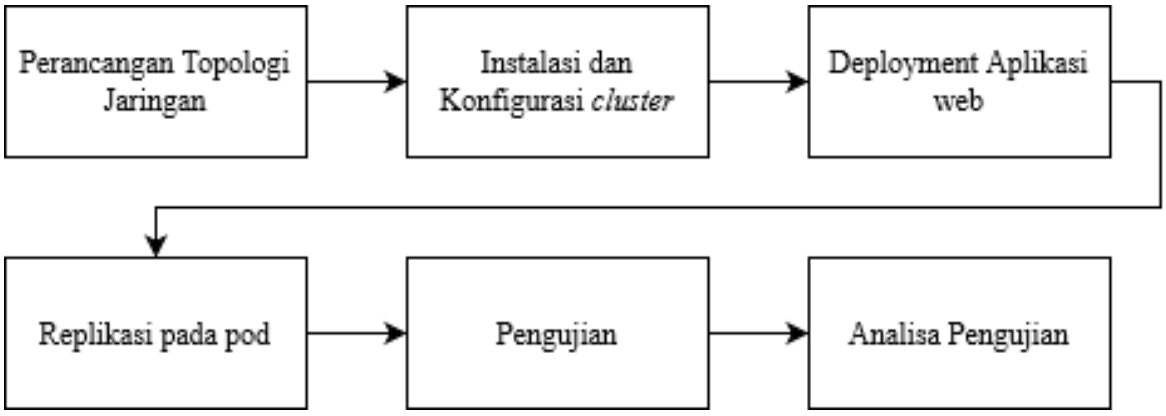
BAB III
METODE PENELITIAN

Dalam melakukan analisis *High Availability* dan *cluster* pada *container orchestrator* swarm yang disimulasikan menggunakan GNS3 (*Graphical Network Simulator 3*). Pada penelitian ini simulasi digunakan 3 *node* dengan sistem operasi ubuntu server 16.04 cloud x64. *Node* pertama akan digunakan menjadi *manager node* dan dua *node* berikutnya akan digunakan menjadi *worker node*.

Node worker pertama digunakan untuk menjalankan *service web server* dan *service database server* pada *container* yang selanjutnya dilakukan proses replikasi kepada *node worker* kedua.

3.1 Tahapan Penelitian

Berikut tahapan penelitian yang dilakukan penulis :



Gambar 3. 1 Tahapan Penelitian

3.1.1 Perancangan Topologi Jaringan

Topologi pada jaringan yang disimulasikan menggunakan satu *node manager* dan dua *node worker*. *Node manager* menjalankan *api-server, scheduler, controller,* dan *dashboard*. *Node worker* menjalankan docker *engine,* dan swarm. *Node worker* juga menjalankan *container* yang berisi *service web server* dan

database server. *Client* dapat mengakses *dashboard* yang sudah terpasang pada *node manager*.

Berikut perangkat keras yang digunakan penulis dalam melakukan simulasi :

1. Spesifikasi laptop GNS3 GUI

a) AMD Ryzen 5

b) RAM 12GB

c) Harddisk 1TB

2. Spesifikasi GNS3 server

a) AMD Ryzen 5

b) RAM 12GB

c) Harddisk 1TB

3. Spesifikasi *node manager*

a) CPU 4 core

b) RAM 4GB

4. Spesifikasi *node worker*

a) CPU 2 core

b) RAM 4GB

Kebutuhan perangkat lunak yang digunakan dalam melakukan simulasi :

1. Kebutuhan komputer simulasi :

a) Sistem operasi Ubuntu 20.04 LTS

b) GNS3

c) *Web Browser* (*mozila firefox*)

d) Httperf

2. Kebutuhan perangkat lunak *node manager*

a) Sistem operasi Ubuntu Server 16.04

b) Docker engine.

c) Docker swarm

3. Kebutuhan perangkat lunak *node worker*

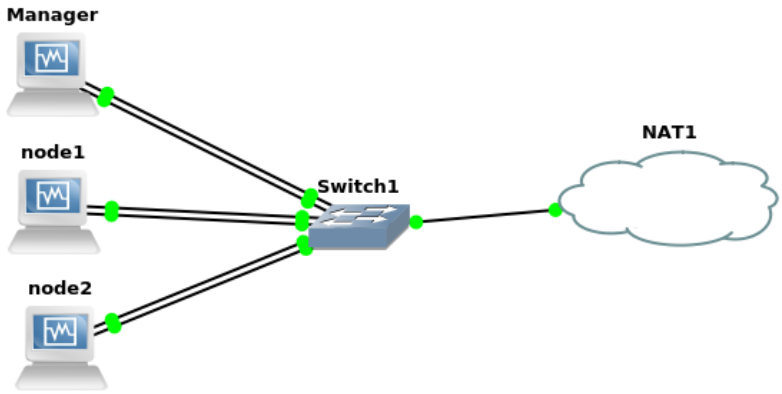
a) Sistem operasi Ubuntu Server 16.04

b) Docker engine.

c) Docker swarm.

