

**LAPORAN
PRAKTIK KERJA LAPANGAN
DI**

PT INDOSTORAGE SOLUSI TEKNOLOGI

Jl. Pramuka Raya RT. 007 RW. 004, Kel. Mampang, Kec. Pancoran Mas, Depok
16433

**RANCANG BANGUN LAYANAN PRIVATE CLOUD MEGGUNAKAN
OPENSTACK PADA UBUNTU 22.04**

Disusun untuk memenuhi salah satu syarat kelulusan dari SMKN 1 Cimahi

Oleh:

NAMA	: PAJRI ZAHRAWAANI AHMAD
NO. INDUK SISWA	: 181113955
TINGKAT	: IV (EMPAT)
PAKET KEAHILAN	: TEKNOLOGI INFORMASI DAN KOMUNIKASI
BIDANG KEAHLIAN	: SISTEM INFORMATIKA JARINGAN DAN APLIKASI



**SEKOLAH MENENGAH KEJURUAN NEGERI 1
KOTA CIMAHI
2023**

PENGESAHAN DARI PIHAK INDUSTRI

**RANCANG BANGUN LAYANAN PRIVATE CLOUD MEGGUNAKAN
OPENSTACK PADA UBUNTU 22.04**

Laporan ini telah disetujui oleh:

Pembimbing

KIKI FAKHRI DERMAWAN, S.T

Mengetahui,

Direktur

ELAN SUHERLAN, S.Kom

PT INDOSTORAGE SOLUSI TEKNOLOGI

2023

PENGESAHAN DARI PIHAK SEKOLAH

**RANCANG BANGUN LAYANAN PRIVATE CLOUD MEGGUNAKAN
OPENSTACK PADA UBUNTU 22.04**

Laporan ini telah disetujui oleh:

Ketua Kopetensi Keahlian SIJA,

Pembimbing,

DIKY RIDWAN, S. Kom.

NIP. 197507032009021001

DIKY RIDWAN, S. Kom.

NIP. 197507032009021001

Mengetahui:

Kepala Sekolah SMK Negeri 1 Cimahi,

AGUS PRIYATMONO NUGROHO, S. Pd, M. Si.

NIP : 196708311990031002



KATA PENGANTAR

Puji serta syukur penulis panjatkan kepada Allah *Subhanahu Wa Ta'ala* yang telah memberikan rahmat serta karunia-Nya sehingga penulis dapat menyelesaikan penyusunan laporan Praktik Kerja Lapangan yang berjudul “RANCANG BANGUN LAYANAN PRIVATE CLOUD MEGGUNAKAN OPENSTACK PADA UBUNTU 22.04” sesuai kemampuan penulis.

Penulisan laporan Praktik Kerja Lapangan ini adalah salah satu syarat untuk kelulusan di Sekolah Menengah Kejuruan Negeri 1 Cimahi serta sebagai bukti tertulis dari hasil Praktik Kerja Industri di PT. Indostorage Solusi Teknologi yang dilaksanakan selama 6 (enam) bulan yang dimulai pada tanggal 1 Agustus 2022 sampai tanggal 31 Januari 2023.

Penulisan laporan Praktik Kerja Lapangan ini tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Penulis mengucapkan terima kasih atas bantuan dan dukungan yang telah diberikan demi kelancaran dan selesainya penulisan laporan ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Orang tua dan seluruh keluarga penulis yang selalu memberikan do'a serta dukungan kepada penulis.
2. Bapak Elan Suherlan, S. Kom, selaku Direktur PT. Indostorage Solusi Teknologi
3. Bapak Kiki Fakhri Dermawan, S.T, selaku Pembimbing dari pihak perusahaan
4. Seluruh rekan di PT. Indostorage Solusi Teknologi yang bersedia memberikan ilmu dan pengalaman kepada penulis selama Praktik Kerja Lapangan (PKL).
5. Bapak Agus Priyatmono Nugroho, S.Pd., M. Si., selaku Kepala Sekolah Menengah Kejuruan Negeri 1 Cimahi.
6. Bapak Diky Ridwan, S.Kom., selaku Ketua Kompetensi Keahlian Sistem Informatika Jaringan dan Aplikasi dan selaku guru pembimbing dari pihak sekolah yang telah membantu dan membimbing dalam pelaksanaan Praktik Kerja Industri serta penyusunan laporan sehingga dapat diselesaikan dengan tuntas oleh penulis.

7. Bapak Antoni Budiman, S.Pd, M.Pd., selaku Kepala Bengkel Sistem Informatika Jaringan dan Aplikasi dan selaku wali kelas XIII Sistem Informatika Jaringan dan Aplikasi B.
8. Seluruh guru dan staf Sekolah Menengah Kejuruan Negeri 1 Cimahi.
9. Seluruh teman tingkat XIII Sekolah Menengah Kejuruan Negeri 1 Cimahi, khususnya pada kompetensi keahlian Sistem Informatika Jaringan dan Aplikasi yang telah memberikan saran, dukungan serta semangat kepada penulis.

Penulis menyadari bahwa penulisan laporan ini jauh dari sempurna, terdapat banyak kelemahan dan kekurangan baik pada materi isi maupun teknik penulisan. Hal ini semata-mata disebabkan keterbatasan kemampuan, pengalaman dan pengetahuan yang penulis miliki, serta keterbatasan dalam memperoleh data dan informasi. Oleh karena itu penulis mengharapkan kritik serta saran yang bersifat membangun demi kesempurnaan laporan Praktik Kerja Industri ini

Akhir kata, penulis mengucapkan terima kasih banyak kepada seluruh pihak yang terkait, semoga Allah *Subhanahu wa Ta'ala* meridhai usaha dan hasil yang penulis lakukan, Aamiin.

Cimahi, Maret 2023

Penulis

DAFTAR ISI

KATA PENGANTAR.....	i
DAFTAR ISI.....	iii
DAFTAR GAMBAR.....	vi
DAFTAR TABEL	xii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang Masalah.....	1
1.2 Tujuan	2
1.3 Pembatasan Masalah	2
1.4 Sistematika Penulisan.....	3
BAB II TINJAUAN PERUSAHAAN	4
2.1 Profil Perusahaan	4
2.2 Visi dan Misi Perusahaan.....	4
2.2.1 Visi	4
2.2.2 Misi	4
2.3 Struktur Organisasi Perusahaan	4
2.4 Lini Bisnis	5
2.5 Layanan dan Produk Perusahaan	5
2.5.1 Solusi Infrastruktur	5
2.5.2 Proeksi Data	6
2.5.3 <i>Cloud and Virtualization</i>	6
2.5.4 <i>Microservices</i>	6
2.5.5 <i>Big Data</i>	6
2.5.6 Data Integration.....	7
2.5.7 DevOps.....	7
BAB III LANDASAN TEORI.....	8
3.1 Jaringan Komputer	8
3.1.1 Jaringan Komputer Berdasarkan Jangkauan Geografis	8
3.1.2 Jaringan Komputer Berdasarkan Topologi	9
3.1.3 Jaringan Komputer Berdasarkan Media Transmisi Data	12
3.1.4 Jaringan Komputer Berdasarkan Hubungan Tiap Komputer.....	12

3.2	Model Referensi OSI.....	13
3.3	Model TCP/IP	15
3.4	IP Address	17
3.4.1	IPv4	17
3.5	Internet	17
3.6	World Wide Web (WWW).....	18
3.7	Web Server.....	18
3.7.1	Apache HTTP Server	18
3.8	Application Programming Interface (API).....	19
3.9	Sistem Operasi	19
3.9.1	Ubuntu.....	19
3.10	Virtualisasi	20
3.11	<i>Hypervisor</i>	20
3.12	Cloud Computing.....	21
3.12.1	Karakteristik <i>Cloud Computing</i>	21
3.12.2	Model Layanan <i>Cloud Computing</i>	22
3.12.3	Model Penyebaran <i>Cloud Computing</i>	23
3.13	<i>Secure Shell</i> (SSH).....	24
3.14	<i>Windows Terminal</i>	24
3.15	<i>Network Time Protocol</i> (NTP).....	25
3.16	<i>Message Broker</i>	25
3.17	MariaDB.....	25
3.18	Container	26
3.19	Memcached	26
3.20	Logical Volume Management (LVM)	26
3.21	Open Virtual Network (OVN)	27
3.22	OpenStack	28
3.22.1	<i>Keystone</i>	29
3.22.2	<i>Placement</i>	29
3.22.3	<i>Glance</i>	30
3.22.4	<i>Nova</i>	30

3.22.5 Neutron.....	30
3.22.6 Cinder.....	32
3.22.7 Skyline	32
BAB IV PEMBAHASAN.....	33
4.1 Perencanaan.....	33
4.1.1 Topologi	33
4.1.2 Spesifikasi Kebutuhan Perangkat.....	34
4.2 Langkah Kerja.....	35
4.2.1 Persiapan <i>Environment</i>	35
4.2.2 Instalasi dan Konfigurasi <i>Keystone</i>	40
4.2.3 Instalasi dan Konfigurasi <i>Glance</i>	42
4.2.4 Instalasi dan Konfigurasi <i>Placement</i>	45
4.2.5 Instalasi dan Konfigurasi <i>Nova</i>	48
4.2.6 Instalasi dan Konfigurasi <i>Neutron</i>	56
4.2.7 Instalasi dan Konfigurasi <i>Cinder</i>	62
4.2.8 Instalasi dan Konfigurasi <i>Skyline</i>	68
4.3 Pengujian.....	71
BAB V PENUTUP.....	85
5.1 Kesimpulan	85
5.2 Saran.....	85
DAFTAR PUSTAKA	86

DAFTAR GAMBAR

Gambar 2.1 Logo PT. Indostorage Solusi Teknologi	4
Gambar 2.2 Struktur Organisasi PT. Indostorage Solusi Teknologi.....	5
Gambar 3.1 Topologi <i>Bus</i>	9
Gambar 3.2 Topologi <i>Ring</i>	10
Gambar 3.3 Topologi <i>Star</i>	10
Gambar 3.4 Topologi <i>Tree</i>	11
Gambar 3.5 Topologi <i>Mesh</i>	11
Gambar 3.6 <i>Layer Model Referensi OSI</i>	13
Gambar 3.7 Perbandingan <i>Layer Model TCP/IP</i> dan <i>Layer Model OSI</i>	16
<i>Gambar 3.8 Logo Apache HTTP Server</i>	18
<i>Gambar 3.9 Logo Ubuntu</i>	20
Gambar 3.10 Tipe <i>Hypervisor</i>	21
<i>Gambar 3.11 Model Layanan Cloud Computing</i>	23
Gambar 3.12 Logo MariaDB	26
Gambar 3.13 Deskripsi LVM.....	27
Gambar 3.14 Arsitektur OVN	28
Gambar 3.15 Logo OpenStack.....	29
Gambar 4.1 Topologi yang Digunakan.....	33
Gambar 4.2 Melakukan <i>Remote Connection</i> pada <i>Controller Node</i>	35
Gambar 4.3 Melakukan <i>Remote Connection SSH</i> pada <i>Compute Node</i>	36
Gambar 4.4 Perintah Instalasi Layanan NTP.....	36
Gambar 4.5 Konfigurasi Layanan NTP	36
Gambar 4.6 Me- <i>restart</i> dan Mengecek Layanan NTP pada <i>Controller Node</i>	36
Gambar 4.7 Sinkronisasi NTP <i>Compute Node</i> dengan <i>Controler Node</i>	37
Gambar 4.8 Me- <i>restart</i> dan Mengecek NTP <i>Client</i> pada <i>Compute Node</i>	37
Gambar 4.9 Instalasi OpenStack <i>Client</i>	37
Gambar 4.10 Melakukan Instalasi <i>Database Server</i>	37
Gambar 4.11 Menambahkan Konfigurasi Layanan MariaDB	37
Gambar 4.12 Melakukan <i>Restart</i> MariaDB	38
Gambar 4.13 Menginstal RabbitMQ.....	38

Gambar 4.14 Menambahkan <i>User</i> OpenStack pada RabbitMQ	38
Gambar 4.15 Penambahan Hak Akses Untuk <i>User</i> OpenStack.....	38
Gambar 4.16 Perintah Instalasi Layanan Memcached.....	38
Gambar 4.17 Konfigurasi Memcached	39
Gambar 4.18 <i>Restart</i> Layanan Memcached	39
Gambar 4.19 Perintah Instalasi Layanan Etcd	39
Gambar 4.20 Konfigurasi Layanan Etcd.....	39
Gambar 4.21 <i>Restart</i> Layanan Etcd	40
Gambar 4.22 Menambahkan <i>User</i> dan <i>Database</i> Layanan <i>Keystone</i>	40
Gambar 4.23 Perintah Instalasi Layanan <i>Keystone</i>	40
Gambar 4.24 Konfigurasi <i>Database</i> <i>Keystone</i>	40
Gambar 4.25 Konfigurasi <i>Fernet Token</i> <i>Keystone</i>	41
Gambar 4.26 Sinkronasi <i>Database</i> dengan <i>Keystone</i>	41
Gambar 4.27 <i>Bootstrap</i> Layanan <i>Keystone</i>	41
Gambar 4.28 <i>Environment Variable</i> <i>User</i> <i>Keystone</i>	41
Gambar 4.29 Perintah Menambah Hak Akses	42
Gambar 4.30 Perintah dan Output dari Pembuatan <i>project service</i>	42
Gambar 4.31 Menambahkan <i>User</i> dan <i>Database</i> Layanan <i>Glance</i>	42
Gambar 4.32 Menambah <i>User</i> Layanan <i>Glance</i> pada OpenStack.....	43
Gambar 4.33 Menambah <i>Service Glance</i> pada OpenStack	43
Gambar 4.34 Menambahkan <i>Endpoint</i> Layanan <i>Glance</i>	44
Gambar 4.35 Perintah Instalasi <i>Glance</i>	44
Gambar 4.36 Konfigurasi <i>Database Glance</i>	44
Gambar 4.37 Konfigurasi untuk Menghubungkan <i>Glance</i> dan <i>Keystone</i>	45
Gambar 4.38 Konfigurasi untuk Penyimpanan <i>Image</i>	45
Gambar 4.39 Perintah Sinkronisasi <i>Database Glance</i>	45
Gambar 4.40 Perintah <i>Restart</i> Layanan <i>Glance</i>	45
Gambar 4.41 Menambahkan <i>User</i> dan <i>Database</i> Layanan <i>Placement</i>	46
Gambar 4.42 Menambah <i>User</i> Layanan <i>Placement</i> pada OpenStack	46
Gambar 4.43 Menambah <i>Service Placement</i> pada OpenStack	46
Gambar 4.44 Menambahkan <i>Endpoint</i> Layanan <i>Placement</i>	47

Gambar 4.45 Perintah Instalasi <i>Placement</i>	47
Gambar 4.46 Konfigurasi <i>Database Placement</i>	47
Gambar 4.47 Konfigurasi Menghubungkan <i>Placement</i> dengan <i>Keystone</i>	48
Gambar 4.48 Perintah Sinkronisasi <i>Database Placement</i>	48
Gambar 4.49 Perintah <i>Restart Web Server</i>	48
Gambar 4.50 Menambah <i>User</i> dan <i>Database</i> Layanan <i>Nova</i>	49
Gambar 4.51 Menambah <i>User</i> Layanan <i>Nova</i> pada <i>OpenStack</i>	49
Gambar 4.52 Menambah <i>Service Nova</i> pada <i>OpenStack</i>	49
Gambar 4.53 Menambahkan <i>Endpoint</i> Layanan <i>Nova</i>	50
Gambar 4.54 Perintah Instalasi <i>Package-package Nova</i>	50
Gambar 4.55 Konfigurasi Koneksi <i>Database Nova</i>	51
Gambar 4.56 Konfigurasi Koneksi <i>Nova Controller</i> dengan <i>Keystone</i>	51
Gambar 4.57 Konfigurasi <i>VNC</i> untuk <i>Web Console</i>	51
Gambar 4.58 Konfigurasi Koneksi <i>Nova Controller</i> dengan <i>Glance</i>	52
Gambar 4.59 Konfigurasi Koneksi <i>Nova Controller</i> dengan <i>Placement</i>	52
Gambar 4.60 Penambahan Baris Konfigurasi Koneksi <i>RabbitMQ</i>	52
Gambar 4.61 Perintah <i>Restart Package</i> Layanan <i>Nova</i>	53
Gambar 4.62 Perintah Instalasi <i>Package nova-compute</i>	53
Gambar 4.63 Penambahan Baris Konfigurasi Koneksi <i>RabbitMQ</i>	53
Gambar 4.64 Konfigurasi Koneksi <i>Nova Compute</i> dengan <i>Keystone</i>	54
Gambar 4.65 Konfigurasi <i>VNC</i> pada <i>Compute Node</i>	54
Gambar 4.66 Konfigurasi Koneksi <i>Nova Compute</i> dengan <i>Glance</i>	54
Gambar 4.67 Konfigurasi Koneksi <i>Nova Compute</i> dengan <i>Placement</i>	55
Gambar 4.68 Konfigurasi untuk <i>sceduling</i> pencarian <i>compute node</i>	55
Gambar 4.69 Menentukan <i>Hypervisor</i> yang Digunakan.....	55
Gambar 4.70 Perintah <i>Restart Package Nova Compute</i>	55
Gambar 4.71 Sinkronisasi Konfigurasi dengan <i>Database</i> Layanan <i>Nova</i>	56
Gambar 4.72 Pengecekan Layanan <i>Nova</i>	56
Gambar 4.73 Menambab <i>User</i> dan <i>Database</i> Layanan <i>Neutron</i>	56
Gambar 4.74 Menambah <i>User</i> Layanan <i>Neutron</i> pada <i>OpenStack</i>	57
Gambar 4.75 Menambah <i>Service Neutron</i> pada <i>OpenStack</i>	57

Gambar 4.76 Perintah Instalasi <i>Package-package Neutron</i>	57
Gambar 4.77 Perintah Pembuatan <i>Endpoint</i> Layanan <i>Neutron</i>	57
Gambar 4.78 Output Pembuatan <i>Endpoint</i> Layanan <i>Neutron</i>	58
Gambar 4.79 Konfigurasi Layanan <i>Neutron</i> bagian [<i>DEFAULT</i>].....	58
Gambar 4.80 Konfigurasi Koneksi <i>Database Neutron</i>	58
Gambar 4.81 Konfigurasi Koneksi <i>Neutron</i> dengan <i>Keystone</i>	59
Gambar 4.82 Konfigurasi Koneksi <i>Neutron</i> dengan <i>Nova</i>	59
Gambar 4.83 Konfigurasi Bawaan <i>Neutron</i>	59
Gambar 4.84 Perintah Konfigurasi <i>OVN</i>	60
Gambar 4.85 Konfigurasi <i>File ML2 Plugin</i>	60
Gambar 4.86 Konfigurasi <i>Metadata Agent</i> Layanan <i>Neutron</i>	61
Gambar 4.87 Konfigurasi Koneksi <i>Nova</i> dengan <i>Neutron</i>	61
Gambar 4.88 Perintah Instalasi <i>Package OVN Controller</i>	61
Gambar 4.89 Perintah Konfigurasi <i>OVN Controller</i> pada <i>Controller Node</i>	61
Gambar 4.90 Pengecekan Layanan <i>Neutron</i>	62
Gambar 4.91 Menambahkan <i>User</i> dan <i>Database</i> Layanan <i>Cinder</i>	62
Gambar 4.92 Menambah Layanan <i>Cinder</i> pada <i>OpenStack</i>	62
Gambar 4.93 Menambah <i>service Cinder</i> pada <i>OpenStack</i>	63
Gambar 4.94 Perintah Instalasi <i>Package</i> Layanan <i>Cinder</i>	63
Gambar 4.95 Perintah Pembuatan <i>Endpoint</i> Layanan <i>Cinder</i>	63
Gambar 4.96 Konfigurasi Koneksi <i>Database Cinder</i>	63
Gambar 4.97 Konfigurasi Koneksi <i>RabbitMQ</i> dan <i>Auth Type</i>	64
Gambar 4.98 Konfigurasi Koneksi <i>Cinder</i> dengan <i>Keystone</i>	64
Gambar 4.99 Perintah Sinkronasi <i>Database Cinder</i>	64
Gambar 4.100 Perintah <i>Restart</i> Layanan <i>Cinder</i>	64
Gambar 4.101 Perintah Instalasi <i>Package</i> Layanan <i>Cinder</i>	65
Gambar 4.102 Perintah Mengecek <i>Disk</i>	65
Gambar 4.103 Konfigurasi <i>Disk</i> untuk <i>LVM</i>	66
Gambar 4.104 Pembuatan <i>Physical Volume</i>	66
Gambar 4.105 Pembuatan <i>Volume Group</i>	66
Gambar 4.106 Konfigurasi Koneksi <i>RabbitMQ & Glance</i> dengan <i>Cinder</i>	67

Gambar 4.107 Konfigurasi Koneksi <i>Cinder</i> dengan <i>Database</i>	67
Gambar 4.108 Konfigurasi Koneksi <i>Cinder</i> dengan <i>Keystone</i>	67
Gambar 4.109 Konfigurasi <i>Cinder Backend</i> LVM	68
Gambar 4.110 Perintah <i>Restart</i> layanan <i>Cinder</i>	68
Gambar 4.111 Mengecek layanan <i>Cinder</i>	68
Gambar 4.112 Menambahkan <i>User</i> dan <i>Database</i> Layanan <i>Skyline</i>	68
Gambar 4.113 Menambah <i>User</i> Layanan <i>Skyline</i> pada OpenStack	69
Gambar 4.114 Perintah Instalasi <i>package</i> pendukung	69
Gambar 4.115 Menambah GPG <i>key</i> untuk repositori <i>Docker</i>	69
Gambar 4.116 Menambah Repositori <i>Docker</i>	69
Gambar 4.117 Perintah Instalasi <i>Docker</i>	69
Gambar 4.118 Perintah <i>Pulling Image</i>	70
Gambar 4.119 Konfigurasi Layanan <i>Skyline</i>	70
Gambar 4.120 <i>Bootstrapping</i> Layanan <i>Skyline</i>	70
Gambar 4.121 Perintah Menghapus <i>Container skyline_bootstrap</i>	71
Gambar 4.122 Perintah Menjalankan <i>Container</i> Layanan <i>Skyline</i>	71
Gambar 4.123 Perintah Pembuatan Membuat <i>External Network</i>	72
Gambar 4.124 Perintah Pembuatan <i>External Subnet</i>	73
Gambar 4.125 Perintah Pembuatan <i>Internal Network</i>	73
Gambar 4.126 Perintah Pembuatan <i>Internal Subnet</i>	74
Gambar 4.127 Perintah Pembuatan <i>Router</i>	74
Gambar 4.128 Perintah Menghubungkan <i>Network</i> dengan <i>Router</i>	75
Gambar 4.129 Perintah Mencari Informasi Alamat IP <i>Router</i>	75
Gambar 4.130 Pengujian Koneksi <i>Router</i>	75
Gambar 4.131 Tampilan Halaman <i>Login Dashboard</i> OpenStack	76
Gambar 4.132 Tampilan <i>Dashboard</i> OpenStack	76
Gambar 4.133 Tampilan Halaman <i>Flavor Admin</i>	77
Gambar 4.134 Tampilan Pembuatan <i>Flavor</i>	77
Gambar 4.135 Menentukan Tipe Akses <i>Flavor</i>	78
Gambar 4.136 Tampilan <i>Dashboard Image</i>	78
Gambar 4.137 Tampilan Pembuatan <i>Image</i>	79

Gambar 4.138 Tampilan Pembuatan <i>Security Group</i>	79
Gambar 4.139 Pembuatan <i>Rule ICMP</i> pada <i>Security Group</i> ubuntu-sg.....	80
Gambar 4.140 Pembuatan <i>Rule SSH</i> pada <i>Security Group</i> ubuntu-sg	80
Gambar 4.141 Memilih <i>Availability Zone</i> dan <i>Flavor</i>	80
Gambar 4.142 Memilih <i>Image</i> untuk VM	81
Gambar 4.143 Menentukan Kapasitas <i>Disk</i> VM	81
Gambar 4.144 Memilih <i>Network</i> untuk VM yang akan Dibuat.....	81
Gambar 4.145 Memilih <i>Security Group</i> untuk VM.....	82
Gambar 4.146 Menentukan Nama dan <i>Password</i> VM	82
Gambar 4.147 Pembuatan <i>floating IP</i>	82
Gambar 4.148 Penambahan <i>Floating IP</i> pada VM	83
Gambar 4.149 VM Berhasil Dibuat	83
Gambar 4.150 Akses SSH Menuju VM Berhasil Dilakukan.....	83
Gambar 4.151 Pengecekan alamat IP dari VM.....	84
Gambar 4.152 Pengujian Koneksi dari VM Menuju <i>Internet</i>	84

DAFTAR TABEL

Tabel 3.1 Fungsi masing-masing <i>layer</i> pada model OSI	14
Tabel 3.2 Pembagian kelas alamat IP	17
Tabel 4.1 Spesifikasi Perangkat yang Digunakan.....	34

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam perkembangan teknologi informasi yang semakin canggih memunculkan rasa kemudahan dan kenyamanan dalam proses penggunaannya. Dalam suatu instansi yang khususnya bergerak dalam bidang teknologi informasi, perangkat keras komputer (*hardware*) akan menjadi suatu kendala dalam pengadaan suatu sistem maupun ketika dilakukan proses *upgrade*. Perusahaan pada umumnya membeli server baru ketika membutuhkan aplikasi baru untuk dijalankan. Lambat laun data center menjadi penuh dengan server yang hanya menggunakan sebagian kecil dari kapasitas total yang tersedia. Meskipun server itu berjalan hanya sebagian kecil dari kapasitas total, perusahaan tetap harus membayar listrik untuk menjalankan server tersebut untuk menghilangkan panas yang dihasilkan (Hidayat, 2016) (Nugraha, Mogi, & Setiawan, 2015).

Adapun cara dalam mengatasi kendala pada perangkat keras tersebut adalah dengan menerapkan teknologi *cloud computing*. Teknologi cloud computing merupakan model dimana sumber daya seperti *proccessor*, *network*, *storage*, dan *software* dijadikan sebagai layanan internet dalam wujud abstrak (Purbo, 2011). Perangkat keras tersebut disediakan dari layanan cloud computing yaitu IaaS (*Infrastructure as a Service*) sehingga tidak perlu menambah perangkat secara fisik, melainkan dalam bentuk *virtual*.

Sistem OpenStack merupakan salah satu *opensource software* yang dapat menyediakan solusi IaaS (*Infrastructure as a Service*) baik untuk *private cloud* maupun *public cloud*. Penulis memiliki alasan dalam penggunaan *software* OpenStack, dikarenakan OpenStack adalah *opensource project* untuk *platform cloud computing* yang dapat diterapkan secara sederhana dan dapat diperbesar kapasitasnya.

