

ANALISIS EFEKTIFITAS IMPLEMENTASI SISTEM APLIKASI DOCKER TERINTEGRASI OPENSTACK

Iqbal Abadilah Umar¹, Nurhadi², Lailis Syafaah³, Khaeruddin⁴

¹²³ Program Studi Teknik Elektro, Fakultas Teknik, Universitas Muhammadiyah Malang

⁴ Program Studi D3 Teknik Elektronika, Fakultas Teknik, Universitas Muhammadiyah Malang

Jln. Raya Tlogomas, No. 246, Malang 65144, Jawa Timur

¹skiperball@webmail.umm.ac.id, ²nurhadi_ft@umm.ac.id, ³lailis@umm.ac.id, ⁴lamone@umm.ac.id

Abstract

The development of information technology is very much needed in today's era where it is supported by the access and speed of searching for information so that large storage media are also required. *Cloud* computing has a very important role in this development. The problem of the flexibility of a system is still an interesting issue among information technology application developers today. This article presents the implementation of an openstack integrated with docker application system running on a virtual machine system. Docker is a solution for developers who often struggle when developing flexible applications. The first installation and configuration is done on a virtual machine, which is then done with the installation and configuration of Docker on OpenStack. Configure Glance and Nova with docker. The results obtained in this implementation study is that a virtual machine platform utilizing an integrated openstack docker container was successfully built and its effectiveness analyzed in terms of using storage and memory data sources.

Keywords : *openstack, docker container, virtual machine, cloud computing*

Abstrak

Perkembangan teknologi informasi menjadi sangat dibutuhkan pada jaman sekarang di mana didukung dengan akses dan kecepatan mencari informasi sehingga dituntut juga media penyimpanan yang besar. *Cloud* computing memiliki peranan yang sangat penting dalam perkembangan tersebut. Masalah fleksibilitas sebuah sistem masih menjadi isu menarik di kalangan pengembang aplikasi teknologi informasi hingga saat ini. Artikel ini menyajikan tentang implementasi sistem aplikasi *docker* terintegrasi dengan *openstack* yang dijalankan dengan sistem mesin virtual. Docker adalah solusi bagi para pengembang yang sering kesulitan saat mengembangkan aplikasi yang fleksibel. Instalasi dan konfigurasi pertama dilakukan adalah pada mesin virtual, yang selanjutnya dilakukan dengan instalasi dan konfigurasi Docker pada OpenStack. Konfigurasi Glance dan Nova dengan docker. Hasil yang didapat pada penelitian ini adalah sebuah platform mesin *virtual* memanfaatkan *docker container* yang terintegrasi *openstack* berhasil diimplementasikan. Selanjutnya, penelitian ini juga menyajikan efektifitas integrasi docker ke dalam openstack dalam hal penggunaan sumber data penyimpanan dan *memory*.

Kata kunci : *openstack, docker container, virtual machine, cloud computing*

1. PENDAHULUAN

Kemajuan teknologi seperti kecepatan *internet* untuk mengakses data dan informasi secara *online* semakin tangguh dan meningkat. Jika sebuah penyedia situs informasi diakses oleh banyak pengguna atau *client* dengan perangkat

yang kurang memadai dapat mengakibatkan *overload* pada perangkat sehingga *server* yang digunakan mengalami *downserver* yang sangat merugikan banyak kalangan. Maka dengan berkembangnya teknologi juga dituntut untuk mengikuti perkembangan perangkat keras sesuai dengan kebutuhan saat ini. Sehingga inovasi

teknologi informasi saat ini mengarah kepada perangkat lunak untuk melakukan pelayanan. Mayoritas penyedia pelayanan saat ini banyak yang menggunakan teknologi *virtual machine* (VM).

Virtual machine (VM) merupakan implementasi perangkat lunak dari sebuah mesin komputer yang dapat menjalankan program sama seperti layaknya sebuah komputer asli. Pada perkembangan jaman sekarang, *virtual machine* dapat digunakan sebagai simulator perangkat keras walaupun secara fisik tidak ada perangkat keras aslinya sama sekali. Namun dalam VM terdapat *hypervisor*. *Hypervisor* merupakan bagian dari perangkat lunak yang membangun VM. *Hypervisor* sendiri dikenal cukup membebani *hardware*, utamanya untuk *web hosting* yang memiliki spesifikasi tidak besar [1].

Penggunaan *cloud computing* pada era sekarang bukan hal tabu lagi bagi para development, banyak dari pengembang yang memanfaatkan fasilitas yang tersedia dari *openstack*. Pada fasilitas *openstack* memiliki teknologi *Docker Container*. *Docker* adalah sebuah *project open-source* yang menyediakan platform terbuka untuk *developer* maupun *sysadmin* untuk dapat membangun, mengemas, dan menjalankan aplikasi dimanapun sebagai sebuah wadah (*Container*) yang ringan [2]. *Docker* sangat ringan dan cepat jika dibandingkan dengan virtual mesin yang berbasis *hypervisor*, sehingga menjadikan *Docker* sebagai alternatif yang efisien untuk *developertooling* [3]. Sebagai *light-weight virtualization*, *Docker* hampir tidak menambah *overhead* pada mesin *host* [4].

