Analisis Kebutuhan Resource Dan Independensi Antara Teknologi Single Server, Virtualisasi Dan Container

Hanif Afrizal[1], Agus Prihanto[2]

[1,2] Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya
[1]hanif.18044@unesa.ac.id
[2]agusprihanto@unesa.ac.id

Abstrak*— Saat ini perkembangan teknologi server semakin pesat, perkembangan ini disertai dengan banyaknya aplikasi web yang dikelola. Teknologi virtualisasi merupakan solusi dari independensi aplikasi. Peningkatan jumlah aplikasi web harus diikuti dengan peningkatan kebutuhan resource. Virtualisasi docker container menjadi trend saat ini. Docker adalah salah satu software yang menggunakan virtualisasi OS untuk menyimpan perangkat lunak ke dalam sebuah container. Penelitian ini dilakukan untuk menganalisis perbandingan kebutuhan resource dan independensi terhadap teknologi single server, virtualisasi dan container. Metode yang digunakan untuk menganalisis perbandingan dengan mengelola banyak aplikasi apache web server. Hasil dari pengujian penggunaan memori, container membutuhkan memori sebesar 521 MB, single server membutuhkan memori sebesar 1330 MB dan virtualisasi membutuhkan memori sebesar 2475 MB. Hasil pengujian penggunaan CPU container membutuhkan CPU 1.85%, single server membutuhkan CPU 2.74% dan virtualisasi membutuhkan CPU 4.95%. Penggunaan memori dan CPU pada docker container lebih sedikit karena pada setiap virtualisasi server docker container tidak memuat kernel, melainkan berbagi kernel host dengan container yang lain. Berbeda dengan virtualisasi dan single server yang memiliki kernel sendiri untuk menjalankan aplikasi dalam hal ini menyebabkan kebutuhan resource memori dan CPU container lebih kecil. Container dan virtualisasi memiliki independensi yang terjaga, karena sumber daya dan aplikasi terisolasi secara terpisah, sehingga tidak menggangu konfigurasi aplikasi yang lain.

Kata Kunci*— Virtualisasi Server, Docker Container, Single Server, Resource, Independensi.

I. PENDAHULUAN

Di era modern ini perkembangan teknologi merupakan tantangan yang tidak dapat dipisahkan dari kehidupan manusia. Adaptasi manusia dan teknologi sangat penting dan secara teknis tidak boleh ditinggalkan oleh generasi berikutnya. Dengan demikian, teknologi dan kehidupan manusia dapat berkembang dengan adanya generasi baru sebagai penerus generasi lama. Era globalisasi saat ini tidak terlepas dari perkembangan teknologi khususnya komputer dan media internet. Sebagian besar kegiatan manusia saat ini dilakukan dengan menggunakan komputer dan internet. Komputer pada dasarnya adalah mesin yang membantu memecahkan masalah matematika atau matematika [1]. Internet seharusnya menjadi jaringan yang dapat menghubungkan banyak komputer untuk mengirim pesan, menerima informasi, dan mentransfer data [2]. Salah satu pemanfaatan perkembangan komputer dan internet adalah aplikasi berbasis web. Perkembangan dari komputer dan internet hingga aplikasi berbasis web saat ini berkembang sangat pesat. Aplikasi berbasis web semakin banyak digunakan karena dapat mengakses berbagai platform komputasi hanya dengan menjalankan browser web. Oleh karena itu, kesederhanaan proses penyebaran aplikasi web, bersama dengan perangkat lunak pendukung seperti server web, server basis data, dependensi, dan lingkungan server lainnya, sangat penting [3]. Sebagai teknologi berkembang, sistem jaringan dan masalah manajemen sumber daya yang terbatas muncul. Sistem teknis sangat dibatasi oleh ruang seperti penawaran server, penawaran penyimpanan, dan peningkatan kapasitas perangkat keras. Tentu saja biaya awal ini tidak murah. Selain itu, server tidak dapat digunakan secara optimal sebagai perangkat keras dasar.

Server adalah pusat kendali otoritas untuk masalah jaringan. Server sering tidak bekerja secara optimal karena beban dan permintaan yang tinggi. Oleh karena itu, server hanya menggunakan semua sumber daya untuk permintaan tertentu [4].