OpenStack dibangun untuk mengelola sumber daya di pusat data, khususnya alat komputasi, penyimpanan, dan jaringan. Apa yang dilakukan OpenStack adalah mengumpulkan sumber daya fisik ke dalam satu kumpulan pusat, dan kemudian mengalokasikan sumber daya virtual yang dibutuhkan,

menggunakan kumpulan fisik ini sebagai sumbernya. Meskipun ini terdengar seperti virtualisasi, OpenStack tidak melakukan virtualisasi itu sendiri. OpenStack menggunakan teknologi virtualisasi yang sudah ada sebelumnya. Arsitektur sistem OpenStack dirancang sebagai modular yang dimana untuk membangunnya terbagi dari beberapa layanan. Setiap layanan yang ditawarkan oleh OpenStack menyediakan API (*Application Programming Interface*) guna memfasilitasi integrasi lingkungan pengguna dengan *tools* dan fitur yang tersedia. Layanan-layanan tersebut dapat di-*install* sesuai dengan kebutuhan penggunanya.

1.2 Tujuan

Berdasarkan latar belakang masalah tersebut, penulis memiliki tujuan yaitu:

1. Implementasi penggunaan *private cloud* OpenStack dapat menghemat biaya perawatan perangkat server dan jaringan fisik dengan menggunakan layanan *virtual*.
2. Dengan membangun *private cloud* menggunakan OpenStack, dapat memaksimalkan penggunaan *resource* seperti RAM, CPU, maupun *storage* pada perangkat *server* fisik yang ada.

1.3 Pembatasan Masalah

Dalam laporan prakerin ini, penulis membatasi beberapa masalah yang akan dibahas. Adapun beberapa batasan masalah tersebut yaitu:

1. Proyek dibuat pada *virtual machine* (VM) yang berada di *server Research & Development* (R&D) milik PT. Indostorage Solusi Teknologi dengan sumber daya yang sudah disediakan.
2. *Private cloud* yang penulis bangun bersifat simulasi dan tidak terkait dengan kebutuhan suatu perusahaan.
3. Dalam pembuatan proyek, penulis menggunakan 2 *node*, yaitu *controller node* dan *compute node*.
4. Layanan yang dibangun pada OpenStack hanya layanan inti.
5. Sistem operasi yang digunakan yaitu Ubuntu 22.04.
6. Penulis menggunakan OpenStack versi Yoga.
7. *Dashboard service* dibangun di atas Docker versi 20.10.

8. *Image* yang digunakan untuk pembuatan *virtual machine* di OpenStack adalah Ubuntu 22.04.

1.4 Sistematika Penulisan

Adapun sistematika penulisan yang digunakan oleh penulis dalam menyusun laporan praktik kerja industri adalah sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini terdapat bahasan mengenai uraian laporan yang disusun yaitu latar belakang masalah, tujuan pembahasan, rumusan masalah serta sistematika penulisan laporan.

BAB II TINJAUAN PERUSAHAAN

Bab ini menjelaskan tentang profil umum PT. Indostorage Solusi Teknologi berupa informasi perusahaan, visi dan misi perusahaan, struktur organisasi perusahaan, serta layanan yang diberikan oleh PT. Indostorage Solusi Teknologi.

BAB III LANDASAN TEORI

Bab ini menjelaskan tentang landasan teori dan ilmu yang penulis gunakan sebagai acuan dalam pembahasan laporan.

BAB IV RANCANG BANGUN LAYANAN PRIVATE CLOUD MENGGUNAKAN OPENSTACK PADA UBUNTU 22.04

Bab ini menjelaskan tentang penjabaran dan langkah kerja dalam merancang dan membangun layanan *private cloud* menggunakan OpenStack, mulai dari topologi, analisis kebutuhan perangkat, tahapan instalasi dan konfigurasi layanan-layanan OpenStack, hingga pengujian.

BAB V PENUTUP

Pada bab ini penulis menuliskan kesimpulan mengenai laporan yang telah disusun serta saran yang berifat membangun yang berhubungan dengan masalah yang penulis bahas.

BAB II

TINJAUAN PERUSAHAAN

2.1 Profil Perusahaan



Gambar 2.1 Logo PT. Indostorage Solusi Teknologi

PT Indostorage Solusi Teknologi merupakan perusahaan yang bergerak di bidang *information technology* (IT).

Nama Instansi : PT Indostorage Solusi Teknologi

Website : www.indostorage.com

Email : info@indostorage.co.id

Telepon : +6221 3042 0653

Fax : +6221 7523 1563

2.2 Visi dan Misi Perusahaan

2.2.1 Visi

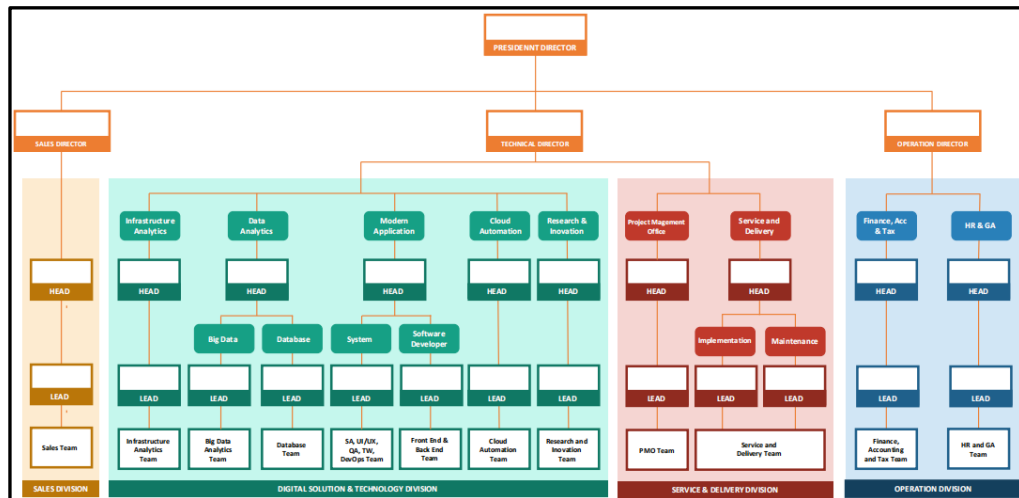
Menjadi perusahaan penyedia layanan teknologi informasi yang profesional dan kompetitif dengan memberikan layanan, solusi yang tepat terencana dan bernilai bagi pelanggan dan *stakeholders* serta mampu berkontribusi dalam memperkenalkan teknologi informasi kepada publik

2.2.2 Misi

1. Memberikan solusi inovatif dan optimal kepada klien yang berorientasi pada kepentingan pelanggan.
2. Berkomitmen untuk memberikan pelayanan terbaik sehingga dapat menjadi mitra bisnis yang terpercaya.
3. Berusaha untuk selalu meningkatkan kapabilitas dan mengoptimalkan pengelolaan sumber daya manusia yang unggul dan mandiri.

2.3 Struktur Organisasi Perusahaan

Berikut merupakan struktur organisasi di PT. Indostorage Solusi Teknologi.



Gambar 2.2 Struktur Organisasi PT. Indostorage Solusi Teknologi

2.4 Lini Bisnis

Bisnis utama PT. Indostorage Solusi Teknologi adalah konsultan jasa dan pengembangan produk IT, baik itu bagi lembaga pemerintahan maupun swasta. Indostorage Solusi Teknologi berpengalaman dalam bidang pengembangan *hardware*.

Dalam pengembangan *hardware* Indostorage Solusi Teknologi melakukan pendekatan teknikal desain dan perencanaan, penyesuaian desain dengan klien, penyuplai *hardware*, pemeliharaan dan manajemen proyek

2.5 Layanan dan Produk Perusahaan

2.5.1 Solusi Infrastruktur

Saat ini bisnis perlu beralih ke bidang teknologi informasi yang lebih cepat, dan untuk mempercepat perjalanan dalam perubahan dengan merencanakan dan mengadopsi teknologi baru.

Dalam penerapan solusi *data center*, solusi komunikasi atau virtualisasi, meningkatkan penyimpanan dan perlindungan data, atau memperbaharui *hardware* dan lisensi aplikasi, Indostorage dapat membantu melakukannya secara efisien, mengikuti standar dan praktik terbaik. Dengan memastikan keamanan, ketersediaan, dan integrasi yang lancar antara semua elemen.

Solusi Indostorage Solusi Teknologi terkait infrastruktur:

1. Storage

2. *Server*
3. *Networking*
4. *IT Security*
5. *IT Management*
6. *Enterprise Collaboration*

2.5.2 Proeksi Data

Indostorage menyediakan strategi perlindungan data terhadap kehilangan dan *corrupt data*, menggunakan produk perlindungan data, Indostorage menyediakan solusi ketersediaan data, replikasi, *snapshot*, pencadangan dan arsip untuk memenuhi tingkat layanan terbaik.

Mencakup semua lingkungan data, lintas lokasi, *cloud*, pribadi, *hybrid*, dan berbagai tingkatan seperti:

1. *Backup Disk ke Disk*
2. *Backup Disk ke Tape*
3. *Backup Disk ke Cloud*
4. *Backup Cloud ke Cloud*

2.5.3 Cloud and Virtualization

Indostorage memberikan konsep virtualisasi untuk memenuhi kebutuhan klien dengan skema standar dan menyesuaikan skenario berdasarkan kebutuhan pelanggan.

2.5.4 Microservices

Indostorage telah membantu perusahaan dengan pengembangan layanan *microservice* untuk memodernisasi sistem IT klien dengan aplikasi *microservice* yang bersifat *independent*, aman, dan cepat. Dengan layanan *microservice*, proyek dapat lebih cepat dan dengan sistem sederhana yang memungkinkan perbaikan cepat, pembaruan, dan pemutakhiran dengan *down time* minimal.

2.5.5 Big Data

Indostorage menyediakan layanan *big data* termasuk konsultasi, implementasi, dukungan untuk membantu klien dengan lingkungan *big data*. Layanan big data membantu perusahaan memaksimalkan nilai dan mencapai tujuan dengan analisis *big data*.

2.5.6 Data Integration

Konsep integrasi data yang menggabungkan data dari berbagai sumber menjadi satu kesatuan tampilan. Indostorage memberikan layanan untuk mengintegrasikan sistem dengan data untuk berbagai keperluan.

2.5.7 DevOps

Indostorage menyediakan layanan DevOps. Menawarkan solusi yang disesuaikan dengan kebutuhan klien. Keuntungan inti dari layanan DevOps adalah waktu *up* yang lebih cepat, kualitas produk yang lebih tinggi, peningkatan efisiensi dan produktivitas.

BAB III

LANDASAN TEORI

3.1 Jaringan Komputer

Jaringan komputer adalah jaringan telekomunikasi yang memungkinkan antar komputer untuk saling berkomunikasi dengan bertukar data. Tujuan dari jaringan komputer adalah agar dapat mencapai tujuannya, setiap bagian dari jaringan komputer dapat meminta dan memberikan layanan. Dua buah komputer yang masing-masing memiliki sebuah kartu jaringan, kemudian dihubungkan melalui kabel maupun nirkabel. Apabila ingin membuat jaringan komputer yang lebih luas lagi jangkauannya, maka diperlukan peralatan tambahan seperti *switch*, *router*, dan lain lain (Astuti, 2020).

Menurut (Pamungkas, Setiawan, & Ramadhani, 2018), terdapat beberapa jenis-jenis jaringan komputer berdasarkan kriterianya sebagai berikut:

3.1.1 Jaringan Komputer Berdasarkan Jangkauan Geografis

a. *Local Area Network* (LAN)

Jaringan LAN adalah jaringan yang menghubungkan beberapa komputer dalam satu area lokal. Misalnya dalam satu gedung, satu rumah, perkantoran, perindustrian, universitas, dan daerah yang sejenis. Kecepatan transmisi data LAN mencapai 100 Mbps dengan cakupan antara 10m sampai 5km. Keunggulan dari LAN antara lain akses data antar komputer berlangsung cepat dan mudah, dapat menghubungkan banyak komputer, serta *backup* data lebih mudah dan cepat.

b. *Metropolitan Area Network* (MAN)

Jaringan MAN adalah jaringan yang menghubungkan beberapa jaringan komputer dalam wilayah yang lebih luas atau kota besar. Misalnya jaringan antar kampus dalam satu universitas, jaringan dalam satu kota, korporasi *Internet Service Provider* (ISP). MAN merupakan jaringan dengan kecepatan tinggi yang memungkinkan *sharing* data pada area yang luas, dengan area cakupannya sekitar antara 5 km sampai 50 km (Astuti, 2020).

c. *Wide Area Network* (WAN)

Jaringan WAN adalah jaringan yang menghubungkan beberapa MAN dari beberapa kota atau negara yang berbeda. WAN biasanya terhubung melalui kabel

optik atau satelit. Jaringan WAN telah banyak dibangun dan digunakan secara publik, misalnya perbankan, jaringan perdagangan *online*, jaringan korporasi yang besar, dan jaringan militer.

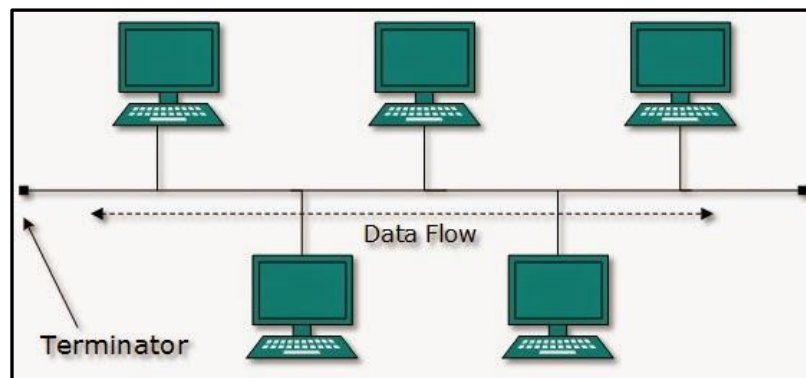
3.1.2 Jaringan Komputer Berdasarkan Topologi

Topologi jaringan komputer adalah suatu cara menghubungkan komputer yang satu dengan komputer lainnya sehingga membentuk jaringan. Dalam suatu jaringan komputer, jenis topologi yang dipilih akan memengaruhi kecepatan komunikasi (Supriyadi & Gartina, 2007).

Topologi jaringan dapat dikategorikan dalam tipe dasar berikut, yakni:

a. Topologi *Bus*

Topologi *bus* menghubungkan komputer stau dengan yang lain secara berantai dengan perantara satu kabel tunggal. Biasanya kabel yang digunakan adalah kabel *coaxial*. Kelebihan topologi ini adalah pengembangan jaringan atau penambahan workstation baru dapat dilakukan dengan mudah tanpa mengganggu workstation lain.



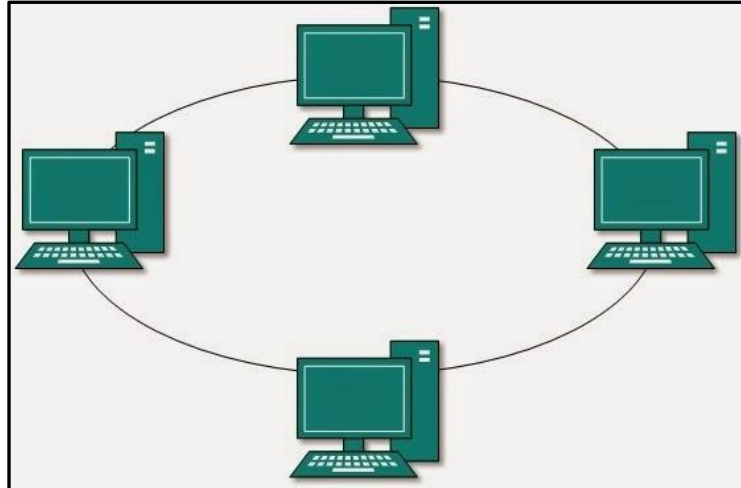
Gambar 3.1 Topologi *Bus*

Sumber: <https://www.nesabamedia.com/topologi-jaringan-komputer/>

b. Topologi *Ring*

Topologi *ring* adalah topologi jaringan yang rangkaiannya membentuk cincin dan berupa titik yang mana masing-masing titik bagian kanan dan kiri terhubung ke dua titik lainnya sampai komputer pertama dan komputer terakhir terhubung. Titik yang ada pada topologi cincin ini berfungsi memperkuat sinyal di setiap rangkaiannya atau bisa juga disebut *repeater*. Dengan metode ini sinyal dan

aliran data akan tetap stabil, serta arah aliran datanya bisa searah jarum jam atau sebaliknya.

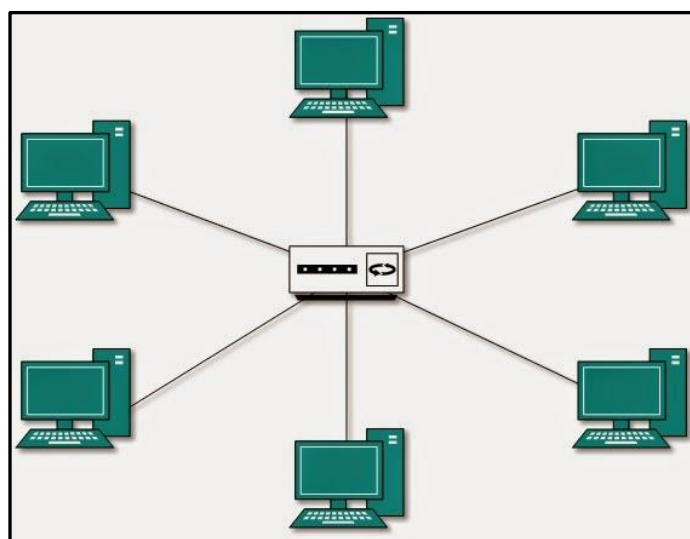


Gambar 3.2 Topologi *Ring*

Sumber: <https://www.nesabamedia.com/topologi-jaringan-komputer/>

c. Topologi *Star*

Topologi *star* mempunyai satu penghubung sebagai pusat (*hub* atau *switch*) dari setiap komputer yang terhubung. *Hub* atau *switch* tersebut posisinya di tengah dan berfungsi untuk menghubungkan komputer ke setiap komputer yang terhubung. Topologi memiliki kelebihan dimana jika ada salah satu komputer putus maka tidak akan mengganggu kinerja jaringan.

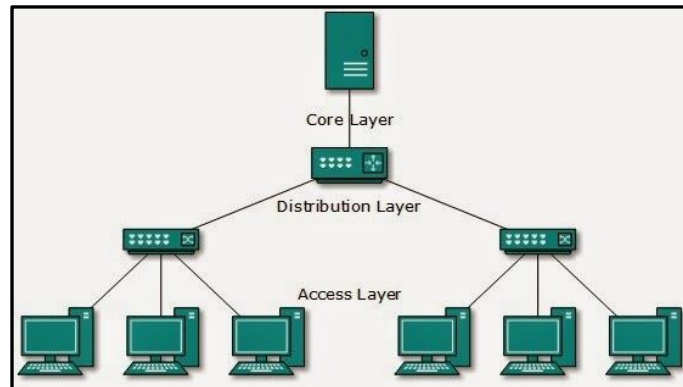


Gambar 3.3 Topologi *Star*

Sumber: <https://www.nesabamedia.com/topologi-jaringan-komputer/>

d. Topologi *Tree*

Topologi jaringan *tree* merupakan kombinasi antara topologi jaringan *star* dan *bus*. Topologi ini terdiri atas beberapa topologi *star* yang dihubungkan dalam satu topologi *bus* sebagai *backbone*. Komputer-komputer dihubungkan ke *hub*, sedangkan *hub* lain dihubungkan sebagai jalur yang mempunyai topologi *bus*.

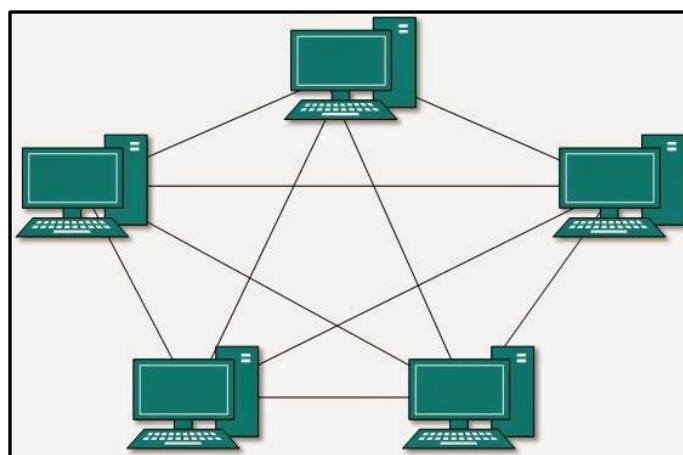


Gambar 3.4 Topologi *Tree*

Sumber: <https://www.nesabamedia.com/topologi-jaringan-komputer/>

e. Topologi *Mesh*

Topologi jaringan ini mempunyai jalur ganda dari setiap perangkat pada jaringan. Jaringan ini rumit dan boros. Semakin banyak komputer yang ada pada jaringan maka semakin rumit dan semakin banyak jumlah kabel yang digunakan. Topologi ini biasanya digunakan untuk jaringan antar *router*.



Gambar 3.5 Topologi *Mesh*

Sumber: <https://www.nesabamedia.com/topologi-jaringan-komputer/>

3.1.3 Jaringan Komputer Berdasarkan Media Transmisi Data

Terdapat dua jenis jaringan komputer berdasarkan media transmisi data, yaitu:

a. Jaringan Berkabel (*Wired Network*)

Jaringan yang menggunakan kabel dalam menghubungkan satu komputer dengan komputer yang lain. Kabel berfungsi sebagai media untuk mengirim data dalam bentuk sinyal listrik atau sinyal cahaya antar komputer jaringan. Kabel yang biasa digunakan antara lain UTP/STP, *fiber optic*, dan *coaxial*.

b. Jaringan Tanpa Kabel (*Wireless Network*)

Jaringan komputer yang menggunakan gelombang elektromagnetik sebagai media untuk menghubungkan antar komputer, gelombang tersebut berfungsi untuk mengirimkan data atau informasi. Jenis-jenis gelombang elektromagnetik yang biasa digunakan yaitu *micro wave* (seluler dan satelit), infra merah, bluetooth, dan wifi.

3.1.4 Jaringan Komputer Berdasarkan Hubungan Tiap Komputer

Terdapat dua jenis jaringan komputer berdasarkan hubungan tiap komputer, yaitu:

a. Jaringan *Client-Server*

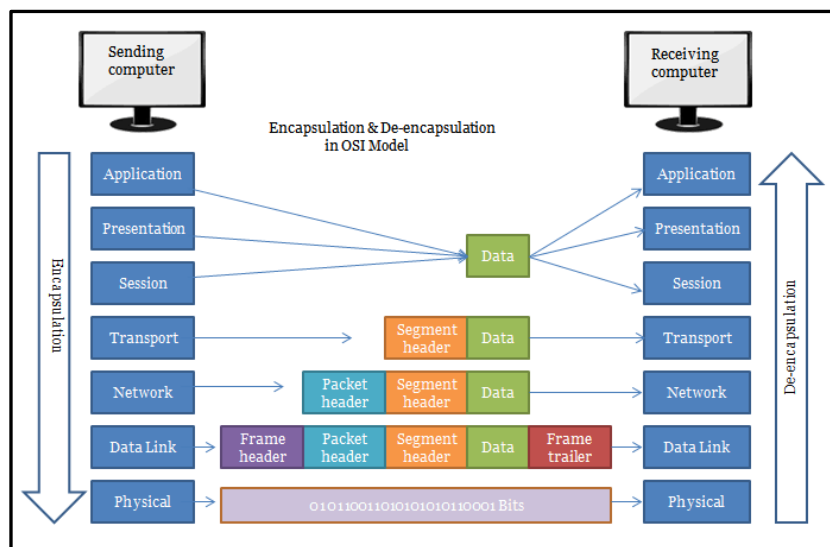
Jaringan *client-server* adalah jaringan komputer yang salah satu komputer dalam satu jaringan digunakan sebagai *server*. Dalam implementasi pada jaringan *client-server*, komputer yang menjadi *server* atau pun *client* dapat berubah-ubah yang diatur menggunakan perangkat lunak pada protokolnya. Komputer *client* berfungsi sebagai perantara pengguna untuk mengakses data pada *server*. Sedangkan komputer *server* menyediakan data atau informasi yang diperlukan oleh komputer *client*.

b. Jaringan *Peer to Peer*

Jaringan *peer to peer* adalah jaringan komputer yang dimana setiap komputer dapat melakukan pengiriman ataupun penerimaan data sehingga semua komputer bisa menjadi *server* sekaligus *client*.

3.2 Model Referensi OSI

Model referensi OSI (*Open System Interconnection*) menggambarkan bagaimana informasi dari suatu aplikasi di sebuah komputer berpindah melewati sebuah media jaringan ke suatu aplikasi di komputer lain. Model referensi OSI terbagi menjadi 7 lapisan dimana masing-masing lapisan memiliki fungsi jaringan yang spesifik. 7 lapisan tersebut yaitu *application layer*, *presentation layer*, *session layer*, *transport layer*, *network layer*, *data link layer*, dan *physical layer*. Model ini diciptakan berdasarkan proposal yang dibuat oleh *The International Standards Organization* (ISO) sebagai langkah awal menuju standarisasi protokol internasional yang digunakan pada berbagai layer. *Open System* dapat diartikan sebagai suatu sistem yang terbuka untuk berkomunikasi dengan sistem-sistem lainnya (Yusnika, 2013).



Gambar 3.6 Layer Model Referensi OSI

Sumber: <https://www.computernetworkingnotes.com/ccna-study-guide/data-encapsulation-and-de-encapsulation-explained.html>

Berdasarkan pada Gambar 3.6, ketika data ditransfer melalui jaringan, sebelumnya data tersebut harus melewati ke-tujuh layer, mulai dari layer aplikasi sampai layer *physical*, pada saat data melewati satu layer dari sisi pengirim, maka akan ditambahkan satu *header* atau dikenal dengan enkapsulasi. Kemudian di sisi penerima, data tersebut melewati layer *physical* sampai aplikasi dan pada saat data melewati satu *layer*, *header* dicopot sesuai dengan layer-nya atau disebut dengan

dekapsulasi (Saputro, 2014). Pada Tabel 3.1, menunjukkan nama dan fungsi dari *layer* OSI.