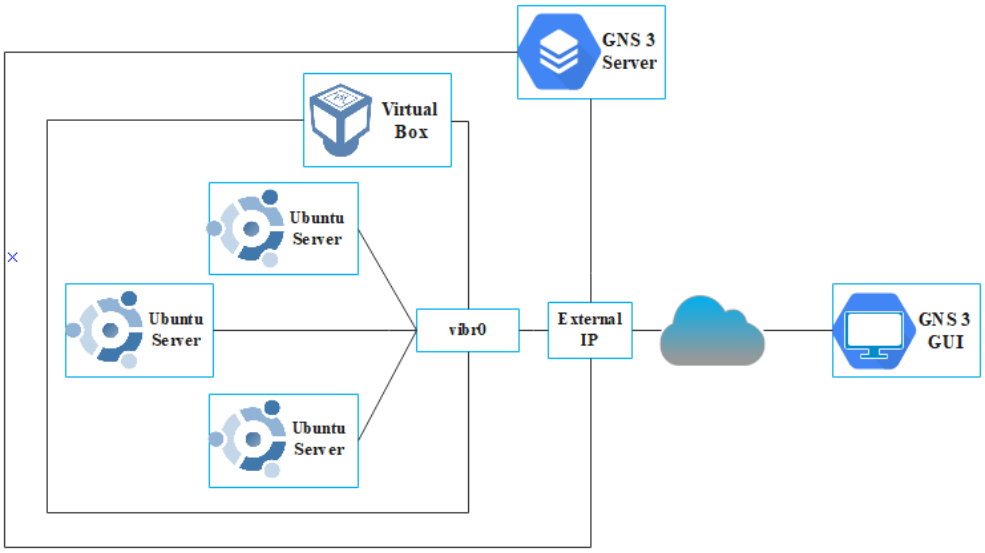
Node manager dilakukan konfigurasi untuk pemasangan *service web server* dan *database server*, selanjutnya dilakukan replikasi pada *web server* dan *database server* yang sudah dipasang.

Platform GNS3 (*Graphical Network Simulator 3*) yang digunakan dalam simulasi merupakan *platform* yang mudah digunakan. Dalam melakukan simulasi jaringan secara nyata GNS3 mendukung dalam penambahan *image* melalui Qemu, Vmware, dan Virtualbox.



Gambar 3. 2 Topologi Jaringan

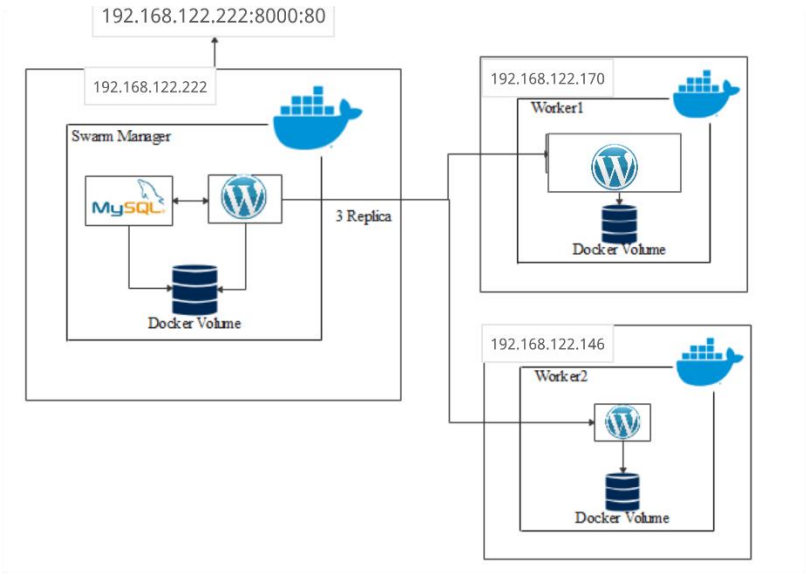
Gambar 3.2 merupakan topologi yang digunakan dalam penelitian dimana terdapat tiga *node*, satu *node* yang dijadikan *manager* dan dua *node* yang dijadikan *worker*. Ketiga *node* mempunyai dua *interface ethernet*. *Interface* ens3 yang digunakan untuk NAT bertujuan untuk berkomunikasi dengan internet. *Interface* ens4 digunakan sebagai akses jaringan *internal* pada ketiga *node*.



