

## BAB III METODE PENELITIAN

Penelitian ini akan menganalisis penerapan *live migration* menggunakan *Proxmox VE* yang berfungsi untuk membuat VM. Penelitian ini menerapkan pendekatan kuantitatif dengan metode eksperimen melalui pengujian dan pengambilan data.

### 3.1 PERANGKAT YANG DIGUNAKAN

Pada penelitian ini perlu adanya perangkat keras dan perangkat lunak yang digunakan untuk perancangan dan pengujian.

#### 3.1.1 Perangkat Keras (*Hardware*)

Dalam penelitian ini, diperlukan suatu perangkat keras berupa komputer untuk menyusun dan menganalisis *live migration*. Berikut adalah tabel 3.1 spesifikasi rinci dari komputer yang diperlukan:

**Tabel 3.1 Spesifikasi Komputer.**

Sistem Operasi	<i>Windows Server 2019 Datacenter 64-bit</i>
<i>Processor</i>	11th Gen Intel(R) Core(TM) i7-11700F @ 2.50GHz (16 CPUs), ~2.5GHz
RAM	64GB
<i>Storage</i>	2TB SSD
Kartu Jaringan	<i>Realtek Gaming GBE Family Controller</i>

#### 3.1.2 Perangkat Lunak (*Software*)

##### 1) Perangkat *Virtual*

Dalam penelitian ini, terdapat dua *server virtual* yang digunakan, beroperasi di bawah *Hypervisor Proxmox VE*. Tabel 3.2 di bawah ini menampilkan spesifikasi dari masing-masing perangkat *server virtual* yang digunakan dalam penelitian ini:

**Tabel 3.2 Spesifikasi perangkat *virtual*.**

Node1	Sistem operasi	<i>Proxmox VE 7.4</i>
	Prosesor	4 Core
	RAM	8GB
	HDD	80GB
	<i>IP Address</i>	192.168.110.128

Node2	Sistem operasi	<i>Proxmox VE 7.4</i>
	Prosesor	4 Core
	RAM	8GB
	HDD	80GB
	<i>IP Address</i>	192.168.110.129
VM100	Sistem Operasi	<i>XenialPup64 7.5</i>
	Prosesor	2 Core
	RAM	4GB
	HDD	8GB
	<i>IP Address</i>	192.168.110.131

## 2) *Software Tools*

*Software tools* yang digunakan dalam penelitian ini akan dijelaskan pengertian dan fungsinya sebagai berikut:

### 1) *Wireshark*

*Wireshark* adalah aplikasi perangkat lunak sumber terbuka yang berfungsi sebagai alat yang berharga bagi administrator jaringan dalam pemantauan jaringan. Aplikasi ini memiliki kemampuan untuk menangkap paket data dan informasi yang ditransmisikan melalui jaringan, sehingga memungkinkan analisis lalu lintas jaringan secara menyeluruh [36]. *Wireshark* pada penelitian ini digunakan untuk menangkap *packet loss* selama proses *live migration* dan pada penelitian ini *wireshark* akan di tempatkan pada *client* untuk pengambilan data.

### 2) *VMware Workstation*

*VMware Workstation* hadir mengatasi tantangan baru ini dengan menggabungkan teknik virtualisasi terkenal, teknik dari domain lain, dan teknik baru ke dalam satu solusi. Untuk memungkinkan virtualisasi dimasukkan ke dalam sistem yang ada, *VMware Workstation* menggabungkan arsitektur yang dihosting dengan monitor mesin virtual (VMM). Arsitektur yang di-*hosting* memungkinkan pengalaman pengguna yang sederhana dan menawarkan kompatibilitas perangkat keras yang luas. Arsitektur ini memungkinkan, dengan gangguan minimal, residenti bersama tingkat sistem dari sistem operasi *host* dan VMM. *VMware Workstation* mengandalkan emulasi perangkat lunak dari perangkat I/O yang dipilih secara kanonik, sehingga juga memungkinkan enkapsulasi mesin virtual yang independen dari perangkat keras [20].