Tabel 3.1 Fungsi masing-masing *layer* pada model OSI

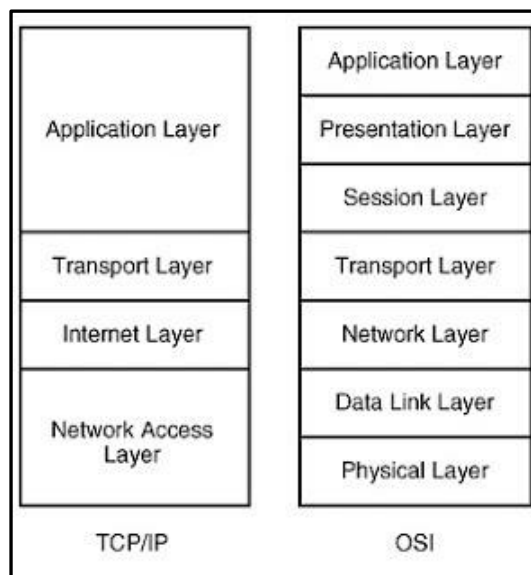
No	Nama <i>Layer</i>	Fungsi <i>Layer</i>
1	<i>Application</i>	<i>Application layer</i> berfungsi sebagai penghubung interaksi antara <i>end user</i> dengan aplikasi yang bekerja menggunakan fungsionalitas jaringan, menggunakan pengaturan bagaimana aplikasi bekerja menggunakan resource jaringan, untuk kemudian memberikan pesan ketika terjadi kesalahan. Beberapa contoh protokol yang terdapat pada <i>application layer</i> yaitu HTTP, FTP, dan DNS.
2	<i>Presentation</i>	<i>Presentation layer</i> berfungsi untuk melakukan translasi data yang hendak ditransmisikan oleh aplikasi melalui jaringan ke format yang dapat ditransmisikan oleh jaringan tersebut. Protokol yang berada dalam level ini adalah perangkat lunak redirektor, seperti <i>network shell</i> , <i>Virtual Network Computing</i> atau VNC, atau <i>Remote Desktop Protocol</i> (RDP).
3	<i>Session</i>	<i>Session layer</i> berfungsi untuk mendefinisikan bagaimana koneksi antar dapat dibuat, dipelihara, atau dihancurkan. Di <i>layer</i> ini terdapat protokol-protokol yang bekerja, seperti NETBIOS yang dikembangkan oleh IBM, PPTP dan RPC.
4	<i>Transport</i>	<i>Transport layer</i> berfungsi melakukan pemecahan data ke dalam paket-paket data serta memberikan nomor urut pada paket-paket data tersebut sehingga dapat disusun kembali ketika sudah sampai pada sisi tujuan. Pada <i>layer</i> ini akan ditentukan protokol yang

		akan digunakan untuk mentransmisi data, misalkan protokol TCP. Protokol ini mengirimkan paket data sekaligus memastikan bahwa paket diterima dengan sukses (<i>acknowledgement</i>).
5	<i>Network</i>	<i>Network layer</i> akan membuat header untuk paket-paket yang berisi informasi IP, baik IP pengirim data maupun IP tujuan data. Pada kondisi tertentu, <i>layer</i> ini juga akan melakukan routing melalui internetworking dengan menggunakan <i>router</i> dan <i>switch layer 3</i> .
6	<i>Data-Link</i>	<i>Data-link layer</i> berfungsi untuk menentukan bagaimana bit-bit data dikelompokkan menjadi format yang disebut <i>frame</i> . Selain itu, pada <i>layer</i> ini berfungsi mendefinisikan jaringan, alamat <i>hardware</i> (MAC <i>address</i>), karakteristik protokol, pendeteksi kesalahan, <i>flow control</i> , dan menentukan bagaimana perangkat-perangkat jaringan seperti <i>hub</i> , <i>bridge</i> , <i>switch</i> beroperasi. Spesifikasi IEEE 802 membagi. Spesifikasi IEEE 802 membagi <i>layer</i> ini menjadi dua <i>level</i> , yakni lapisan LLC (<i>Logical Link Control</i>) dan MAC (<i>Media Access Control</i>).
7	<i>Physical</i>	<i>Physical layer</i> bekerja dengan mendefinisikan media transmisi jaringan, metode pensinyalan, sinkronasi bit, arsitektur jaringan, topologi jaringan dan pengabelan. Selain itu, <i>layer</i> ini juga mendefinisikan bagaimana <i>Network Interface Card</i> (NIC) dapat berinteraksi dengan media kabel atau nirkabel.

3.3 Model TCP/IP

TCP/IP atau *Transmission Control Protocol/Internet Protocol* merupakan rangkaian protokol komunikasi yang digunakan untuk menghubungkan perangkat

jaringan di internet. TCP/IP juga digunakan sebagai protokol komunikasi dalam jaringan komputer pribadi. Model TCP/IP dirancang dan digunakan oleh Departemen Pertahanan AS pada 1960an dan didasarkan pada protokol standar. Model ini adalah versi ringkas dari model OSI karena model TCP/IP memiliki empat lapisan atau *layer*, yaitu *application layer*, *transport layer*, *internet layer*, dan *network access layer*.



Gambar 3.7 Perbandingan *Layer Model TCP/IP* dan *Layer Model OSI*

Sumber: <http://nguprek.com/perbandingan-model-osi-layer-dan-tcp-ip/>

Pada Gambar 3.7, *network access layer* memiliki fungsi yang identik dengan *physical layer* dan *data-link layer* OSI. *Internet layer* memiliki fungsi yang identik dengan *network layer* pada OSI. *Transport layer* memiliki fungsi yang identik dengan *transport layer* OSI. *Application layer* mendefinisikan aplikasi-aplikasi yang dijalankan pada jaringan yang berinteraksi langsung dengan pengguna (Pamungkas, Setiawan, & Ramadhani, 2018).

Dua protokol utama, TCP dan IP melayani fungsi tertentu. TCP mendefinisikan bagaimana aplikasi dapat membuat saluran komunikasi di seluruh jaringan. Selani itu, juga mengatur bagaimana pesan dirakit menjadi paket-paket yang lebih kecil sebelum kemudian dikirim melalui internet dan disusun kembali dalam urutan yang benar di alamat tujuan. IP mendefinisikan cara menangani dan

merutekan setiap paket untuk memastikan paket tersebut mencapai tujuan yang benar (Trivusi, 2022).

3.4 IP Address

IP *address* adalah serangkaian nomor yang dimiliki perangkat dimana nomor tersebut digunakan untuk menghubungkan suatu perangkat dengan perangkat lainnya melalui jaringan, IP *address* disusun dengan kisaran antara 32bit (untuk IPv4) sampai 128 bit (untuk IPv6). IP *address* memiliki 2 fungsi khusus, yakni sebagai alat identifikasi *host* dimana tidak boleh ada yang memiliki IP *address* yang sama dan sebagai alamat lokasi jaringan pada setiap komputer supaya data dapat sampai pada komputer yang tepat (Riyadi, 2022).

3.4.1 IPv4

IPv4 adalah kependekan dari IP versi 4. IP versi ini merupakan alamat IP yang paling umum digunakan. IPv4 menggunakan alamat 32bit yang ditulis dalam empat segmen angka yang dipisahkan oleh tanda titik. IPv4 terbagi menjadi beberapa kelas, seperti pada Tabel 3.2.

Tabel 3.2 Pembagian kelas alamat IP

Kelas	Oktet Pertama (Desimal)	Penggunaan
A	1-126	Jaringan komputer skala besar
B	128-191	Jaringan komputer skala menengah sampai besar
C	192-223	Jaringan komputer skala kecil
D	224-239	Alamat <i>multicast</i>
E	240-255	Digunakan sebagai percobaan atau eksperimen

3.5 Internet

Internet (*Interconnection Networking*) merupakan jaringan yang dapat menghubungkan satu media elektronik dengan elektronik lainnya. Internet melibatkan berbagai jenis komputer serta topologi jaringan yang berbeda. Standar teknologi pendukung yang dipakai secara global adalah TCP/IP. Internet pertama kali muncul pada tahun 1969 dengan bentuk sebuah jaringan komputer yang dibuat

oleh Departemen Pertahanan AS dan diberi nama ARPANET yang awalnya digunakan untuk riset dan keperluan militer (Darmawan, 2012).

3.6 World Wide Web (WWW)

World Wide Web (WWW) adalah sebuah sistem untuk membuat, mengatur, dan menghubungkan dokumen sehingga dokument tersebut dapat mudah diakses. Secara umum, WWW berfungsi sebagai penyedia data dan informasi yang diperlukan oleh pengguna internet. Untuk mengakses dokumen tersebut, pengguna memerlukan sebuah perangkat lunak yang bernama *web browser*.

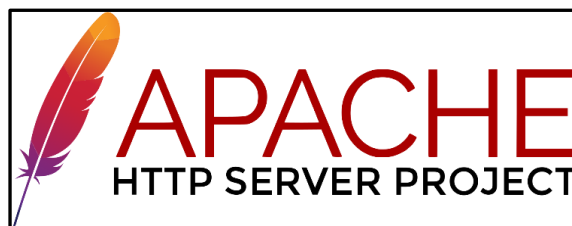
3.7 Web Server

Web server adalah sebuah perangkat lunak yang berfungsi untuk memberikan layanan berupa data, dan berfungsi menerima permintaan HTTP atau HTTPS dari *client* yang menggunakan *web browser*, nantinya *web server* merespon permintaan tersebut dengan menampilkan halaman *website* yang diminta (Prayoga, 2021).

3.7.1 Apache HTTP Server

Apache adalah salah satu jenis *web server* yang dapat dijalankan di berbagai sistem operasi yang digunakan untuk melayani dan melakukan pengaturan fasilitas web menggunakan protokol yang dikenal dengan HTTP (*Hypertext Transfer Protocol*). Nama Apache sendiri dipilih sebagai penghormatan terhadap suku Indian Apache yang menggunakan keterampilan dan strategis yang luar biasa dalam peperangan.

Pada awalnya, Apache merupakan perangkat lunak *opensource* yang hanya digunakan sebagai alternatif *web server* Netscape. Namun sejak April 1996, Apache menjadi *web server* yang populer. Hingga pada Mei 1999, Apache mulai banyak digunakan di berbagai *web server* dunia (Syafitri, 2022).



Gambar 3.8 Logo Apache HTTP Server

Sumber: https://id.wikipedia.org/wiki/Apache_HTTP_Server

3.8 Application Programming Interface (API)

Application Programming Interface (API) merupakan sebuah antarmuka yang berfungsi sebagai perantara bagi beberapa aplikasi atau klien dan server baik pada satu platform yang sama atau lintas platform agar bisa saling berkomunikasi. API menciptakan integrasi agar fitur-fitur diantara dua aplikasi bisa saling terkoneksi. Dengan menggunakan API, akan lebih mudah untuk membuat aplikasi yang fungsional dan kompleks, aplikasi tersebut tidak perlu menambahkan data secara manual, melakukan komunikasi langsung dengan aplikasi lain, ataupun menyimpan data yang dibutuhkan di server sendiri, cukup meminta API untuk melakukan semua fungsi tersebut (Lawrence, 2020).

3.9 Sistem Operasi

Sistem operasi atau sering disebut *operating system* (OS) adalah seperangkat program yang mengelola sumber daya *hardware* komputer, dan menyediakan layanan umum untuk aplikasi *software*. Secara umum, *operating system* merupakan *software* pada lapisan pertama yang ditempatkan pada memori komputer pada saat komputer dinyalakan/ *booting*. Sedangkan *software-software* lainnya dijalankan setelah sistem operasi berjalan, dan sistem operasi akan melakukan layanan inti untuk *software-software* itu (Fadhilah, 2013).

3.9.1 Ubuntu

Ubuntu adalah open source, linux *distribution* yang gratis. Ubuntu merupakan sistem operasi untuk komputasi awan, sesuai dengan dukungan OpenStack. Ubuntu dikembangkan oleh Canonical *Community* dan tersedia secara bebas. Juga, Canonical Ltd. bertanggung jawab atas pendanaan Ubuntu. Pada dasarnya, Ubuntu dirilis setiap enam bulan sekali. Dukungan gratis tersedia selama sembilan bulan untuk setiap rilis dan dukungan jangka panjang (LTS) yang dirilis setiap dua tahun. Rilis pertama Ubuntu pada Oktober 2004. Rilis terbaru adalah 23.04, juga dikenal sebagai, *Lunar Lobster*. Tiga edisi Ubuntu adalah *Desktop Edition*, *Server Edition*, dan *Core Edition*. Nama Ubuntu diberikan setelah filsuf Afrika, yang diterjemahkan menjadi "*Humanity to Others*". Arsitektur Ubuntu didasarkan pada server Debian dan Linux.



Gambar 3.9 Logo Ubuntu

Sumber: <https://design.ubuntu.com/brand>

3.10 Virtualisasi

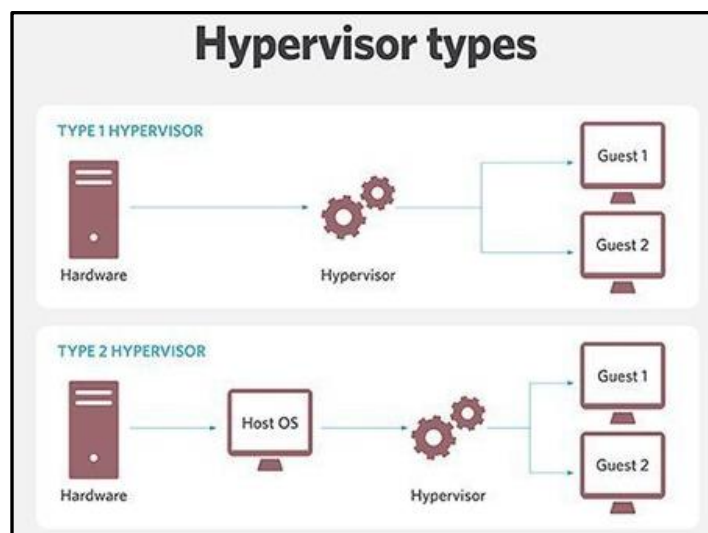
Virtualisasi merupakan pembagian *server* fisik menjadi beberapa *virtual server* yang lebih kecil dengan tujuan untuk mengoptimalkan penggunaan *resource* *server* fisik. Dalam virtualisasi *server*, *resource* dari *server* fisik disembunyikan dari pengguna *virtual server*, dan hanya admin yang bisa melihat *resource* asli dari *server* fisik. Virtualisasi *server* menggunakan *Hypervisor* yang digunakan untuk membagi *resource* fisik ke dalam banyak *virtual environment* atau sering disebut *virtual private server* (VPS), *guests*, *instance*, *container*, atau *emulation*. (Arianto, 2022).

3.11 Hypervisor

Hypervisor adalah sebuah teknik virtualisasi yang memungkinkan beberapa *operating system* untuk berjalan bersamaan pada sebuah *host*. Dikatakan teknik virtualisasi karena OS yang ada bukanlah sebuah OS yang sesungguhnya, hanya sebuah *virtual machine* saja. Tugas dari *hypervisor* adalah untuk mengatur setiap *operating system* tersebut sesuai dengan gilirannya agar tidak mengganggu satu dengan yang lainnya. Terkadang, *hypervisor* juga disebut sebagai *virtual machine management* (VMM), sesuai dengan tugasnya dalam mengatur beberapa *virtual machine* (Efendi, 2016).

Hypervisor dapat diklasifikasikan menjadi dua jenis yaitu *hypervisor* tipe 1 (*bare-metal*) dan *hypervisor* tipe 2 (*hosted*). *Hypervisor* tipe 1 berjalan langsung pada perangkat keras fisik dan mengontrol sumber daya fisik, sehingga mesin *virtual* berjalan pada lingkungan yang terisolasi. Contoh dari *hypervisor* tipe 1 adalah VMWare ESXi, Microsoft Hyper-V, Xen, dan KVM. Sedangkan *hypervisor* tipe 2 berjalan pada sistem operasi *host* dan memanfaatkan sumber daya perangkat

keras melalui sistem operasi *host*. *Hypervisor* tipe 2 digunakan untuk pengembangan dan pengujian perangkat lunak atau aplikasi yang berbeda pada mesin *virtual*. Contoh dari *hypervisor* tipe 2 adalah QEMU, Oracle VirtualBox dan VMWare Workstation (Gillis & Posey, 2021).



Gambar 3.10 Tipe *Hypervisor*

Sumber: <https://www.techtarget.com/searchitoperations/definition/hypervisor>

3.12 Cloud Computing

Cloud computing adalah sebuah model *client-server*, dimana sumber daya yang dapat diakses oleh penggunaanya setiap saat secara remote dan inti dari teknologi ini adalah virtualisasi. Pada *cloud computing*, mesin fisik digantikan dengan mesin *virtual*, *network* dan *storage* fisik pun digantikan dengan yang *virtual*.

National Institute of Standards and Technology (NIST) mendefinisikan *cloud computing* sebagai sebuah model untuk kenyamanan, akses jaringan *on-demand* untuk pengaturan konfigurasi sumber daya komputasi (seperti jaringan, *server*, media penyimpanan, aplikasi dan layanan) yang dapat dengan cepat ditetapkan dan dirilis dengan usaha manajemen yang minimal atau interaksi dengan penyedia layanan (Ashari & Setiawan, 2011).

3.12.1 Karakteristik *Cloud Computing*

Terdapat lima karakteristik penting dari *cloud computing* (Ashari & Setiawan, 2011), yaitu:

- a. *On-demand self-service*. Konsumen dapat menentukan kemampuan komputasi secara sepihak, seperti *server time* dan *storage*, secara otomatis sesuai kebutuhan tanpa memerlukan interaksi manusia dengan masing-masing penyedia layanan.
- b. *Broad network access*. Kemampuan yang tersedia melalui jaringan dan diakses melalui mekanisme standar yang mengenalkan penggunaan berbagai *platform* (misal telepon selular, laptop).
- c. *Resource pooling*. Penyatuan sumber daya komputasi yang dimiliki penyedia untuk melayani beberapa konsumen virtual yang berbeda, ditetapkan secara dinamis dan ditugaskan sesuai dengan permintaan konsumen. Contoh sumber daya termasuk penyimpanan, pemrosesan, memori, bandwidth jaringan, dan mesin virtual.
- d. *Measured service*. Penyedia layanan menyediakan layanan untuk mengoptimasi dan memonitor layanan yang dipakai. Dengan layanan ini, pengguna dapat melihat berapa *resource* yang telah terpakai. Layanan ini sebagai bentuk transparansi antar *provider* dan *customer*.

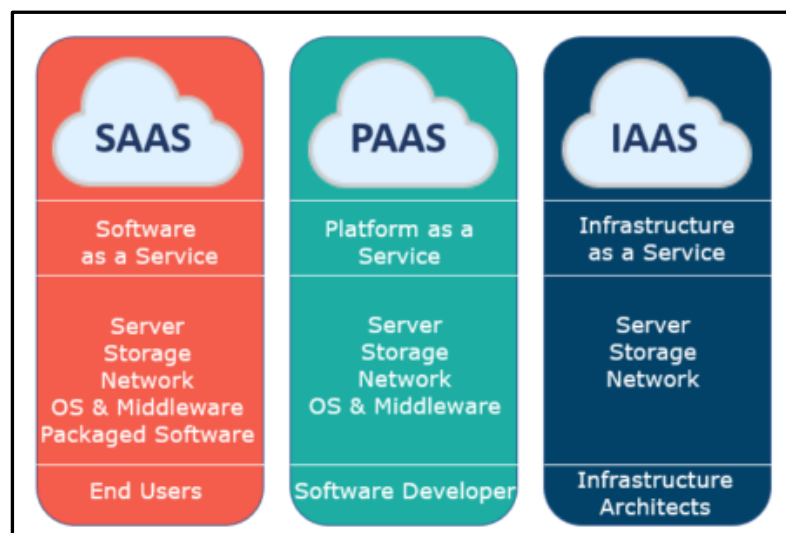
3.12.2 Model Layanan Cloud Computing

Terdapat tiga model layanan *cloud computing* (Ashari & Setiawan, 2011), yaitu:

- a. *Software as a Service* (SaaS). Kemampuan yang diberikan kepada konsumen untuk menggunakan aplikasi penyedia dapat beroperasi pada infrastruktur *cloud*. Aplikasi penyedia dapat diakses melalui perangkat klien dengan antarmuka seperti *web browser*. Konsumen tidak mengelola atau mengendalikan infrastruktur cloud yang mendasar termasuk jaringan, *server*, sistem operasi, penyimpanan, atau bahkan kemampuan aplikasi. Contoh dari model layanan ini adalah Google Apps, Facebook, Twitter, dan lain-lain.
- b. *Platform as a Service* (PaaS). Model ini memberikan kemampuan kepada konsumen untuk menyebarkan aplikasi yang dibuat oleh konsumen ke infrastruktur *cloud computing* menggunakan bahasa pemrograman dan peralatan yang didukung oleh penyedia. Pada model layanan ini, konsumen tidak mengelola infrastruktur *cloud* seperti jaringan, *server*, sistem operasi, atau

penyimpanan, namun masih memiliki kontrol atas aplikasi yang di-*deploy* dan melakukan konfigurasi *environment* aplikasinya. Contoh dari model layanan ini adalah Heroku, Kubernetes, Openshift, dan lain-lain.

- c. *Infrastructure as a Service* (IaaS). Model ini memberikan kemampuan kepada konsumen untuk memproses, menyimpan, dan sumber komputasi penting yang lainnya, dimana konsumen dapat menyebarkan dan menjalankan perangkat lunak secara bebas, yang dapat mencakup sistem operasi. Konsumen tidak mengelola atau mengontrol infrastruktur *cloud* yang mendasarinya tetapi hanya memiliki kontrol terbatas dari komponen jaringan tertentu seperti *firewall*. Contoh dari model layanan ini adalah AWS, Microsoft Azure, OpenStack, dan lain-lain.



Gambar 3.11 Model Layanan Cloud Computing

Sumber: <https://www.edureka.co/blog/cloud-computing-services-types/>

3.12.3 Model Penyebaran *Cloud Computing*

Menurut (Mell & Grance, 2011), Terdapat 4 model penyebaran *cloud computing*, yaitu:

a. *Private Cloud*

Pada model ini, infrastruktur cloud disediakan untuk penggunaan eksklusif oleh suatu organisasi yang memiliki konsumen. Organisasi ini bisa berupa perusahaan ataupun yang lainnya. Model ini bisa dimiliki, dikelola, dan dioperasikan oleh organisasi, pihak ketiga, ataupun keduanya.

b. *Public Cloud*

Kemudian ada *public cloud*, model ini disediakan secara terbuka untuk masyarakat umum. model ini dikelola oleh penyedia layanan *public cloud* yang memiliki kepemilikan penuh atas *public cloud* dengan kebijakan, dan keuntungannya sendiri. Beberapa contoh layanan *public cloud* yang populer adalah AWS dari Amazon, Google Cloud dari Google, dan lain-lain.

c. *Community Cloud*

Pada model ini, infrastruktur cloud disediakan untuk penggunaan eksklusif oleh suatu komunitas konsumen dari sebuah organisasi yang memiliki kepentingan dan tujuan yang sama. Model penyebaran ini bisa dimiliki, dikelola, dan dioperasikan oleh komunitas, pihak ketiga, ataupun keduanya. Bisa dibilang pada model ini, beberapa organisasi secara bersama-sama membangun dan berbagi infrastruktur cloud.

d. *Hybrid Cloud*

Hybrid cloud merupakan gabungan dari layanan *public cloud* dan *private cloud* yang diimplementasikan oleh suatu perusahaan. Dalam *hybrid cloud* ini, perusahaan dapat memilih proses bisnis mana yang ingin dipindahkan ke *public cloud* dan proses bisnis mana yang harus tetap berjalan di *private cloud*.

3.13 *Secure Shell (SSH)*

Secure shell (SSH) adalah protokol transfer yang memungkinkan penggunaannya untuk mengontrol sebuah perangkat secara *remote* atau dari jarak jauh melalui koneksi internet ataupun lokal. SSH menggunakan teknologi enkripsi yang menjamin keamanan koneksi yang digunakan selama proses transfer dan menjamin semua data terlibat dalam transfer terenkripsi.

3.14 *Windows Terminal*

Windows Terminal adalah sebuah aplikasi terminal modern yang dikembangkan oleh *Microsoft* untuk *Windows 10* dan *Windows 11*. *Windows Terminal* menyediakan antarmuka pengguna yang dapat dikonfigurasi secara fleksibel dan memungkinkan pengguna untuk mengakses beberapa jenis konsol dan lingkungan *shell* dalam satu jendela, termasuk *Command Prompt*, *PowerShell*, dan *Windows Subsystem for Linux (WSL)*.

3.15 *Network Time Protocol (NTP)*

Network Time Protocol (NTP) membantu waktu jam komputer untuk disinkronkan dalam jaringan. Protokol ini adalah protokol aplikasi yang bertanggung jawab untuk sinkronisasi host pada jaringan TCP/IP. NTP dikembangkan oleh David Mills pada tahun 1981 di University of Delaware. Ini diperlukan dalam mekanisme komunikasi sehingga koneksi tanpa batas hadir di antara komputer. Proses sinkronisasi ini dilakukan didalam jalur komunikasi data yang biasanya menggunakan protokol komunikasi TCP/IP (Oluwademilade, 2022).

3.16 *Message Broker*

Menurut (IBM, n.d.), *message broker* adalah perangkat lunak yang bertugas untuk mengirim dan menerima pesan di antara aplikasi atau sistem yang berbeda, baik secara internal maupun eksternal. Dengan menggunakan *message broker*, komunikasi antar aplikasi menjadi lebih efisien dan dapat diatur dengan lebih baik. Salah satu contoh perangkat lunak *message broker* yang paling umum digunakan adalah RabbitMQ.

RabbitMQ adalah *message broker* yang memberikan aplikasi sebuah *platform* umum untuk mengirim dan menerima pesan, dan pesan-pesan tersebut disimpan dengan aman sampai pesan tersebut berhasil diterima. RabbitMQ mendukung berbagai protokol *messaging* seperti AMQP, STOMP, dan MQTT. Contoh penggunaan RabbitMQ adalah pada sistem yang memerlukan sinkronisasi data antar server, misalnya pada sistem *multi-server*. Dalam sistem tersebut, RabbitMQ dapat digunakan untuk mengirim pesan dari satu *server* ke *server* lainnya secara *asynchronous* (RabbitMQ, n.d.).

3.17 *MariaDB*

MariaDB adalah sistem manajemen basis data relasional (RDBMS) sumber terbuka yang merupakan cabang dari MySQL. MariaDB dikembangkan oleh pengembang MySQL yang mengkhawatirkan akuisisi MySQL oleh Oracle Corporation. MariaDB mengubah data menjadi informasi terstruktur dalam beragam aplikasi, mulai dari perbankan hingga *website*. Awalnya dirancang sebagai pengganti drop-in yang disempurnakan untuk MySQL, MariaDB digunakan karena cepat, dapat diskalakan, dan kuat, dengan ekosistem mesin penyimpanan yang

kaya, *plugin*, dan banyak alat lainnya membuatnya sangat serbaguna untuk berbagai kasus penggunaan (MariaDB, n.d.).



