Eksplorasi Pemanfaatan Docker untuk Mempermudah Pengelolaan Instalasi Komputer di Laboratorium Komputer Teknik Informatika Universitas Kristen Petra

Adi Kurniawan[1], Henry Novianus Palit [2], Justinus Adjarwirawan[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-mail: adi_kurnia_wan@yahoo.com[1], hnpalit@petra.ac.id[2], justin@petra.ac.id[3]

ABSTRAK

Docker adalah sebuah aplikasi yang berbasiskan teknologi *open source* yang memungkinkan *developer* atau siapapun untuk membuat, menjalankan, melakukan percobaan dan meluncurkan aplikasi didalam sebuah *container*. Docker membuat proses pemaketan aplikasi bersama komponennya secara cepat dalam sebuah kontainer yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal tanpa melakukan perbuahan konfigurasi pada kontainer. Docker juga sangat ringan dan cepat jika dibandingkan dengan mesin virtual yang berbasis *hypervisor*.

Jurusan Teknik Informatika Universitas Kristen Petra memiliki jumlah mahasiswa yang cukup banyak, yang jumlahnya ratusan, sedangkan jumlah komputer yang ada di laboratorium Jurusan Teknik Informatika jumlahnya tidak sebanyak mahasiswa tersebut. Dengan jumlah yang tidak sebanding ini, maka beberapa mahasiswa memakai komputer yang sama bergantian. Hal ini yang membuat mahasiswa tidak bisa nyaman bekerja pada 1 komputer, dikarenakan pengkonfigurasian komputer pada setiap pelajaran berbeda. Oleh karena itu, dilakukan implementasi aplikasi Docker pada laboratorium.

Implementasi menggunakan Docker dan sistem penyimpanan RAID dan setelah implementasi, pengujian dilakukan dengan cara mengukur kecepatan *transfer* docker *image*. Data yang dihasilkan digunakan untuk pengembangan sistem *container* dan mengevaluasi kecepatan antara sistem penyimpanan RAID yang digunakan.

Kata Kunci: *Container, Virtual Machine,* Docker, *storage,* RAID.

ABSTRACT

Docker is an application that is based on open source technology that allows developers or anyone to create, run, do experiments and launch application in a container. Docker make the process of packaging application components together quickly in an isolated container, so it can run in local infrastructure without changing configuration on the container. Docker also very light and fast when compared to hypervisor-based virtual machine.

Department of Information Engineering of Petra Christian University has a considerable number of students, which the number usually hundreds, while the number of computers in the laboratory of Department of Information number is not as much as the students. With this amount is not comparable, then some students work in the same computer alternately. It makes the students cannot be comfortable working on one computer, because configuring a computer on every lesson is different. Therefore, implementation of Docker application carried out in the laboratory.

Docker implementation and RAID storage system and after implementation, testing was done by measuring the speed of docker image transfer. The resulting data were used to development of container system and evaluate the speed of RAID storage system are used.

Keywords: *Containers, Virtual Machine, Docker, storage, RAID.*

1. PENDAHULUAN

Docker merupakan sebuah platform terbuka untuk developer, yang berguna untuk membangun, mendistribusikan dan menjalankan aplikasi dimanapun. Docker membuat proses pemaketan aplikasi bersama komponennya secara cepat dalam sebuah kontainer yang terisolasi, sehingga dapat dijalankan dalam infrastruktur lokal tanpa melakukan perbuahan/konfigurasi lagi pada kontainer. Docker juga sangat ringan dan cepat jika dibandingkan dengan mesin virtual yang berbasis *hypervisor*.

Jurusan Teknik Informatika Universitas Kristen Petra memiliki banyak sekali mahasiswa yang jumlahnya ratusan, sedangkan komputer yang ada di laboratorium jumlahnya tidak sebanyak mahasiswa. Dengan jumlah yang tidak sebanding ini maka setiap 1 komputer dapat dipakai oleh beberapa mahasiswa. Dengan cara berbagi komputer ini, seorang mahasiswa tidak bisa dengan aman menyimpan data dalam komputer tersebut, karena orang lain dapat membaca, mengedit dan menghapus.

Untuk jurusan Teknik Informatika memiliki laboratorium sendiri, yang setiap pergantian semester, tiap-tiap komputernya di install ulang kembali, karena untuk membersihkan data yang sudah tidak terpakai secara bersih. Hal ini sangat membutuhkan waktu lama untuk menginstal ulang satu persatu komputer dan menginstal aplikasinya.