Gambar 3. 3 Topologi GNS3

3.1.2 Instalasi dan Konfigurasi *cluster*

Dalam *cluster* swarm terdapat *docker volume* yang berfungsi sebagai penyimpanan data dan juga untuk berbagi data antar *container*. Untuk membuat layanan web yang memerlukan penyimpanan persisten maka penulis menggunakan aplikasi Wordpress yang digunakan untuk manajemen konten blog dan *website*. Aplikasi Wordpress dibangun dengan bahasa pemrograman PHP dan basis data MySQL. *Service* Wordpress dan *service* MySQL berkomunikasi menggunakan *cluster* IP, kemudian aplikasi Wordpress menggunakan *node port* untuk membangun aplikasi bersama dengan *node worker* lainnya melalui IP dari *node manager*. Topologi GNS3 dapat dilihat pada gambar 3.4.



Gambar 3. 4 Topologi Dalam *Cluster* Swarm

Cluster menggunakan 3 *node*, satu *node* sebagai *manager* dan 2 *node* sebagai *worker*. Dalam setiap *node* memiliki dua *interface* yaitu, *interface internal* yang digunakan untuk berkomunikasi antar sesama *node* dan *interface external* yang digunakan untuk mengakses *internet*.

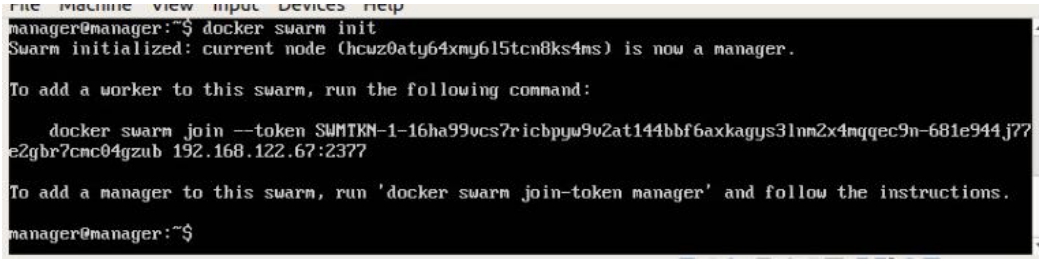
Tabel 3. 1 Konfigurasi *Node* pada *Cluster*

Node	Hostname	IP External	IP Internal
Manager	Manager	192.168.122.222	10.10.10.10
Node 1	Worker 1	192.168.122.170	10.10.10.20
Node 2	Worker 2	192.168.122.146	10.10.10.30

Cluster swarm mempunyai Docker sebagai *engine*-nya, maka dari itu perlu dilakukan pemasangan docker pada masing-masing *node* sebelum melakukan pembuatan *cluster* swarm.

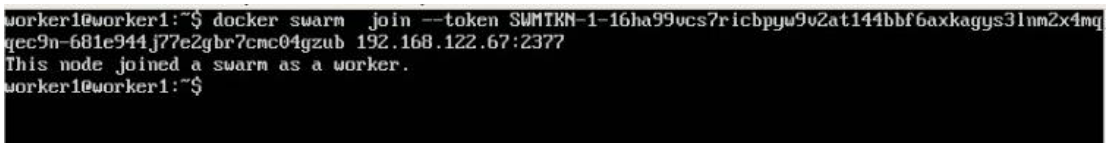
Pemasangan docker *engine* harus terlebih dahulu mengatur repositori docker, untuk kemudahan dalam pemasangan dan jika ingin melakukan peningkatan atau memperbaharui docker.

Pembuatan *cluster* swarm terlebih dahulu untuk melakukan inisiasi *node manager*.