### 3.2 ALUR PENELITIAN

Penelitian ini dilakukan dalam beberapa tahap seperti yang ada pada Gambar 3.1



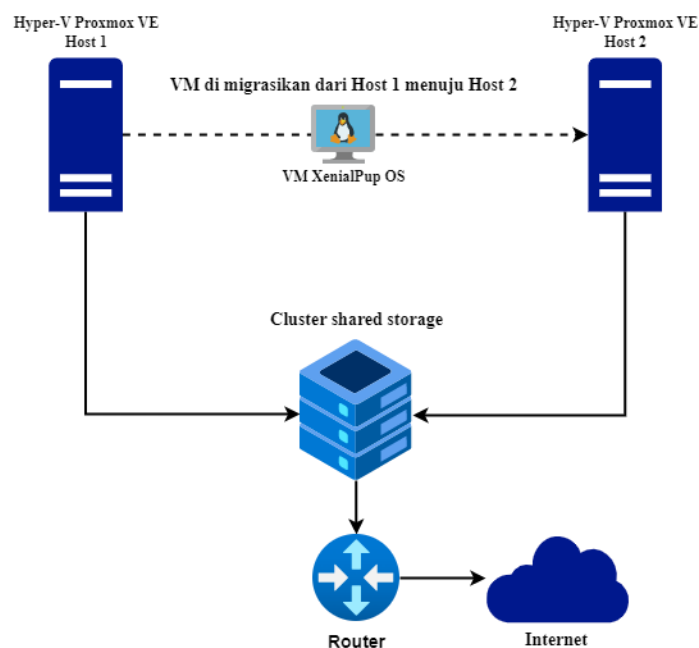
**Gambar 3.1 Diagram alur penelitian.**

Pada Gambar 3.1 menunjukkan diagram penelitian pada perancangan sistem agar hasil dari penelitian dapat tercapai dengan baik. Tahap pertama adalah menemukan landasan teoritis untuk penelitian melalui publikasi dan jurnal terkait penelitian, selanjutnya perlu untuk memeriksa tuntutan dalam penelitian, seperti yang diperlukan untuk implementasinya. Kemudian merancang sistem *server live migration*. Selanjutnya yaitu pengujian *live migration* menggunakan metode *pre-copy*. Jika berhasil maka dapat melakukan pengujian terhadap sistem dan melakukan pengambilan data berdasarkan parameter data transfer, *downtime*,

*migration time, throughput, delay, jitter, dan packetloss*. Jika pada saat pengujian *live migration* gagal maka akan mengulang pada konfigurasi *server*.

Tahapan terakhir pada penelitian ini adalah menganalisis data-data yang diperoleh untuk mengetahui kinerja *live migration*. Untuk memahami penelitian ini dan untuk mendapatkan hasil terbaik, penting untuk memperhatikan tujuan penelitian saat menarik kesimpulan. Pembaca dimaksudkan untuk memahami kekurangan dalam penelitian ini dengan menggunakan saran-saran yang telah disediakan. Selain itu, menjadi inspirasi untuk penelitian lain dengan tema terkait.

### 3.3 TOPOLOGI JARINGAN



**Gambar 3.2 Topologi jaringan.**

Topologi jaringan yang digunakan pada penelitian ini ditunjukkan pada Gambar 3.2. Topologi pada penelitian ini terdiri dari satu *router* yang berfungsi untuk menghubungkan *cluster* dengan *internet*. Terdapat satu *cluster shared storage* digunakan untuk manajemen terpusat dari beberapa *server* dan merujuk pada penyimpanan yang dapat diakses oleh semua *hyper-v Proxmox VE* dalam sebuah *cluster* tanpa mengganggu operasi yang berlangsung. Ini berarti bahwa semua *hyper-v Proxmox VE* dalam *cluster* dapat membaca dan menulis data ke penyimpanan bersama tanpa perlu mengganggu *server* lainnya. Dalam *cluster* terdiri dua *server hyper-v Proxmox VE* yang *storage*-nya saling terhubung dan

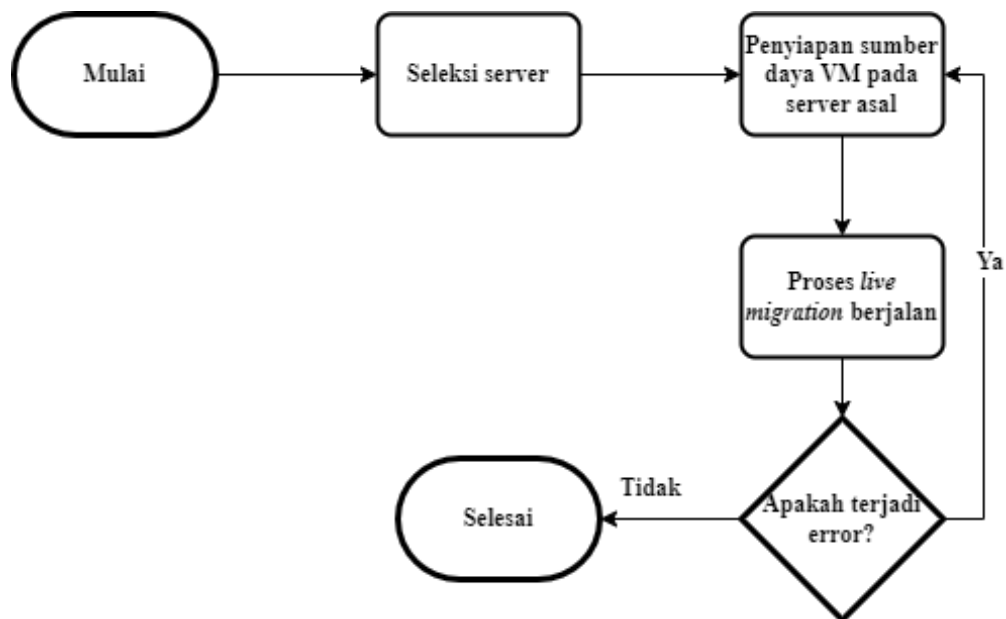
menggunakan sistem operasi *Proxmox VE 7.4*. Terdapat satu VM yang menggunakan sistem operasi *Xenialpup* dan sebagai objek untuk dimigrasikan dari *hyper-v Proxmox VE host 1* menuju *hyper-v Proxmox VE host 2*.

### 3.4 PERANCANGAN PROSES MIGRASI

Pada penelitian ini menggunakan metode dalam *live migration* yaitu *pre-copy*. Berikut proses pengimplementasi dari metode *pre-copy*:

#### 1) Alur *Pre-copy*

Pada Gambar 3.3 dijelaskan proses *live migration* metode *pre-copy* dengan 5 tahapan.