Teknologi virtualisasi semakin banyak digunakan dalam pengembangan web karena kemudahan konfigurasi dan isolasi mesin virtual. Virtualisasi dapat didefinisikan sebagai pemecahan server fisik menjadi beberapa server virtual untuk mengoptimalkan penggunaan sumber daya yang ada pada server fisik [5]. Virtualisasi juga digunakan untuk meniru perangkat komputasi fisik. Virtualisasi memungkinkan beberapa sistem operasi dan layanan berjalan secara bersamaan pada satu perangkat keras fisik. Ini termasuk membuat satu sumber daya, seperti server, sistem operasi, aplikasi, atau perangkat penyimpanan tampak bertindak sebagai beberapa sumber daya logis. Ini juga termasuk mendefinisikan beberapa sumber daya fisik (seperti beberapa perangkat penyimpanan dan server) sebagai satu sumber daya logis. Sumberdaya [6]. Ini adalah dasar untuk menerapkan virtualisasi server. Ini adalah solusi hebat untuk menghemat sumber daya penyebaran infrastruktur dan meningkatkan efisiensi energi dan biaya.

Teknologi virtualisasi adalah perangkat lunak yang memungkinkan perangkat keras untuk menjalankan beberapa sistem operasi dan layanan lainnya secara bersamaan, dengan tujuan untuk mengoptimalkan penggunaan prosesor dan menghindari pemborosan energi listrik dan biaya server yang berlebihan. Teknologi virtualisasi memudahkan dalam mengelola dan mengamankan server fisik jika rusak, sehingga cepat dan tidak perlu diinstal ulang [7]. Keuntungan lain dari teknologi virtualisasi adalah teknologi yang sangat kuat yang

digunakan untuk menyederhanakan pengembangan dan pengujian perangkat lunak dan memungkinkan integrasi server, sebagai pemisahan perangkat lunak dan perangkat keras, telah terbukti dapat digunakan [8]. Virtualisasi mesin memiliki beberapa kelemahan, termasuk pengelompokan server web yang lambat. Ini telah menciptakan berbagai teknik dan model untuk meningkatkan efisiensi, skalabilitas, dan ketahanan.

Container adalah virtualisasi tingkat *OS*, dan semua proses atau aplikasi yang menjalankan semua wadah berbagi kernel yang sama. Itu sendiri lebih baik daripada virtualisasi tingkat mesin (mesin virtual), yang membutuhkan mesin virtual untuk memiliki kernel sistem operasi yang berbeda untuk setiap aplikasi yang berjalan. Anda dapat secara optimal menggunakan perangkat keras yang ada dengan menggunakan wadah Docker dalam desain server Anda. Wadah Docker di mana kernel digunakan sendiri merupakan bagian dari sistem operasi *host* dan oleh karena itu tidak membahayakan kinerja server *host* [6].

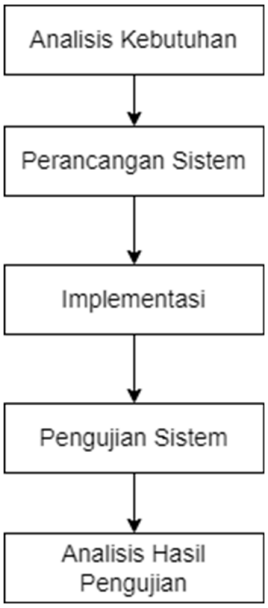
Docker adalah cara untuk menempatkan layanan di lingkungan terpisah yang disebut wadah, sehingga Anda dapat menggabungkan layanan dengan perpustakaan atau perangkat lunak lain yang diperlukan [3]. Docker memiliki sifat *containerization* yang ringan dengan berbagai komponen dan fitur yang memudahkan pengembang untuk mengembangkan dan memantau kinerja aplikasi yang dibangunnya, memungkinkan mereka untuk mendistribusikan dan mengembangkan aplikasi dengan lebih cepat. Docker dapat dikonfigurasi untuk membuat objek. Objek yang disertakan dalam Docker termasuk gambar, wadah, dan layanan. Untuk menyebarkan aplikasi Anda ke platform Docker, cukup buat file instruksi (bernama dockerfile) tanpa membuat perubahan yang tidak perlu pada infrastruktur aplikasi Anda. Klien Docker dan daemon Docker dapat berjalan di *host* yang sama dan berkomunikasi melalui permintaan API, soket, atau jaringan. Fungsi dari klien Docker adalah mengambil sekumpulan perintah dari pengguna dan mengirimkannya untuk diproses oleh daemon Docker [9]. Docker meminimalkan penyediaan sumber daya yang tidak diperlukan untuk menerapkan aplikasi Anda. Docker adalah solusi standar untuk memecahkan salah satu area yang paling mahal dari siklus pengembangan perangkat lunak: penyebaran [10] Miel. Selama proses penerapan, Docker menjalankan wadah menggunakan gambar dasar dengan sistem file sebagai metode lapisan. Artinya, Docker menyalin perubahan lapisan dan menjalankannya sebagai wadah replikasi terpisah yang menggunakan gambar dasar yang sama [6]. Secara teknis, layanan web disediakan oleh aplikasi server web seperti Apache dan Nginx. Secara default, setiap server web melayani situs web untuk domain tertentu. Anda kemudian dapat mengembangkan aplikasi website Anda menggunakan bahasa pemrograman seperti PHP dan Javascript. Semua dependensi yang diperlukan dikemas dalam wadah Docker dengan beberapa konfigurasi termasuk Dockerfile dan YAML.