Melihat kemampuan docker yang ringan ini, maka docker memiliki potensial untuk membantu meningkatkan keamanan data yang dimiliki oleh mahasiswa dan mempermudah kinerja asisten lab. Dengan cara menyediakan *virtual machine* untuk masing-masing mahasiswa, tiap mahasiswa dapat dengan nyaman menyimpan data-data mereka, dan untuk asisten lab tidak harus menginstal ulang tiap-tiap komputer yang ada di labnya setiap semester.DASAR TEORI

2.1. Docker

Docker adalah sebuah *platform* terbuka untuk siapapun yang bertujuan menggunakan sebuah *platform* untuk membangun,

mendistribusikan dan menjalankan aplikasi dimanapun seperti laptop, *data center*, *virtual machine* ataupun *cloud*. [1]

Docker merupakan *open source software* di bawah Lisensi Apache Versi 2.0 yang bisa dipergunakan secara gratis. Saat ini Docker hanya bisa berjalan pada Linux, tetapi bisa menggunakan *virtual machine* pada *operating system* windows, atau menggunakan Boo2docker.

Docker menggunakan arsitektur *client-server*. Docker *client* menghubungi Docker *daemon*, yang melakukan pekerjaan berat, menjalankan, dan mendistribusikan Docker *container* anda. Kedua Docker *client* dan *daemon* dapat berjalan pada sistem yang sama. Docker *client* dan *daemon* berkomunikasi via *sockets* atau lewat API yang disediakan Docker. [2]

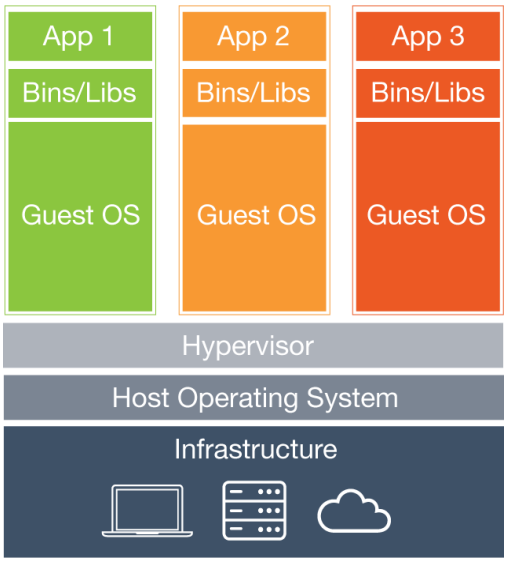
2.2. *Virtual Machine*

Virtual machine adalah sebuah komputer perangkat lunak, seperti komputer fisik, menjalankan sistem operasi dan aplikasi. Mesin virtual terdiri dari satu set spesifikasi dan konfigurasi *file* dan didukung oleh sumber daya fisik dari sebuah *host*. Setiap mesin virtual memiliki perangkat virtual yang menyediakan fungsi yang sama dengan perangkat keras fisik dan memiliki manfaat tambahan dalam hal portabilitas, pengelolaan, dan keamanan. [6]

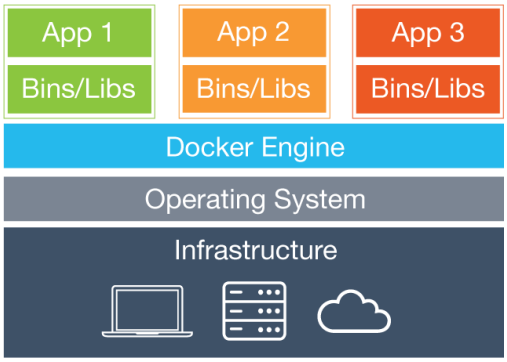
Dengan memiliki beberapa sistem operasi "virtual", kita juga dapat menjalankan beberapa eksperimen dan pengujian sebuah aplikasi. Setelah semua diinstal, sebuah mesin virtual dan harddisk virtual dapat dianggap sebagai *"container"* yang dapat sewaktu-waktu *frozen*, disalin, di-*back up*, dan di kirim antara komputer host.

Perbedaan Docker dengan *virtual machine* adalah *containers* dalam Docker memiliki sumbar daya yang terisolasi sama dan manfaat alokasi seperti *virtual machine* tetapi perbedaan arsitektur yang berbeda memungkinkan *containers* untuk menjadi lebih portabel dan efisien.