Gambar 3.12 Logo MariaDB

Sumber: <https://mariadb.com/about-us/logos/>

3.18 Container

Container merupakan sebagai sebuah *unit* perangkat lunak yang berisi semua elemen yang diperlukan untuk menjalankan suatu aplikasi, termasuk *code*, *runtime*, *sistem* perpustakaan, variabel lingkungan, dan konfigurasi. *Container* diisolasi dari lingkungan *host* dan *container* lainnya, tetapi berbagi *kernel host* dengan *container-container* lainnya. Isolasi ini memungkinkan container berjalan dengan konsisten di lingkungan yang berbeda, termasuk di lingkungan pengembangan dan produksi.

3.19 Memcached

Memcached adalah sebuah sistem *caching* yang sering digunakan untuk meningkatkan kinerja aplikasi web dengan menyimpan data di dalam memori RAM. Data yang disimpan di dalam Memcached dapat diakses dengan cepat dan mudah dari aplikasi web, tanpa perlu melakukan akses ke *database* yang lebih lambat dan mahal (Memcached, n.d.).

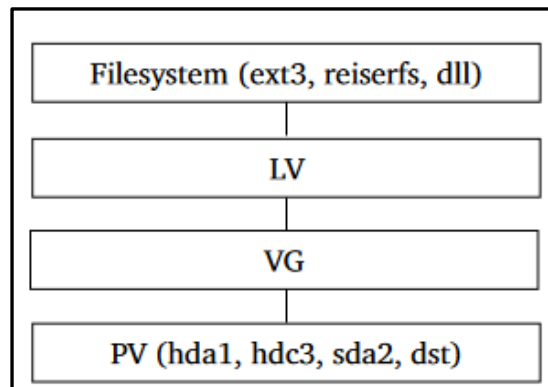
3.20 Logical Volume Management (LVM)

LVM (*Logical Volume Manager*) adalah sebuah perangkat lunak yang digunakan untuk mengatur dan manajemen partisi *hard disk* pada sistem operasi Linux. Dengan menggunakan LVM, pengguna dapat membuat partisi *virtual*, mengubah ukuran partisi, dan membagi partisi ke dalam beberapa *volume grup*.

Berikut adalah beberapa konsep penting dalam LVM:

1. *Volume Group* (VG): Kumpulan partisi fisik yang dikelompokkan menjadi satu kesatuan logis.

2. *Logical Volume (LV)*: Partisi virtual yang dibuat di atas Volume Group, dan dapat diubah ukurannya dengan mudah.
3. *Physical Volume (PV)*: Partisi fisik pada hard disk yang dikelola oleh LVM.
4. *Logical Extents (LE)*: Blok data yang digunakan oleh LVM untuk mengatur data dalam *logical volume*.



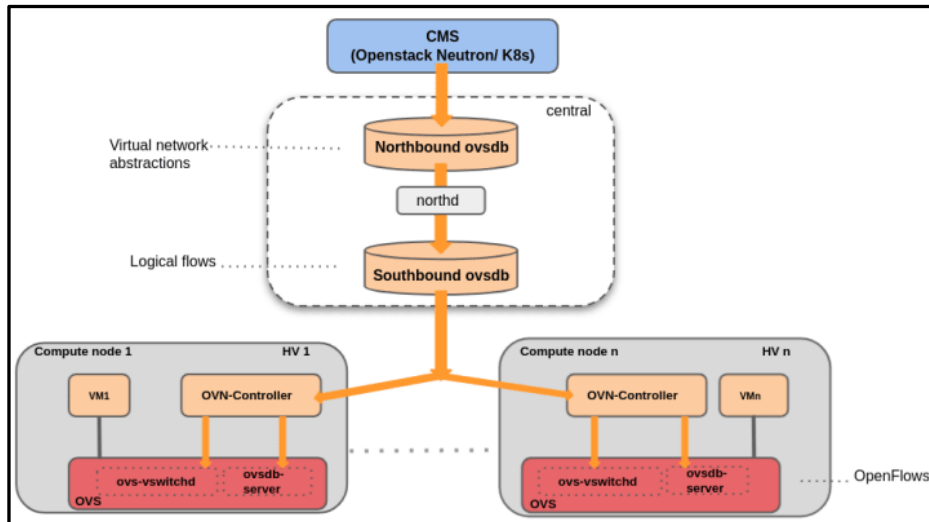
Gambar 3.13 Deskripsi LVM

Cara kerja LVM adalah dengan melakukan pengelolaan pada partisi *hard disk* yang terbagi ke dalam *physical volume*. Kemudian, LVM akan mengelompokkan *physical volume* ke dalam *volume group*, dan kemudian membentuk *logical volume* di atas *volume group*. Pengguna dapat dengan mudah membuat, mengubah ukuran, atau menghapus *logical volume* dengan menggunakan perintah LVM. Dengan menggunakan LVM, pengguna dapat mengalokasikan ruang *hard disk* secara fleksibel, memudahkan pengelolaan partisi, dan meningkatkan performa dengan mengelola ukuran *volume grup* dan *logical volume* (Setiawan, 2005).

3.21 Open Virtual Network (OVN)

OVN (Open Virtual Network) adalah sebuah proyek opensource yang memungkinkan pengguna untuk mengatur dan mengelola jaringan virtual di dalam data center mereka dengan lebih efisien. OVN berfungsi sebagai penghubung antara *hypervisor* dan *network controller*, sehingga memungkinkan pengguna untuk membuat dan mengatur jaringan *virtual* dengan mudah. Fitur-fitur OVN meliputi *overlay network*, *logical switching*, dan *logical routing*. *Overlay network* memungkinkan pengguna untuk membuat jaringan *virtual* yang terisolasi secara logis, sehingga memudahkan pengaturan dan konfigurasi jaringan. *Logical*

switching memungkinkan pengguna untuk menghubungkan mesin *virtual* ke jaringan *virtual*, sedangkan *logical routing* memungkinkan pengguna untuk mengatur *routing* antar *jaringan virtual*.



Gambar 3.14 Arsitektur OVN

Sumber: <https://ubuntu.com/blog/data-centre-networking-what-is-ovn>

OVN memiliki arsitektur yang modular dan terdiri dari beberapa komponen, termasuk OVN *controller*, OVN *northbound database*, dan OVN *southbound database*. Setiap komponen ini memiliki tugas dan fungsi yang berbeda-beda dalam mengatur dan mengelola jaringan *virtual*. Pengguna dapat menginstalasi dan mengkonfigurasi OVN di lingkungan mereka dengan menggunakan beberapa metode instalasi, termasuk menggunakan *Ubuntu Server* atau menggunakan *container Docker*. Secara keseluruhan, OVN adalah solusi *opensource* yang dapat membantu pengguna dalam mengatur dan mengelola jaringan *virtual* di dalam *data center* mereka dengan lebih efisien (Bouguerra, 2021).

3.22 OpenStack

OpenStack merupakan *platform opensource*, *open design*, *open development*, dan dapat terbuka bagi semua *community* yang ingin berpartisipasi dalam pengembangannya. Visi jangka panjang OpenStack adalah untuk menghasilkan *platform* komputasi awan yang bersifat terbuka dan dapat dipakai oleh siapa saja yang memenuhi kebutuhan penyedia *cloud* yang bersifat *public* maupun *private* terlepas dari ukurannya.

Layanan OpenStack mengontrol kumpulan besar sumber daya komputasi, penyimpanan, dan jaringan di seluruh *data center*. Teknologi dibalik OpenStack terdiri dari serangkaian komponen yang saling terkait untuk solusi infrastruktur cloud. Setiap layanan menyediakan API yang terbuka sehingga semua sumber daya dapat dikelola melalui antarmuka web, CLI, atau perangkat lunak yang mendukung API (Fifield, et al., 2014).



Gambar 3.15 Logo OpenStack

Sumber: <https://www.openstack.org/brand/openstack-logo/logo-download/>

OpenStack memiliki beberapa layanan inti, yaitu *Keystone*, *Placement*, *Glance*, *Neutron*, *Cinder*, dan *Skyline*.

3.22.1 *Keystone*

Keystone bertanggung jawab untuk mengatur dan memastikan bahwa setiap pengguna dan layanan yang terhubung dengan OpenStack dikenali secara akurat, dan hanya diberikan akses ke sumber daya yang sesuai dengan hak akses yang diberikan. *Keystone* juga berfungsi sebagai sistem manajemen identitas, yang memungkinkan pengguna untuk mengelola identitas dan hak aksesnya secara terpusat dan aman. Dengan menggunakan *Keystone*, pengguna OpenStack dapat mengelola identitas dan hak aksesnya dengan lebih efisien dan aman, tanpa harus mengelola setiap layanan secara terpisah (Khedher & Chowdhury, 2017).

3.22.2 *Placement*

Placement adalah layanan manajemen sumber daya yang memungkinkan *administrator* untuk mengalokasikan sumber daya fisik dan virtual pada lingkungan OpenStack. *Placement* menyediakan mekanisme untuk mengatur dan mengelola sumber daya seperti CPU, RAM, dan penyimpanan, sehingga dapat mengalokasikan sumber daya secara efektif dan efisien. *Placement* juga memungkinkan seorang *administrator* untuk mengatasi masalah yang berkaitan dengan pengalokasian sumber daya, seperti konflik dan kelebihan kapasitas,

sehingga dapat meminimalkan gangguan yang mungkin terjadi pada lingkungan OpenStack (OpenStack, 2022).

3.22.3 Glance

Glance merupakan layanan *image* yang menyediakan cara yang gesit dan nyaman untuk menyalin dan menjalankan suatu *instance*. Dengan *Glance*, pengguna dapat meng-*upload*, menemukan, mendaftarkan, dan mengambil *virtual machine image* dengan mudah. *Glance* menyediakan fitur pencarian untuk memudahkan administrator dalam menemukan gambar yang tepat, serta mendukung berbagai format *image*, seperti *qcow2*, *VHD*, dan *RAW*.

3.22.4 Nova

Nova merupakan komponen utama dalam *platform* OpenStack yang bertanggung jawab untuk mengelola *virtual machine* dan menyediakan antarmuka untuk mengelola sumber daya komputasi di lingkungan *cloud* (Khedher & Chowdhury, 2017). Layanan *Nova* memiliki komponen-komponen utama, yaitu:

- a. Nova-api: Komponen ini bertanggung jawab untuk menerima dan merespon *end user* kemudian komponen ini berkomunikasi dengan komponen lainnya untuk membuat *instance*.
- b. Nova-compute: Komponen ini pada dasarnya adalah *daemon* yang membuat dan menghapus *virtual machine* pada *Hypervisor*. Komponen ini dapat mengelola *virtual machine* pada *host* dan *hypervisor* yang berbeda, seperti *KVM*, *Xen*, atau *VMWare*.
- c. Nova-scheduler: Komponen ini menangani penentuan dimana *virtual machine* yang baru akan disediakan pada *hypervisor* berdasarkan berbagai filter *resource* seperti *RAM*, *CPU*, ataupun penyimpanan yang tersedia dari *hypervisor*.
- d. Nova-conductor: Komponen ini berfungsi sebagai “jembatan” supaya komponen nova-compute dapat mengakses *database* pada *controller node* untuk mengelola informasi *virtual machine*.

3.22.5 Neutron

Menurut (Khedher & Chowdhury, 2017), layanan *Neutron* menyediakan kapabilitas *Network as a Service* yang mengelola jaringan *virtual* pada OpenStack. *Neutron* memberikan kemampuan untuk membuat jaringan *virtual*, *subnet*, *router*,

dan memungkinkan *virtual machine* yang dibuat dapat terhubung dengan jaringan *virtual*. Ada beberapa ekstensi pada layanan *Neutron*, berikut diantaranya:

- a. *Router*: *Router* menyediakan *gateway* antara jaringan *virtual*.
- b. *Tenant network*: *Tenant Network* merupakan jaringan internal yang terlihat oleh *virtual machine* dan memungkinkan *virtual machine* tersebut berkomunikasi secara terisolasi dari *virtual machine* yang berbeda pemilik.
- c. *External network*: Jaringan ini merupakan jaringan yang disediakan oleh *provider* supaya *virtual machine* pada OpenStack dapat terkoneksi dengan jaringan luar atau *internet*.
- d. *Floating IP*: *Floating IP* merupakan alamat IP yang dialokasikan pada *external network* yang dipetakan oleh layanan *Neutron* ke *tenant network* suatu *virtual machine*. *Floating IP* ditetapkan ke sebuah *virtual machine* sehingga dapat diakses oleh jaringan luar.

Layanan *Neutron* memiliki komponen-komponen utama, yaitu:

- a. *Neutron server*: Komponen ini merupakan antarmuka pengguna untuk mengelola jaringan *virtual* dan sumber daya jaringan lainnya di lingkungan OpenStack, bertanggung jawab untuk mengoordinasikan fungsi *Neutron API*, *Neutron plugin*, dan *Neutron agent*.
- b. *Neutron plugin*: *Neutron plugin* bertanggung jawab untuk menyediakan fungsi dasar untuk layanan *Neutron*, termasuk manajemen jaringan dan layanan jaringan. *Plugin* dapat dipilih untuk berbagai teknologi jaringan, seperti VLAN, VXLAN, GENEVE, atau GRE. *Plugin* juga dapat memanfaatkan layanan jaringan dari *vendor* jaringan eksternal untuk menyediakan layanan jaringan yang lebih canggih, contohnya seperti OVN/OVS.
- c. *Neutron Agent*: *Neutron agent* bertanggung jawab untuk menyediakan jaringan di setiap *hypervisor* yang menjalankan *virtual machine*. *Neutron agent* dapat berjalan di *host* yang sama dengan *Nova compute* ataupun berbeda. OVN *metadata agent*, OVN *controller*, OVN *controller metadata agent* merupakan contoh *Neutron agent* yang menggunakan *backend* OVN.

3.22.6 Cinder

Menurut (Khedher & Chowdhury, 2017) Layanan *Cinder* menyediakan manajemen penyimpanan persisten untuk *hard drive* dari *virtual machine*. Tidak seperti penyimpanan sementara, *virtual machine* yang didukung oleh *Cinder* dapat dengan mudah dihidupkan, dipindahkan ataupun dievakuasi. Layanan *Cinder* menggunakan LVM & iSCSI, NFS, dan fiber untuk menghadirkan *block device* ke *virtual machine*. Layanan *Cinder* juga menyediakan limitasi penggunaan bagi *user*. Layanan *Cinder* memiliki beberapa komponen utama, yaitu:

- a. *Cinder API*: Komponen ini merupakan antarmuka pengguna dalam pengelolaan sumber daya penyimpanan pada lingkungan OpenStack. *Cinder API* menyediakan antarmuka program untuk berninteraksi antara pengguna dengan layanan *Cinder* yang dapat digunakan oleh *developer* aplikasi ataupun sistem otomatisasi untuk mengelola *volume block*.
- b. *Cinder volume*: Komponen ini bertanggung jawab dalam membuat, menghapus, dan mengubah *volume* penyimpanan. *Cinder volume* dapat diintegrasikan dengan berbagai teknologi, seperti LVM & iSCSI, NFS, Ceph, dan *fibre channel*.
- c. *Cinder scheduler*: *Cinder scheduler* berfungsi untuk menempatkan *volume block* ke dalam sumber daya penyimpanan yang tersedia pada OpenStack. *Cinder scheduler* dapat melakukan keputusan tentang penempatan *volume block* berdasarkan kebijakan yang ditentukan.

3.22.7 Skyline

Layanan *Skyline* merupakan antarmuka pengguna web berupa *dashboard* untuk pengelolaan sumber daya OpenStack. *Skyline* ini memiliki UI dan UX yang dioptimalkan dan tampilan yang modern, sehingga lebih mudah bagi pengguna untuk melakukan pemeliharaan dan pengoperasian. *Skyline* merupakan layanan yang tergolong baru, dengan fungsi yang sama dengan layanan *dashboard* yang sebelumnya, yaitu *Horizon*.

4. BAB IV

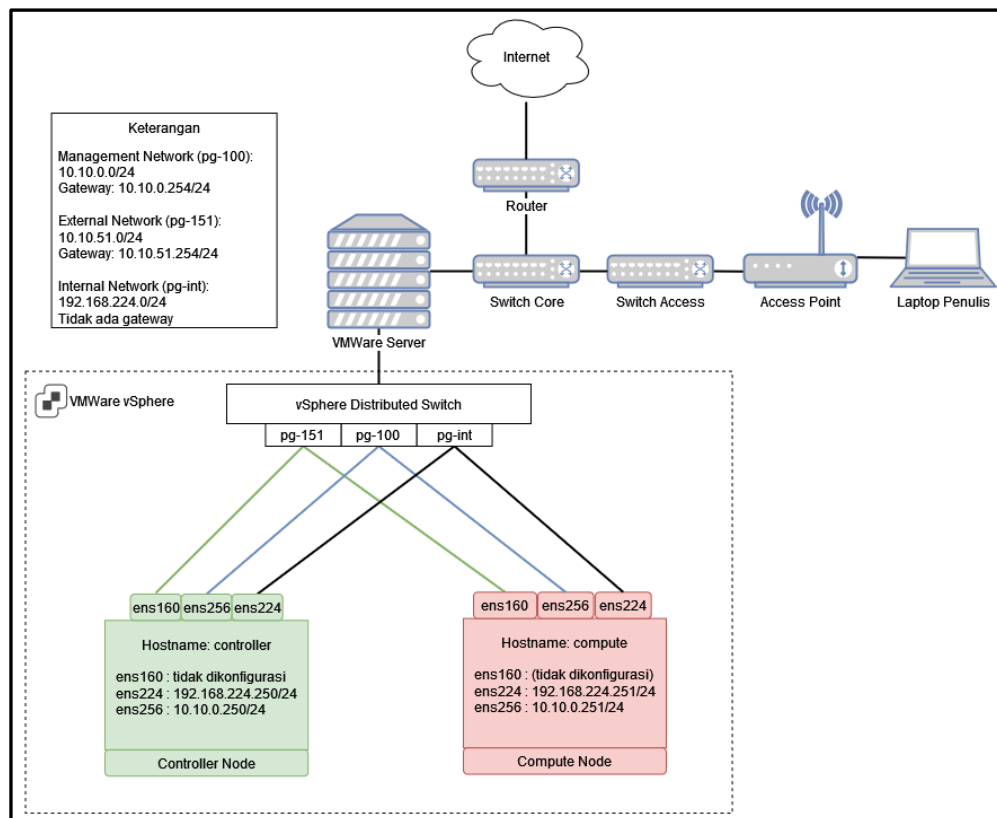
PEMBAHASAN

4.1 Perencanaan

Dalam pembahasan ini, Penulis akan melakukan perancangan dan membangun *private cloud* menggunakan OpenStack. *Private cloud* yang penulis buat bersifat simulasi, dibangun pada *virtual machine* (VM) menggunakan sistem operasi Ubuntu 22.04 yang terdapat di *server R&D (Research & Development)* milik perusahaan tempat penulis melakukan prakerin.

Penulis akan melakukan instalasi dan konfigurasi pada 2 buah VM, yaitu *controller node* dan *compute node*. Untuk membuktikan bahwa layanan *private cloud* berhasil dibangun, penulis akan melakukan pengujian mulai dari pembuatan *external network*, *security group*, *router*, *internal network*, *image*, *flavor*, *key pair*, dan pembuatan VM. Penulis juga akan melakukan pengujian konektivitas pada VM yang dibuat tersebut.

4.1.1 Topologi



Gambar 4.1 Topologi yang Digunakan

Pada Gambar 4.1, menunjukkan topologi pada perusahaan PT. Indostorage Solusi Teknologi, penulis menggunakan topologi tersebut dalam rancang bangun *private cloud* menggunakan OpenStack. Berdasarkan topologi tersebut, penulis melakukan instalasi dan konfigurasi pada *controller node* dan *compute node* yang terletak di *server* perusahaan. *Controller node* berfungsi sebagai pengontrol yang menjalankan *identity service* (Keystone), *image service* (Glance), Manajemen *compute service* (Nova), manajemen *network service* (Neutron), dan *dashboard service* (Skyline). *Compute node* menjalankan bagian *hypervisor* untuk mengoperasikan *virtual machine* (Nova) dan sebagai *block storage service* (Cinder).

Pada topologi tersebut, masing-masing *node* memiliki tiga jaringan yang tersambung ke *port group* berbeda pada *virtual distributed switch* yang tersedia pada *server* perusahaan. Setiap *port group* memiliki jaringan yang berbeda. Yang pertama yaitu *external network* dengan *port group* pg-151 memiliki *network address* 10.10.51.0/24 dan *default gateway*, digunakan sebagai jaringan khusus untuk alokasi *floating IP* yang dipakai oleh *instance* pada OpenStack. Jaringan kedua yaitu *management network* dengan *port group* pg-100 memiliki *network address* 10.10.0.0/24 dan *default gateway*, digunakan untuk mengelola OpenStack dan *remote access*. Kemudian jaringan yang ketiga yaitu *internal network* dengan *port group* pg-int memiliki *network address* 192.168.224.0/24 namun tidak memiliki *default gateway*, digunakan oleh *Neutron backend* yaitu OVN yang menyediakan *internal network* VM di dalam OpenStack.

4.1.2 Spesifikasi Kebutuhan Perangkat

Dalam membangun proyek ini, penulis menggunakan beberapa perangkat, ditunjukkan pada Tabel 4.1:

Tabel 4.1 Spesifikasi Perangkat yang Digunakan

No.	Perangkat	Spesifikasi
1	<i>Controller Node</i>	a. Sistem operasi Ubuntu 22.04 b. RAM 8 GB c. Harddisk 50 GB

		d. 6 <i>Core</i> CPU
2	<i>Compute Node</i>	a. Sistem operasi Ubuntu 22.04 b. RAM 24 GB c. <i>Harddisk</i> 1) 80 GB di /dev/sda 2) 100 GB di /dev/sdb (untuk layanan <i>Cinder</i>) 3) 100 GB di /dev/sdc (untuk layanan <i>Cinder</i>) d. 12 <i>Core</i> CPU
2	Laptop Penulis	a. Sistem operasi Windows 10 b. RAM 8 GB c. <i>Harddisk</i> 500 GB d. <i>Processor</i> Intel Core i3 5005U 2 CPUs

4.2 Langkah Kerja

Untuk membangun *private cloud* menggunakan OpenStack diperlukan beberapa tahapan kerja, yaitu:

4.2.1 Persiapan *Environment*

Sebelum melakukan instalasi dan konfigurasi layanan OpenStack, penulis melakukan persiapan *environment* yang dibutuhkan oleh masing-masing *node*, dan melakukan *remote connection* SSH pada kedua *node* secara *parallel* atau bersamaan menggunakan *Windows Terminal* dengan memasukkan kredensial (*username*, alamat IP, dan *password*) masing-masing *node*. Untuk melakukan instalasi dan konfigurasi, penulis menggunakan hak akses *root* dengan mengetikkan perintah “*sudo -i*”.

```
C:\Users\ADMIN>ssh phi@10.10.0.250
phi@10.10.0.250's password:
Last login: Tue Feb 14 01:48:10 2023 from 10.100.1.14
phi@controller:~$ |
```

Gambar 4.2 Melakukan *Remote Connection* pada *Controller Node*

```
C:\Users\ADMIN>ssh phi@10.10.0.251
phi@10.10.0.251's password:
Last login: Tue Feb 14 01:50:56 2023 from 10.100.1.14
phi@compute:~$ |
```

Gambar 4.3 Melakukan *Remote Connection* SSH pada *Compute Node*

Penulis melakukan instalasi dan konfigurasi *package* prasyarat OpenStack yang dibutuhkan *controller node*, yaitu Chrony sebagai NTP *server* dan NTP *client* bagi *compute node*, MariaDB sebagai *database service* untuk menyimpan data, RabbitMQ untuk mengkoordinasikan operasi dan informasi status antar *service*, Memcached untuk *caching* token layanan Keystone, dan Etcd untuk penyimpanan konfigurasi. Berikut merupakan tahapan konfigurasi dan instalasi dan konfigurasi *package* yang dibutuhkan.