Beberapa peneliti sebelumnya telah mencoba masalah di atas. Penelitian [11] telah sampai pada kesimpulan bahwa teknologi virtualisasi dapat digunakan untuk meningkatkan penggunaan sumber daya perangkat keras, terutama memori, hingga 80% dari memori yang ada. Selain itu, teknologi virtualisasi dapat meningkatkan kinerja dan kemudahan penggunaan, yang diwujudkan dalam beberapa server virtual.

Selama penyelidikan [1], saat menguji penggunaan sumber daya, sistem operasi memiliki batas mesin virtual yang dapat diinstal pada platform Proxmox. Ini adalah 16 mesin virtual, tetapi platform VMWare ESXi dan XenServer hanya dapat menampung 12 mesin virtual. Ini karena Proxmox menggunakan memori virtual per server virtual. Berbeda dengan penelitian sebelumnya, penelitian ini melakukan uji perbandingan kebutuhan *resource* antara teknologi *single-server*, virtualization, dan *container*, serta menguji perbandingan kebutuhan independensi teknologi dengan *single-server* virtualization dan *container*.

II. METODE PENELITIAN

Jenis penelitian ini adalah penelitian rekayasa yang berbasis eksperimen, yaitu metode penelitian yang bertujuan untuk meneliti dan menganalisis perbandingan kebutuhan *resource* pada teknologi *single server*, virtualisasi dan *container* jika digunakan untuk *web server*. Metode merupakan suatu petunjuk yang dilakukan untuk mencapai suatu tujuan yang diinginkan. Beberapa tahap yang dilakukan pada analisis *single server*, virtualisasi dan *container* sebagai berikut :



Gbr 1. Metode penelitian

A. Analisa Kebutuhan

Tahap pertama dalam penelitian adalah analisis kebutuhan yang akan digunakan untuk eksperimen. Kebutuhan tersebut akan digunakan sebagai modal untuk menyelesaikan analisis perbandingan *single server*, virtualisasi dan *container*. Analisa kebutuhan dapat dijabarkan menjadi beberapa bagian, yaitu:

1. Kebutuhan Data

Data yang dibutuhkan dalam penelitian ini diambil dari berbagai sumber referensi. Sumber refrensi yang diambil berasal dari jurnal internasional dan jurnal nasional. Penggunaan data yang lengkap dan akurat dapat membuat hasil penelitian ini menjadi lengkap dan terarah. Pengumpulan data untuk penelitian ini diambil dari dua jenis sumber, yaitu studi litelatur dan observasi.

a. *Studi literatur*

Pada penelitian ini mengumpulkan berbagai referensi dari beberapa macam literatur yang mendukung dengan analisis perbandingan *resource* dan independentsi teknologi *single server*, virtualisasi dan *container*. Untuk mengetahui lebih banyak pengetahuan. Literatur yang diambil dari laporan, jurnal nasional, jurnal internasional, makalah, tesis, video dari youtube, web resmi dan sumber lainnya dari internet.

b. *Observasi*

Penelitian ini juga melakukan observasi analisis dengan mempelajari beberapa situs refrensi mengenai penggunaan teknologi *single server*, virtualisasi dan *container*.

