dalam format CSV yang nantinya akan dilakukan analisis. Berikut merupakan Gambar 3. 17 contoh visualisasi data pada Grafana yang diimpor melalui Prometheus.

Gambar 3. 17 Visualisasi penggunaan CPU dan RAM

3.3.9 Penyusunan Hasil Penelitian

Proses penyusunan laporan hasil penelitian dapat dimulai dengan pembuatan outline dan struktur penulisan laporan. Outline pada laporan dapat berupa latar belakang, tujuan penelitian, metodologi, hasil penelitian, dan kesimpulan. Selanjutnya dapat Menyusun laporan sesuai dengan struktur penulisan yang sebelumnya telah dibuat dengan menyertakan informasi yang penting dan relevan dan dijelaskan secara jelas. Selain itu ditambahkan beberapa informasi berupa grafik dan tabel untuk memvisualisasikan data hasil agar lebih jelas dan dapat mudah dipahami.

Terakhir dapat dilakukan pembuatan kesimpulan mengenai penelitian yang telah dilakukan dengan menjelaskan bagaimana hasil dari penelitian yang dilakukan. Selain dari itu, daftar Pustaka menjadi hal yang penting pada proses penyusunan laporan penelitian, ditambahkan daftar Pustaka dengan mencantumkan sumber penelitian yang digunakan.

3.4 HIPOTESIS PENELITIAN

Hipotesis penelitian merupakan suatu asumsi yang dapat dijadikan dasar untuk melakukan pengujian kebenaran dari suatu penelitian. Hipotesis penelitian harus dapat

diuji dan harus dapat divalidasi melalui pengujian yang nantinya akan dilakukan selama penelitian. Hipotesis penelitian dari penelitian ini dapat berupa rumusan hipotesis penelitian seperti berikut ini.

1. Aplikasi berbasis arsitektur *microservices* yang dijalankan pada *computer cluster* Raspberry Pi 4 menggunakan Kubernetes memiliki performa lebih baik dengan tingkat ketersediaan yang tinggi dibandingkan dengan aplikasi yang hanya dijalankan pada satu komputer.
2. Penggunaan Kubernetes pada *computer cluster* Raspberry Pi 4 meningkatkan kemudahan *deployment* aplikasi berbasis arsitektur *microservices*.