**Gambar 3.3** Alur *Pre-copy*

Proses migrasi menggunakan *pre-copy* dimulai dengan menyeleksi VM yang ada di *server* asal. Kemudian dilakukannya proses penyiapan sumber daya *server* asal, seperti status CPU, RAM, dan *disk* pada VM100. Setelahnya dilakukan proses *live migration* dari *server* asal menuju *server* tujuan. Pada saat proses *live migration pre-copy*, data yang di migrasikan berupa status CPU, RAM, *disk*, dan layanan yang berjalan pada VM100. Pada saat proses *live migration* menggunakan metode *pre-copy* pertama-tama yang ditransfer adalah *disk*, kemudian dilanjut dengan status CPU, RAM, dan yang terakhir layanan. Lalu apabila selama proses *live migration* ada kegagalan maka

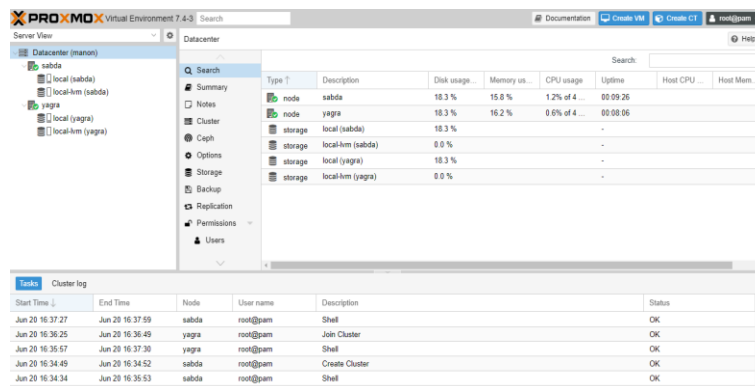
proses *live migration* akan berhenti, tetapi jika tidak adanya kegagalan pada proses *live migration* maka prosesnya akan tetap berjalan.

### 3.5 IMPLEMENTASI DAN KONFIGURASI SISTEM

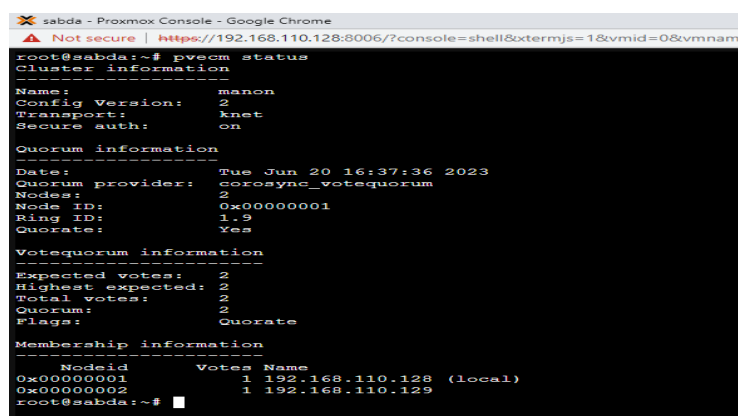
Pengimplementasian dan konfigurasi sistem adalah langkah penting dalam memastikan sistem komputer atau perangkat lunak dapat berfungsi secara optimal. Berikut implementasi dan konfigurasi untuk membangun *server live migration*.

#### 1) Implementasi Sistem

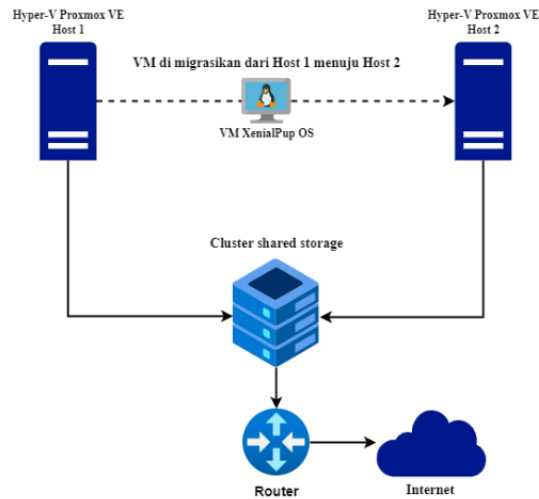
Penelitian ini membutuhkan dua *server Proxmox VE 7.4* untuk melakukan *live migration* dan *server* dibangun menggunakan *software VMware Workstation*. Pada gambar 3.4 merupakan implementasi sistem dengan keterangan gambar (a) implementasi *server proxmox ve*, (b) implementasi *cluster*, dan (c) merupakan topologi jaringan.



(a)



(b)



(c)

### Gambar 3.4 Implementasi Sistem *Live Migration*

(a) Implementasi *server proxmox VE 7.4*

(b) Implementasi *cluster pada kedua server*

(c) Topologi jaringan VM

#### 2) Konfigurasi *Cluster*

*Cluster* merupakan bagian yang penting dalam penelitian ini. *Cluster* adalah suatu fitur dari *proxmox VE* yang berfungsi untuk melakukan manajemen terpusat dari banyak *server proxmox VE*. *Cluster* terdiri dari minimal satu master *cluster* dan satu node. Manfaat *cluster* pada *proxmox VE* adalah 1).Manajemen terpusat melalui *webGUI* 2).Satu login dan password untuk mengakses semua node dan 3).Melakukan *live migration* VM.