Gambar 3. 5 Inisiasi *Manager* Swarm

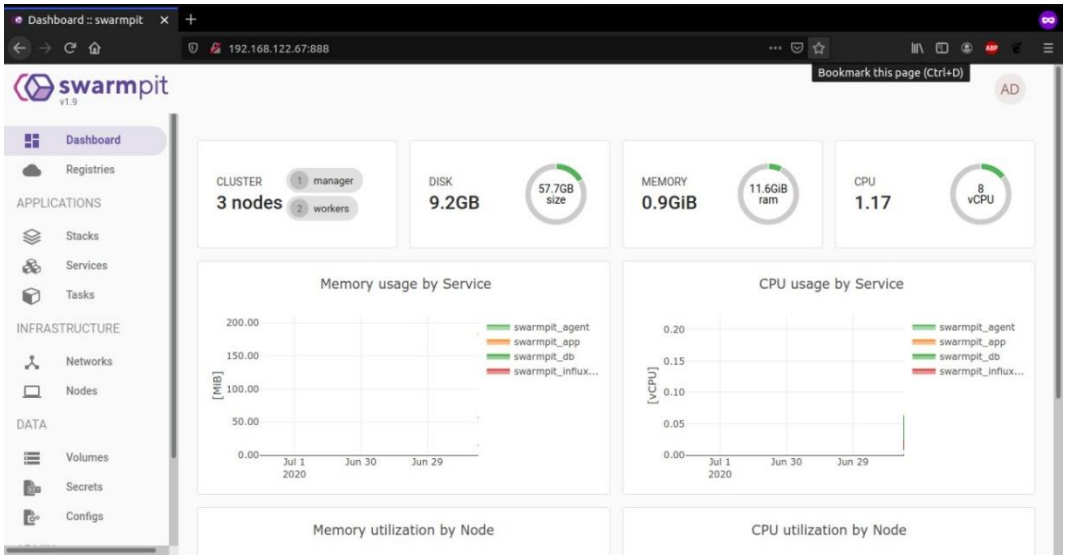
Untuk meringankan beban *node manager* diperlukan *node* yang digunakan sebagai *worker*. *Node worker* bertugas untuk meringkan beban kerja aplikasi yang dijalankan pada *node manager*.



Gambar 3. 6 Menambahkan *Worker*

Swarm menggunakan --token sebagai *autentikasi* bagi *worker* yang bekerja dengan *manager*. Jika token berhasil dikonfirmasi *worker* masuk kedalam mode swarm, dan *worker* bisa menjalankan *services* dan *deployment* yang dibuat. Semua *node* sudah memasuki mode swarm selanjutnya pemasangan *overlay network* yang berfungsi sebagai media komunikasi antar *container* yang berbeda *node*.

Selain menggunakan *command-line* untuk konfigurasi, swarm menyediakan *web user interface* yang bernama *Swarmpit* bisa dilihat pada gambar 3.7



Gambar 3. 7 *Dashboard* Swarmpit

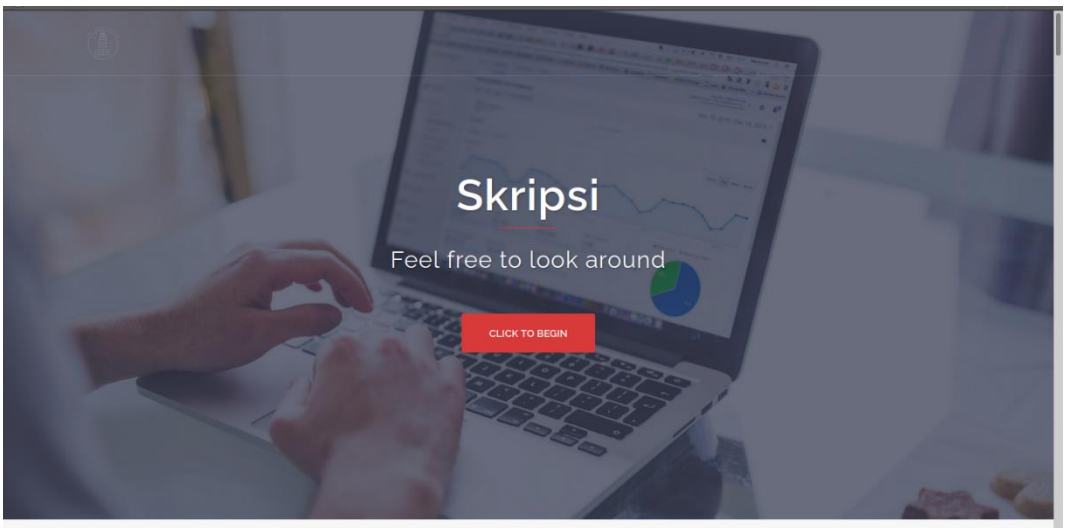
Gambar 3.7 merupakan tampilan *web user interface* dari *cluster swarm*. *Web user interface* berguna sebagai pusat kendali dan *monitoring cluster*. Pada *Web user* dapat melihat aplikasi yang sedang berjalan, informasi masing-masing *services*, melihat informasi *node* yang terhubung. Untuk menjalankan *Dashboard Swarmpit* pada swarm diperlukan *deployment* sendiri. Berikut *command-line* dalam pemasangan *dashboard swarmpit*.

```
docker run -it --rm \
  --name swarmpit-installer \
  --volume /var/run/docker.sock:/var/run/docker.sock \
  swarmpit/install:1.9
```