Perbedaan antara *virtual machine* dan docker dapat kita lihat pada gambar 1, *virtual machine* membutuhkan banyak ruang untuk "*Guest OS*" atau *operating system*. Untuk docker, dapat kita lihat pada gambar 2, docker tidak membutuhkan banyak ruang untuk *operating system*. Karena *operating system* pada docker dapat digunakan untuk menjalankan beberapa *container* bersama-sama.



Gambar 1. Sistem kerja *virtual machine*



Gambar 2. Sistem kerja docker

2.3. Ubuntu

Linux adalah platform server perusahaan yang ditetapkan pada tahun 2004, tetapi perangkat lunak bebas bukan bagian dari kehidupan sehari-hari bagi sebagian besar pengguna komputer. Itu sebabnya Mark Shuttleworth mengumpulkan tim kecil pengembang dari salah satu proyek Linux paling mapan yaitu Debian dan berangkat untuk membuat desktop Linux yang mudah digunakan, yaitu Ubuntu. [4]

Ubuntu dibuat di atas dasar Linux, yang merupakan anggota dari keluarga Unix. Unix sendiri adalah salah satu jenis tertua dari sistem operasi, dan bersama-sama dengan Linux telah memberikan keandalan dan keamanan untuk aplikasi profesional selama hampir setengah abad. Banyak server di seluruh dunia yang menyimpan data untuk situs populer, seperti YouTube dan Google menjalankan beberapa varian dari Linux atau Unix. Yang sedang populer, yaitu Android *system* untuk *smartphone* merupakan varian dari Linux, bahkan Apple OS X berdasarkan pada Unix.

2.4. Windows Server

Windows Server 2016 adalah sistem operasi server yang akan datang, yang dikembangkan oleh Microsoft sebagai bagian dari keluarga sistem operasi Windows NT, dikembangkan bersamaan dengan Windows 10. Versi *Technical Preview* pertama telah tersedia pada tanggal 1 Oktober 2014 bersama dengan *Technical Preview* dari System Center dan saat ini berada dalam tahap pengujian. [3]

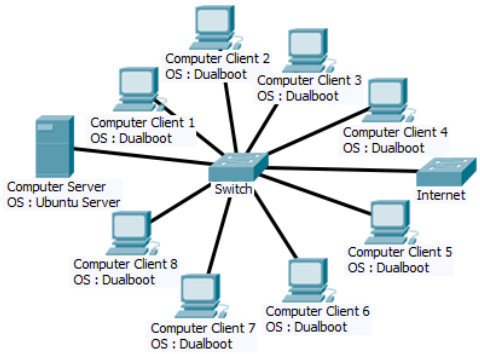
2.5. RAID

RAID adalah *system storage* menggabungkan beberapa harddisk dan menciptakan sebuah virtual harddisk yang besar. Seiring berjalannya waktu, RAID terus bertumbuh, mempunyai banyak solusi yang baru untuk problem yang lama, seperti *disk performance, redudancy, dan scalability.* Dengan RAID, kita juga dapat memastikan bila harddisk akan rusak, komputer server tidak akan berhenti bekerja. Kita dapat menggunakan *mirror*, tipe RAID yang menggunakan 2 atau lebih harddisk yang berisi sama sebagai *backup*. Bila sebuah harddisk mati, maka *mirror* harddisk dapat mengambil alih tugas. [5]

3. DESAIN SISTEM

3.1. Desain Jaringan

Topologi yang digunakan nantinya adalah topologi *star*. Topologi ini pada dasarnya memanfaatkan salah satu hardware penting dalam pembentukan jaringan komputer, yaitu hub ataupun switch yang ada pada gambar 3 berikut.



Gambar 3. Bentuk topologi *star*

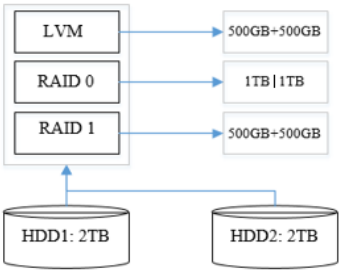
Kelebihan menggunakan topologi ini yaitu apabila komputer mengalami kerusakan, tidak akan mempengaruhi jaringan komputer yang lain. Pendeteksian masalah akan lebih mudah bila terjadi kerusakan pada jaringan. Kekurangan topologi ini adalah ketika switch rusak, semua komputer tidak bisa terhubung ke internet, dan akan lebih membutuhkan banyak kabel LAN.