- a. Penulis melakukan instalasi layanan NTP dengan menggunakan perintah seperti pada Gambar 4.4. Perintah ini dieksekusi pada kedua *node*.

```
root@controller:~# apt-get install chrony|
```

Gambar 4.4 Perintah Instalasi Layanan NTP

- b. Kemudian, penulis mengonfigurasi layanan NTP pada *controller node* dengan menambahkan baris seperti pada Gambar 4.5 di dalam file */etc/chrony/chrony.conf*.

```
server 0.id.pool.ntp.org iburst
allow 10.10.0.0/24
```

Gambar 4.5 Konfigurasi Layanan NTP

- c. Selanjutnya, penulis melakukan *restart* layanan NTP di *controller node* untuk menyimpan konfigurasi dan pengecekan layanan NTP dengan mengetikkan perintah seperti pada Gambar 4.6.

```
root@controller:~# systemctl restart chrony
root@controller:~# chronyc sources
MS Name/IP address          Stratum Poll Reach LastRx Last sample
=====
^? 230.subnet-8.helium.co.id 3      6      3      0  +4148us[+4148us] +/- 95ms
root@controller:~# |
```

Gambar 4.6 Me-restart dan Mengecek Layanan NTP pada *Controller Node*

- d. Untuk *compute node*, sinkronasi jam berdasarkan layanan NTP dari *controller node*, dengan menambahkan konfigurasi “*server 10.10.0.250 iburst*” pada file */etc/chrony.conf*.

```
server 10.10.0.250 iburst
```

Gambar 4.7 Sinkronisasi NTP *Compute Node* dengan *Controller Node*

- e. Selanjutnya penulis melakukan *restart* NTP *client* di *compute node* untuk menyimpan konfigurasi dan pengecekan layanan NTP dengan mengetikkan perintah seperti pada Gambar 4.8.

```
root@compute:~# systemctl restart chrony
root@compute:~# chronyc sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
=====
^? 10.10.0.250              4      6    3    1   -16us[ -16us] +/-  71ms
root@compute:~#
```

Gambar 4.8 Me-*restart* dan Mengecek NTP *Client* pada *Compute Node*

- f. Pada Ubuntu 22.04, repositori OpenStack versi Yoga sudah tersedia secara bawaan, jadi penulis tidak perlu menambahkan repositorinya. Selanjutnya pada *controller node*, dilakukan instalasi OpenStack *client* supaya dapat menggunakan perintah-perintah OpenStack.

```
root@controller:~# apt-get install python3-openstackclient
```

Gambar 4.9 Instalasi OpenStack *Client*

- g. Berikutnya, penulis melakukan instalasi *database server* MariaDB pada *controller node* dengan mengetikkan perintah berikut.

```
root@controller:~# apt install mariadb-server python3-pymysql
```

Gambar 4.10 Melakukan Instalasi *Database Server*

- h. Pada *controller node*, penulis menambahkan file */etc/mariadb.conf.d/99-openstack.cnf* dan mengonfigurasi file tersebut seperti pada Gambar 4.11.

```
[mysqld]
bind-address = 10.10.0.250

default-storage-engine = innodb
innodb_file_per_table = on
max_connections = 4096
collation-server = utf8_general_ci
character-set-server = utf8
```

Gambar 4.11 Menambahkan Konfigurasi Layanan MariaDB

Berdasarkan Gambar 4.11, terdapat konfigurasi “bind-address = 10.10.0.250”, menunjukkan bahwa *database service* MariaDB berjalan pada *controller node* dan dapat diakses oleh *compute node* melalui *management network*. Setelah melakukan konfigurasi *database service*, penulis me-restart dengan mengetikkan “*systemctl restart mariadb*” untuk menyimpan konfigurasi.

```
root@controller:~# systemctl restart mariadb
```

Gambar 4.12 Melakukan *Restart* MariaDB

- i. Selanjutnya penulis melakukan instalasi RabbitMQ pada *controller node* dengan menggunakan perintah berikut.

```
root@controller:~# apt-get install rabbitmq-server
```

Gambar 4.13 Menginstal RabbitMQ

- j. Pada Gambar 4.14, pada *controller node* penulis membuat *user* untuk layanan OpenStack di RabbitMQ.

```
root@controller:~# rabbitmqctl add_user openstack RABBIT_PASS
Adding user "openstack" ...
Done. Don't forget to grant the user permissions to some virtual hosts!
See 'rabbitmqctl help set_permissions' to learn more.
root@controller:~#
```

Gambar 4.14 Menambahkan *User* OpenStack pada RabbitMQ

- k. Setelah pembuatan user, penulis menambahkan *permission* untuk hak akses konfigurasi, akses tulis dan baca pada *user* OpenStack.

```
root@controller:~# rabbitmqctl set_permissions openstack ".*" ".*" ".*"
Setting permissions for user "openstack" in vhost "/" ...
root@controller:~#
```

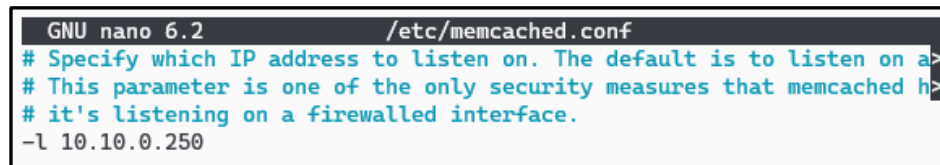
Gambar 4.15 Penambahan Hak Akses Untuk *User* OpenStack

- l. Selanjutnya, penulis menginstal layanan Memcached pada *controller node*, ditunjukkan pada Gambar 4.16.

```
root@controller:~# apt-get install python3-memcache
```

Gambar 4.16 Perintah Instalasi Layanan Memcached

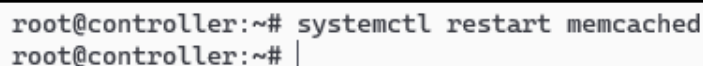
- m. Kemudian mengedit *file* */etc/memcached.conf* supaya layanan Memcached dapat diakses oleh *compute node* melalui *management network* dengan mengubah baris konfigurasi “-l 127.0.0.1” dengan baris “-l 10.10.0.250”.



```
GNU nano 6.2 /etc/memcached.conf
# Specify which IP address to listen on. The default is to listen on all
# This parameter is one of the only security measures that memcached has
# it's listening on a firewalled interface.
-l 10.10.0.250
```

Gambar 4.17 Konfigurasi Memcached

Untuk menyimpan konfigurasi tersebut, dilakukan *restart* layanan Memcached dengan menggunakan perintah “*systemctl restart memcached*”.



```
root@controller:~# systemctl restart memcached
root@controller:~#
```

Gambar 4.18 Restart Layanan Memcached

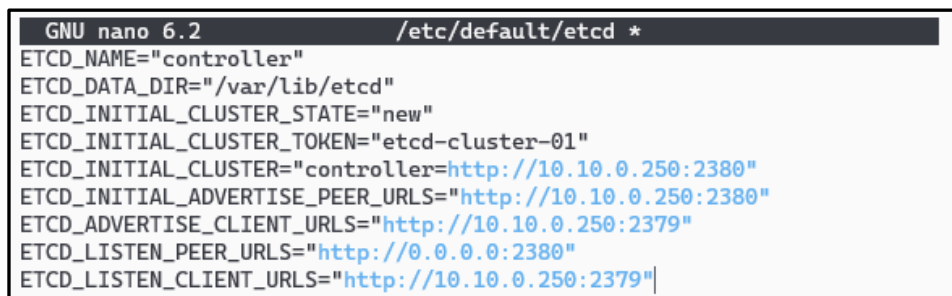
- n. Berikutnya, penulis melakukan instalasi layanan Etcd pada *controller node*, seperti pada Gambar 4.19.



```
root@controller:~# apt-get install etcd
```

Gambar 4.19 Perintah Instalasi Layanan Etcd

- o. Mengganti nama *file* */etc/default/etcd* menjadi */etc/default/etcd.backup* untuk *backup file* konfigurasi bawaan dan membuat *file* baru bernama */etc/default/etcd*, kemudian penulis menambahkan konfigurasi pada *file* */etc/default/etcd* seperti pada Gambar 4.20.



```
GNU nano 6.2 /etc/default/etcd *
ETCD_NAME="controller"
ETCD_DATA_DIR="/var/lib/etcd"
ETCD_INITIAL_CLUSTER_STATE="new"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-01"
ETCD_INITIAL_CLUSTER="controller=http://10.10.0.250:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://10.10.0.250:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://10.10.0.250:2379"
ETCD_LISTEN_PEER_URLS="http://0.0.0.0:2380"
ETCD_LISTEN_CLIENT_URLS="http://10.10.0.250:2379"
```

Gambar 4.20 Konfigurasi Layanan Etcd

Berdasarkan gambar di atas, *value* dari *ETCD_INITIAL_CLUSTER*, *ETCD_INITIAL_ADVERTISE_PEER_URLS*, *ETCD_ADVERTISE_CLIENT_URLS*, dan *ETCD_LISTEN_CLIENT_URLS* menggunakan alamat IP *management* supaya layanan Etcd dapat diakses oleh *compute node*. Setelah melakukan konfigurasi, dilakukan *restart* layanan Etcd untuk menyimpan konfigurasi dengan mengetikkan “*systemctl restart etcd*”.

```
root@controller:~# systemctl restart etcd
root@controller:~# |
```

Gambar 4.21 Restart Layanan Etcd

4.2.2 Instalasi dan Konfigurasi Keystone

Instalasi dan konfigurasi layanan *Keystone* dilakukan pada *controller node*. Berikut langkah-langkah pengerjaan yang penulis lakukan.

- a. Penulis menambah *user* dan *database* untuk layanan Keystone di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.22.

```
MariaDB [(none)]> CREATE DATABASE keystone;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
-> IDENTIFIED BY 'KEYSTONE_DBPASS';
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%'
IDENTIFIED BY 'KEYSTONE_DBPASS';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> |
```

Gambar 4.22 Menambahkan User dan Database Layanan Keystone

- b. Selanjutnya, penulis melakukan instalasi *package* Keystone dengan menggunakan perintah seperti pada Gambar 4.23.

```
root@controller:~# apt-get install keystone|
```

Gambar 4.23 Perintah Instalasi Layanan Keystone

- c. Penulis mengedit *file* konfigurasi Keystone di */etc/keystone/keystone.conf*. Pada *section [database]* penulis menambahkan konfigurasi yang mengoneksikan Keystone dengan *database* yang sebelumnya dibuat, seperti pada Gambar 4.24.

```
GNU nano 6.2 /etc/keystone/keystone.conf

[database]
#connection = sqlite:///var/lib/keystone/keystone.db
connection = mysql+pymysql://keystone:KEYSTONE_DBPASS@10.10.0.250/keystone
```

Gambar 4.24 Konfigurasi Database Keystone

Kemudian penulis menambahkan *value "fernet" provider* pada *section [tx`oken]*.


```

GNU nano 6.2 /etc/keystone/keystone.conf
# reason to change this option unless you are providing a custom entry point.
# (string value)
#driver = sql
[token]
provider = fernet

```

Gambar 4.25 Konfigurasi *Fernet Token* Keystone

- d. Penulis melakukan sinkronasi *database* dengan konfigurasi Keystone menggunakan perintah seperti pada Gambar 4.26.

```

root@controller:~# su -s /bin/sh -c "keystone-manage db_sync" keystone
root@controller:~# keystone-manage fernet_setup --keystone-user keystone --keysto
ne-group keystone
root@controller:~# keystone-manage credential_setup --keystone-user keystone --ke
ystone-group keystone
root@controller:~# |

```

Gambar 4.26 Sinkronasi *Database* dengan Keystone

- e. Selanjutnya, penulis melakukan *bootstrapping* layanan Keystone untuk pembuatan *user* yang akan digunakan pada layanan-layanan OpenStack. Pada Gambar 4.27, menunjukkan penggunaan perintah *bootstrapping password user* “*admin*”, *endpoint*, dan *region id*.

```

root@controller:~# keystone-manage bootstrap --bootstrap-password ADMIN_PASS \
--bootstrap-admin-url http://10.10.0.250:5000/v3/ \
--bootstrap-internal-url http://10.10.0.250:5000/v3/ \
--bootstrap-public-url http://10.10.0.250:5000/v3/ \
--bootstrap-region-id RegionOne
root@controller:~# |

```

Gambar 4.27 *Bootstrap* Layanan Keystone

- f. Penulis melakukan *restart* layanan *web server* Apache2 dengan menggunakan perintah “*systemctl restart apache2*”.
- g. Selanjutnya penulis menambahkan file baru *~/admin-openrc* untuk menyimpan Keystone *administrative account* dalam bentuk *environment variable*. File ini digunakan untuk pemakaian layanan OpenStack dengan *user* admin melalui *command line*.

```

GNU nano 6.2 admin-openrc
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_PROJECT_NAME=admin
export OS_USER_DOMAIN_NAME=Default
export OS_PROJECT_DOMAIN_NAME=Default
export OS_AUTH_URL=http://10.10.0.250:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_VOLUME_API_VERSION=3

```

Gambar 4.28 *Environment Variable User* Keystone

Supaya file yang dibuat dapat digunakan, file tersebut ditambahkan hak akses *executeable* dengan menggunakan perintah seperti pada Gambar 4.29.

```
root@controller:~# chmod +x ~/admin-openrc
root@controller:~# |
```

Gambar 4.29 Perintah Menambah Hak Akses

Untuk menggunakan perintah-perintah layanan OpenStack, penulis mengeksekusi file *admin-openrc* dengan melakukan perintah “*source ~/admin-openrc*”.

- h. Kemudian, penulis membuat *project* “*service*” di OpenStack yang digunakan oleh layanan-layanan OpenStack dengan melakukan perintah seperti pada Gambar 4.30.

```
root@controller:~# openstack project create --domain default \
--description "Service Project" service
```

Field	Value
description	Service Project
domain_id	default
enabled	True
id	7907996049ea49358abf6ed2a38cef41
is_domain	False
name	service
options	{}
parent_id	default
tags	[]

```
root@controller:~# |
```

Gambar 4.30 Perintah dan Output dari Pembuatan *project service*

4.2.3 Instalasi dan Konfigurasi Glance

Instalasi dan konfigurasi layanan *Glance* dilakukan pada *controller node*. Berikut langkah-langkah pengerjaan yang penulis lakukan.

- a. Penulis menambah *user* dan *database* untuk layanan *Glance* di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.31.

```
MariaDB [(none)]> CREATE DATABASE glance;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
-> IDENTIFIED BY 'GLANCE_DBPASS';
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
-> IDENTIFIED BY 'GLANCE_DBPASS';
Query OK, 0 rows affected (0.002 sec)
```

Gambar 4.31 Menambahkan *User* dan *Database* Layanan *Glance*

- b. Selanjutnya penulis membuat *user* layanan *Glance* pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti pada Gambar 4.32. Untuk *password*, penulis memasukkan *value* “GLANCE_PASS”.

```

root@controller:~# openstack user create --domain default --password-prompt glance
User Password:
Repeat User Password:
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | eblade58da604bcb81db060bd3e342df |
| name | glance |
| options | {} |
| password_expires_at | None |
+-----+-----+
root@controller:~# openstack role add --project service --user nova admin
root@controller:~#

```

Gambar 4.32 Menambah *User* Layanan *Glance* pada OpenStack

- c. Penulis menambah entitas *service* untuk *Glance* pada OpenStack, perintah ditunjukkan pada Gambar 4.33.

```

root@controller:~# openstack service create --name glance \
--description "OpenStack Image" image
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Image |
| enabled | True |
| id | 7e70696fb86e4adbaff857bbc4c0e57c |
| name | glance |
| type | image |
+-----+-----+
root@controller:~#

```

Gambar 4.33 Menambah *Service Glance* pada OpenStack

- d. Penulis menambahkan tiga *endpoint* untuk layanan *Glance*. *Public endpoint*, *internal endpoint*, dan *admin endpoint*. Pada Gambar 4.34 menunjukkan perintah pembuatan *endpoint* layanan *Glance* beserta outputnya.

```

root@controller:~# openstack endpoint create --region RegionOne image public http://10.10.0.250:9292 && \
openstack endpoint create --region RegionOne image internal http://10.10.0.250:9292 && \
openstack endpoint create --region RegionOne image admin http://10.10.0.250:9292

```

Field	Value
enabled	True
id	7174a13477cb4e30b8223b1d676cbde8
interface	public
region	RegionOne
region_id	RegionOne
service_id	3eb717bf67f84f7c96aa3803061267c9
service_name	glance
service_type	image
url	http://10.10.0.250:9292

Field	Value
enabled	True
id	890151aa6a9c4c2880c2e19a16b05a56
interface	internal
region	RegionOne
region_id	RegionOne
service_id	3eb717bf67f84f7c96aa3803061267c9
service_name	glance
service_type	image
url	http://10.10.0.250:9292

Field	Value
enabled	True
id	7d7db07575114cfe80d6fa028a0a2468
interface	admin
region	RegionOne
region_id	RegionOne
service_id	3eb717bf67f84f7c96aa3803061267c9
service_name	glance
service_type	image
url	http://10.10.0.250:9292

Gambar 4.34 Menambahkan *Endpoint* Layanan *Glance*

- e. Penulis melakukan instalasi *package Glance* dengan menggunakan perintah seperti pada Gambar 4.35.

```
root@controller:~# apt-get install glance
```

Gambar 4.35 Perintah Instalasi *Glance*

- f. Kemudian penulis melakukan konfigurasi layanan *Glance* pada file */etc/glance-api.conf*. Pada file tersebut ditambahkan beberapa konfigurasi, diantaranya:
- 1) Menambah konfigurasi pada *section [database]* untuk mengoneksikan layanan *Glance* dengan *database* seperti pada Gambar 4.36.

```

GNU nano 6.2 /etc/glance/glance-api.conf *
[database]
connection = mysql+pymysql://glance:GLANCE_DBPASS@10.10.0.250/glance

```

Gambar 4.36 Konfigurasi *Database Glance*

- 2) Penulis menambah konfigurasi pada *section* `[keystone_authtoken]` untuk menghubungkan layanan *Glance* dengan layanan *Keystone* seperti pada Gambar 4.37.

```
GNU nano 6.2 /etc/glance/glance-api.conf *

[keystone_authtoken]
www_authenticate_uri = http://10.10.0.250:5000
auth_url = http://10.10.0.250:5000
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = glance
password = GLANCE_PASS
```

Gambar 4.37 Konfigurasi untuk Menghubungkan *Glance* dan *Keystone*

- 3) Penulis menambahkan konfigurasi pada *section* `[glance_store]` untuk lokasi dan cara penyimpanan *image* seperti pada Gambar 4.38.

```
GNU nano 6.2 /etc/glance/glance-api.conf *

[glance_store]
stores = file,http
default_store = file
filesystem_store_datadir = /var/lib/glance/images/
```

Gambar 4.38 Konfigurasi untuk Penyimpanan *Image*

- g. Penulis menyinkronisasi *database* dengan konfigurasi *Glance* dengan menggunakan perintah seperti pada Gambar 4.39.

```
root@controller:~# su -s /bin/sh -c "glance-manage db_sync" glance|
```

Gambar 4.39 Perintah Sinkronisasi *Database Glance*

- h. Untuk memastikan layanan *Glance* telah terkonfigurasi, penulis melakukan *restart* layanan dengan menggunakan perintah seperti pada Gambar 4.40.

```
root@controller:~# service glance-api restart|
```

Gambar 4.40 Perintah *Restart* Layanan *Glance*

4.2.4 Instalasi dan Konfigurasi *Placement*

Instalasi dan konfigurasi layanan *Placement* dilakukan pada *controller node*. Berikut langkah-langkah pengerjaan yang penulis lakukan:

- a. Penulis menambahkan *user* dan *database* untuk layanan *Placement* di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.41.

```
MariaDB [(none)]> CREATE DATABASE placement;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'localhost' \
-> IDENTIFIED BY 'PLACEMENT_DBPASS';
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON placement.* TO 'placement'@'%' \
-> IDENTIFIED BY 'PLACEMENT_DBPASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> |
```

Gambar 4.41 Menambahkan *User* dan *Database* Layanan *Placement*

- b. Penulis membuat *user* layanan *Placement* pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti pada Gambar 4.42. Untuk *password*, penulis memasukkan *value* “*PLACEMENT_PASS*”.

```
root@controller:~# openstack user create --domain default --password-prompt placement
User Password:
Repeat User Password:

+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 792e05907856490da72bc559367d70e4 |
| name | placement |
| options | {} |
| password_expires_at | None |
+-----+-----+
root@controller:~# openstack role add --project service --user placement admin
root@controller:~# |
```

Gambar 4.42 Menambah *User* Layanan *Placement* pada OpenStack

- c. Penulis menambahkan entitas *service* untuk *Placement* pada OpenStack, perintah ditunjukkan pada Gambar 4.43.

```
root@controller:~# openstack service create --name placement --description "Placement API" placement
+-----+-----+
| Field | Value |
+-----+-----+
| description | Placement API |
| enabled | True |
| id | 313873be11d647dd9a3a914626ee2fdc |
| name | placement |
| type | placement |
+-----+-----+
root@controller:~# |
```

Gambar 4.43 Menambah *Service* *Placement* pada OpenStack

- d. Penulis menambah tiga *endpoint* untuk layanan *Placement*. *Public endpoint*, *internal endpoint*, dan *admin endpoint*. Pada Gambar 4.44 menunjukkan perintah pembuatan *endpoint* layanan *Glance* beserta outputnya.

```

root@controller:~# openstack endpoint create --region RegionOne placement public http://10.10.0.250:8778 && \
openstack endpoint create --region RegionOne placement internal http://10.10.0.250:8778 && \
openstack endpoint create --region RegionOne placement admin http://10.10.0.250:8778

```

Field	Value
enabled	True
id	004255580ade40b58a29077a12ec715f
interface	public
region	RegionOne
region_id	RegionOne
service_id	ee990e8c6bdc40a1ab923b0b76442459
service_name	placement
service_type	placement
url	http://10.10.0.250:8778

Field	Value
enabled	True
id	596b56e373eb4625acea98b3ad0247c2
interface	internal
region	RegionOne
region_id	RegionOne
service_id	ee990e8c6bdc40a1ab923b0b76442459
service_name	placement
service_type	placement
url	http://10.10.0.250:8778

Field	Value
enabled	True
id	35bb1529c8d84963ac8b3109967195fa
interface	admin
region	RegionOne
region_id	RegionOne
service_id	ee990e8c6bdc40a1ab923b0b76442459
service_name	placement
service_type	placement
url	http://10.10.0.250:8778

Gambar 4.44 Menambahkan *Endpoint Layanan Placement*

- e. Selanjutnya, penulis meng-*install package Placement* dengan menggunakan perintah seperti pada Gambar 4.45.

```
root@controller:~# apt-get install placement-api
```

Gambar 4.45 Perintah Instalasi *Placement*

- f. Penulis mengonfigurasi layanan *Placement* pada file */etc/placement.conf*. Pada file tersebut ditambahkan beberapa konfigurasi, diantaranya:

- 1) Menambah konfigurasi pada *section [placement_database]* untuk mengoneksikan layanan *Placement* dengan *database* seperti pada Gambar 4.46.

```

GNU nano 6.2 /etc/placement/placement.conf *

[placement_database]
connection = mysql+pymysql://placement:PLACEMENT_PASS@10.10.0.250/placement

```

Gambar 4.46 Konfigurasi *Database Placement*

- 2) Penulis menambahkan konfigurasi pada *section* `[keystone_authtoken]` dan `[api]` untuk mengoneksikan layanan *Placement* dengan layanan *Keystone* seperti pada Gambar 4.47.

```
GNU nano 6.2 /etc/placement/placement.conf *

[keystone_authtoken]
auth_url = http://10.10.0.250:5000/v3
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = placement
password = PLACEMENT_PASS

[api]
auth_strategy = keystone
```

Gambar 4.47 Konfigurasi Menghubungkan *Placement* dengan *Keystone*

- g. Penulis menyinkronisasi *database* dengan konfigurasi *Glance* dengan melakukan perintah seperti pada Gambar 4.48.

```
root@controller:~# su -s /bin/sh -c "placement-manage db sync" placement|
```

Gambar 4.48 Perintah Sinkronisasi *Database Placement*

- h. Kemudian, penulis memuat ulang layanan *web server* untuk menyesuaikan konfigurasi *Placement* dengan pengaturan konfigurasi yang baru.

```
root@controller:~# service apache2 restart|
```

Gambar 4.49 Perintah *Restart Web Server*

4.2.5 Instalasi dan Konfigurasi *Nova*

Instalasi dan konfigurasi layanan *Nova* dilakukan pada *controller node* dan *compute node*. Penulis melakukan instalasi dan konfigurasi layanan *Nova* pada *controller node* terlebih dahulu, berikut langkah pengerjaannya:

- a. Penulis menambahkan *user* dan *database* yang dibutuhkan layanan *Nova* di MariaDB. Pembuatan *user* dan *database* ditunjukkan pada Gambar 4.50.


```

MariaDB [(none)]> CREATE DATABASE nova_api;
MariaDB [(none)]> CREATE DATABASE nova;
MariaDB [(none)]> CREATE DATABASE nova_cell0;
MariaDB [(none)]>
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]>
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'localhost'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON nova_cell0.* TO 'nova'@'%'
-> IDENTIFIED BY 'NOVA_PASS';
MariaDB [(none)]> |

```

Gambar 4.50 Menambah User dan Database Layanan Nova

- b. Penulis membuat *user* layanan Nova pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti Gambar 4.51. Untuk *password*, penulis memasukkan *value* “NOVA_PASS”.

```

root@controller:~# openstack user create --domain default --password-prompt nova
User Password:
Repeat User Password:

```

Field	Value
domain_id	default
enabled	True
id	8a7e9fd059243e0a5e5f4ace777c287
name	nova
options	{}
password_expires_at	None

```

root@controller:~# openstack role add --project service --user nova admin
root@controller:~# |

```

Gambar 4.51 Menambah User Layanan Nova pada OpenStack

- c. Selanjutnya, penulis membuat entitas *service* untuk Nova pada OpenStack, perintah ditunjukkan pada Gambar 4.52.

```

root@controller:~# openstack service create --name nova \
--description "OpenStack Compute" compute

```

Field	Value
description	OpenStack Compute
enabled	True
id	d24db86c19c5496e8bb70a76e7587083
name	nova
type	compute

```

root@controller:~# |

```

Gambar 4.52 Menambah Service Nova pada OpenStack

- d. Penulis menambahkan tiga *endpoint* untuk layanan *Nova*. *Public endpoint*, *internal endpoint*, dan *admin endpoint*. Pada Gambar 4.53 menunjukkan perintah pembuatan *endpoint* layanan *Nova* beserta outputnya.

```
root@controller:~# openstack endpoint create --region RegionOne compute public http://10.10.0.250:8774/v2.1 && \
openstack endpoint create --region RegionOne compute internal http://10.10.0.250:8774/v2.1 && \
openstack endpoint create --region RegionOne compute admin http://10.10.0.250:8774/v2.1
```

Field	Value
enabled	True
id	7ee60fdbae6f4c3598c0ce144032e2ed
interface	public
region	RegionOne
region_id	RegionOne
service_id	d3014f92bb804ed0b71a6883ee051218
service_name	nova
service_type	compute
url	http://10.10.0.250:8774/v2.1

Field	Value
enabled	True
id	92e8772a36cb49c5a38bb165f6a4a9fff
interface	internal
region	RegionOne
region_id	RegionOne
service_id	d3014f92bb804ed0b71a6883ee051218
service_name	nova
service_type	compute
url	http://10.10.0.250:8774/v2.1

Field	Value
enabled	True
id	0ad8df3fc2ce44369f27d0a5408ae842
interface	admin
region	RegionOne
region_id	RegionOne
service_id	d3014f92bb804ed0b71a6883ee051218
service_name	nova
service_type	compute
url	http://10.10.0.250:8774/v2.1

Gambar 4.53 Menambahkan *Endpoint* Layanan *Nova*

- e. Penulis meng-*install package-package* layanan *Nova* untuk *controller node* dengan menggunakan perintah seperti pada Gambar 4.54.

```
root@controller:~# apt-get install nova-api nova-conductor \
> nova-novncproxy nova-scheduler
```

Gambar 4.54 Perintah Instalasi *Package-package* *Nova*

- f. Selanjutnya, penulis mengonfigurasi layanan *Nova* untuk *controller node* pada file */etc/nova/nova.conf*. Pada file tersebut ditambahkan beberapa konfigurasi, diantaranya:
- 1) Penulis menambahkan konfigurasi pada *section [api_database]* dan *[database]* untuk mengoneksikan layanan *Nova* dengan *database* seperti pada Gambar 4.55.

```

GNU nano 6.2 /etc/nova/nova.conf *

[api_database]
connection = mysql+pymysql://nova:NOVA_PASS@10.10.0.250/nova_api

[database]
connection = mysql+pymysql://nova:NOVA_PASS@10.10.0.250/nova

```

Gambar 4.55 Konfigurasi Koneksi Database Nova

- 2) Penulis menambahkan konfigurasi pada *section* `[keystone_authtoken]` dan `[api]` untuk mengoneksikan layanan Nova pada *controller node* dengan layanan *Keystone* seperti pada Gambar 4.56.

```

GNU nano 6.2 /etc/nova/nova.conf *

[keystone_authtoken]
www_authenticate_uri = http://10.10.0.250:5000/
auth_url = http://10.10.0.250:5000/
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[api]
auth_strategy = keystone