2. *Kebutuhan Alat*

Spesifikasi perangkat yang digunakan untuk melakukan penelitian yang berupa perancangan dan analisis adalah :

a. Prosessor Intel Core I5-825U
b. RAM 12 GB
c. Harddisk 512 GB
d. Sistem Operasi Windows 10 64-bit (yang mendukung virtualisasi atau terinstal windows subsystem for linux)

Adapun perangkat lunak yang digunakan untuk mendukung penelitian ini adalah :

a. Docker desktop yang berguna untuk membangun sebuah *container.*
b. Virtualbox sebagai media virtualisasi perangkat.
c. WSL2(Windows Subsystem for Linux) untuk menghubungkan *backend* linux dengan *host OS* windows
d. Notepad++ memiliki fungsi sebagai alat untuk menulis kode program

B. Perancangan Sistem

Perancangan system adalah tahap mempersiapkan kebutuhan system untuk melakukan implementasi penelitian. Yaitu system *single server*, virtualiasasi server dan docker :

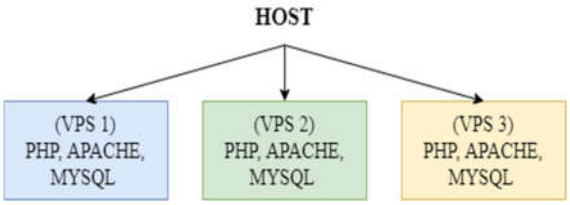
1. *Perancangan Containers*

Gbr 2. Perancangan Containers

Image yang diambil dari Docker berisi PHP dengan menggunakan Apache sebagai *web server* untuk menjalankan website yang menggunakan bahasa pemrograman PHP. Selanjutnya, mysql berfungsi sebagai manajemen database relasional dan adminer digunakan untuk mengelola database. Konfigurasi pengembangan *image* Docker mencakup setiap lingkungan yang dirancang dan *image* yang dibangun melalui Dockerfile yang berjalan di Docker.

2. *Desain Sistem*

Berikut ini desain sistem yang akan dibuat untuk analis perbandingan kebutuhan reasource dan independentsi *single server*, virtualisasi dan *container* :



Gbr 3. Desain Sistem

Terdapat 3 *container* yang ada pada docker dan 3 virtualisasi yang diimplementasikan pada analisis perbandingan *resource* dan independensi.

C. Implementasi Sistem

Setelah perancangan dan Analisa kebutuhan system. Tahap selanjutnya adalah menerapkan hasil rancangan install beberapa aplikasi web yang telah dibuat untuk analisis perbandingan *resource* dan independentsi teknologi *single server*, virtualisasi dan *container* sesuai dengan Analisa kebutuhan yang didapat.

D. Pengujian Sistem

Pengujian sistem dijalankan setelah implementasi selesai. Pengujian sistem ini memiliki tujuan untuk memastikan bahwa hasil implementasi sistem yang diimplementasikan sesuai dengan perancangan yang sudah dibuat. Jika terjadi kesalahan yang mengarah pada kegagalan percobaan, desain dimodifikasi dan implementasi diulang.

E. Analisis Hasil Pengujian

Kesimpulan atas hasil pengujian diambil dari *resource* yang dibutuhkan saat menjalankan *web server* apache pada *single server*, virtualisasi dan *container*. Berdasarkan data tersebut ada beberapa hasil, seperti :

1. *Memori*

Kebutuhan *resource* memori yang diukur adalah penggunaan memori tertinggi di setiap pengujian sistem *web server* Apache yang sedang berjalan. Data yang diambil dari statistik penggunaan sumber daya *web server* yang dilakukan selama pengujian.

2. *CPU*

Kebutuhan *resource* CPU yang diukur adalah penggunaan CPU tertinggi dari setiap pengujian sistem *web server* Apache yang sedang berjalan. Data yang diambil dari statistik penggunaan sumber daya *web server* yang dilakukan selama pengujian.

III. HASIL DAN PEMBAHASAN

Hasil dan pembahasan dilambil setelah perancangan system dan Analisa kebutuhan selesai diimplementasikan. Hal yang akan dibahas pada hasil dan pembahasan adalah analisis kebutuhan *resource* dan independentsi antara teknologi *single server*, virtualisasi dan *container*.