3.2. Desain *Storage*

Pada tugas akhir ini akan menggunakan 3 buah bentuk *system storage*. Yang pertama adalah LVM, *system storage* ini disediakan secara otomatis dari Ubuntu sendiri. Cara kerja LVM ini adalah dengan menggabungkan 2 buah partisi dari harddisk yang berbeda menjadi 1 partisi, tetapi pengisian datanya adalah dengan memenuhi 1 harddisk terlebih dahulu, baru akan mengisi harddisk yang kedua. *System storage*

RAID ini merupakan fitur yang disediakan untuk komputer server dengan kegunaan ada bisa memanfaatkan harddisk dengan baik maupun menggunakan harddisk sebagai backup. Yang pertama adalah RAID0. RAID0 ini adalah merupakan RAID yang bekerja secara *mirroring*. Sistem kerja RAID ini adalah menggabungkan 2 buah partisi yang berbeda harddisk menjadi 1, tetapi jumlahnya tetap. Kegunaan RAID0 adalah ketika data dimasukkan ke dalam partisi, data tersebut akan dituliskan ke dua buah harddisk yang berbeda tersebut.

RAID yang kedua adalah RAID1, yaitu *striping*. RAID ini bekerja dengan cara menggabungkan 2 buah harddisk yang berbeda menjadi 1 dengan cara menjumlahnya. Sistem kerjanya hampir sama dengan LVM, tetapi dalam RAID1 ini, data dipecah-pecah dan dimasukkan secara bergantian ke harddisk 1 lalu ke harddisk yang kedua.



Gambar 4. Bentuk desain RAID

Pada gambar 4 di atas, menjelaskan pembagian dari 2 harddisk yang jumlahnya sama 2 *terabytes*, untuk partisi pertama yang akan digunakan untuk *system storage* LVM menggunakan 500 GB dari harddisk pertama dan ditambah 500 GB dari harddisk kedua, yang nantinya partisi akan berkapasitas 1 TB. Untuk yang kedua adalah partisi untuk RAID 0, menggunakan 1 TB dari harddisk pertama dan 1 TB dari harddisk kedua, yang nantinya akan digunakan untuk *mirroring* sehingga kapasitas partisi nantinya berjumlah 1TB. Partisi ketiga yaitu untuk RAID 1 menggunakan 500 GB dari harddisk pertama dan 500 GB dari harddisk kedua yang pada akhirnya kapasitas partisi menjadi 1 TB.

4. IMPLEMENTASI SISTEM

4.1. Konfigurasi Komputer Server

Dalam skripsi ini akan membutuhkan sebuah server yang akan digunakan sebagai tempat penyimpanan data-data *container*, sehingga tidak membebankan pada harddisk *computer client* melainkan pada harddisk *computer server*.

4.1.1. Instalasi Docker Private Registry

```
$ docker run -d -p 5000:5000 --restart=always --
name registry registry:2
```

4.1.2. Membuat Domain Registry

Lakukan lah perintah berikut untuk membuat direktori yang akan digunakan untuk menyimpan sertifikat koneksi

```
$ mkdir -p certs
```

Setelah membuat direktori certs, masuk ke direktori certs dan lakukanlah perintah berikut untuk membuat sertifikat.

```
$ openssl req -newkey rsa:4096 -nodes -sha256 -
keyout certs/domain.key -x509 -days 365 -out
certs/domain.crt
```

Setelah melakukan perintah tersebut, isilah terserah menurut masing-masing, tetapi pada bagian CN (*Common Name*) gunakanlah nama domain yang akan digunakan, pada skripsi ini akan menggunakan nama domain "registry.infor".

4.1.3. Instalasi Apache untuk Forwarding Port

Karena *registry* yang standart berjalan pada port 5000 dan setiap akan *push* atau *pull image* menggunakan port 5000 di akhir, dengan apache, kita akan mengalihkan port 5000 ke port 443 yaitu port https, sehingga tidak perlu menggunakan port lagi saat melakukan *push* atau *pull image*.