Untuk mengkonfigurasi *cluster* pada *proxmox VE* adalah *install* terlebih dahulu minimal dua *server proxmox VE*. Pastikan masing-masing *server* memiliki *hostname* yang berbeda. Konfigurasi *cluster* dapat dilakukan menggunakan *terminal proxmox VE*, berikut konfigurasinya:

- a) Buka *terminal* pada *proxmox VE* yang akan dijadikan master *cluster*-nya kode perintah ini akan membuat *cluster* baru dengan nama yang ditentukan dan menjadikan *node* tersebut sebagai *node* utama dengan kode perintah sebagai berikut:

*Input:*

```
#pvecm create manon
```

*Output:*

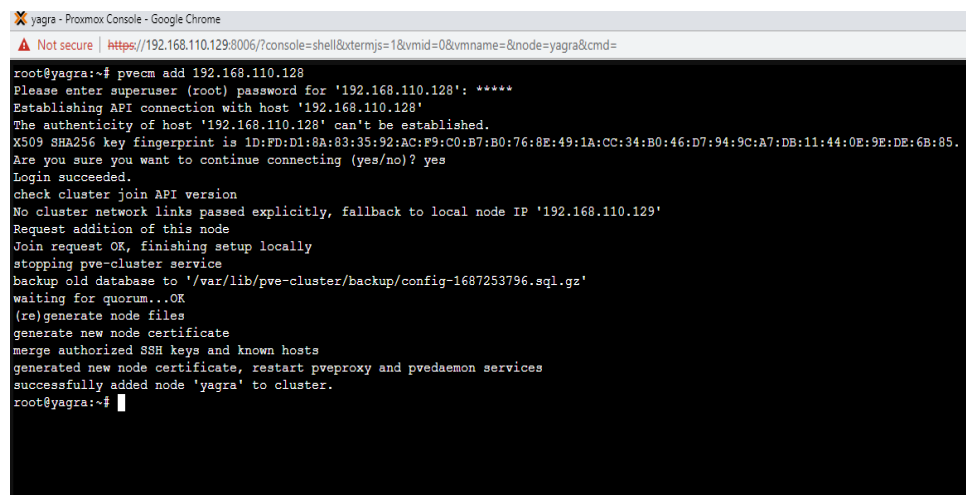
```
Corosync Cluster Engine Authentication key generator
Gathering 2048 bits for key from /dev/urandom.
Writing corosync key to /etc/corosync/authkey.
Writing corosync config to /etc/pve/corosync.conf
Restart corosync and cluster filesystem
```

- b) Perintah ini akan memberikan informasi yang diperlukan untuk bergabung dengan kluster, termasuk alamat IP node utama dan port komunikasi kluster. Kode perintahnya sebagai berikut dan pada gambar 3.5 adalah *output*::

*Input:*

```
#pvecm add 192.168.110.128
```

*Output:*



```
root@yagra:~# pvecm add 192.168.110.128
Please enter superuser (root) password for '192.168.110.128': ****
Establishing API connection with host '192.168.110.128'
The authenticity of host '192.168.110.128' can't be established.
X509 SHA256 key fingerprint is 1D:FD:D1:8A:83:35:92:AC:F9:C0:B7:B0:76:8E:49:1A:CC:34:B0:46:D7:94:9C:A7:DB:11:44:0E:9E:DE:6B:85.
Are you sure you want to continue connecting (yes/no)? yes
Login succeeded.
check cluster join API version
No cluster network links passed explicitly, fallback to local node IP '192.168.110.129'
Request addition of this node
Join request OK, finishing setup locally
stopping pve-cluster service
backup old database to '/var/lib/pve-cluster/backup/config-1687253796.sql.gz'
waiting for quorum...OK
(re)generate node files
generate new node certificate
merge authorized SSH keys and known hosts
generated new node certificate, restart pveproxy and pvedaemon services
successfully added node 'yagra' to cluster.
root@yagra:~#
```

**Gambar 3.5** Penambahan node2 ke *master cluster*.

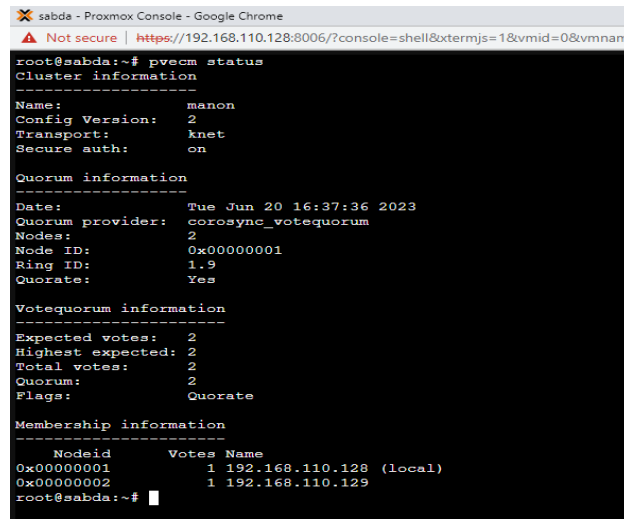
- c) melihat informasi tentang status kluster, seperti nama kluster, status node, status *service*, *ring id*, *quorum* status, dan *node meassges* status apakah node-node dalam kluster berfungsi dengan baik atau ada masalah yang perlu diatasi. Kode perintah untuk mengecek ketersediaan tersebut sebagai berikut dan gambar 3.6 merupakan *output*-nya:

*Input:*

```
#pvecm status
```



Output:



```
root@sabda:~# pvecm status
Cluster information
-----
Name:          manon
Config Version: 2
Transport:     knet
Secure auth:   on

Quorum information
-----
Date:          Tue Jun 20 16:37:36 2023
Quorum provider: corosync_votequorum
Nodes:         2
Node ID:       0x00000001
Ring ID:       1.9
Quorate:       Yes

Votequorum information
-----
Expected votes: 2
Highest expected: 2
Total votes:    2
Quorum:         2
Flags:          Quorate

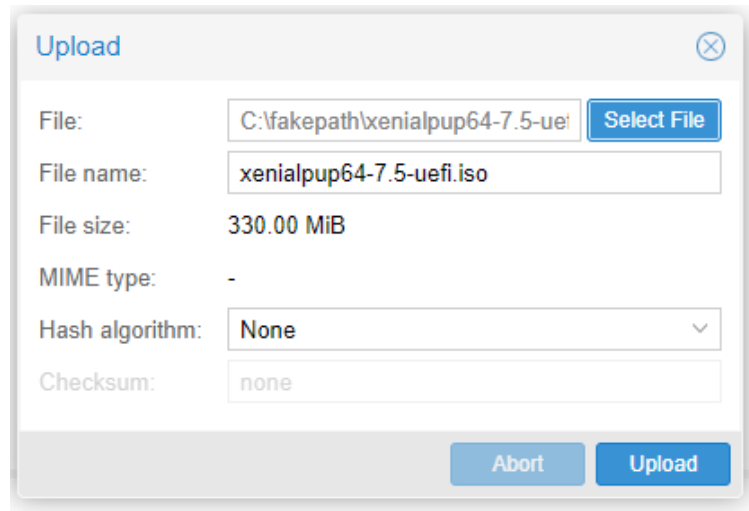
Membership information
-----
Nodeid      Votes Name
0x00000001   1 192.168.110.128 (local)
0x00000002   1 192.168.110.129
root@sabda:~#
```

Gambar 3.6 Status Cluster.

### 3) Konfigurasi VM

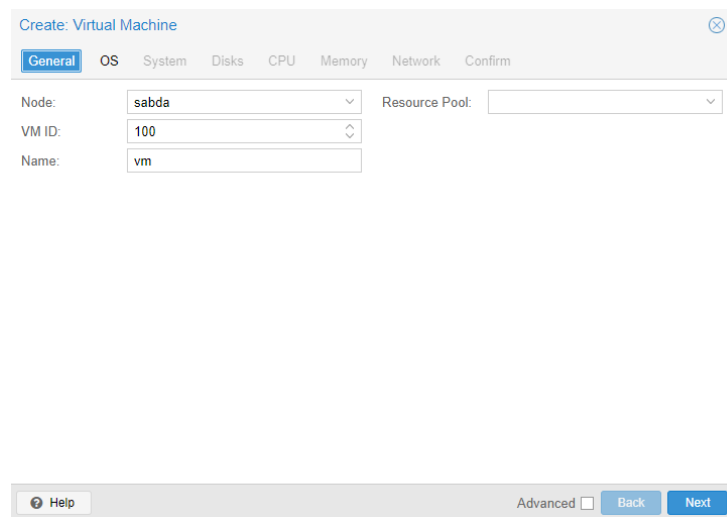
Konfigurasi VM dilakukan untuk pada satu *server proxmox VE*. VM yang dibuat berguna untuk objek *live migration*, nama VM dari penelitian ini adalah VM100 dan menggunakan sistem operasi *PuppyLinux*. Berikut Langkah-langkah untuk mengkonfigurasi VM pada *server proxmox VE*:

- a) Sebelum mengkonfigurasi VM terlebih dahulu unggah *file iso bootable* dari *PuppyLinux* agar dapat terdeteksi oleh *proxmox* dan berfungsi untuk menginstall sistem operasi *puppylinux* pada VM100. Mengunggah *file iso* dapat dilihat pada Gambar 3.7.



**Gambar 3.7 Mengunggah file iso.**

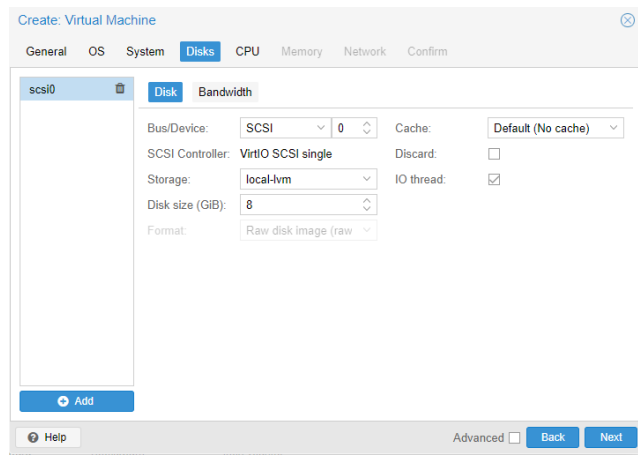
- b) Kemudian untuk membuat VM dan mengkonfigurasinya, pada Gambar 3.8 terdapat beberapa sub-menu pada menu *general* yaitu, node merujuk pada *server proxmox VE* yang akan dibuat sebuah VM, VM ID merupakan kode pengenal VM, dan nama merupakan nama pengenal untuk VM.