A. Hasil Implementasi Virtualisasi Server

1. *Virtualisasi Server pada Oracle VirtualBox*

Setelah kebutuhan sudah terpenuhi selanjutnya adalah implementasi virtualisasi server pada Oracle VirtualBox sebagai webserver untuk pengujian. Untuk membuat server lokal virtualisasi menyiapkan iso server yang menggunakan ubuntu-server diinstal pada Oracle VirtualBox.

Sebelum membuat server virtualisasi harus menginstal beberapa software pendukung kebutuhan virtualisasi server yang berisi berisi php dengan apache sebagai *web server* mysql sebagai manajemen database.

```
hanif@hanif:~$ sudo apt install php libapache2-mod-php php-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.4 php-common php7.4 php7.4-cli php7.4-common php7.4-json php7.
  php7.4-opcache php7.4-readline
Suggested packages:
  php-pear
hanif@hanif:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
hanif@hanif:~$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-7
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mecab-ipadic mecab-ipadic-utf8
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0
  mysql-server-core-8.0
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyca
```

Gbr 4. Install software

Server akan menginstall php dan apache *web server* yang kemudian memberikan nama server 'localhost' pada pengaturan 'apache2.conf'. Server akan menyalin konfigurasi apache pada folder yang bernama '/etc/apache2/sites-available/'. Setelah itu akan mengambil file di folder 'src' yang ada ke konfigurasi apache '/var/www/html', serta akan menambahkan username 'root' untuk memberi izin user local dalam server apache.

```
<VirtualHost *:80>

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html/Qgis

        <Directory /var/www/html>
         Options Indexs FollowSymLinks
         AllowOverride All
         Require all granted
        </Directory>_


        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

Gbr 5. Konfigurasi apache 000-default.conf

Konfigurasi apache dilakukan untuk mengkonfigurasi virtual *host* apache, konfigurasi akan diprioritaskan untuk server. Dalam konfigurasi 'apache 000-default.conf' menampilkan port *host*, admin server dan dokumen root yang merujuk pada folder Qgis. Direktori 'var/www/html/' berfungsi agar apache mengikuti link sesuai pada sistem dan menampilkan list direktori jika tidak terdapat file index. Pada 'Require all granted' Apache mengizinkan semua ip mengakses *web server*. 'ErrorLog' berfungsi untuk menyimpan error log pada apache.

Setelah itu peneliti dapat memasukkan web dan memasukkan database web ke dalam mysql pada virtualisasi server tersebut.

2. *Virtualisasi Single Server pada Oracle VirtualBox*

Pada impelementasi *single server* di Oracle Virtualbox hampir sama dengan implementasi virtualisasi. Menentukan terlebih dahulu spek dari *single server* yang akan dilakukan virtualisasi, seperti RAM dan memori. Agar bisa digunakan sebagai *single server* harus melengkapi kebutuhan software yang ingin digunakan. Dengan menginstal apache yang digunakan untuk *web server*, php dan mysql yang digunakan untuk mengelola database yang akan diupload pada *single server*.

```
hanif@hanif:~$ sudo apt install php libapache2-mod-php php-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libapache2-mod-php7.4 php-common php7.4 php7.4-cli php7.4-common php7.4-json php7.
  php7.4-opcache php7.4-readline
Suggested packages:
  php-pear
hanif@hanif:~$ sudo apt install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser
hanif@hanif:~$ sudo apt install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
```

Gbr 6. Install kebutuhan aplikasi

Setelah menginstal semua software yang dibutuhkan, selanjutnya akan download web yang akan dijalankan pada server dengan menggunakan perintah wget. Dan harus mengkonfigurasi dokumen root untuk penyimpanan webnya. Untuk dokumen rootnya menggunakan default dari pengaturan 'var/www/html' agar sistem apache membaca list direktori berikut.

```
hanif@hanif:/var/www/html$ ls
cloudHanif1  cloudHanif2  cloudHanif3  csFVZ  index.html
hanif@hanif:/var/www/html$
```

Gbr 7. DocumentRoot

Setelah web berada pada documentroot maka web sudah dapat diakses pada browser dengan nama file yang sesuai dengan documentroot.

3. *Virtualisasi Docker Container*

Untuk membuat server dari docker *container* peneliti membuat folder dan direktori file yang tersusun dengan rapi terlebih dahulu agar dapat dipahami oleh docker dan pengguna. Karena akan menggunakan perintah 'docker-compose up' yang mempermudah peneliti untuk membuat virtualisasi *web server* pada docker *container*.