Hapus terlebih dahulu *container* dengan nama registry, setelah itu kita akan membuat direktori baru di home sebagai autentikasi apache nantinya

```
$ mkdir auth
$ mkdir data
```

Kita akan membuat konfigurasi apache dengan cara script, script ini sudah disediakan secara langsung oleh docker sendiri. Pertama kita akan membuat konfigurasi pada

```
$ sudo nano auth/httpd.conf
```

File httpd.conf ini nantinya akan kita isikan dengan script berikut

```
LoadModule headers_module modules/mod_headers.so
LoadModule authn_file_module
modules/mod_authn_file.so
LoadModule authn_core_module
modules/mod_authn_core.so
LoadModule authz_groupfile_module
modules/mod_authz_groupfile.so
LoadModule authz_user_module
modules/mod_authz_user.so
LoadModule authz_core_module
modules/mod_authz_core.so
LoadModule auth_basic_module
modules/mod_auth_basic.so
LoadModule access_compat_module
modules/mod_access_compat.so
LoadModule log_config_module
modules/mod_log_config.so
```

```
LoadModule ssl_module modules/mod_ssl.so
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module
modules/mod_proxy_http.so
LoadModule unixd_module modules/mod_unixd.so
<IfModule ssl_module>
    SSLRandomSeed startup builtin
    SSLRandomSeed connect builtin
</IfModule>
<IfModule unixd_module>
    User daemon
    Group daemon
</IfModule>
ServerAdmin you@example.com
ErrorLog /proc/self/fd/2
LogLevel warn
<IfModule log_config_module>
    LogFormat "%h %l %u %t \"%r\" %>s %b
\"%{Referer}i\" \"%{User-Agent}i\"" combined
    LogFormat "%h %l %u %t \"%r\" %>s %b" common
    <IfModule logio_module>
      LogFormat "%h %l %u %t \"%r\" %>s %b
\"%{Referer}i\" \"%{User-Agent}i\" %I %O"
combinedio
    </IfModule>
    CustomLog /proc/self/fd/1 common
</IfModule>
ServerRoot "/usr/local/apache2"
Listen 443
<Directory />
    AllowOverride none
    Require all denied
</Directory>
<VirtualHost *:443>
  ServerName myregistrydomain.com
  SSLEngine on
  SSLCertificateFile
/usr/local/apache2/conf/domain.crt
  SSLCertificateKeyFile
/usr/local/apache2/conf/domain.key
  ## SSL settings recommandation from:
https://raymii.org/s/tutorials/Strong_SSL_Security_
On_Apache2.html
  # Anti CRIME
  SSLCompression off
  # POODLE and other stuff
  SSLProtocol all -SSLv2 -SSLv3 -TLSv1
  # Secure cypher suites
  SSLCipherSuite
EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH
  SSLHonorCipherOrder on
  Header always set "Docker-Distribution-Api-
Version" "registry/2.0"
  Header onsuccess set "Docker-Distribution-Api-
Version" "registry/2.0"
  RequestHeader set X-Forwarded-Proto "https"
  ProxyRequests     off
  ProxyPreserveHost on
  # no proxy for /error/ (Apache HTTPd errors
messages)
  ProxyPass /error/ !
  ProxyPass        /v2 http://registry:5000/v2
  ProxyPassReverse /v2 http://registry:5000/v2
  <Location /v2>
    Order deny,allow
    Allow from all
    AuthName "Registry Authentication"
    AuthType basic
    AuthUserFile
"/usr/local/apache2/conf/httpd.htpasswd"
    AuthGroupFile
"/usr/local/apache2/conf/httpd.groups"
    # Read access to authentified users
    <Limit GET HEAD>
      Require valid-user
    </Limit>
    # Write access to docker-deployer only
    <Limit POST PUT DELETE PATCH>
      Require group pusher
    </Limit>
  </Location>
```

```
</VirtualHost>
```

Setelah script sudah selesai dibuat, lakukanlah perintah berikut untuk menambahkan pengguna, yang akan dipakai saat mengakses server dari komputer lain

```
$ docker run --entrypoint htpasswd httpd:2.4 -Bbn
testuser testpassword > auth/httpd.htpasswd
```

Salinlah sertifikat yang tadi sudah dibuat ke dalam folder milik apache, karena port sudah di *forwarding* melalui apache

```
$ cp domain.crt auth
$ cp domain.key auth
```

Kembali ke direktori home, lalu buatlah file compose baru dengan nama "docker-compose.yml", yang isinya

```
apache:
  image: "httpd:2.4"
  hostname: myregistrydomain.com
  ports:
    - 443:443
  links:
    - registry:registry
  volumes:
    - `pwd`/auth:/usr/local/apache2/conf

registry:
  image: registry:2
  ports:
    - 127.0.0.1:5000:5000
  volumes:
    - `pwd`/data:/var/lib/registry
```