Docker menyatukan beberapa perangkat lunak dalam *file* sistem dan berisi semua yang diperlukan menjalankannya: source code, paket sistem untuk proses runtime, perangkat sistem, sistem pustaka software – apapun yang dapat diinstal pada *server*. Hal ini menjamin bahwa perangkat lunak akan selalu berjalan sama, tidak tergantung pada lingkungannya. Seperti pada penelitian oleh Nugroho dan Kartadie [5] dalam implementasi *docker container* untuk *load balancing web server* menggunakan raspberry pi. Sementara itu, contoh penerapan yang lain adalah pada [6] untuk membangun cluster pada Kubernetes dengan *google cloud*.

Sementara itu, *Openstack* adalah sebuah platform awan yang terdiri dari software *open source* untuk menyediakan basis menjalankan *cloud Infrastructure as a Service* (IaaS), baik pribadi maupun perusahaan yaitu berupa sumber daya untuk komputasi dan penyimpanan data dalam bentuk mesin virtual, contohnya seperti yang dilakukan oleh [7]. *Openstack* juga

diterapkan sebagai *Metal as a Service* (Maas) [8] untuk mendukung layanan multitenant infrastruktur jaringan VM. Pada perkembangannya, *openstack* versi panjang (*extended version*) telah dikenalkan oleh Haifeng Li, dkk sebagai *platform* emulasi *real-time* untuk *delay tolerant network* [9].

Meskipun sudah banyak implementasi teknologi virtualisasi, namun isu fleksibilitas dan integrasi berkelanjutan (*continuous integration*) atau kontinuitas dalam model pengembangan aplikasi teknologi informasi masih menjadi perbincangan menarik hingga saat ini [3], [4]. Dengan berbagai contoh penerapan VM dengan *docker container* di atas, serta layanan *openstack* yang menyediakan basis untuk menjalankan platform awan, pada studi ini, sebuah platform VM memanfaatkan *docker container* yang terintegrasi *openstack* berhasil dibangun dan dianalisis efektifitasnya dalam hal penggunaan sumber data penyimpanan dan memory. Tujuan daripada penelitian ini adalah implementasi dan membuktikan efektifitas teknologi virtualisasi *Docker* terintegrasi dengan *openstack* dalam hal penggunaan *storage* dan *memory*.

Ide penelitian ini sesuai dengan rujukan referensi [3] dan [4] tentang efektifitas *docker* sebagai *container*, dan *openstack* sebagai *node storage* [7] dan [8]. Sehingga melengkapi efektifitas *container* dan *storage*, penelitian ini membuktikan bahwa integrasi *docker* sebagai *container* ke dalam *openstack* sebagai *node storage* berhasil diimplementasikan.

2. TEORI DASAR

2.1 Docker

Sistem pada *Docker* merupakan *project open-source* yang menyediakan platform terbuka dalam bentuk teknologi virtualisasi berbasis *container*, dimana sistem ini ditujukan untuk para pengembang *cloud computing* maupun development karena memiliki kehandalan dan dapat dijalankan aplikasi dimanapun dalam satu *container* yang ringan. *Docker container* ini mirip seperti *virtual machine* namun memiliki sistem yang lebih ringan karena sistem *docker* tidak membawa keseluruhan sistem operasi melainkan berbagi sistem dengan *host* [2].

Docker menyatukan beberapa perangkat lunak dalam filesystem dan berisi semua yang diperlukan menjalankannya: source code, paket sistem untuk proses runtime, perangkat sistem, sistem pustaka software – apapun yang dapat diinstal pada *server*. Hal ini menjamin bahwa perangkat lunak akan selalu berjalan sama, tidak tergantung pada lingkungannya [2].

2.2 Openstack

Openstack adalah sebuah platform awan yang terdiri dari software open source untuk menyediakan basis menjalankan *cloud* IaaS (Infrastructure as a Service), baik pribadi maupun perusahaan yaitu berupa sumber daya untuk komputasi dan penyimpanan data dalam bentuk mesin virtual. Openstack mempunyai kemampuan skalabilitas yang lebih besar dibandingkan kerangka kerja awan lainnya [10].

Openstack tersusun dari beberapa komponen. Adapun komponen-komponen tersebut adalah sebagai berikut:

Nova (Compute Service). Semua kegiatan yang diperlukan untuk mendukung siklus hidup dari instance dalam OpenStack *cloud* yang ditangani oleh Nova. Hal ini membuat Nova sebagai Platform Manajemen yang mengelola sumber daya komputasi, jaringan, otorisasi, dan kebutuhan skalabilitas dari OpenStack *cloud*.