```
D :
|       .env
|       docker-compose.yml
+-----docker
|       +--apache
|       |       000-default.conf
|       |       dockerfile
|       \--mysql
|               database.sql
|               dockerfile
|
|
\--src
```

Gbr 8. Direktori file dan folder

Pada direktorinya berisi file '.env', 'docker-compose.yml', folder docker dan folder src. Untuk di dalam folder docker terdapat dua sub folder apache dan mysql. Folder apache yang di dalamnya ada file '000-default.conf' dan 'dockerfile'. Folder mysql memuat database mysql 'database.sql' dan 'dockerfile'. Namun file 'dockerfile' pada apache dan mysql isinya berbeda.

```yaml
version: '3'
services:
    db:
        build:
            context: .
            dockerfile: docker/mysql/dockerfile
        environment:
            MYSQL_DATABASE: ${MYSQL_DATABASE}
            MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
        command: --default-authentication-plugin=mysql_native_password
        restart: unless-stopped
        volumes:
            - ./db_data:/usr/data
        ports:
            - 3308:3306
    web:
        build:
            context: .
            dockerfile: docker/apache/Dockerfile
            args:
                uid: ${UID}
        environment:
            - APACHE_RUN_USER=#${UID}
            - APACHE_RUN_GROUP=#${UID}
        restart: unless-stopped
        volumes:
            - ./src:/var/www/html
            - ./apache_log:/var/log/apache2
        ports:
            - 8000:80
        depends_on:
            - db
        links:
            - db
    adminer:
        image: adminer
        restart: unless-stopped
        ports:
            - 8080:8080
volumes:
    db_data:
    src:
```

Gbr 9. docker-compose.yml

File 'docker-compose.yml' berisi suatu konfigurasi yang memiliki format YAML berfungsi untuk mengimplementasikan docker dengan *multi-container*. Dalam file tersebut terdapat perintah untuk mendefinisikan dua *image* mysql (db) dan php-apache server (web) yang akan dijalankan sebagai docker *container* dan adminer.

Untuk membangun sebuah *multi-container* yang pertama docker akan mengambil *image* mysql (db) pada dockerfile yang sudah ada di sub folder 'mysql', kemudian mengisi environment nama dan password yang telah ada. Docker volume digunakan untuk membuat jalan folder 'usr/data' pada *image* docker ke lokal computer. *Container* pada docker memiliki port 3306 dan akan diteruskan pada port 3308 di lokal komputer.

Kemudian *web server* apache (web) untuk membangun imagenya diambil dari menjalankan dockerfile yang telah disiapkan pada sub folder 'apache' dan konfigurasi UID yang telah didefinisikan pada file env. Docker volume akan membuat jalur file dari 'src/var/www/html' ke folder 'src' yang sudah dibuat. Docker akan membuka port 80 pada *container* dan akan dilanjutkan ke port 8000 pada lokal komputer. Pada saat menjalankan *multi-container web server* apache akan running setelah *container* mysql.

Container adminer juga dijalankan dari *image* adminer di docker hub yang telah disediakan dan akan disimpan pada lokal komputer. Docker akan mengekspos port 8080 pada *container* yang sudah dikonfigurasi pada 'docker-compose.yml' untuk dilanjutkan ke port 8080 pada lokal komputer.