Gambar 3. 8 Pemasangan *Dashboard*

3.1.3 Deployment Aplikasi web

Pengujian menggunakan apache *web server* dengan aplikasi wordpress dan MySQL . Gambar 3.9 merupakan tampilan halaman depan *website* yang diuji. Gambar 3.9 merupakan contoh tampilan dari aplikasi wordpress yang biasa digunakan sebagai *blog* maupun *website* lainnya.



Gambar 3. 9 Tampilan Halaman Depan Wordpress

Penggunaan aplikasi wordpress merupakan aplikasi web yang paling banyak digunakan, menurut data w3techs.com pada bulan februari 2020, wordpress berada di peringkat pertama dalam CMS (*Content Management Systems*) paling populer, dengan nilai cakupan pasar mencapai 62.6%[13]. Sangat layak untuk dijadikan sebagai bahan aplikasi web yang diuji.

3.1.4 Replikasi pada *service*

Pengujian ini menggunakan tiga replikasi pada aplikasi wordpress. Replikasi berguna untuk mendukung *availability* pada *cluster*. Berikut merupakan *command-line* replika *service* wordpress yang terdapat pada gambar 3.10.

```
manager@manager:~$ docker service scale App_wordpress=3
App_wordpress scaled to 3
overall progress: 3 out of 3 tasks
1/3: running
2/3: running
3/3: running
verify: Service converged
manager@manager:~$ docker service ls
ID                  NAME                MODE                REPLICAS            IMAGE
    PORTS
py58v2wuwvmz        App_db              replicated          1/1                 mysql:5.7

ok1fc4olykv4        App_wordpress       replicated          3/3                 wordpress:latest
    *:8000->80/tcp
```

Gambar 3. 10 Replika *Service* Wordpress

3.1.5 Pengujian

Pada tahap pengujian dilakukan pengujian *High Availability* dan performa dari simulasi *clustering* pada swarm. Pengujian dilakukan dari sisi *node manager* dan *client*

3.1.5.1 Pengujian *Availability*

Digunakannya istilah *nine* karena didalam suatu sistem tidak ada yang dapat mencapai kesempurnaan *availability* seratus persen. *Availability* dapat digunakan sebagai salah satu parameter dalam *Service Level Agreement* (SLA). SLA adalah perjanjian yang disepakati antara penyedia layanan dengan pengguna dalam ruang lingkup untuk menentukan karekteristik dan kualitas layanan yang diberikan[8].

3.1.5.2 Pengujian fitur *self-healing*

Self-healing merupakan faktor pendukung yang penting dalam *availability*. Pengujian *self-healing* dilakukan dengan beberapa skenario, yaitu dengan mematikan salah satu *node worker* pada swarm dan mengamati perubahan pada *services* yang berjalan pada *node worker* yang dimatikan.

3.1.5.3 Pengujian CPU *Utilization*

CPU *usage* dari sebuah program adalah lama waktu penggunaan prosesor yang diperlukan program untuk menjalankan instruksinya[8]. Pengujian CPU *Utilization* dilakukan dengan menggunakan aplikasi *httperf*. Penggunaan aplikasi *httperf* sudah banyak digunakan dalam menguji performa *web server*, dikarenakan *httperf* dapat mensimulasikan jumlah koneksi dengan jumlah *request* per-detik. Jumlah inilah yang digunakan untuk mensimulasikan beban komputasi sehingga didapatkanlah nilai CPU *utilization*. Hasil pengujian CPU *utilization* dapat dilihat pada tabel 3.2 dibawah ini

Tabel 3. 2 Pengujian CPU *Utilization*

No	**Jumlah Koneksi**	***Request* per-detik**
1	100	10
2	500	50
3	1000	100
4	2500	250
5	5000	500

Pengujian dilakukan dengan membuat koneksi sebanyak 100, 500, 1000, 2500, dan 5000 dengan banyaknya *request* yang dilakukan selamat satuan detik sebanyak 10, 50, 100, 250, 500. Selanjut dilakukan pengamatan CPU *usage* pada setiap *node* dengan menggunakan program htop. Htop merupakan program untuk memantau proses sistem dan melihat proses, sehingga dapat dilihat CPU *usage* pada setiap *node* dengan menggunakan program htop.