Glance (Image Service) OpenStack Imaging Service adalah salah satu produk dari OpenStack yang digunakan untuk layanan virtual disk images.

Keystone (Identity Service) menyediakan layanan identitas dan akses kebijakan untuk semua komponen dalam keluarga OpenStack. Keystone menerapkan itu di REST-nya sendiri yang berbasis API (Identity API). Keystone menyediakan otentikasi dan otorisasi untuk semua komponen OpenStack. Otorisasi akan memverifikasi apakah pengguna yang terotentikasi memiliki akses ke layanannya yang dia minta atau tidak.

Neutron (Networking Service) adalah salah satu komponen openstack yang menyediakan layanan *cloud* Network as a Service. Neutron menyediakan API yang memungkinkan Anda menentukan konektivitas jaringan dan menangani di awan.

Cinder (Block Storage Service) adalah komponen penyimpanan blok yang lebih analog dengan gagasan tradisional komputer yang dapat mengakses lokasi tertentu pada disk drive serta menyediakan perangkat penyimpanan untuk digunakan dengan instances pada OpenStack.

Horizon (User Interface Service) merupakan suatu layanan user interface dalam infrastruktur Openstack yang memberikan akses visualisasi bagi user dalam menciptakan *cloud*.

3. METODOLOGI PENELITIAN

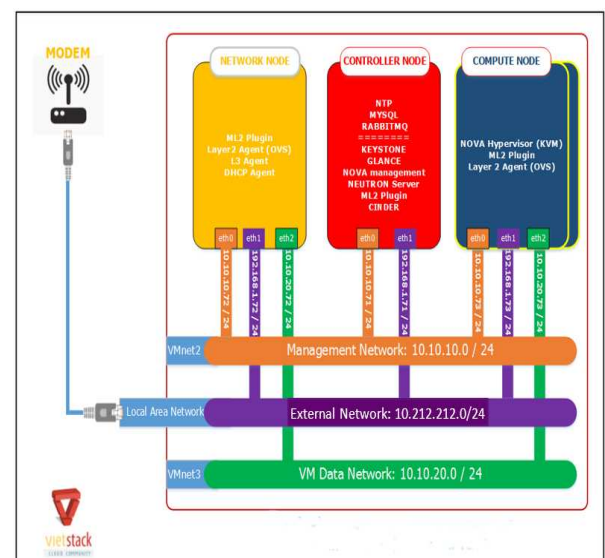
Dalam perancangan sistem ini pembahasannya meliputi proses instalasi, konfigurasi dan integrasi perangkat lunak agar dapat berjalan pada jaringan yang dirancang. Sistem *Docker* terintegrasi dengan *Openstack* yang

dibangun menggunakan *openstack* dirancang menggunakan tiga unit *server* dalam bentuk virtualisasi, di mana tiga unit *server* difungsikan sebagai *server controller node*, *network node*, dan *compute node*. Pada semua *server* menggunakan sistem operasi Ubuntu Server 14.04.

Untuk menginstall *openstack* ada beberapa cara, salah satu diantaranya adalah instalasi dengan *Single Node* atau *multi node*. *Single node* atau sering dikenal dengan AIO VM (All in One VM) yaitu teknik install *openstack* dengan menaruh semua project *openstack* pada satu VM, biasanya untuk proses intallasinya menggunakan *script* otomation (*DevStack*), *devstack* adalah *platform* untuk membuat *cloud openstack* dengan cepat karena menggunakan *script automation* dan dapat diinstall pada VM maupun hardware yang sudah didesain.

Rancangan berikut ini merupakan rancangan pembangunan *system openstack* secara keseluruhan mulai dari proses instalasi sampai dengan penggunaan, dimulai dengan proses konfigurasi interface *Vmware* pada sistem operasi ubuntu yang digunakan sampai dengan proses instalasi *Cinder*, akan disela dengan proses instalasi *openstack* Kilo yang ada pada *server* *Network* dan *Compute*. Setelah proses instalasi konfigurasi *server* *Network* dan *Compute* selesai, proses konfigurasi pada *Controller* akan dilanjutkan lagi dengan instalasi *Horizon* dashboard sampai proses *create instance*.