Docker volume untuk membuat penyimpanan dapat dikelola dengan mudah dan juga berfungsi mengelola file secara permananen pada docker *container*. Setelah 'docker-compose.yml' siap dieksekusi, selanjutnya menjalankan perintah 'docker-compose up' untuk menjalankan dan menghubungkan semua file dan folder yang ada pada dorektori yang telah dibuat.

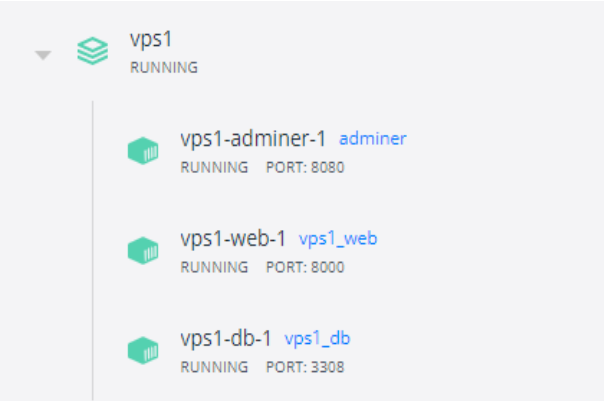
```
PS D:\kuliah\semester8\VPS1> docker-compose up
 - Container vps1-adminer-1  Created
 - Container vps1-db-1       Created
 - Container vps1-web-1      Created
```

Gbr. 10 docker-compose up

Setelah perintah dijalankan docker akan membuat *container* yang berisi tiga *image* yang ada pada 'docker-compose.yml' dan saling tehubung dalam satu jaringan. Berikut adalah *image* dan docker *container* yang dijalankan setelah melakukan perintah 'docker-compose up' pada docker desktop.

NAME		TAG	IMAGE ID
adminer	IN USE	latest	6261a306e265
vps1_db	IN USE	latest	08d3b33599cc
vps1_web	IN USE	latest	80256f1dfba2

Gbr. 11 Image Container



Gbr. 12 Docker container

B. Hasil Pengujian Resource dan Independensi

1. Hasil Pengujian Penggunaan Memori

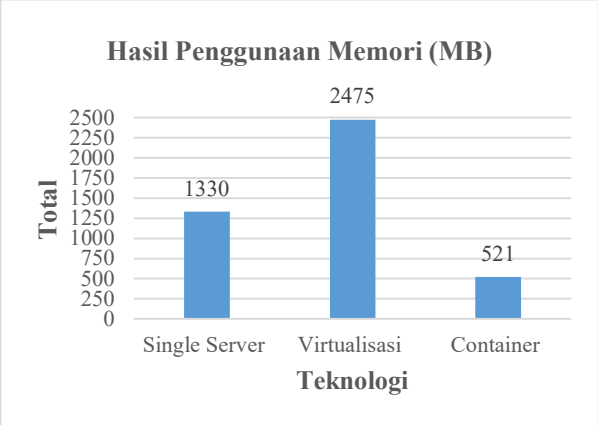
Penggunaan kebutuhan memori yang diambil dengan menjalankan 3 multi docker *container*, virtualisasi server dan *single server* dengan menjalankan *web server* apache.

Data yang diambil dari *resource* yang dibutuhkan untuk menjalankan *web server* ketika proses pengujian berlangsung.

Untuk hasil pengujian penggunaan memori pada penelitian ini dapat dilihat pada table 1 dan gambar 10.

Tabel 1. Hasil Penggunaan Memori (MB)

Teknologi	NAMA WEB			Total
	WEB 1	WEB 2	WEB 3	
Single Server	1.330 MB			1.330 MB
Virtualisasi	819 MB	832 MB	824 MB	2.475 MB
Container	171 MB	167 MB	183 MB	521 MB



Gbr 10. Grafik Hasil Pengujian Penggunaan Memori

Pada grafik hasil pengujian penggunaan kebutuhan memori menunjukkan bahwa kebutuhan *resource container* lebih sedikit dibandingkan dengan teknologi *single server* dan virtualisasi. Untuk hasil pada *container* membutuhkan memori 521 MB , *single server* membutuhkan memori 1330 MB dan virtualisasi membutuhkan memori 2475 MB. Penggunaan memori pada docker *container* lebih sedikit karena pada setiap virtualisasi server docker *container* tidak memuat kernel, melainkan berbagi kernel *host* dengan *container* yang lain. Berbeda dengan virtualisasi dan *single server* yang memiliki kernel sendiri untuk menjalankan aplikasi dalam hal ini menyebabkan kebutuhan *resource* memori *container* lebih kecil.

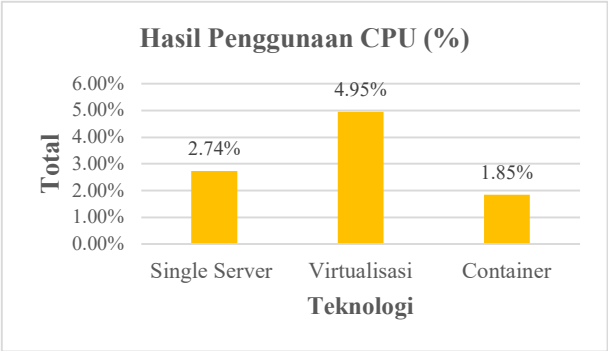
2. *Hasil Pengujian Penggunaan CPU*

Penggunaan kebutuhan CPU yang diambil dengan menjalankan 3 multi docker *container*, virtualisasi server dan *single server* dengan menjalankan *web server* apache. Data yang diambil dari *resource* yang dibutuhkan untuk menjalankan *web server* yang sedang berjalan.

Untuk hasil pengujian kebutuhan CPU untuk pengujian ini mendapatkan hasil pada table 1 dan gambar 10.

Tabel 2. Hasil Penggunaan CPU(%)

Teknologi	NAMA WEB			Total
	WEB 1	WEB 2	WEB 3	
Single Server	2.74 %			2.74 %
Virtualisasi	1.59 %	1.72 %	1.64 %	4.95 %