```

Gambar 4.56 Konfigurasi Koneksi Nova Controller dengan Keystone

- 3) Pada *section* `[vnc]`, penulis menambahkan konfigurasi VNC supaya *vm* yang terdapat di *compute node* dapat diakses melalui *web browser* menggunakan alamat IP *controller node*. Gambar 4.57 menunjukkan konfigurasi yang ditambahkan.

```

GNU nano 6.2 /etc/nova/nova.conf *

[vnc]
enabled = true
server_listen = 10.10.0.250
server_proxyclient_address = 10.10.0.250

```

Gambar 4.57 Konfigurasi VNC untuk Web Console

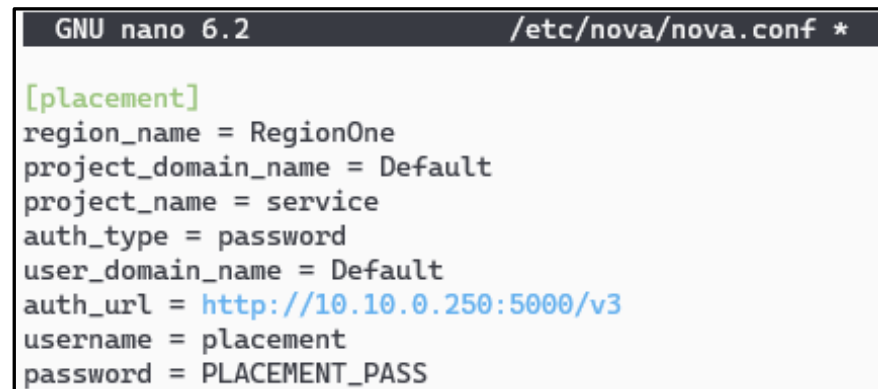
- 4) Kemudian penulis menambahkan konfigurasi pada *section [glance]* supaya layanan *Nova* di *controller node* dapat berkomunikasi dengan layanan *Glance*. Konfigurasi ditunjukkan pada Gambar 4.58.



```
GNU nano 6.2 /etc/nova/nova.conf *
[glance]
api_servers = http://10.10.0.250:9292
```

Gambar 4.58 Konfigurasi Koneksi *Nova Controller* dengan *Glance*

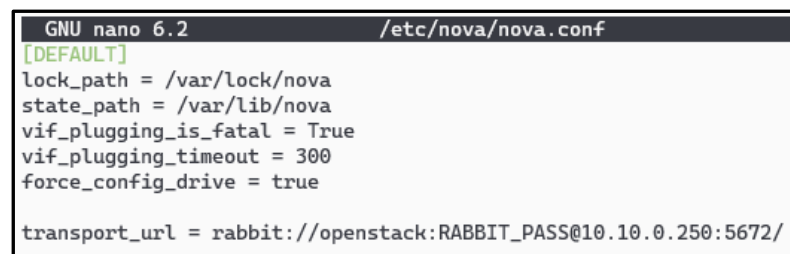
- 5) Selanjutnya penulis menambahkan konfigurasi pada *section [placement]* supaya layanan *Nova* pada *controller node* dapat berkomunikasi dengan layanan *Placement*.



```
GNU nano 6.2 /etc/nova/nova.conf *
[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://10.10.0.250:5000/v3
username = placement
password = PLACEMENT_PASS
```

Gambar 4.59 Konfigurasi Koneksi *Nova Controller* dengan *Placement*

- 6) Pada *section [DEFAULT]*, penulis menambahkan baris konfigurasi untuk mengoneksikan layanan *Nova* pada *controller node* dengan *RabbitMQ*, seperti pada Gambar 4.60.



```
GNU nano 6.2 /etc/nova/nova.conf
[DEFAULT]
lock_path = /var/lock/nova
state_path = /var/lib/nova
vif_plugging_is_fatal = True
vif_plugging_timeout = 300
force_config_drive = true

transport_url = rabbit://openstack:RABBIT_PASS@10.10.0.250:5672/
```

Gambar 4.60 Penambahan Baris Konfigurasi Koneksi *RabbitMQ*

- g. Setelah itu, penulis melakukan *restart package-package* layanan *Nova* menggunakan perintah seperti pada Gambar 4.61.

```
root@controller:~# systemctl restart nova-api.service nova-scheduler.service \
> nova-conductor.service nova-novncproxy.service
root@controller:~# |
```

Gambar 4.61 Perintah *Restart Package* Layanan Nova

Setelah mengonfigurasi layanan *Nova* pada *controller node*, penulis melakukan konfigurasi *hypervisor* layanan *Nova*, yaitu *compute node*. Berikut langkah pengerjaannya:

- a. Penulis melakukan instalasi *package* untuk *compute node* menggunakan perintah seperti pada Gambar 4.62.

```
root@compute:~# apt-get install nova-compute|
```

Gambar 4.62 Perintah Instalasi *Package nova-compute*

- b. Berikutnya, penulis mengonfigurasi file */etc/nova/nova.conf*. Pada file tersebut ditambahkan beberapa konfigurasi, diantaranya:
 - 1) Pada *section [DEFAULT]*, penulis menambahkan baris konfigurasi untuk mengoneksikan layanan *Nova* pada *compute node* dengan *RabbitMQ*, seperti pada Gambar 4.63.

```
GNU nano 6.2 /etc/nova/nova.conf *
#from compute
[DEFAULT]
log_dir = /var/log/nova
lock_path = /var/lock/nova
state_path = /var/lib/nova
use_neutron = true
force_config_drive = true

transport_url = rabbit://openstack:RABBIT_PASS@10.10.0.250:5672/
```

Gambar 4.63 Penambahan Baris Konfigurasi Koneksi *RabbitMQ*

- 2) Menambah konfigurasi pada *section [keystone_auth_token]* dan *[api]* untuk mengoneksikan layanan *Nova* pada *compute node* dengan layanan *Keystone* seperti pada Gambar 4.64.

```

GNU nano 6.2 /etc/nova/nova.conf *

[keystone_authtoken]
www_authenticate_uri = http://10.10.0.250:5000/
auth_url = http://10.10.0.250:5000/
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = Default
user_domain_name = Default
project_name = service
username = nova
password = NOVA_PASS

[api]
auth_strategy = keystone

```

Gambar 4.64 Konfigurasi Koneksi *Nova Compute* dengan *Keystone*

- 3) Kemudian pada *section [vnc]*, penulis menambahkan konfigurasi VNC *server* dan *men-setting* supaya *vm* di *compute node* dapat diakses melalui *website* dengan menagakses alamat IP *controller node*. Konfigurasi yang penulis tambahkan terdapat pada Gambar 4.65.

```

GNU nano 6.2 /etc/nova/nova.conf *

[vnc]
enabled = true
server_listen = 0.0.0.0
server_proxyclient_address = 10.10.0.251
novncproxy_base_url = http://10.10.0.250:6080/vnc_auto.html
#

```

Gambar 4.65 Konfigurasi VNC pada *Compute Node*

- 4) Kemudian penulis menambahkan konfigurasi pada *section [glance]* supaya layanan *Nova* di *compute node* dapat berkomunikasi dengan layanan *Glance*. Konfigurasi ditunjukkan pada Gambar 4.66.

```

GNU nano 6.2 /etc/nova/nova.conf *

[glance]
api_servers = http://10.10.0.250:9292

# Configuration options for the Image service

```

Gambar 4.66 Konfigurasi Koneksi *Nova Compute* dengan *Glance*

- 5) Selanjutnya penulis menambahkan konfigurasi pada *section [placement]* supaya layanan *Nova* pada *compute node* dapat berkomunikasi dengan layanan *Placement*.

```

GNU nano 6.2 /etc/nova/nova.conf

[placement]
region_name = RegionOne
project_domain_name = Default
project_name = service
auth_type = password
user_domain_name = Default
auth_url = http://10.10.0.250:5000/v3
username = placement
password = PLACEMENT_PASS

```

Gambar 4.67 Konfigurasi Koneksi *Nova Compute* dengan *Placement*

- 6) Penulis menambahkan konfigurasi pada *section [scheduler]* untuk *discovery compute node* yang baru dikonfigurasi, supaya *controller node* dapat men-*discover compute node* baru setiap *interval* yang ditentukan. Penulis melakukan konfigurasi seperti pada Gambar 4.68.

```

GNU nano 6.2 /etc/nova/nova.conf *

[scheduler]
discover_hosts_in_cells_interval = 300

```

Gambar 4.68 Konfigurasi untuk *sceduling* pencarian *compute node*

- 7) Kemudian penulis mengonfigurasi *file /etc/nova/nova-compute.conf* untuk menentukan *software hypervisor* apa yang akan digunakan. Penulis menggunakan *software QEMU* untuk *hypervisornya*. Konfigurasi ditambahkan pada *section [libvirt]* seperti pada Gambar 4.69.

```

GNU nano 6.2 /etc/nova/nova-compute.conf *

[DEFAULT]
compute_driver=libvirt.LibvirtDriver
[libvirt]
virt_type=qemu

```

Gambar 4.69 Menentukan *Hypervisor* yang Digunakan

- 8) Kemudian, penulis melakukan *restart package nova-compute* dengan menggunakan perintah seperti pada Gambar 4.70.

```

root@compute:~# systemctl restart nova-compute.service |

```

Gambar 4.70 Perintah *Restart Package Nova Compute*

Setelah mengonfigurasi layanan *Nova* di kedua *node*, penulis melakukan sinkronisasi *database* layanan *Nova* dengan konfigurasi yang ditambahkan

sebelumnya dengan menggunakan perintah seperti pada Gambar 4.71. Perintah tersebut dieksekusi pada *controller node*.

```
root@controller:~# su -s /bin/sh -c "nova-manage cell_v2 map_cell0" nova && \
> su -s /bin/sh -c "nova-manage cell_v2 create_cell --name=cell1" nova && \
> su -s /bin/sh -c "nova-manage db sync" nova && \
> su -s /bin/sh -c "nova-manage cell_v2 list_cells" nova
```

Gambar 4.71 Sinkronisasi Konfigurasi dengan *Database Layanan Nova*

Untuk mengecek layanan *Nova* yang berjalan, dapat menggunakan perintah seperti pada Gambar 4.72.

```
root@controller:~# openstack compute service list
```

ID	Binary	Host	Zone	Status	State	Updated At
351b3944-b852-4c57-803c-2f0ff20460bf	nova-scheduler	controller	internal	enabled	up	2023-02-19T06:15:04.000000
5beac64b-60e6-4117-b0ab-dfdddf6d8cfd	nova-conductor	controller	internal	enabled	up	2023-02-19T06:15:09.000000
fc0b27e8-6b45-495b-ad1a-7ac0806bbfe2	nova-compute	compute	nova	enabled	up	2023-02-19T06:15:02.000000

```
root@controller:~#
```

Gambar 4.72 Pengecekan Layanan *Nova*

4.2.6 Instalasi dan Konfigurasi *Neutron*

Instalasi dan konfigurasi layanan *Nova* dilakukan pada *controller node* dan *compute node*. Penulis melakukan instalasi dan konfigurasi layanan *Neutron* pada *controller node* terlebih dahulu, berikut langkah pengerjaannya:

- Penulis menambahkan *user* dan *database* untuk layanan *Neutron* di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.73.

```
MariaDB [(none)]> CREATE DATABASE neutron;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost'
-> IDENTIFIED BY 'NEUTRON_PASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%'
-> IDENTIFIED BY 'NEUTRON_PASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> |
```

Gambar 4.73 Menambab *User* dan *Database Layanan Neutron*

- Penulis membuat *user* layanan *Neutron* pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti pada Gambar 4.74. Untuk *password*, penulis memasukkan *value* “NEUTRON_PASS”.


```

root@controller:~# openstack user create --domain default --password-prompt neutron
User Password:
Repeat User Password:
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 363b928eac844d8290d8df70a94d3886 |
| name | neutron |
| options | {} |
| password_expires_at | None |
+-----+-----+
root@controller:~# openstack role add --project service --user neutron admin
root@controller:~#

```

Gambar 4.74 Menambah *User Layanan Neutron* pada OpenStack

- c. Kemudian, penulis menambah entitas *service* untuk *Neutron* pada OpenStack, perintah ditunjukkan pada Gambar 4.75.

```

root@controller:~# openstack service create --name neutron \
> --description "OpenStack Networking" network
+-----+-----+
| Field | Value |
+-----+-----+
| description | OpenStack Networking |
| enabled | True |
| id | eae54b0547144bd8b54073c47aac5c6c |
| name | neutron |
| type | network |
+-----+-----+
root@controller:~#

```

Gambar 4.75 Menambah *Service Neutron* pada OpenStack

- d. Penulis melakukan instalasi *package-package* kebutuhan layanan *Neutron* untuk *controller node* dengan menggunakan perintah seperti pada Gambar 4.76.

```

root@controller:~# apt-get install neutron-server neutron-common \
> neutron-ovn-metadata-agent openvswitch-switch ovn-central

```

Gambar 4.76 Perintah Instalasi *Package-package Neutron*

- e. Penulis menambahkan tiga *endpoint* untuk layanan *Neutron*. *Public endpoint*, *internal endpoint*, dan *admin endpoint*. Pada Gambar 4.77 menunjukkan perintah pembuatan *endpoint* layanan *Neutron*, dan Gambar 4.78 menunjukkan outputnya.

```

root@controller:~# openstack endpoint create --region RegionOne network public http://10.10.0.250:9696 && \
openstack endpoint create --region RegionOne network internal http://10.10.0.250:9696 && \
openstack endpoint create --region RegionOne network admin http://10.10.0.250:9696

```

Gambar 4.77 Perintah Pembuatan *Endpoint Layanan Neutron*

Field	Value
enabled	True
id	6e61523d04044bba90150339e26a7add
interface	public
region	RegionOne
region_id	RegionOne
service_id	65ad14e29c674506bb9a1d2bf75a479b
service_name	neutron
service_type	network
url	http://10.10.0.250:9696

Field	Value
enabled	True
id	99a6225113354995a5e03b7133f05c9d
interface	internal
region	RegionOne
region_id	RegionOne
service_id	65ad14e29c674506bb9a1d2bf75a479b
service_name	neutron
service_type	network
url	http://10.10.0.250:9696

Field	Value
enabled	True
id	06fedd6ac3264f638203790476d901fc
interface	admin
region	RegionOne
region_id	RegionOne
service_id	65ad14e29c674506bb9a1d2bf75a479b
service_name	neutron
service_type	network
url	http://10.10.0.250:9696

Gambar 4.78 Output Pembuatan *Endpoint* Layanan *Neutron*

- f. Penulis mengganti nama *file* konfigurasi */etc/neutron/neutron.conf* menjadi */etc/neutron/neutron.conf.backup* untuk *backup file* konfigurasi bawaan dan membuat *file* baru bernama */etc/neutron/neutron.conf*, kemudian pada *file* tersebut ditambahkan beberapa konfigurasi, diantaranya:

- 1) Pada *section [DEFAULT]*, penulis menambah beberapa baris konfigurasi seperti pada Gambar 4.79.

```
GNU nano 6.2 /etc/neutron/neutron.conf *
[DEFAULT]
core_plugin = neutron.plugins.ml2.plugin.ML2Plugin
service_plugins = ovn-router
transport_url = rabbit://openstack:RABBIT_PASS@10.10.0.250
auth_strategy = keystone
notify_nova_on_port_status_changes = true
notify_nova_on_port_data_changes = true
```

Gambar 4.79 Konfigurasi Layanan *Neutron* bagian *[DEFAULT]*

- 2) Kemudian, penulis mengonfigurasi pada *section [database]* untuk mengoneksikan layanan *Neutron* dengan *database* seperti pada Gambar 4.80.

```
[database]
connection = mysql+pymysql://neutron:NEUTRON_PASS@10.10.0.250/neutron
```

Gambar 4.80 Konfigurasi Koneksi *Database Neutron*

- 3) Selanjutnya, penulis menambah konfigurasi pada *section* `[keystone_authtoken]` untuk mengoneksikan layanan *Neutron* dengan layanan *Keystone* seperti pada Gambar 4.81.

```
[keystone_authtoken]
www_authenticate_uri = http://10.10.0.250:5000
auth_url = http://10.10.0.250:5000
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = neutron
password = NEUTRON_PASS
```

Gambar 4.81 Konfigurasi Koneksi *Neutron* dengan *Keystone*

- 4) Berikutnya, penulis menambah konfigurasi pada *section* `[nova]` untuk mengoneksikan layanan *Neutron* dengan layanan *Nova* seperti pada Gambar 4.82.

```
[nova]
auth_url = http://10.10.0.250:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = nova
password = NOVA_PASS
```

Gambar 4.82 Konfigurasi Koneksi *Neutron* dengan *Nova*

- 5) Penulis menambahkan konfigurasi bawaan yang terdapat pada *file* `/etc/neutron/neutron.conf.backup` ke dalam *file* `/etc/neutron/neutron.conf`, yaitu *section* `[oslo_concurrency]` dan `[agent]` berfungsi sebagai *helper* untuk proses layanan *Neutron* seperti pada Gambar 4.83.

```
[oslo_concurrency]
lock_path = /var/lib/neutron/tmp

[agent]
root_helper = "sudo /usr/bin/neutron-rootwrap /etc/neutron/rootwrap.conf"
```

Gambar 4.83 Konfigurasi Bawaan *Neutron*

- g. Selanjutnya, penulis konfigurasi OVN untuk layanan *Neutron* pada *controller node* menggunakan perintah seperti pada Gambar 4.84. OVN pada *controller node* difungsikan sebagai *gateway* untuk *external network* dari VM pada OpenStack.

```

root@controller:~# ovn-nbctl set-connection tcp:6641:192.168.224.195 -- \
set connection . inactivity_probe=60000
root@controller:~# ovn-sbctl set-connection tcp:6642:192.168.224.195 -- \
set connection . inactivity_probe=60000
root@controller:~# ovs-vsctl set open . external-ids:ovn-remote=tcp:192.168.224.195:6642
root@controller:~# ovs-vsctl set open . external-ids:ovn-encap-type=geneve
root@controller:~# ovs-vsctl set open . external-ids:ovn-encap-ip=192.168.224.195
root@controller:~# ovs-vsctl add-br br-provider -- add-port br-provider ens160
root@controller:~# ovs-vsctl set open . external-ids:ovn-bridge-mappings=provider:br-provider
root@controller:~# ovs-vsctl set open . external-ids:ovn-cms-options=enable-chassis-as-gw
root@controller:~#

```

Gambar 4.84 Perintah Konfigurasi OVN

- h. Penulis mengganti nama *file konfigurasi ml2_conf.ini* yang terletak di */etc/neutron/plugins/ml2/* menjadi *ml2_conf.ini.backup* untuk *backup file* konfigurasi bawaan dan membuat file baru dengan nama file *ml2_conf.ini* dengan ditambahkan konfigurasi seperti pada Gambar 4.85.

```

GNU nano 6.2 /etc/neutron/plugins/ml2/ml2_conf.ini *
[ml2]
type_drivers = flat,geneve
tenant_network_types = geneve
mechanism_drivers = ovn
extension_drivers = port_security
overlay_ip_version = 4

[ml2_type_flat]

flat_networks = provider

[ml2_type_geneve]

vni_ranges = 1:65536
max_header_size = 38

[securitygroup]
firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver
enable_security_group = true
enable_ipset = true

[ovn]
ovn_nb_connection = tcp:192.168.224.195:6641
ovn_sb_connection = tcp:192.168.224.195:6642
ovn_l3_scheduler = leastloaded
ovn_metadata_enabled = true

```

Gambar 4.85 Konfigurasi File ML2 Plugin

- i. Penulis menambah konfigurasi untuk OVN *metadata agent layanan Neutron* pada *file /etc/neutron/neutron_ovn_metadata_agent.conf*. Konfigurasi ditambahkan seperti pada Gambar 4.86.

```

GNU nano 6.2 /etc/neutron/neutron_ovn_metadata_agent.ini *
[DEFAULT]
nova_metadata_host = 10.10.0.250
metadata_proxy_shared_secret = METADATA_SECRET

[ovn]
ovn_nb_connection = tcp:192.168.224.195:6641
ovn_sb_connection = tcp:192.168.224.195:6642
ovn_metadata_enabled = True

```

Gambar 4.86 Konfigurasi *Metadata Agent* Layanan *Neutron*

- j. Penulis menambah konfigurasi pada file `/etc/nova/nova.conf` di section `[neutron]` supaya layanan *Nova* dapat terhubung dengan layanan *Neutron*. Konfigurasi ditunjukkan pada Gambar 4.87.

```

GNU nano 6.2 /etc/nova/nova.conf *

[neutron]
auth_url = http://10.10.0.250:5000
auth_type = password
project_domain_name = default
user_domain_name = default
region_name = RegionOne
project_name = service
username = neutron
password = NEUTRON_PASS
service_metadata_proxy = true
metadata_proxy_shared_secret = METADATA_SECRET

```

Gambar 4.87 Konfigurasi Koneksi *Nova* dengan *Neutron*

Setelah mengonfigurasi layanan *Neutron* pada *controller node*, penulis melakukan konfigurasi OVN untuk layanan *Neutron* pada *compute node*. Berikut langkah pengerjaannya.

- a. Penulis melakukan instalasi *package* OVN *Controller* untuk layanan *Neutron* pada *compute node* dengan menggunakan perintah seperti pada Gambar 4.88.

```

root@controller:~# apt-get install openvswitch-switch ovn-host

```

Gambar 4.88 Perintah Instalasi *Package* OVN *Controler*

- b. Selanjutnya, penulis mengonfigurasi OVN *Controller* pada *compute node* untuk menghubungkan OVN pada *compute node* dengan OVN pada *controller node* menggunakan perintah seperti pada Gambar 4.89.

```

root@controller:~# ovs-vsctl set open . external-ids:ovn-remote=tcp:192.168.224.195:6642
root@controller:~# ovs-vsctl set open . external-ids:ovn-encap-type=geneve
root@controller:~# ovs-vsctl set open . external-ids:ovn-encap-ip=192.168.224.196
root@controller:~#

```

Gambar 4.89 Perintah Konfigurasi OVN *Controller* pada *Controller Node*

Untuk mengecek layanan *Neutron* yang berjalan, dapat menggunakan perintah seperti pada Gambar 4.90. Perintah dieksekusi pada *controller node*.

```
root@controller:~# openstack network agent list
```

ID	Agent Type	Host	Availability Zone	Alive	State	Binary
e0317099-206c-58df-9301-2aa0dadf904c	OVN Metadata agent	controller		:-)	UP	neutron-ovn-metadata-agent
a128fa7f-9d01-4d2a-a38b-e9afe7df4589	OVN Controller Gateway agent	controller		:-)	UP	ovn-controller
5268ff17-c615-4ce4-a804-f73b4a0ebf4e	OVN Controller agent	compute		:-)	UP	ovn-controller

```
root@controller:~#
```

Gambar 4.90 Pengecekan Layanan *Neutron*

4.2.7 Instalasi dan Konfigurasi *Cinder*

Instalasi dan konfigurasi layanan *Cinder* dilakukan pada *controller node* dan *compute node*. Penulis melakukan instalasi dan konfigurasi layanan *Cinder* pada *controller node* terlebih dahulu, berikut langkah pengerjaannya:

- Penulis menambahkan *user* dan *database* untuk layanan *Cinder* di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.91.

```
MariaDB [(none)]> DROP DATABASE cinder;
Query OK, 0 rows affected (0.003 sec)

MariaDB [(none)]> CREATE DATABASE cinder;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'localhost' \
-> IDENTIFIED BY 'CINDER_PASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON cinder.* TO 'cinder'@'%' \
-> IDENTIFIED BY 'CINDER_PASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]>
```

Gambar 4.91 Menambahkan *User* dan *Database* Layanan *Cinder*

- Penulis membuat *user* layanan *Cinder* pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti pada Gambar 4.42. Untuk *password*, penulis memasukkan *value* “*CINDER_PASS*”.

```
root@controller:~# openstack user create --domain default --password-prompt cinder
User Password:
Repeat User Password:

+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | 611505213dcb42178498e1c2d30facf0 |
| name | cinder |
| options | {} |
| password_expires_at | None |
+-----+-----+

root@controller:~# openstack role add --project service --user cinder admin
root@controller:~#
```

Gambar 4.92 Menambah Layanan *Cinder* pada OpenStack

- c. Penulis menambah entitas *service* untuk *Cinder* pada OpenStack, perintah ditunjukkan pada Gambar 4.93.

```
root@controller:~# openstack service create --name cinderv3 \
> --description "OpenStack Block Storage" volumev3
```

Field	Value
description	OpenStack Block Storage
enabled	True
id	054ddf3bf6034ca1983681e8cd5bbdbe
name	cinderv3
type	volumev3

```
root@controller:~# |
```

Gambar 4.93 Menambah *service Cinder* pada OpenStack

- d. Penulis melakukan instalasi *package-package* layanan *Cinder* untuk *controller node* dengan menggunakan perintah seperti pada Gambar 4.94.

```
root@controller:~# apt-get install apt install cinder-api cinder-scheduler|
```

Gambar 4.94 Perintah Instalasi *Package Layanan Cinder*

- e. Penulis menambahkan tiga *endpoint* untuk layanan *Cinder*. *Public endpoint*, *internal endpoint*, dan *admin endpoint*. Pada Gambar 4.95 menunjukkan perintah pembuatan *endpoint* layanan *Cinder*.

```
root@controller:~# openstack endpoint create --region RegionOne \
volumev3 public http://10.10.0.250:8776/v3/%(project_id)s && \
openstack endpoint create --region RegionOne \
volumev3 internal http://10.10.0.250:8776/v3/%(project_id)s && \
openstack endpoint create --region RegionOne \
volumev3 admin http://10.10.0.250:8776/v3/%(project_id)s|
```

Gambar 4.95 Perintah Pembuatan *Endpoint Layanan Cinder*

- f. Penulis melakukan konfigurasi layanan *Cinder* pada file */etc/glance-api.conf*. Pada file tersebut ditambahkan beberapa konfigurasi, diantaranya:
- 1) Menambah konfigurasi pada *section [database]* untuk mengoneksikan layanan *Cinder* dengan *database* seperti pada Gambar 4.96.

```
[database]
connection = mysql+pymysql://cinder:CINDER_PASS@10.10.0.250/cinder
```

Gambar 4.96 Konfigurasi Koneksi *Database Cinder*

- 2) Pada *section [DEFAULT]*, penulis menambahkan baris konfigurasi untuk mengoneksikan layanan *Cinder* dengan RabbitMQ serta menambahkan *auth type Keystone* seperti pada Gambar 4.97.

```
#add to [DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@10.10.0.250
auth_strategy = keystone
```