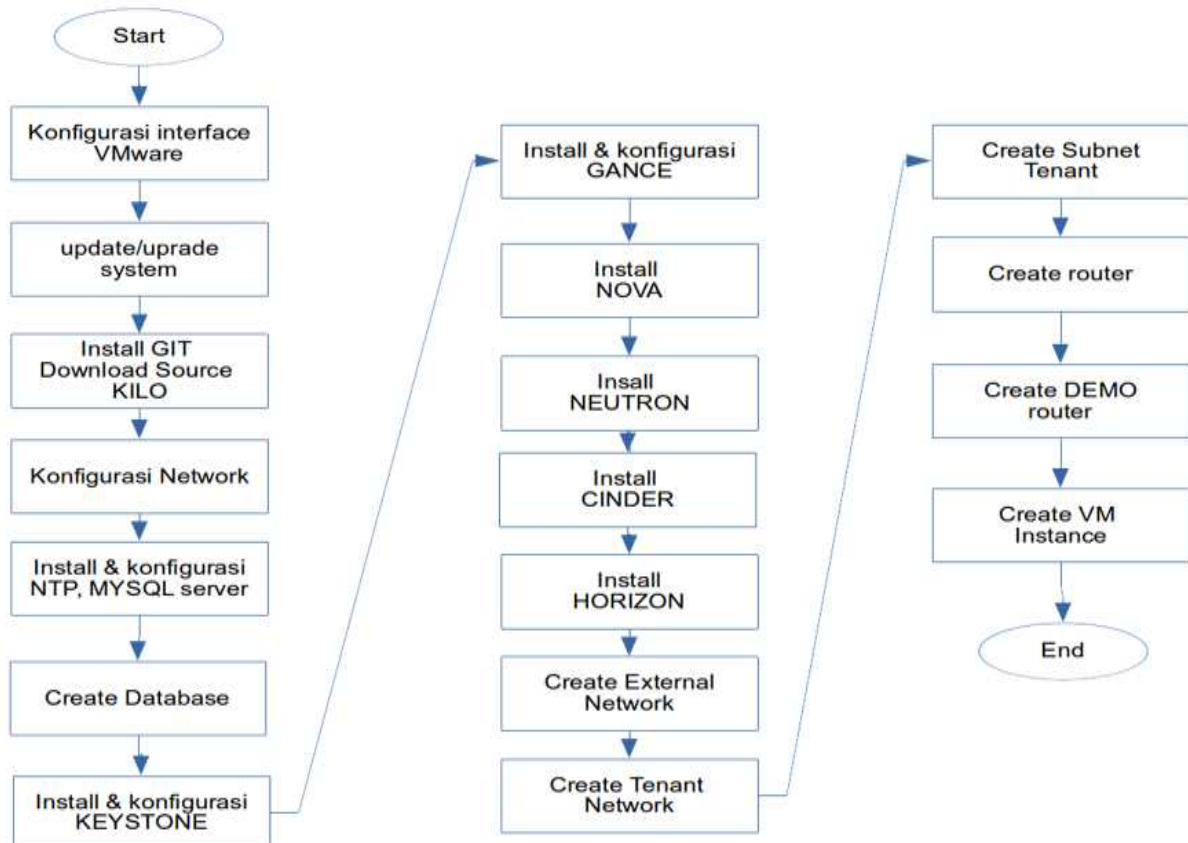
Sistem operasi yang digunakan dalam studi ini adalah *Ubuntu 14.04 server*, selain *open-source*, *Ubuntu server* dipilih karena *familiar* dengan berbagai pengguna ataupun pengembang sistem teknologi informasi.



Gambar 1. Topologi Jaringan Openstack terintegrasi Docker

Topologi jaringan yang digunakan dalam studi ini adalah seperti pada Gambar 1, di mana ada tiga buah *server* yang difungsikan sebagai *server* Openstack, satu *server* difungsikan sebagai *server controller*, satu *server* difungsikan sebagai *server* Network, dan satu *server* sebagai Compute.

Tiga *server* ini merupakan satu kesatuan yang mempunyai peran berbeda, dimana *server* *controller* sebagai pusat kontroller dari openstack, untuk *server* network berfungsi sebagai pengatur jaringan di openstack, untuk *server* computer berfungsi sebagai pengatur penyimpanan pada system openstack.



Gambar 2. Rancangan pembangunan sistem *docker* terintegrasi dengan *openstack*

4. HASIL DAN PEMBAHASAN

Untuk mengetahui instalasi dan konfigurasi yang dilakukan sudah sesuai, maka pengujian keseluruhan sistem dilakukan pertahap, seperti pengujian pada Gambar 3 pada *server controller* berfungsi untuk mengetahui semua fungsi yang ada pada *server* berfungsi dengan semestinya, misalnya *service glance*, *keystone*, *swift*, *cinder*, *neutron* juga untuk mengetahui service-serice ini berjalan pada *server controller*. *Keystone* untuk mengetahui *user* yang terdaftar di *server controller*, *glance* untuk mengetahui list image dalam *server*, *Neutron* untuk mengetahui fungsi ketersediaan virtual LAN.

```

root@controller:~# keystone user-list
+-----+-----+-----+-----+
| id | name | enabled | email |
+-----+-----+-----+-----+
| d13919b207214a94af81645de7a71c23 | admin | True | congth@vietstack.vn |
| fc7b1d6593a144dc907d28b2a29b1b35 | cinder | True | cinder@vietstack.vn |
| 42bad176759943ad080ae65b0e1a77127 | demo | True | congth@vietstack.vn |
| 8bde0197c5054a029b026c9ab60c12 | glance | True | glance@vietstack.vn |
| f1c5a15c73c943438aeebde0d55c4617 | neutron | True | neutron@vietstack.vn |
| e4ba13e6f58740798e2e9983db9e69d5 | nova | True | nova@vietstack.vn |
| 127382cdcd142b6b2bb0f7784956ceb | swift | True | swift@vietstack.vn |
+-----+-----+-----+-----+
root@controller:~#
  