Setelah semua selesai dilakukan, kita akan menyalakan *container* registry dan apache secara bersamaan dengan menggunakan script dengan nama "docker-compose.yml" dengan perintah berikut

```
$ docker-compose up
```

4.2. Konfigurasi *Client* berbasis Ubuntu Server 14.04.4 Trusty Tahr

Komputer *client* ini yang nantinya digunakan oleh para mahasiswa untuk bekerja dalam kesehariannya didalam laboratorium. Komputer ini digunakan hanya untuk mengedit *image*, tetapi bukan sebagai tempat penyimpanan *image.*

4.2.1. Konfigurasi untuk Koneksi ke Insecure Registry

Setelah proses instalasi aplikasi docker-engine selesai, hal berikutnya yang dilakukan adalah dengan menambahkan konfigurasi *private registry* dengan mengedit "/etc/default/docker" dengan menambahkan baris berikut pada bagian "DOCKER OPTS"

```
DOCKER_OPTS="--insecure-registry registry.infor"
```

4.2.2. Menambahkan Sertifikat Domain

Setelah menambahkan *insecure registry* dan hosts, langkah terakhir yang kita lakukan adalah menempatkan sertifikat yang sudah kita buat pada komputer server. Kita salin file "domain.crt" dari komputer server, letakkanlah pada direktori

```
/etc/docker/certs.d/registry.infor/ca.crt
```

4.3. Konfigurasi *Client* berbasis Windows Server Technical Preview 5 Datacenter

Windows server yang akan digunakan ini akan memakai versi yang *Techincal Preview* ke 5. Versi ini adalah versi windows server 2016 yang terbaru saat ini, karena *operating system* ini

masi baru. Penulis menggunakan *operating system* ini karena, di saat penulis mengerjakan skripsi ini, fitur *container* yang sudah berjalan cukup stabil pada *operating system* ubuntu server di jalankan pada windows server 2016 ini. Fitur *container* ini merupakan fitur baru dalam windows server 2016. Meski belum berjalan sepenuhnya sempurna, kita akan mencoba mengeksplorasi *container* ini lebih lanjut.

4.3.1. Instalasi Docker Image

Instalasi *Operating System Image* ini untuk digunakan sebagai *base image* dari *containers*. Langkah pertama yaitu bukalah *powershell*. Lakukanlah konfigurasi berikut

```
PS C:\Users\Administrator> Install-PackageProvider
ContainerImage -Force
```

Perintah di atas adalah perintah untuk menginstal modul *container provider* pada *powershell*. Setelah instalasi selesai, ketiklah perintah berikut untuk mengetahui *image* apa saja yang tersedia yang dapat kita install

```
PS C:\Users\Administrator> Find-ContainerImage
```

Lakukanlah perintah berikut untuk instalasi *image* tersebut

```
PS C:\Users\Administrator> Install-ContainerImage –
Name WindowsServerCore
```

Sedangkan yang kedua adalah untuk instalasi *image* windows server core

4.3.2. Menambahkan Hosts

Menambahkan hosts dilakukan agar komputer *client* dapat mengenal sebuah domain lokal dengan nama domainnya, bukan dengan IP *address*-nya.

Bukalah notepad secara administrator, lalu pilih *open*, pilihlah hosts pada

```
C:\Windows\System32\drivers\etc\hosts
```

Tambahkanlah 1 baris lagi, yaitu IP *address private registry* dan nama domainnya

```
192.168.202.131          registry.infor
```

5. PENGUJIAN DAN EVALUASI SISTEM

5.1. Pengujian Docker

Pengujian ini akan dimulai dengan menjalankan sebuah *container*. Dalam pengujian ini, penulis akan menginstall beberapa aplikasi di dalam *container* yang akan digunakan untuk dikirim ke *private registry*. Setelah aplikasi selesai diinstall, *container* akan dikembalikan lagi menjadi sebuah *image* baru.