Gambar 4.97 Konfigurasi Koneksi RabbitMQ dan *Auth Type*

- 3) Penulis menambah konfigurasi pada *section [keystone_authtoken]* untuk menghubungkan layanan *Cinder* dengan layanan *Keystone* seperti pada Gambar 4.98.

```
[keystone_authtoken]
# ...
www_authenticate_uri = http://10.10.0.250:5000
auth_url = http://10.10.0.250:5000
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS
```

Gambar 4.98 Konfigurasi Koneksi *Cinder* dengan *Keystone*

- g. Selanjutnya, penulis melakukan sinkronisasi *database* dengan konfigurasi *Cinder* dengan mengeksekusi perintah seperti pada Gambar 4.99.

```
root@controller:~# su -s /bin/sh -c "cinder-manage db sync" cinder
```

Gambar 4.99 Perintah Sinkronasi *Database Cinder*

- h. Terakhir, penulis me-restart layanan *Cinder* dengan mengetikkan perintah seperti pada Gambar 4.100.

```
root@controller:~# systemctl restart cinder-scheduler.service apache2.service
root@controller:~# |
```

Gambar 4.100 Perintah *Restart Layanan Cinder*

Setelah mengonfigurasi layanan *Cinder* pada *controller node*, penulis melakukan konfigurasi untuk *storage* layanan *Cinder* pada *compute node*. Berikut langkah pengerjaannya:

- a. Penulis melakukan instalasi *package* untuk layanan *Cinder* menggunakan perintah seperti pada Gambar 4.101.

```
root@compute:~# apt-get install lvm2 thin-provisioning-tools \
> cinder-volume tgt|
```

Gambar 4.101 Perintah Instalasi *Package* Layanan *Cinder*

- b. Dalam *controller node*, terdapat 2 *disk* yang disediakan untuk layanan *Cinder*. Untuk mengeceknya, penulis menggunakan perintah “*lsblk /dev/sd**”. Pada Gambar 4.102, terdapat 3 *disk* yang terdapat pada *compute node*. *Disk* /dev/sda digunakan untuk penggunaan sistem operasi, *disk* /dev/sdb dan /dev/sdc disediakan sebagai *storage* untuk layanan *Cinder*.

```
root@compute:~# lsblk /dev/sd*
NAME                                MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda                                  8:0    0   80G  0 disk
├─sda1                               8:1    0    1M  0 part
├─sda2                               8:2    0    2G  0 part /boot
├─sda3                               8:3    0   48G  0 part
│   └─ubuntu--vg-ubuntu--lv         253:0    0   78G  0 lvm /
├─sda4                               8:4    0   30G  0 part
│   └─ubuntu--vg-ubuntu--lv         253:0    0   78G  0 lvm /
├─sda1                               8:1    0    1M  0 part
├─sda2                               8:2    0    2G  0 part /boot
├─sda3                               8:3    0   48G  0 part
│   └─ubuntu--vg-ubuntu--lv         253:0    0   78G  0 lvm /
├─sda4                               8:4    0   30G  0 part
│   └─ubuntu--vg-ubuntu--lv         253:0    0   78G  0 lvm /
├─sdb                                8:16    0  100G  0 disk
└─sdc                                8:32    0  100G  0 disk
root@compute:~#
```

Gambar 4.102 Perintah Mengecek *Disk*

- c. Supaya kedua *disk* tersebut dapat di-*manage* oleh LVM, penulis menambahkan konfigurasi pada file */etc/lvm.conf*. Konfigurasi yang ditambahkan yang bergaris merah pada Gambar 4.103.

```

GNU nano 6.2                               /etc/lvm/lvm.conf

# Configuration section devices.
# How LVM uses block devices.
devices {

    filter = [ "a/sda/", "a/sdb/", "a/sdc/", "r/.*/"]

```

Gambar 4.103 Konfigurasi *Disk* untuk LVM

- d. Kemudian, penulis membuat *physical volume* dari dua disk tersebut menggunakan perintah seperti pada Gambar 4.104. Dan mengecek apakah *physical volume* sudah terbuat atau belum, penulis menggunakan perintah “*pvs*”.

```

root@compute:~# pvcreate /dev/sdb /dev/sdc
Physical volume "/dev/sdb" successfully created.
Physical volume "/dev/sdc" successfully created.
root@compute:~# pvs
PV          VG          Fmt  Attr  PSize   PFree
/dev/sda3   ubuntu-vg   lvm2 a--   <48.00g    0
/dev/sda4   ubuntu-vg   lvm2 a--   <30.00g    0
/dev/sdb          lvm2 ---   100.00g 100.00g
/dev/sdc          lvm2 ---   100.00g 100.00g
root@compute:~# |

```

Gambar 4.104 Pembuatan *Physical Volume*

- e. Kemudian, penulis membuat *volume group* yang bernama *cinder-volumes* dari *physical volume* yang sebelumnya dibuat. Gambar 4.105 menunjukkan perintah dan output dari pembuatan *volume group*.

```

root@compute:~# vgcreate cinder-volumes /dev/sdb
Volume group "cinder-volumes" successfully created
root@compute:~# vgextend cinder-volumes /dev/sdc
Volume group "cinder-volumes" successfully extended
root@compute:~# vgs
VG          #PV #LV #SN Attr   VSize   VFree
cinder-volumes  2  0  0 wz--n- 199.99g 199.99g
ubuntu-vg      2  1  0 wz--n-  77.99g    0
root@compute:~# |

```

Gambar 4.105 Pembuatan *Volume Group*

- f. Selanjutnya penulis menambahkan beberapa konfigurasi layanan *Cinder* pada file */etc/cinder/cinder.conf*, diantaranya:

- 1) Pada *section [DEFAULT]*, penulis menambahkan baris konfigurasi untuk mengoneksikan layanan *Cinder* dengan RabbitMQ, menambahkan *auth type Keystone*, dan mengonfigurasi lokasi layanan *Glance* seperti pada Gambar 4.106.

```
#add ke [DEFAULT]
transport_url = rabbit://openstack:RABBIT_PASS@10.10.0.250
glance_api_servers = http://10.10.0.250:9292
auth_strategy = keystone
```

Gambar 4.106 Konfigurasi Koneksi RabbitMQ & *Glance* dengan *Cinder*

- 2) Penulis menambah konfigurasi pada *section [database]* untuk mengoneksikan layanan *Cinder* pada *compute node* dengan *database*.

```
[database]
connection = mysql+pymysql://cinder:CINDER_PASS@10.10.0.250/cinder
```

Gambar 4.107 Konfigurasi Koneksi *Cinder* dengan *Database*

- 3) Selanjutnya, penulis menambah konfigurasi pada *section [keystone_authtoken]* untuk menghubungkan layanan *Cinder* pada *compute node* dengan layanan *Keystone* seperti pada Gambar 4.108.

```
GNU nano 6.2 /etc/cinder/cinder.conf

[keystone_authtoken]
# ...
www_authenticate_uri = http://10.10.0.250:5000
auth_url = http://10.10.0.250:5000
memcached_servers = 10.10.0.250:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = cinder
password = CINDER_PASS
```

Gambar 4.108 Konfigurasi Koneksi *Cinder* dengan *Keystone*

- 4) Di bagian *[lvm]*, penulis mengonfigurasi *backend LVM* dengan *driver LVM*, *volume-group cinder-volumes*, protokol *iSCSI*, dan layanan *iSCSI* yang sesuai.

```
[lvm]
# ...
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
target_protocol = iscsi
target_helper = tgtadm
```

Gambar 4.109 Konfigurasi *Cinder Backend LVM*

- 5) Selanjutnya, penulis me-restart layanan *Cinder* dengan dependensinya untuk menyimpan konfigurasinya. Gambar 4.110 menunjukkan perintah untuk melakukan *restart* pada layanan *Cinder*.

```
root@compute:~# systemctl restart cinder-volume.service tgt.service
```

Gambar 4.110 Perintah *Restart* layanan *Cinder*

Untuk mengetahui *storage node* untuk layanan *Cinder* sudah berjalan atau tidak, dapat menggunakan perintah seperti pada Gambar 4.111. Perintah tersebut dieksekusi pada *controller node*.

```
root@controller:~# openstack volume service list
```

Binary	Host	Zone	Status	State	Updated At
cinder-scheduler	controller	nova	enabled	up	2023-02-22T19:43:35.000000
cinder-volume	compute@lvm	nova	enabled	up	2023-02-22T19:43:37.000000

Gambar 4.111 Mengecek layanan *Cinder*

4.2.8 Instalasi dan Konfigurasi *Skyline*

Instalasi dan konfigurasi layanan *Skyline* dilakukan pada *controller node*. Berikut langkah-langkah pengerjaan yang penulis lakukan:

- a. Penulis menambah *user* dan *database* untuk layanan *Skyline* di MariaDB. pembuatan *user* dan *database* ditunjukkan pada Gambar 4.112.

```
MariaDB [(none)]> CREATE DATABASE skyline DEFAULT CHARACTER SET \
-> utf8 DEFAULT COLLATE utf8_general_ci;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON skyline.* TO 'skyline'@'localhost' \
-> IDENTIFIED BY 'SKYLINE_DBPASS';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON skyline.* TO 'skyline'@'%' \
-> IDENTIFIED BY 'SKYLINE_DBPASS';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> |
```

Gambar 4.112 Menambahkan *User* dan *Database* Layanan *Skyline*

- b. Penulis membuat *user* layanan *Skyline* pada OpenStack. *User* ditambahkan ke *project* “*service*” dan *role* “*admin*”, seperti pada Gambar 4.32. Untuk *password*, penulis memasukkan *value* “*SKYLINE_PASS*”.

```
root@controller:~# openstack user create --domain default --password-prompt skyline
User Password:
Repeat User Password:
+-----+-----+
| Field | Value |
+-----+-----+
| domain_id | default |
| enabled | True |
| id | c0e0cfc67daf4dbbba6971308a71e9f8 |
| name | skyline |
| options | {} |
| password_expires_at | None |
+-----+-----+
root@controller:~# openstack role add --project service --user skyline admin
root@controller:~#
```

Gambar 4.113 Menambah *User* Layanan *Skyline* pada OpenStack

- c. Supaya dapat menggunakan repositori HTTPS, penulis melakukan instalasi *package* pendukung seperti pada Gambar 4.114.

```
root@controller:~# apt-get install ca-certificates \
> curl gnupg lsb-release
```

Gambar 4.114 Perintah Instalasi *package* pendukung

- d. Selanjutnya, penulis menambah GPG *key* supaya dapat menggunakan repositori *Docker*. Perintah yang dieksekusi seperti pada Gambar 4.115.

```
root@controller:~# mkdir -m 0755 -p /etc/apt/keyrings
root@controller:~# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | gpg
--dearmor -o /etc/apt/keyrings/docker.gpg
File '/etc/apt/keyrings/docker.gpg' exists. Overwrite? (y/N) y
root@controller:~#
```

Gambar 4.115 Menambah GPG *key* untuk repositori *Docker*

- e. Kemudian penulis menambah repositori *Docker* menggunakan perintah seperti pada Gambar 4.116.

```
root@controller:~# echo \
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Gambar 4.116 Menambah Repositori *Docker*

- f. Selanjutnya penulis meng-*update* konfigurasi repositori dengan menggunakan perintah “*apt-get update*” dan melakukan instalasi *Docker* menggunakan perintah seperti pada Gambar 4.117.

```
root@controller:~# apt-get install docker-ce docker-ce-cli containerd.io
```

Gambar 4.117 Perintah Instalasi *Docker*

- g. Penulis men-download Docker image Skyline API Server dari Docker Hub, dan memastikan folder Skyline API Server sudah terbuat. Perintah-perintah tersebut dieksekusi seperti pada Gambar 4.118.

```

root@controller:~# docker pull 99cloud/skyline:latest
latest: Pulling from 99cloud/skyline
b549f31133a9: Pull complete
f3828da355dd: Pull complete
d04a9623a81d: Pull complete
a730a582037b: Pull complete
df891be359a0: Pull complete
1d10db8d8519: Pull complete
Digest: sha256:e1dea7a5072d5e9967ee27bc8242205e06bcfb72900d9ead12c10bb14d38d4f8
Status: Downloaded newer image for 99cloud/skyline:latest
docker.io/99cloud/skyline:latest
root@controller:~# mkdir -p /etc/skyline /var/log/skyline /var/lib/skyline /var/
log/nginx

```

Gambar 4.118 Perintah *Pulling Image*

- h. Kemudian, penulis menambahkan konfigurasi untuk layanan Skyline dengan menambahkan baris untuk koneksi database, lokasi logfile, koneksi dengan Keystone, dan password untuk user layanan Skyline. Konfigurasi yang ditambahkan seperti pada Gambar 4.119.

```

GNU nano 6.2 /etc/skyline/skyline.yaml
default:
  database_url: mysql://skyline:SKYLINE_DBPASS@10.10.0.250:3306/skyline
  debug: true
  log_dir: /var/log/skyline
openstack:
  keystone_url: http://10.10.0.250:5000/v3/
  system_user_password: SKYLINE_PASS

```

Gambar 4.119 Konfigurasi Layanan Skyline

- i. Kemudian, penulis melakukan bootstrap layanan Skyline menggunakan perintah pada Gambar 4.120. Bootstrapping berfungsi untuk membuat database dan menyinkronisasikannya konfigurasi yang telah dilakukan.

```

root@controller:~# docker run -d --name skyline_bootstrap \
> -e KOLLA_BOOTSTRAP="" \
> -v /etc/skyline/skyline.yaml:/etc/skyline/skyline.yaml \
> -v /var/log:/var/log \
> --net=host 99cloud/skyline:latest
60c2e16dbb3fc0c385c72790f66245aa78e56b3416857f81140ccc8876ea8f31
root@controller:~# docker ps -a
CONTAINER ID   IMAGE               PORTS          COMMAND                  CREATED
STATUS        60c2e16dbb3f      99cloud/skyline:latest  "start_service.sh"      6 seconds ago
go Exited (0) 4 seconds ago          skyline_bootstrap
root@controller:~#

```

Gambar 4.120 Bootstrapping Layanan Skyline

- j. Pada gambar Gambar 4.120, *container* memiliki status *Exited* (0) menunjukkan bahwa proses *bootstrapping* selesai. Selanjutnya *container* tersebut dapat dihapus dengan menggunakan perintah “*docker rm -f skyline_bootstrap*”.

```
root@controller:~# docker rm skyline_bootstrap
skyline_bootstrap
root@controller:~#
```

Gambar 4.121 Perintah Menghapus *Container skyline_bootstrap*

- k. Selanjutnya, menjalankan layanan *Skyline* pada *Docker* menggunakan perintah seperti pada Gambar 4.122. Setelah *containter* terbuat, *dashboard* dapat diakses pada URL <http://10.10.0.250:9999/>.

```
root@controller:~# docker run -d --name skyline --restart=always \
> -v /etc/skyline/skyline.yaml:/etc/skyline/skyline.yaml \
> -v /var/log:/var/log \
> --net=host 99cloud/skyline:latest
0df14d0555ad063d22efb3e4fe035bb6d7728371ff5d9df3c786dd6a8262b826
root@controller:~#
```

Gambar 4.122 Perintah Menjalankan *Container Layanan Skyline*

4.3 Pengujian

Pada tahapan ini, penulis melakukan pengujian layanan *private cloud* yang telah dibangun menggunakan OpenStack. Penulis akan menguji layanan dengan melakukan pembuatan *virtual machine* hingga pengujian koneksi.

Sebelum dibuatnya VM pada OpenStack, penulis menyiapkan infrastruktur jaringan dalam OpenStack dengan membuat *external network* dan *internal network* kemudian menghubungkannya dengan *router*.

Dalam pembuatan *external network* pada OpenStack, penulis mengeksekusi perintah “*openstack network create --provider-physical-network provider --provider-network-type flat --external external-network*”. Perintah tersebut memiliki keterangan sebagai berikut:

- physical network* memiliki *value* “*provider*”, *value* tersebut didapat dari konfigurasi *Neutron* dan *bridge mapping* dari OVN.
- Network* yang dibuat bertipe *flat* supaya VM yang berada di dalam *compute node* dapat berkomunikasi dengan jaringan luar dengan melalui *adapter controller node* (ens160).

Gambar 4.123 menunjukkan eksekusi perintah pembuatan *external network* OpenStack beserta outputnya.

```
root@controller:~# openstack network create --provider-physical-network
provider --provider-network-type flat --external external-network
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2023-02-28T04:47:01Z
description	
dns_domain	None
id	5cf03c1c-8724-484e-9be4-2ac1290467f3
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1500
name	external-network
port_security_enabled	True
project_id	36909d2958924e2ba82d50cb52602e87
provider:network_type	flat
provider:physical_network	provider
provider:segmentation_id	None
qos_policy_id	None
revision_number	1
router:external	External
segments	None
shared	False
status	ACTIVE
subnets	
tags	
updated_at	2023-02-28T04:47:01Z

```
root@controller:~#
```

Gambar 4.123 Perintah Pembuatan Membuat *External Network*

Kemudian, penulis membuat *subnet* pada *external network* OpenStack yang sebelumnya dibuat dengan mengeksekusi perintah” *openstack subnet create --network external-network --allocation-pool start=10.10.51.239,end=10.10.51.249 --dns-nameserver 1.1.1.1 --gateway 10.10.51.254 --subnet-range 10.10.51.0/24 external-subnet*”. Perintah tersebut memiliki keterangan sebagai berikut:

- Subnet* untuk *external network* OpenStack memiliki *network address* 10.10.51.0/24 dan *gateway* 10.10.51.254. *Network address* ini harus sama dengan *network* yang tersambung dengan *adapter external network* (ens160) milik *node-node* OpenStack.
- Subnet* memiliki alamat *IP allocation pool* mulai dari 10.10.51.239 sampai 10.10.51.249. Alamat IP ini digunakan oleh *router* OpenStack dan sebagai *floating IP* untuk VM pada OpenStack.
- Subnet* memiliki *dns nameserver* supaya VM pada OpenStack dapat *me-resolve* nama domain seperti google.com.

Gambar 4.124 menunjukkan eksekusi perintah pembuatan *external network* OpenStack beserta outputnya.


```

root@controller:~# openstack subnet create --network external-network \
--allocation-pool start=10.10.51.239,end=10.10.51.249 \
--dns-nameserver 1.1.1.1 --gateway 10.10.51.254 \
--subnet-range 10.10.51.0/24 external-subnet

```

Field	Value
allocation_pools	10.10.51.239-10.10.51.249
cidr	10.10.51.0/24
created_at	2023-02-28T04:54:54Z
description	
dns_nameservers	1.1.1.1
dns_publish_fixed_ip	None
enable_dhcp	True
gateway_ip	10.10.51.254
host_routes	
id	1c4a9c94-0a10-4b99-b584-ce042f5d394e
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	external-subnet
network_id	5cf03c1c-8724-484e-9be4-2ac1290467f3
project_id	36909d2958924e2ba82d50cb52602e87
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2023-02-28T04:54:54Z

```

root@controller:~#

```

Gambar 4.124 Perintah Pembuatan *External Subnet*

Untuk Pembuatan *internal network* dan *internal subnet* OpenStack, penulis menggunakan perintah “*openstack network create internal-network*”. Gambar 4.125 menunjukkan eksekusi perintah beserta outputnya.

```

root@controller:~# openstack network create internal-network

```

Field	Value
admin_state_up	UP
availability_zone_hints	internal
availability_zones	
created_at	2023-02-28T05:56:11Z
description	
dns_domain	None
id	ef37825d-7f9d-4abc-a941-6e2d65ffd320
ipv4_address_scope	None
ipv6_address_scope	None
is_default	False
is_vlan_transparent	None
mtu	1442
name	internal-network
port_security_enabled	True
project_id	36909d2958924e2ba82d50cb52602e87
provider:network_type	geneve
provider:physical_network	None
provider:segmentation_id	49718
qos_policy_id	None
revision_number	1
router:external	Internal
segments	None
shared	False
status	ACTIVE
subnets	
tags	
updated_at	2023-02-28T05:56:11Z

```

root@controller:~#

```

Gambar 4.125 Perintah Pembuatan *Internal Network*

Selanjutnya, ditambahkan *subnet* pada *internal network* yang sebelumnya dibuat, dengan mengeksekusi perintah “*openstack subnet create --network internal-network --subnet-range 172.16.69.0/24 internal-subnet*”. Untuk *internal subnet* penulis menambahkan *network address* 172.16.69.0/24. Gambar 4.126 menunjukkan eksekusi perintah beserta outputnya.

```
root@controller:~# openstack subnet create --network internal-network
--subnet-range 172.16.69.0/24 internal-subnet
```

Field	Value
allocation_pools	172.16.69.2-172.16.69.254
cidr	172.16.69.0/24
created_at	2023-02-28T06:00:05Z
description	
dns_nameservers	
dns_publish_fixed_ip	None
enable_dhcp	True
gateway_ip	172.16.69.1
host_routes	
id	c57b7b3c-d945-4a6a-a752-79cd5fc1b22f
ip_version	4
ipv6_address_mode	None
ipv6_ra_mode	None
name	internal-subnet
network_id	ef37825d-7f9d-4abc-a941-6e2d65ffd320
project_id	36909d2958924e2ba82d50cb52602e87
revision_number	0
segment_id	None
service_types	
subnetpool_id	None
tags	
updated_at	2023-02-28T06:00:05Z

```
root@controller:~#
```

Gambar 4.126 Perintah Pembuatan *Internal Subnet*

Untuk pembuatan *router*, penulis menggunakan perintah “*openstack router create router*”. Gambar 4.127 menunjukan eksekusi perintah beserta outputnya.

```
root@controller:~# openstack router create router
```

Field	Value
admin_state_up	UP
availability_zone_hints	
availability_zones	
created_at	2023-02-28T06:25:09Z
description	
external_gateway_info	null
flavor_id	None
id	ede51102-f961-4c34-a2ec-b865208e270f
name	router
project_id	36909d2958924e2ba82d50cb52602e87
revision_number	0
routes	
status	ACTIVE
tags	
updated_at	2023-02-28T06:25:09Z

```
root@controller:~#
```

Gambar 4.127 Perintah Pembuatan *Router*

Kemudian, penulis menghubungkan *router* dengan *external network* OpenStack yang sebelumnya dibuat dengan menggunakan perintah “*openstack*

`router set router --external-gateway external-network`". Untuk menghubungkan *router* dengan *subnet* pada *internal network*, penulis menggunakan perintah "`openstack router add subnet router internal-subnet`". Perintah-perintah ditunjukkan pada Gambar 4.128.

```
root@controller:~# openstack router set router --external-gateway external-network
root@controller:~# openstack router add subnet router internal-subnet
root@controller:~# openstack router list
```

ID	Name	Status	State	Project
ede51102-f961-4c34-a2ec-b865208e270f	router	ACTIVE	UP	36909d2958924e2ba82d50cb52602e87

```
root@controller:~#
```

Gambar 4.128 Perintah Menghubungkan *Network* dengan *Router*

Penulis melakukan perintah "`openstack router show router | grep external`" untuk mencari informasi alamat IP *external router*. Pada Gambar 4.129 diketahui bahwa *router* memiliki alamat IP *external* 10.10.51.244. Penulis melakukan uji koneksi dengan melakukan *ping* menggunakan *windows terminal* pada laptop penulis menuju alamat IP *router*. Gambar 4.130 menunjukkan bahwa alamat IP *router* dapat terhubung dengan laptop penulis yang berada di luar jaringan.

```
root@controller:~# openstack router show router | grep "external"
| external_gateway_info | {"network_id": "5cf03c1c-8724-484e-9be4-2ac1
290467f3", "external_fixed_ips": [{"subnet_id": "1c4a9c94-0a10-4b99-b584
-ce042f5d394e", "ip_address": "10.10.51.244"}], "enable_snat": true} |
root@controller:~#
```

Gambar 4.129 Perintah Mencari Informasi Alamat IP Router

```
C:\Users\ADMIN\PZA>ping 10.10.51.244

Pinging 10.10.51.244 with 32 bytes of data:
Reply from 10.10.51.244: bytes=32 time=8ms TTL=253
Reply from 10.10.51.244: bytes=32 time=13ms TTL=253
Reply from 10.10.51.244: bytes=32 time=10ms TTL=253
Reply from 10.10.51.244: bytes=32 time=31ms TTL=253

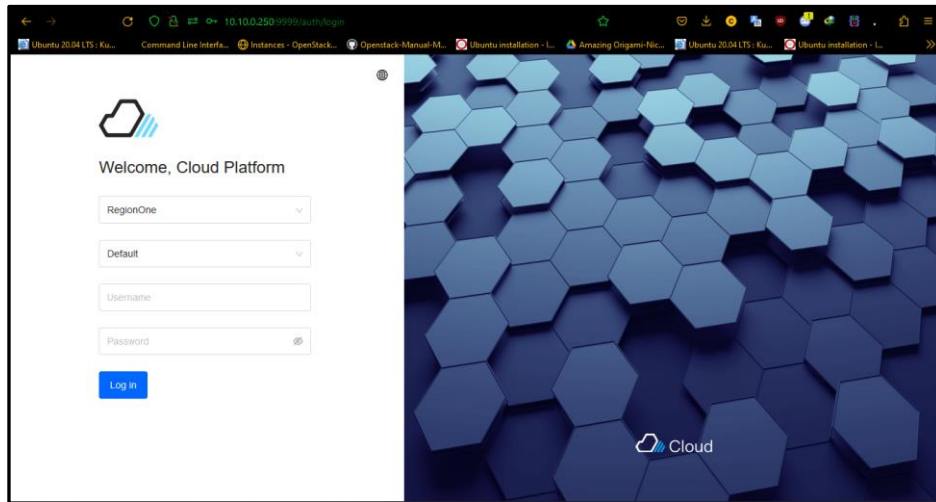
Ping statistics for 10.10.51.244:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 8ms, Maximum = 31ms, Average = 15ms

C:\Users\ADMIN\PZA>
```

Gambar 4.130 Pengujian Koneksi *Router*

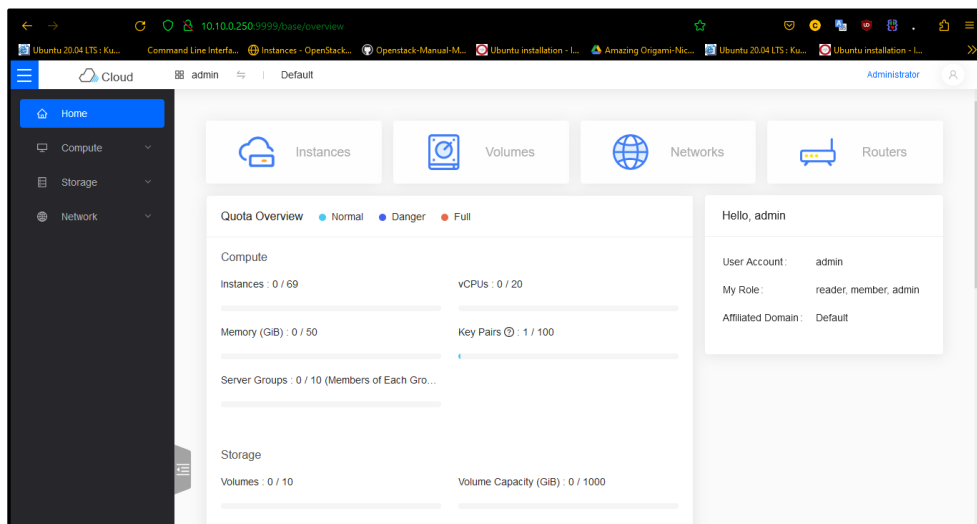
Selanjutnya, penulis melakukan pembuatan VM melalui *Dashboard Service Skyline* dengan mengakses URL `http://10.10.0.250:9999/` pada *web browser*

menggunakan laptop milik penulis. Untuk membuat *virtual machine* pada OpenStack, diperlukan *flavor*, *image*, dan *security group*. Gambar 4.131 menunjukkan tampilan *login* ke *Dashboard* OpenStack.