**Gambar 3.8 Menu General**

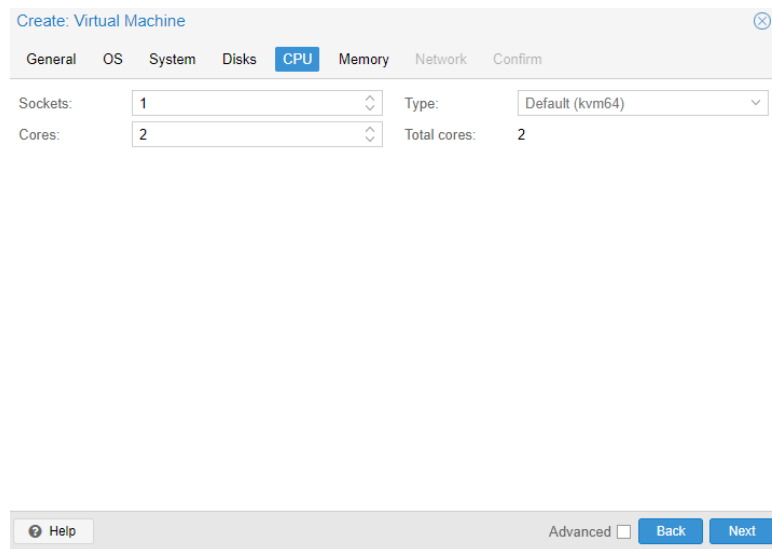
- c) Konfigurasi penyimpanan pada VM *proxmox VE*, pada gambar 3.9 terdapat beberapa sub-menu yaitu, SCSI (*Small Computer System Interface*) merupakan tipe penyimpanan *hardisk*, *VirtIO SCSI Single* merupakan tipe penyimpanan yang berbentuk *virtual*, *storage* merupakan tempat penyimpanan *virtual disk* yaitu pada *local-vm*, dan *disk size*

merupakan kapasitas *virtual disk* yang akan dijadikan tempat penyimpanan sistem operasi.



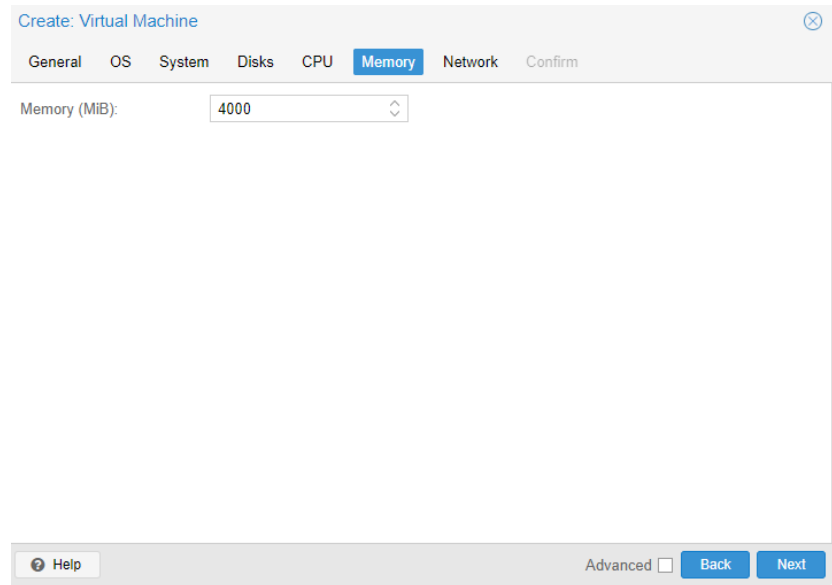
**Gambar 3.9 Menu Disk.**

- d) Konfigurasi *virtual cpu* yang akan digunakan dalam sistem operasi, pada gambar 3.10 ada beberapa sub-menu yaitu, *sockets* merupakan jumlah prosesor yang digunakan, dan *core* unit pemrosesan inti dalam cpu. Dalam penelitian ini menggunakan satu *sockets* dan dua *core virtual cpu*.



**Gambar 3.10 Menu CPU.**

- e) Konfigurasi pada *virtual memory* atau RAM, pada gambar 3.11 ada sub-menu yaitu *memory* merujuk pada kapasitas *memory* yang digunakan pada VM100. Pada penelitian ini menggunakan kapasitas *memory* sebesar 4GB.