```

Gambar 3. Keystone User List

Setelah melakukan proses pengujian Keystone user-list pada Gambar 4 di mana berfungsi untuk mengecek beberapa user sistem yang terdaftar di *server* controller, terlihat pada pengujian user yang muncul admin, cinder, demo, glance, neutron, nova, swift, masing-masing user melayani servicenya masing-masing. Keystone memberikan layanan identitas (authentication dan authorization) untuk servis *Openstack* lainnya. Keystone juga menyediakan katalog endpoint untuk semua servis *Openstack*.

```
root@controller:~# keystone endpoint-list
```

id	region	publicurl	service_id
0c9ff8d23f9c14ac488f6c3ac5cae81ac	regionOne	http://10.10.10.71:5000/v2.0	ht
1357c055e6b04f18a4d7a1c3e3e5df9	regionOne	http://10.10.10.71:8776/v1/(tenant_ids)	ht
194894d9157941b4880a8a26c2a1eb8	regionOne	http://10.10.10.71:9696	ht
1da32069f1564303b05eccf843cc34d3	regionOne	http://10.10.10.71:9292	ht
74b74b50e194548c9d143120ff9f75	regionOne	http://10.10.10.71:8774/v2/(tenant_ids)	ht
acc20746e49a4051a318f076fa462a9	regionOne	http://10.10.10.71:8776/v2/(tenant_ids)	ht
bcbbdce26584fd39c1d68	regionOne	http://10.10.10.71:8776/v2/(tenant_ids)	ht

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

Gambar 4. Keystone Endpoint List

```
root@controller:~# nova image-list
```

ID	Name	Status	Server
d22ab5bc-600a-4cc0-944d-9ceb31213d65	cirros-0.3.3-x86_64	ACTIVE	

```
root@controller:~# nova network-list
```

ID	Label	Cidr
639ce3c8-652e-4072-8069-dbc3c9c90a27	demo-net	-
ac3503e4-5d4c-4476-bcb5-9d7c24f5e30	ext-net	-

```
root@controller:~# nova service-list
```

Id	Binary	Host	Zone	Status	State	Updated_at	Dis
1	nova-cert	controller	internal	enabled	up	2019-05-07T11:40:43.000000	-
2	nova-consoleauth	controller	internal	enabled	up	2019-05-07T11:40:41.000000	-
3	nova-scheduler	controller	internal	enabled	up	2019-05-07T11:40:41.000000	-
4	nova-conductor	controller	internal	enabled	up	2019-05-07T11:40:41.000000	-
5	nova-compute	compute1	nova	enabled	up	2019-05-07T11:40:42.000000	-

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

Gambar 5. Nova Server Kontroller

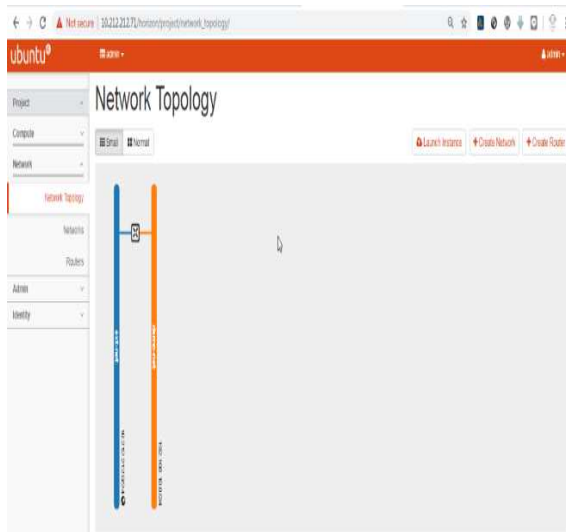
Pengujian proses sebelumnya maka di lanjutkan pengujian dari Horizon yang digunakan untuk menampilkan sistem *openstack* dalam bentuk web iterface, agar mudah dikelola dan atur. Admin tidak direpotkan dengan masuk lagi kedalam terminal cukup mengakses dari web interfacenya saja, pada tugas akhir ini menggunakan ubuntu *openstack* dashboard sebagai template horizonnya sesuai dengan Gambar 6.

Pada proses selanjutnya yaitu pengujian service nova digunakan untuk mengelola *image* dari sistem operasi yang ada pada sistem *openstack*, dapat diupload langsung ke dalam sistem atau juga didownload langsung dari *source* filenya dan hasilnya dapat dilihat pada Gambar 5. Pada sistem yang telah dibuat ada satu sistem operasi didownload langsung source filenya yaitu cirros.