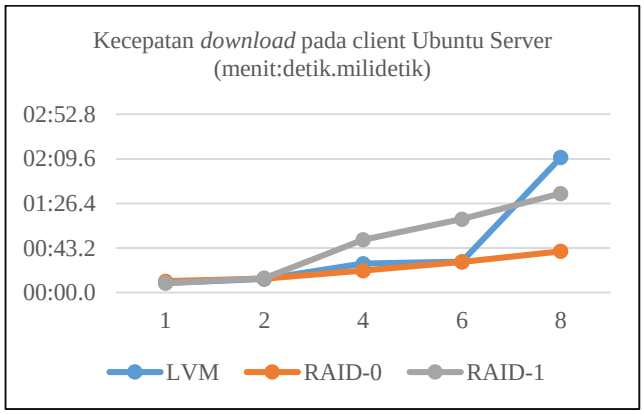
Pada server sudah ada *base image* sebelumnya, sehingga saat penulis memasukkan *image* yang telah dibuat tadi berdasarkan *image* ubuntu yang ada, *image* yang terkirim hanya *image* bagian lapisan atas, bisa kita lihat pada layer paling atas yang terdapat hanya kata-kata "Pushed", sedangkan yang lain adalah "*Layer already exist*" yang menyatakan dalam server sudah ada data tersebut.

Ketika penulis melakukan penarikan *image* kembali, docker hanya menarik bagian layer yang terakhir. Ini dikarenakan pada komputer *client* masih memiliki *image* ubuntu yang layernya sama, sehingga muncul keterangan "*Already exists*".

Ini menunjukkan bahwa aplikasi docker ini sangat ringan. Docker hanya membungkus aplikasi yang kita install saja, docker juga membungkus konfigurasi yang kita lakukan dalam sebuah *container* yang sangat ringan karena tidak memerlukan *base image*.

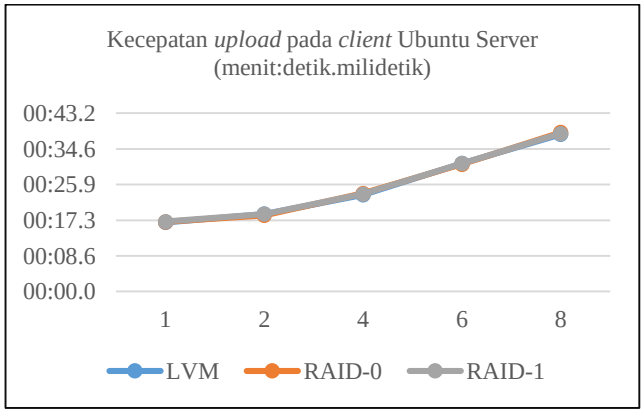
5.2. Evaluasi Kecepatan Transfer Docker *Image* dengan *System Storage* RAID

Berikut merupakan grafik waktu yang dibutuhkan untuk *download* sebuah *image* sebesar 120 MB (*megabytes)* pada *operating system* Ubuntu Server.



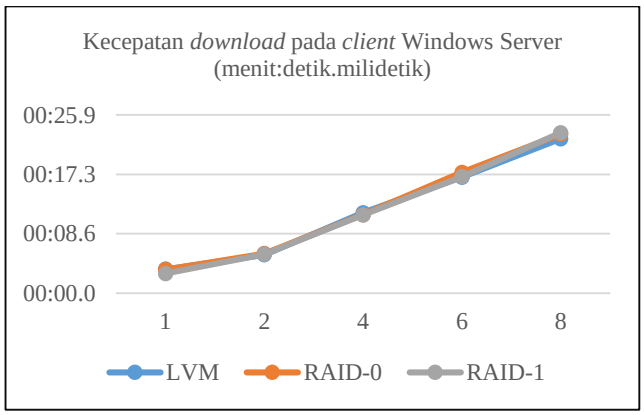
Gambar 5. Kecepatan *download* pada *client* Ubuntu Server

Berikut merupakan grafik waktu yang dibutuhkan untuk *upload* sebuah *image* sebesar 120 MB (*megabytes)* pada *operating system* Ubuntu Server.