**Gambar 3.11 Menu *Memory*.**

#### 4) Implementasi *Live Migration*

Implementasi *live migration* pada penelitian ini menggunakan *proxmox VE 7.4* sebagai *platform* virtualisasi *open-source* berbasis *linux Debian*. *Proxmox* memiliki fitur *Live migration* untuk memindahkan VM yang berjalan pada suatu *server* ke *server* tujuan tanpa memutuskan *client*. Sebelum mengimplementasikan *live migration*, diperlukan konfigurasi *cluster* sebagai manajemen terpusat dan mengkonfigurasi VM pada *server proxmox VE*. Berikut implementasi *live migration*:

- a) Setelah membuat dan mengkonfigurasi VM pada *server proxmox VE*. Untuk melakukan *live migration* pada VM, seleksi *server node1* kemudian klik *shell* pada *webgui proxmox VE* akan muncul jendela terminal. Lalu masukkan kode perintah berikut:

```
#qm migrate 100 yagra --online --force --with-local-disks
```

Keterangan kode perintah: “*qm migrate*” adalah kode perintah migrasi VM, “100” merupakan kode pengenalan (VM ID) yang dibuat pada konfigurasi VM, “yagra” merujuk pada *server proxmox ve* yang akan dijadikan tujuan untuk melakukan *live migration*, “--online” digunakan untuk *live migration* jika VM sedang berjalan, “--force” merupakan kode perintah untuk mengizinkan *live migration* VM yang menggunakan perangkat lokal, dan

“--with-local-disks” digunakan untuk mengaktifkan *live storage migration* pada penyimpanan lokal.

- b) *Output* kode perintah tersebut dapat dilihat pada gambar 3.12 dengan gambar (a) proses *live migration*, dan gambar (b) merupakan hasil akhir, dimana akan menghasilkan data *migration time*, *downtime*, dan data transfer.

```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun 20 16:37:28 WIB 2023 on pts/0
root@sabda:~# qm migrate 100 yagca --online --force --with-local-disk
2023-06-20 17:02:48 starting migration of VM 100 to node 'yagca' (192.168.
8.310:152)
2023-06-20 17:02:48 found local disk 'local-lvm:vm-100-disk-0' (in curru
at VM context)
2023-06-20 17:02:48 starting VM 100 on remote node 'yagca'
2023-06-20 17:02:54 volume 'local-lvm:vm-100-disk-0' is 'local-lvm:vm-10
0-disk-0' on the target
2023-06-20 17:02:54 start remote tunnel
2023-06-20 17:02:57 start tunnel over
2023-06-20 17:02:57 starting storage migration
2023-06-20 17:02:57 starting live migration to sdb:unix:/run/qemu-server/
100_mbd.migrate:exportname=drive-ncs10
drive-ncs10: starting live migration
drive-ncs10: transferred 0.0 B of 8.0 GiB (0.00%) in 0s
drive-ncs10: transferred 18.0 MiB of 8.0 GiB (0.22%) in 3s
drive-ncs10: transferred 63.0 MiB of 8.0 GiB (0.77%) in 2s
drive-ncs10: transferred 115.0 MiB of 8.0 GiB (1.43%) in 2s
drive-ncs10: transferred 145.0 MiB of 8.0 GiB (1.77%) in 2s
drive-ncs10: transferred 161.0 MiB of 8.0 GiB (1.97%) in 2s
drive-ncs10: transferred 185.0 MiB of 8.0 GiB (2.28%) in 2s
drive-ncs10: transferred 209.0 MiB of 8.0 GiB (2.58%) in 2s
drive-ncs10: transferred 236.0 MiB of 8.0 GiB (2.88%) in 2s
drive-ncs10: transferred 261.0 MiB of 8.0 GiB (3.19%) in 10s
drive-ncs10: transferred 287.0 MiB of 8.0 GiB (3.53%) in 11s
drive-ncs10: transferred 295.0 MiB of 8.0 GiB (3.60%) in 12s
drive-ncs10: transferred 298.0 MiB of 8.0 GiB (3.64%) in 13s
drive-ncs10: transferred 309.0 MiB of 8.0 GiB (3.77%) in 14s

```

(a)

```

drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 38s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 39s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 40s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 41s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 42s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 43s
drive-ncs10: transferred 7.8 GiB of 8.0 GiB (97.56%) in 4m 44s
drive-ncs10: transferred 7.9 GiB of 8.0 GiB (98.14%) in 4m 45s
drive-ncs10: transferred 7.9 GiB of 8.0 GiB (98.57%) in 4m 46s
drive-ncs10: transferred 8.0 GiB of 8.0 GiB (100.00%) in 4m 47s
drive-ncs10: transferred 8.0 GiB of 8.0 GiB (100.00%) in 4m 48s, ready
all 'mirror' jobs are ready
2023-06-20 22:32:18 starting online/live migration on unix:/run/qemu-server/100.migrate
2023-06-20 22:32:18 see migration capabilities
2023-06-20 22:32:18 migration downtime limit: 100 ms
2023-06-20 22:32:18 migration overhead: 812.0 MiB
2023-06-20 22:32:18 set migration parameters
2023-06-20 22:32:18 start migrate command to unix:/run/qemu-server/100.migrate
2023-06-20 22:32:19 migration active, transferred 44.9 MiB of 3.9 GiB VM-state, 897.7 MiB/s
2023-06-20 22:32:20 migration active, transferred 126.7 MiB of 3.9 GiB VM-state, 76.2 MiB/s
2023-06-20 22:32:21 migration active, transferred 213.1 MiB of 3.9 GiB VM-state, 125.1 MiB/s
2023-06-20 22:32:22 migration active, transferred 290.7 MiB of 3.9 GiB VM-state, 110.2 MiB/s
2023-06-20 22:32:23 migration active, transferred 354.4 MiB of 3.9 GiB VM-state, 59.3 MiB/s
2023-06-20 22:32:24 migration active, transferred 433.9 MiB of 3.9 GiB VM-state, 127.4 MiB/s
2023-06-20 22:32:25 migration active, transferred 493.6 MiB of 3.9 GiB VM-state, 75.1 MiB/s
2023-06-20 22:32:26 migration active, transferred 593.2 MiB of 3.9 GiB VM-state, 119.1 MiB/s
2023-06-20 22:32:27 migration active, transferred 670.6 MiB of 3.9 GiB VM-state, 122.8 MiB/s
2023-06-20 22:32:28 migration active, transferred 742.4 MiB of 3.9 GiB VM-state, 65.4 MiB/s
2023-06-20 22:32:29 migration active, transferred 817.4 MiB of 3.9 GiB VM-state, 110.7 MiB/s
2023-06-20 22:32:31 average migration speed: 309.0 MiB/s - downtime 69 ms
2023-06-20 22:32:31 migration status: completed
all 'mirror' jobs are ready
drive-ncs10: Completing block job id...
drive-ncs10: Completed successfully.
drive-ncs10: mirror-job finished
2023-06-20 22:32:32 stopping NBD storage migration server on target.
logical volume 'vm-100-disk-0' successfully removed
2023-06-20 22:32:38 migration finished successfully (duration 00:05:16)
root@sabda:~#