Gambar 6. Openstack login portal

Pada Gambar 7 merupakan tampilan dari *network* yang ada pada sistem, ada dua *network* yang digunakan external *network* dengan *network* (10.212.212.0/24) dan demo *network* dengan *network* (192.168.10.0/24).

Jika model jaringan sudah terkoneksi akan muncul dalam bentuk *network* topology seperti berikut, seperti pada Gambar 7 akan memudahkan dalam mengelola jaringannya karena secara visual kelihatan *topology* jaringannya.



Gambar 7. Tampilan Network Topologi

Setelah pengujian API yang telah dilakukan sebelumnya, maka perlu dilakukan pengujian pengujian pada server Network Node berfungsi untuk mengetahui semua fungsi yang ada pada server, pertama kali yang dilakukan pengujian adalah koneksi antara node server, mulai dengan melakukan pengujian dengan dari node network ke node compute dan node controller.

Selain pengujian *network node*, diperlukan juga pengujian pada server Compute Node pada berfungsi untuk mengetahui semua fungsi yang ada pada server, pertama kali yang dilakukan pengujian adalah koneksi antara node server, mulai dengan melakukan pengujian dengan dari node compute1 ke node network dan node controller.

Pada uji *ping* ada balasan dari node controller dan node network ditandai dengan reply dari protocol ICMP dari node yang dituju. Pengujian fungsi node compute1 yang lain dilakukan dari node controller, karena antara node controller dan node compute1 sudah terkoneksi satu sama lain. Command untuk melakukan pengujian yaitu nova di mana nova ini merupakan fungsi yang ada pada node compute1 yaitu untuk mengatur sistem storage yang ada pada jaringan *openstack*.

Dari pengujian *openstack* juga diperlukan Untuk pengujian pada server docker yang berfungsi untuk mengetahui semua fungsi yang ada pada server yang diinstall dan dikonfigurasi didalamnya ada docker, pertama kali yang dilakukan pengujian adalah melihat images sistem operasi maupun container yang terdownload dan tersimpan dengan mengetikkan *docker images* seperti pada Gambar 8.

```
root@compute1:/KILO-U14.04# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
busybox	latest	3d756b52deb2	11 weeks ago	1.199 MB
hello-world	latest	cc813f1ee102	5 months ago	1.84 kB
rastasheep/ubuntu-sshd	latest	afae1982680a	13 months ago	234.3 MB
rastasheep/ubuntu-sshd	14.04	3e8ac31c1694	13 months ago	286.1 MB
ubuntu-upstart	latest	2182e291d511	3 years ago	253.3 MB
larsks/thttpd	latest	2da2b5d59824	4 years ago	1.058 MB

Gambar 8. Server Docker

```
root@compute1:/KILO-U14.04# docker run -it -d ubuntu-upstart
0f3c873b2c32454ba41f1f4b09abcc2c922d1a0001f0f276b89e8a9343a3f473
root@compute1:/KILO-U14.04#
root@compute1:/KILO-U14.04# docker run -it -d larsks/thttpd
d548e7f176881702000e5bc915a28a4db2d478efbc08bf76e3626c21a5f6846c
root@compute1:/KILO-U14.04#
root@compute1:/KILO-U14.04# docker run -it -d hello-world
5f886c949eb0667ed5d25a6973c00b2cd34ad04880e25d05c96ff04600c9e171
root@compute1:/KILO-U14.04#
root@compute1:/KILO-U14.04# docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
d548e7f17688	larsks/thttpd	"/thttpd -D -l /dev/s"	25 seconds ago	Up 23 seconds
0f3c873b2c32	ubuntu-upstart	"/sbin/init"	42 seconds ago	Up 40 seconds
22/tcp	fervent_kare			

```
root@compute1:/KILO-U14.04#
root@compute1:/KILO-U14.04#
```

Gambar 9. Container Docker

```
root@compute1:/KILO-U14.04# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
5f886c949eb0	hello-world	"/hello"	30 seconds ago	Exited (0)
4548e7f17688	larsks/hthttpd	"/hthttpd -D -l /dev/s"	46 seconds ago	Up 44 sec
0f3c873b2c32	ubuntu-upstart	"/sbin/init"	About a minute ago	Up About
abf67c457f9c	rastasheep/ubuntu-sshd	"/usr/sbin/sshd -D"	8 hours ago	Exited (1)
870f1ea45fdc	larsks/hthttpd	"/hthttpd -D -l /dev/s"	8 hours ago	Exited (1)
d994c1e9456c	hello-world	"/hello"	9 hours ago	Exited (0)