Gambar 4.131 Tampilan Halaman *Login Dashboard* OpenStack

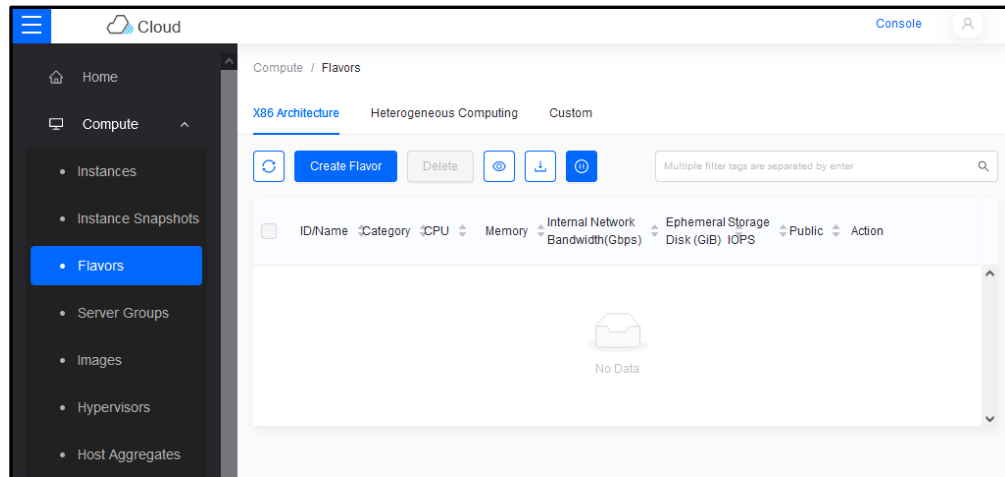
Untuk mengakses halaman *dashboard* OpenStack, penulis memasukkan kredensial admin yang sebelumnya dibuat dengan *username* “admin” dan *password* “ADMIN_PASS”. Setelah *login*, tampilan *dashboard* OpenStack akan seperti pada Gambar 4.132.



Gambar 4.132 Tampilan *Dashboard* OpenStack

Untuk pembuatan *flavor*, penulis mengakses halaman admin dengan menekan tombol yang bertuliskan “Administrator” terletak di pojok kanan atas, lalu

mengakses ke submenu “*Flavors*” pada menu “*Compute*”, kemudian menekan tombol “*Create Flavor*”. Gambar 4.133 menunjukkan halaman *flavor admin*.

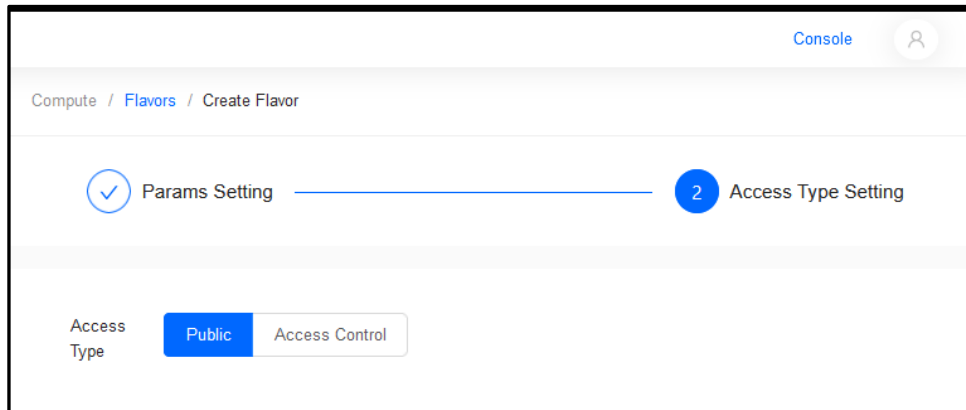


Gambar 4.133 Tampilan Halaman *Flavor Admin*

Penulis mengisi *parameter-parameter flavor* seperti *architecture*, *type*, *name*, jumlah CPU, dan RAM size. Gambar 4.134 menunjukkan pembuatan *flavor*.

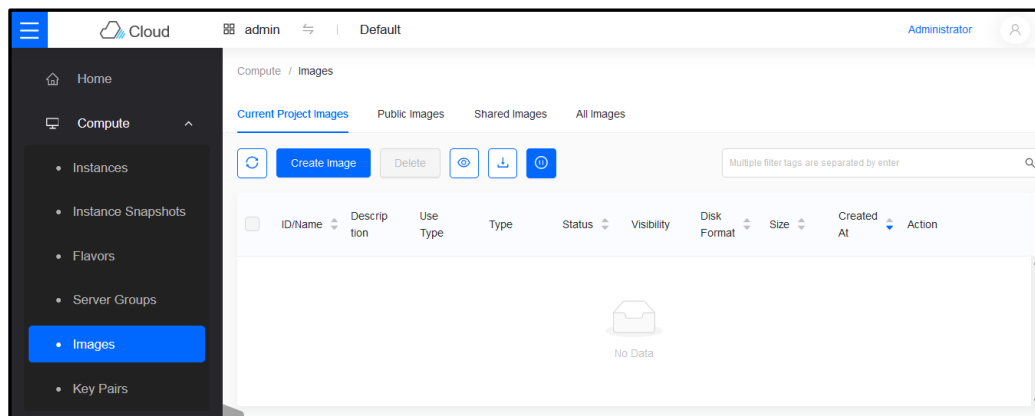
Gambar 4.134 Tampilan Pembuatan *Flavor*

Kemudian, penulis menentukan tipe akses untuk *flavor* yang dibuat. Gambar 4.135 menunjukkan bahwa *flavor* yang dibuat memiliki tipe akses “*public*”.



Gambar 4.135 Menentukan Tipe Akses *Flavor*

Untuk pembuatan *image*, penulis mengakses halaman dashboard utama, mengakses ke submenu “*Images*” pada menu “*Compute*”, kemudian menekan tombol “*Create Image*”. Gambar 4.136 menunjukkan halaman *dashboard* untuk *image*.



Gambar 4.136 Tampilan *Dashboard Image*

Pada pembuatan *image*, penulis menambahkan informasi yang dibutuhkan yaitu nama *image*, *upload type* berupa “*File URL*” dengan memasukkan URL *cloud image* Ubuntu 22.04 yang didapat dari repositori resmi Ubuntu, format *image*, sistem operasi yang akan dibuat, versi dari sistem operasi, dan admin sistem operasi. Penulis mengisi informasi *image* tersebut seperti pada Gambar 4.137.

admin | Default Administrator

Compute / Images / Create Image

* Name: ubuntu-22.04
The name should start with upper letter, lower letter or chinese, and be a string of 1 to 128, characters can only contain "0-9, a-z, A-Z, '-'_()".

Upload Type: Upload File File URL

* File URL: com/jammy/current/jammy-server-cloudimg-amd64.img

* Format: QCOW2 - QEMU image format

* OS: Ubuntu

* OS Version: 22.04

* OS Admin: root
In general, administrator for Windows, root for Linux, please fill by image uploading

Cancel Confirm

Gambar 4.137 Tampilan Pembuatan *Image*

Untuk pembuatan *security group*, mengakses ke submenu “*Security Group*” pada menu “*Network*”, kemudian menekan tombol “*Create Security Group*”. Gambar 4.138 menunjukkan *form* pembuatan *security group*. Pembuatan *security group* dengan mengisi nama *security group*.

Create Security Group

This operation creates a security group with default security group rules for the IPv4 and IPv6 ether types.

* Name: ubuntu-sg
The name should start with upper letter or lower letter, and be a string of 1 to 128, characters can only contain "0-9, a-z, A-Z, '-'_()".

Description: security group Ubuntu

Security Group Quota: 2/10

Security Group Rule Quota: 6/100

Cancel OK

Gambar 4.138 Tampilan Pembuatan *Security Group*

Pada *security group* yang dibuat, penulis menambahkan *rule* untuk mengizinkan komputer yang berada di luar untuk melakukan *ping* kepada VM yang menggunakan *security group* ini dengan mengisi “*Protocol*” dengan value “*Custom ICMP Rule*”. Penulis mengisi informasi pembuatan *rule* seperti pada Gambar 4.139.

Penulis juga menambahkan *rule* untuk mengizinkan SSH seperti pada Gambar 4.140.

Gambar 4.139 Pembuatan *Rule* ICMP pada *Security Group* ubuntu-sg

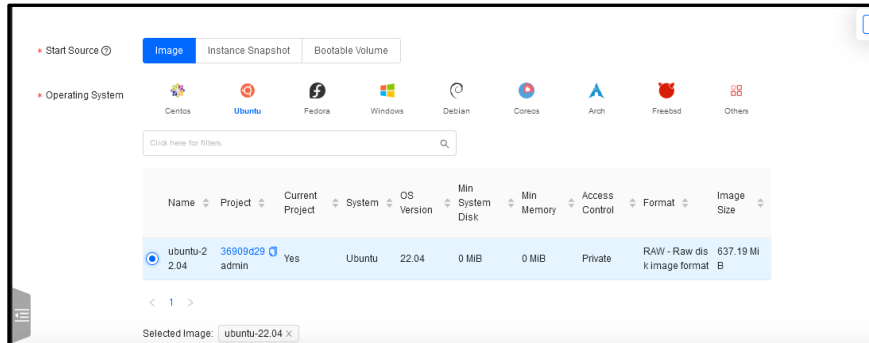
Gambar 4.140 Pembuatan *Rule* SSH pada *Security Group* ubuntu-sg

Penulis memulai pembuatan VM di Openstack, dengan memilih *available zone* dan memilih *flavor* yang sebelumnya dibuat, seperti pada Gambar 4.141.

Name	CPU	Memory	Internal Network Bandwidth(Gbps)	Architecture
m1.medium	2	4.00 GiB	-	X86 Architecture - General Purpose

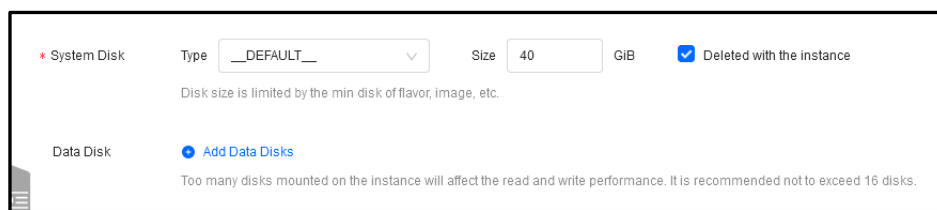
Gambar 4.141 Memilih *Availability Zone* dan *Flavor*

Selanjutnya, penulis memilih Sumber OS yang akan dijalankan yaitu “*image*” dan menentukan *image* yang dipakai, ditunjukkan pada Gambar 4.142.



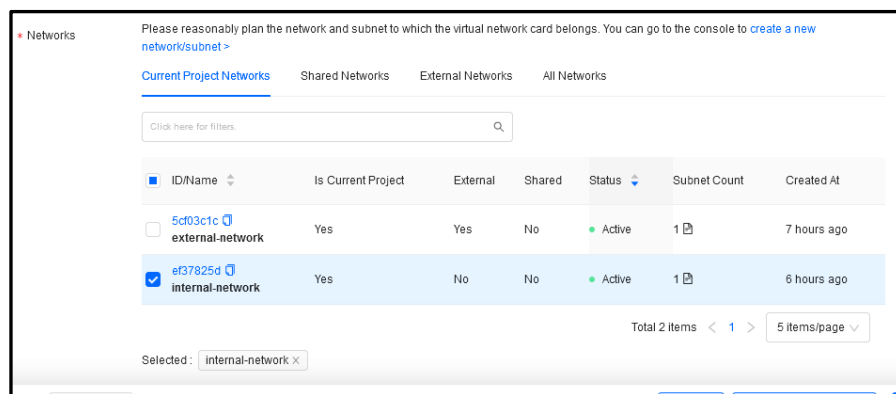
Gambar 4.142 Memilih *Image* untuk VM

Selanjutnya, penulis menentukan kapasitas *Disk* untuk VM yang akan dibuat ditunjukkan pada



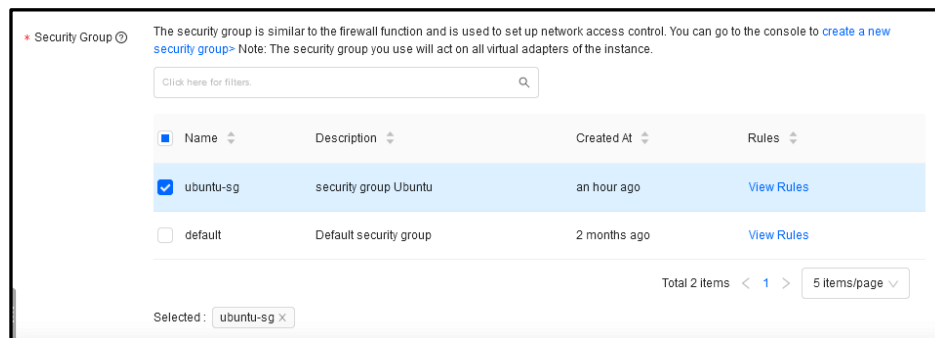
Gambar 4.143 Menentukan Kapasitas *Disk* VM

Selanjutnya, penulis menentukan *network* yang digunakan. Pada Gambar 4.144 menunjukkan pengalokasian *internal network* untuk VM yang dibuat.



Gambar 4.144 Memilih *Network* untuk VM yang akan Dibuat

Selanjutnya, penulis memilih *security group* yang akan digunakan oleh VM yang dibuat. Gambar 4.145 menunjukkan pengalokasian *security group* “ubuntu-sg”.



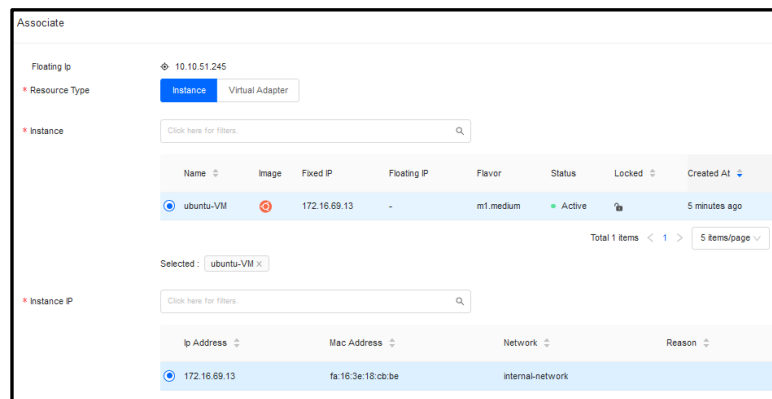
Gambar 4.145 Memilih *Security Group* untuk VM

Selanjutnya, penulis menentukan nama dan *login type* VM yang akan dibuat. Penulis memilih *login type* berupa *password*, dan mengisi *password* untuk *login* kedalam VM, ditunjukkan pada Gambar 4.146.

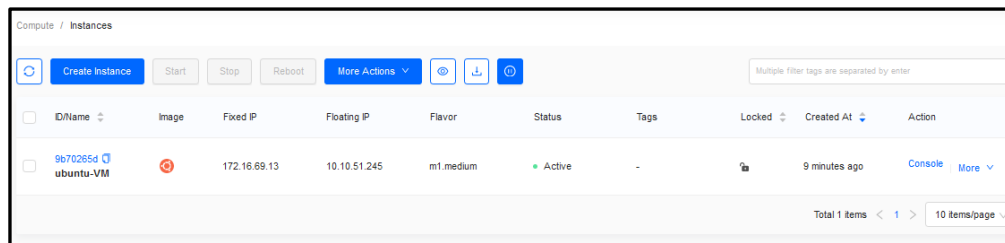
Gambar 4.146 Menentukan Nama dan *Password* VM

Penulis menambahkan *floating IP* supaya VM yang dibuat dapat terhubung dengan jaringan luar. Gambar 4.147 menunjukkan proses pembuatan *floating IP* dan Gambar 4.148 menunjukkan penambahan *floating IP* untuk VM yang sebelumnya dibuat.

Gambar 4.147 Pembuatan *floating IP*

Gambar 4.148 Penambahan *Floating* IP pada VM

Gambar 4.x menampilkan bahwa VM berhasil dibuat dengan status “*active*” dan *floating* IP pun sudah berhasil digunakan oleh VM.



Gambar 4.149 VM Berhasil Dibuat

Untuk menguji VM yang dibuat, penulis melakukan akses *remote* SSH menuju VM di *Windows Terminal*. Sistem Operasi Ubuntu memiliki *default user* yaitu “ubuntu”. Penulis melakukan akses *remote* menggunakan perintah “*ssh ubuntu@10.10.51.245*” dan memasukkan *password* “ubuntu-VM1”, ditunjukkan pada Gambar 4.150.

```
C:\Users\ADMIN\PTA>ssh ubuntu@10.10.51.245
ubuntu@10.10.51.245's password:
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-56-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Mar 1 11:08:47 UTC 2023

System load:  0.080078125   Processes:      99
Usage of /:   4.3% of 38.58GB Users logged in:    1
Memory usage: 6%           IPv4 address for ens3: 172.16.69.13
Swap usage:   0%

0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your I
nternet connection or proxy settings

Last login: Wed Mar 1 11:08:57 2023 from 10.200.1.34
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ubuntu-vm:~$
```

Gambar 4.150 Akses SSH Menuju VM Berhasil Dilakukan

Penulis melakukan pengecekan alamat IP yang dimiliki oleh VM yang dibuat, ditunjukkan pada gambar 4.x. Pada gambar tersebut, ditunjukkan bahwa VM memiliki alamat IP *internal* 172.16.69.13. Alamat IP tersebut sesuai dengan *network* yang dibuat pada *internal subnet*. Untuk *floating* IP pada VM tidak akan muncul karena bersifat *floating*, yang berarti saat ada *traffic data* yang menuju *floating* IP, *traffic* tersebut akan diteruskan oleh OpenStack menuju VM yang sudah di-associate-kan dengan *floating* IP yang dituju.

```
ubuntu@ubuntu-vm:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1442 qdisc fq_codel state UP group
    default qlen 1000
    link/ether fa:16:3e:18:cb:be brd ff:ff:ff:ff:ff:ff
    altname enp0s3
    inet 172.16.69.13/24 metric 100 brd 172.16.69.255 scope global dynamic ens3
        valid_lft 24948sec preferred_lft 24948sec
    inet6 fe80::f816:3eff:fe18:cbbe/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@ubuntu-vm:~$
```

Gambar 4.151 Pengecekan alamat IP dari VM

Penulis melakukan pengujian dengan mengeksekusi perintah “*ping 1.1.1.1*” untuk memastikan bahwa VM yang dibuat dapat terkoneksi dengan *internet*. Pengujian koneksi ditunjukkan pada Gambar 4.152.

```
ubuntu@ubuntu-vm:~$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=55 time=5.63 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=55 time=6.76 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=55 time=6.63 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=55 time=6.87 ms
^C
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3009ms
rtt min/avg/max/mdev = 5.629/6.472/6.870/0.494 ms
ubuntu@ubuntu-vm:~$
```

Gambar 4.152 Pengujian Koneksi dari VM Menuju *Internet*

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan isi dari pembahasan “Rancang Bangun Layanan *Private Cloud* Menggunakan OpenStack”, penulis mendapatkan kesimpulan yaitu:

1. Menerapkan model *private cloud* dapat mengurangi penggunaan perangkat *server* fisik dan jaringan fisik dengan dibuatnya *virtual network* dan *virtual machine* pada *platform* OpenStack.
2. Fleksibilitas dari OpenStack dapat memaksimalkan penggunaan *resource* dari perangkat fisik yang ada, dengan terpusatnya manajemen *resource* perangkat *server* fisik dan *resource* tersebut dapat dialokasikan menjadi *resource virtual* sesuai dengan kebutuhan.

5.2 Saran

Setelah melakukan perancangan, instalasi dan konfigurasi OpenStack serta pengujian yang telah dilakukan penulis, didapatkan beberapa saran yaitu:

1. Gunakan spesifikasi perangkat yang direkomendasikan seperti yang tertera pada dokumentasi OpenStack, supaya layanan OpenStack memiliki performa yang lebih baik.
2. Gunakan *node* yang difungsikan sebagai *storage node* supaya terpisah dengan *compute node*.

DAFTAR PUSTAKA

- Arianto. (2022, Oktober 11). *Mengenal Teknologi Virtualisasi Pada Modern Komputer*. Retrieved from Tembolok ID: <https://www.tembolok.id/mengenal-teknologi-virtualisasi-pada-modern-komputer/>
- Ashari, A., & Setiawan, H. (2011). Cloud Computing : Solusi ICT ? *Jurnal Sistem Informasi (JSI)*, 336-341.
- Astuti, I. K. (2020). Jaringan Komputer. *Fakultas Komputer Universitas Mitra Indonesia*.
- Bouguerra, F. (2021, December 21). *Data centre networking: What is OVN?* Retrieved from Ubuntu: <https://ubuntu.com/blog/data-centre-networking-what-is-ovn>
- Darmawan, A. (2012). Sejarah Internet. *Jurnal Pendidikan Sejarah*, 1.
- Efendi, I. (2016). *Apa Itu Hypervisor ?* Retrieved from IT-JURNAL.COM: <https://www.it-jurnal.com/apa-itu-hypervisor/>
- Fadhilah, R. R. (2013, 6). Definisi Sistem Operasi. *IlmuKomputer.Com*, 1-2. Retrieved from Komunitas eLearning IlmuKomputer.Com: <https://ilmukomputer.org/wp-content/uploads/2013/06/Raihana-Definisi-Sistem-Operasi.pdf>
- Fifield, T., Fleming, D., Gentle, A., Hochstein, L., Proulx, J., Toews, E., & Topjian, J. (2014). *OpenStack Operations Guide*. California: O'Reilly Media.
- Gillis, A., & Posey, B. (2021, March). *What is server virtualization? The ultimate guide*. Retrieved from TechTarget: <https://www.techtarget.com/searchitoperations/definition/hypervisor>
- Hidayat, D. P. (2016). IMPLEMENTASI DAN ANALISA PERBANDINGAN KINERJA VIRTUALISASI SERVER MENGGUNAKAN VMWARE ESXI DAN MICROSOFT HYPER V. *Jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember*, 1.

- IBM. (n.d.). *What is a message broker?* . Retrieved from IBM:
<https://www.ibm.com/topics/message-brokers>
- Khedher, O., & Chowdhury, C. D. (2017). *Mastering OpenStack Second Edition*.
 Birmingham: Packt Publishing Ltd.
- Lawrence, A. (2020, Oktober 24). *API: Pengertian, Fungsi, dan Cara Kerjanya*.
 Retrieved from Niagahoster: <https://www.niagahoster.co.id/blog/api-adalah/>
- MariaDB. (n.d.). *About MariaDB Server*. Retrieved from MariaDB:
<https://mariadb.org/about/>
- Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. *National Institute of Standards and Technology*, 2-3.
- Memcached. (n.d.). *About Memcached*. Retrieved from Memcached:
<https://memcached.org/about/>
- Nugraha, P. S., Mogi, A. I., & Setiawan, I. A. (2015). Implementasi Private Cloud Computing Sebagai Layanan Infrastructure as a Service (IaaS) Menggunakan OpenStack. *Jurnal Ilmiah Ilmu Komputer Universitas Udayana*, Vol. 8, No. 2.
- Oluwademilade, A. (2022, November 18). *What Is Network Time Protocol? Why Is It Important?* Retrieved from Make Us Of:
<https://www.makeuseof.com/what-is-network-time-protocol/>
- OpenStack. (2022, May 24). *OpenStack Documentation*. Retrieved from Placement API: <https://docs.openstack.org/nova/rocky/user/placement.html>
- Pamungkas, D. P., Setiawan, A. B., & Ramadhani, R. A. (2018). *Jaringan Komputer Dasar*. Jombang: CV. Kasih Inovasi Teknologi.
- Prayoga, J. (2021, November 16). *Pengertian Web Server, Fungsi, dan Cara Kerjanya*. Retrieved from Gudang SSL: <https://gudangssl.id/blog/web-server-adalah/>
- Purbo, O. W. (2011). *Pentunjuk Praktis Cloud Computing Menggunakan Open Source*. Jakarta.
- RabbitMQ. (n.d.). *What can RabbitMQ do for you?* Retrieved from RabbitMQ:
<https://www.rabbitmq.com/features.html>

- Riyadi, H. (2022, Juni 12). *Pengertian IP Address Beserta Fungsi dan Kelas IP Address pada Jaringan Komputer*. Retrieved from Nesabamedia: <https://www.nesabamedia.com/pengertian-ip-address-dan-fungsi-ip-address/>
- Saputro, Y. F. (2014). OSI LAYER. *IlmuKomputer.Com*, 1-5.
- Setiawan, I. (2005). Mengenal Logical Volume Manager (LVM). *Jurnal UNPAD*, 1-2.
- Supriyadi, A., & Gartina, D. (2007). Memilih Topologi Jaringan dan Hardware dalam Desain Sebuah Jaringan Komputer. *Informatika Pertanian*, 1037-1039.
- Syafitri, I. (2022, Juni 10). *Pengertian Apache Beserta Fungsi, Kelebihan dan Kekurangan Apache yang Perlu Anda Ketahui*. Retrieved from NesabaMedia: <https://www.nesabamedia.com/pengertian-apache/>
- Trivusi. (2022, September 18). *Mengenal TCP/IP Model pada Jaringan Komputer*. Retrieved from Trivusi: <https://www.trivusi.web.id/2022/08/tcp-ip-model.html>
- Yusnika, K. P. (2013). Model Referensi OSI. *IlmuKomputer.Com*, 1-10.