```

(b)

Gambar 3.12 Implementasi *Live Migration* VM.

(a) Proses *Live Migration*

(b) Hasil akhir *Live Migration*

### 3.6 PENGUJIAN PERFORMANSI *LIVE MIGRATION*

Pada pengujian performansi *live migration* pada penelitian ini dilakukan dengan dua skenario pengujian yaitu *live migration* pada saat bermain game dan juga menonton video *offline*. Setiap skenario pengujian dilakukan sebanyak tiga puluh kali pengambilan *sample* untuk mendapatkan hasil yang valid dan tepat. Pengujian performansi *live migration* dengan parameter luaran *migration time*,

*downtime*, dan data transfer bertujuan untuk melihat kualitas proses *live migration* VM menggunakan metode *pre-copy*. Kemudian untuk menguji kualitas jaringan pada proses *live migration* maka luaran parameternya adalah *throughput*, *delay*, *jitter*, dan *packet loss*.

**Tabel 3.3 Skenario pengujian.**

No	Skenario pengujian	Parameter pengujian
1	Bermain game	<i>Migration time</i> , <i>downtime</i> , data transfer, dan QoS
2	Menonton video <i>offline</i>	<i>Migration time</i> , <i>downtime</i> , data transfer, dan QoS

Pengujian *live migration* yang dilakukan dengan mentransfer VM dari *server* asal ke *server* tujuan menggunakan fitur *proxmox VE*. Pada Tabel 3.3 *Live migration* pada penelitian ini menggunakan skenario bermain *game* dan menonton video *offline*, dilakukan pada saat VM menyala pada *server* asal. Pengujian *live migration* dilakukan dengan perintah “*qm migrate 100 yagra --online --force --with-local-disks*”. Perintah tersebut digunakan untuk melakukan proses *live migration*, dimana “100” merupakan VMID *server* asal dan “yagra” merupakan nama node pada *server* tujuan. Penggunaan perintah tersebut mendapatkan parameter *downtime*, *migration time*, dan data transfer. Kemudian untuk mendapatkan parameter QoS atau kualitas suatu jaringan pada proses *live migration* menggunakan *software wireshark*. *Software* tersebut digunakan untuk menangkap *traffic* jaringan pada proses *live migration*. Hasil dari *software wireshark* akan mendapatkan parameter *throughput*, *delay*, *jitter*, dan *packetloss*.

### 3.6.1 Parameter Pengujian *Live Migration*

#### 1) *Migration time*

*Migration time* adalah waktu yang dibutuhkan dalam menyelesaikan transfer VM dari *server* asal menuju *server* tujuan [20]. Kategori *migration time* tercantum pada tabel 3.4.

**Tabel 3.4 Kategori *Migration time* menurut Gustafsson [37].**

<b>Kategori</b>	<b><i>Migration time</i> (s)</b>
Sangat baik	1-30 <i>second</i>
Baik	31-80 <i>second</i>
Buruk	81-150 <i>second</i>
Sangat Buruk	>151 <i>second</i>

## 2) *Downtime*

Waktu yang diambil oleh proses migrasi untuk menghentikan *virtual machine* di sumber dan melanjutkan di *host* target. Ini secara langsung mempengaruhi ketersediaan layanan [20]. Kategori *downtime* tercantum pada tabel 3.5.

**Tabel 3.5 Kategori *Downtime* menurut Gustafsson [37].**

<b>Kategori</b>	<b><i>Downtime</i> (ms)</b>
Sangat baik	0-10 ms
Baik	11-60 ms
Buruk	61-99 ms
Sangat buruk	> 100 ms

## 3) Data transfer

Data Transfer adalah jumlah total data yang ditransfer selama migrasi. Untuk kinerja yang lebih baik, nilai ini harus sama dengan jumlah total data VM [20].

*Downtime* terjadi pada saat VM dinyalakan kembali setelah proses mentransfer status CPU, RAM, dan *disk*. Sedangkan *migration time* dihitung sejak awal proses *live migration* berjalan dan pada saat VM kembali berjalan pada *server* tujuan.