```
root@compute1:/KILO-U14.04#
```

Gambar 10. Container Docker Running

```
root@compute1:~# docker ps -q | xargs docker stats --no-stream
```

CONTAINER	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O
0f3c873b2c32	0.00%	6.918 MB / 763.7 MB	0.91%	3.45 kB / 648 B
968f69700325	0.00%	162.3 MB / 763.7 MB	21.25%	1.338 kB / 648 B
9c2d61c9d40a	0.00%	20.5 MB / 763.7 MB	2.68%	26.02 MB / 521.7 k
4548e7f17688	1.60%	165.6 MB / 763.7 MB	21.68%	2.154 kB / 648 B

```
root@compute1:~#
```

Gambar 11. Monitoring RAM Container

Pada hasil yang terlihat adalah beberapa *images* ataupun *container docker* seperti *busybox*, *hello-world*, *ubuntu*, dan lainnya. *Images/container* yang ada dijalankan dengan command *docker run -it -d <nama images/container>*, setelah *container* dijalankan diverifikasi dengan perintah *docker ps* untuk memastikan apakah *container* sudah berjalan atau tidak terlihat hal tersebut pada Gambar 9.

Terlihat *container* yang sudah dijalankan sesuai dengan yang diketikkan pada command *docker run*, seperti *container ubuntu-upstart* dan *larsks/httpd*. Untuk melihat semua *container* yang berjalan dapat dengan mengetikkan *docker ps -a*, akan menampilkan semua *container* yang sedang berjalan seperti pada Gambar 10.

Setelah beberapa *container* dijalankan dan dilakukan proses login serta beberapa konfigurasi dilakukan pengujian penggunaan RAM, CPU dan *network interface*. Pada Gambar 10 terlihat ada empat *container* yang dijalankan yang dijadikan sample, terlihat penggunaan RAM pada masing-masing *container* bervariasi sesuai dengan seberapa banyak aktifitas yang terjadi pada *container* tersebut.

Dari keseluruhan sistem *docker* juga dapat dilakukan pengujian terhadap *docker daemon*. Berikut adalah penggunaan secara keseluruhan RAM, CPU maupun *storage* pada *server*, di mana penggunaan *docker daemon* cenderung lebih sedikit dalam menggunakan *sources* baik itu RAM maupun CPU terlihat seperti pada Gambar 12.