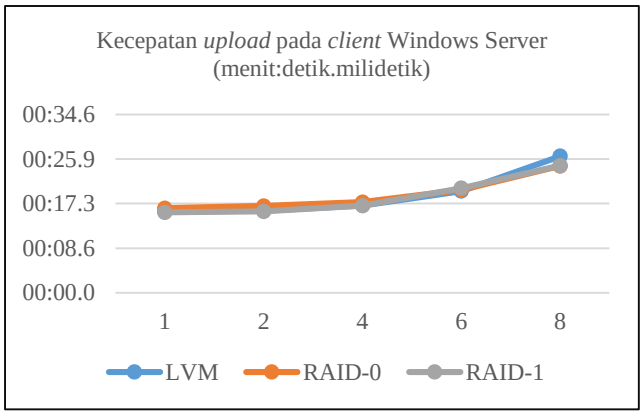
Gambar 6. Kecepatan *upload* pada *client* Ubuntu Server

Berikut merupakan grafik waktu yang dibutuhkan untuk *download* sebuah *image* sebesar 130 MB (*megabytes)* pada *operating system* Windows Server



Gambar 7. Kecepatan *download* pada *client* Windows Server

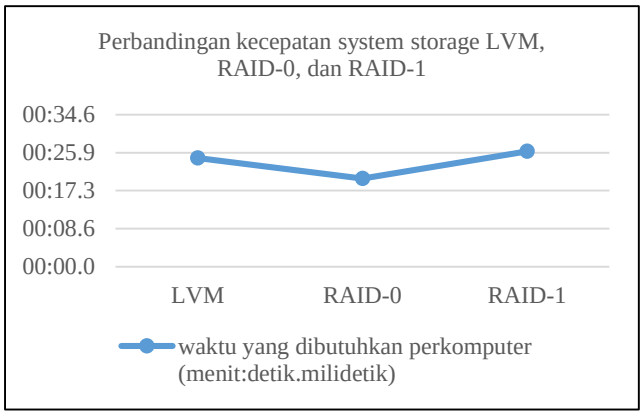
Berikut merupakan grafik waktu yang dibutuhkan untuk *upload* sebuah *image* sebesar 130 MB (*megabytes)* pada *operating system* Windows Server



Gambar 8. Kecepatan *upload* pada *client* Windows Server

Tabel 1. Perbandingan kecepatan rata-rata *system storage* LVM, RAID-0 dan RAID-1

LVM	RAID-0	RAID-1
0:24.730	0:20.060	0:26.250



Gambar 9. Perbandingan kecepatan rata-rata *system storage* LVM, RAID-0 dan RAID-1

Setelah melihat hasil dari tabel 1 dan gambar 9 di atas, kecepatan transfer data yang tercepat adalah dengan menggunakan *system storage* RAID-0.

6. KESIMPULAN

Dengan melakukan eksplorasi aplikasi ini, dapat disimpulkan bahwa implementasi aplikasi Docker dan evaluasi *system storage* RAID memiliki hasil sebagai berikut:

- Aplikasi Docker ini mampu menyediakan kebutuhan mahasiswa dalam memiliki *operating system* atau konfigurasinya sendiri-sendiri.
- Pada tahap pengujian *system storage*, RAID-0 memiliki kecepatan yang lebih cepat bila dibandingkan dengan *system storage* LVM dan RAID-1.

Di lain sisi, implementasi aplikasi Docker ini juga memiliki beberapa kekurangan, yaitu:

- *Container* dapat berjalan pada *operating system* Ubuntu Server dan Windows Server, tetapi keduanya masih belum bisa memberikan GUI (*Graphical User Interface*) yang bisa dipakai oleh *user*, dan pada *operating system* Windows Server masih belum bisa melakukan instalasi aplikasi sendiri karena Windows Server tidak memiliki *store*-nya sendiri.
- Docker belum bisa melakukan autentikasi per-*user* ketika akan melakukan *push* atau *pull* sebuah *images.*

7. DAFTAR PUSTAKA

[1] Docker. 2014. *Understanding Docker*. URI= https://docs.docker.com/engine/introduction/understanding-docker.

[2] Goasguen, Sebastien. 2015. *Docker Cookbook*. United States of America. O'Reilly Media, Inc.

[3] Microsoft. 2016. *Windows Server Containers*. URI= https://msdn.microsoft.com/en-us/virtualization/windowscontainers/about/about_overview.

[4] Ubuntu. 2016. *The Ubuntu Story*. URI= http://www.ubuntu.com/about/about-ubuntu/.

[5] Vadala, Derek. 2003. *Managing* RAID *on* LINUX. United States of America. O'Reilly Media, Inc.

[6] VMWare. 2012. *What Is a Virtual Machine?* URI= https://pubs.vmware.com/vsphere-50/index.jsp?topic=%2Fcom.vmware.vsphere.vm_admin.doc_50%2FGUID-CEFF6D89-8C19-4143-8C26-4B6D6734D2CB.html