```

CPU: 5.2% Tasks: 44, 93 thr: 1 running
Mem: 1502/722MB Load average: 0.72 0.48 0.25
Swap: 22/1023MB Uptime: 02:46:18

```

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
5734	root	20	0	25904	3748	3088	R	3.3	0.5	0:01.40	htop
4264	root	20	0	173M	156M	324	S	1.3	21.4	0:27.65	/hthttpd -D -l /dev/stderr
4607	root	20	0	173M	144M	324	S	0.7	19.8	0:46.75	/hthttpd -D -l /dev/stderr
1836	root	10	-10	5776	3632	2500	S	0.7	0.5	0:02.25	/usr/sbin/iscsid
1013	root	20	0	703M	17688	5228	S	0.0	2.4	0:39.52	/usr/bin/docker daemon
2387	root	10	-10	237M	32128	6568	S	0.0	4.3	0:05.00	ovs-vsctchd unix:/var/run/openvswi
2338	root	10	-10	237M	32128	6568	S	0.0	4.3	0:10.54	ovs-vsctchd unix:/var/run/openvswi
4963	root	20	0	200M	10512	4412	S	0.0	1.4	0:00.46	docker exec -it 9c2d61c9d40a bash
4950	root	20	0	200M	10512	4412	S	0.0	1.4	0:01.32	docker exec -it 9c2d61c9d40a bash
1	root	20	0	33784	3888	2496	S	0.0	0.5	0:28.85	/sbin/init
425	root	20	0	19488	1792	1632	S	0.0	0.2	0:00.75	upstart-udev-bridge --daemon
445	root	20	0	52660	3664	2344	S	0.0	0.5	0:15.29	/lib/systemd/systemd-udev --daemon
468	root	20	0	15288	116	0	S	0.0	0.0	0:00.47	upstart-file-bridge --daemon
503	syslog	20	0	249M	1952	1940	S	0.0	0.3	0:00.37	rsyslogd
504	syslog	20	0	249M	1952	1940	S	0.0	0.3	0:00.05	rsyslogd
505	syslog	20	0	249M	1952	1940	S	0.0	0.3	0:00.31	rsyslogd
501	syslog	20	0	249M	1952	1940	S	0.0	0.3	0:00.76	rsyslogd
507	messagebu	20	0	39252	2240	2060	S	0.0	0.3	0:00.54	dbus-daemon --system --fork
526	root	20	0	43464	3164	2796	S	0.0	0.4	0:01.35	/lib/systemd/systemd-logind
895	root	20	0	15272	204	0	S	0.0	0.0	0:00.36	upstart-socket-bridge --daemon
1320	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.85	/usr/bin/docker daemon
1334	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.10	/usr/bin/docker daemon
1575	root	20	0	703M	17688	5228	S	0.0	2.4	0:01.44	/usr/bin/docker daemon
2263	root	20	0	703M	17688	5228	S	0.0	2.4	0:01.92	/usr/bin/docker daemon
2740	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.14	/usr/bin/docker daemon
2855	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.00	/usr/bin/docker daemon
4223	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.01	/usr/bin/docker daemon
4224	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.23	/usr/bin/docker daemon
4332	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.20	/usr/bin/docker daemon
4337	root	20	0	703M	17688	5228	S	0.0	2.4	0:00.88	/usr/bin/docker daemon

Gambar 12. Docker Daemon

5. Kesimpulan dan Saran

Pada penelitian ini, implementasi openstack terintegrasi docker telah berhasil dijalankan. Meskipun dalam skala yang relatif kecil, yakni membangun teknologi virtualisasi berbasis mesin virtual, efektifitas docker dan openstack sebagai solusi teknologi virtualisasi yang efektif

bagi pengembang. Efektifitas dan fleksibilitas itu terlihat pada implementasi dan pengujian yang dilakukan, bahwa informasi penggunaan media penyimpanan dan penggunaan memory secara *real-time*. Hasil pengujian juga menunjukkan bahwa penggunaan media penyimpanan dan penggunaan memory pada docker yang terintegrasi openstack tidak memerlukan *resources* yang besar. Berturut-turut menunjukkan beban penggunaan sebesar 0.91%, 21.25%, 2.68%, dan 21.68% untuk docker container server, dan tiga nodes server di belakangnya.

Ke depan, implementasi docker dan openstack sebagai alternatif solusi teknologi virtualisasi yang efektif perlu dikembangkan dan diimplementasi pada skala yang lebih besar. Dan dapat diujikan pada jaringan yang sesungguhnya untuk membuktikan efektifitas dan fleksibilitas sistem virtualisasi docker.

6. UCAPAN TERIMA KASIH

Penulis dan tim peneliti mengucapkan terima kasih kepada Fakultas Teknik Universitas Muhammadiyah Malang (FT UMM), atas dukungan pendanaan melalui skema penelitian Pusat Kajian dan Rekayasa (Puskareka) FT UMM tahun 2020-2021. Juga kepada Laboratorium Jaringan Komputer Program Studi Teknik Elektro UMM atas dukungan fasilitas sehingga terlaksana kegiatan penelitian ini.

Daftar Pustaka:

- [1] M. I. Djomi, R. Munadi, and R. M. Negara, "Analisis Performansi Layanan FTP dan Video Streaming berbasis Network Function Virtualization menggunakan Docker Containers," *ELKOMIKA J. Tek. Energi Elektr. Tek. Telekomun. Tek. Elektron.*, vol. 6, no. 2, p. 180, 2018.
- [2] S. Goasguen, *Docker Cookbook*. O'Reilly Media, Inc., 2015.
- [3] A. M. Joy, "Performance comparison between Linux containers and virtual machines," *Conf. Proceeding - 2015 Int. Conf. Adv. Comput. Eng. Appl. ICACEA 2015*, pp. 342–346, 2015.
- [4] M. Raho, A. Spyridakis, M. Paolino, and D. Raho, "KVM, Xen and Docker: A performance analysis for ARM based NFV and cloud computing," *Adv. Information, Electron. Electr. Eng. AIEEE 2015 - Proc. 2015 IEEE 3rd Work.*, pp. 3–10, 2015.
- [5] M. A. Nugroho and R. Kartadie, "Analisis Kinerja Penerapan Container untuk Load Balancing Web Server," *JUPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 1, no. 02, pp. 7–15, 2016.
- [6] M. A. Nugroho, "Analisis Cluster Container Pada Kubernetes Dengan Infrastruktur Google Cloud Platform," *JUPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 3, no. 2, pp. 84–93, 2018.
- [7] B. Arifwidodo, W. R. Baskoro, J. Gusti, and A. Ginting, "Video Conference Pada Openstack Menggunakan Openmeeting Sebagai Infrastructure As A Service (IaaS)," *TECHNO*, vol. 21, no. 2, pp. 99–110, 2020.
- [8] J. R. Panggabean, A. B. Prasetyo, and E. D. Widiyanto, "Layanan Infrastruktur Komputasi Multitenant dengan OpenStack di Lingkungan MaaS," *J. Teknol. dan Sist. Komput.*, vol. 5, no. 4, p. 142, 2017.
- [9] H. Li, H. Zhou, H. Zhang, B. Feng, and W. Shi, "EmuStack: An OpenStack-Based DTN Network Emulation Platform (Extended Version)," *Mob. Inf. Syst.*, vol. 2016, 2016.
- [10] J. Castro León, "Advanced features of the CERN OpenStack Cloud," *EPJ Web Conf.*, vol. 214, p. 07026, 2019.