Container	0.47 %	0.62 %	0.76 %	1.85 %



Gbr 11. Grafik Hasil Pengujian Penggunaan CPU

Pada grafik hasil pengujian penggunaan kebutuhan CPU menunjukkan bahwa kebutuhan *resource container* lebih sedikit dibandingkan dengan teknologi *single server* dan virtualisasi. Untuk hasil pada *container* membutuhkan CPU 1.85 % , *single server* membutuhkan CPU 2.74 % dan virtualisasi membutuhkan CPU 4.95 % .Penggunaan CPU pada docker *container* lebih sedikit karena pada setiap virtualisasi server docker *container* tidak memuat kernel, melainkan berbagi kernel *host* dengan *container* yang lain. Berbeda dengan virtualisasi dan *single server* yang memiliki kernel sendiri untuk menjalankan aplikasi dalam hal ini menyebabkan kebutuhan *resource* CPU *container* lebih kecil.

3. *Hasil Pengujian Independensi*

Hasil pengujian independensi dilakukan untuk mendapatkan hasil perbandingan dari masing-masing teknologi. Dengan menerapkan beberapa kondisi untuk mengetahui pengaruh antar node yang telah diinstal. Berikut ini adalah perbandingan indepensi yang telah dilakukan pada table 3.

Tabel 3. Hasil Perbandingan Independensi

Parameter	Container	Virtualisasi	Single Server
Restart Service	Tidak berpengaruh pada *container* lainnya karena setiap *container* dimuat pada wilayah memorinya sendiri. Jadi *container* yang lain masih berjalan.	Menrestart salah satu service tidak mempengaruhi yang lain karena setiap kernel virtual berada pada memori masing-masing.	Jika salah satu service direstart maka akan mempengaruhi lainnya karena direktori file jadi 1 antar *host* atau klien

Parameter	Container	Virtualisasi	Single Server
Serangan Flooding	Hanya *container* yang terkena serangan flooding yang mengalami performa tingi tidak mempengaruhi performa *container* lain.	Serangan flooding hanya akan mempengaruhi web yang sedang diserang, tidak mempengaruhi web lain yang tidak di serang.	Dengan adanya serangan flooding pada salah satu web, akan mempengaruhi performa semua web yang ada *single server*.
Install beda versi	Dapat di install *image* dengan versi yang berbeda tiap *container*nya	Dapat diinstal aplikasi yang berbeda versi karena penempatannya pada VMnya masing-masing.	Tidak bisa diinstal dua aplikasi yang sama. Hanya bisa diinstal satu aplikasi dengan versi hanya satu

Hasil yang didapatkan setelah melakukan uji indenpendensi pada setiap teknologi adalah *container* dan virtualisasi mempunyai tingkat independensi yang aman karena setiap sumber daya dan aplikasi terisolasi secara terpisah, sehingga tidak menggangu konfigurasi aplikasi yang lain. Berbeda dengan *single server* yang setiap file atau folder tidak terisolasi sehingga dapat mempengaruhi atau mengganggu konfigurasi satu sama lain.

IV. KESIMPULAN

Berdasarkan dari hasil penelitian yang telah dilakukan dengan cara mengukur kebutuhan *resource* memori dan CPU pada teknologi *single server*, virtualisasi dan *container* diperoleh hasil bahwa dari pengujian penggunaan memori, *container* membutuhkan sebesar memori 521 MB, *single server* membutuhkan memori 1330 MB dan virtualisasi membutuhkan memori 2475 MB. Hasil pengujian penggunaan CPU *container* membutuhkan CPU 1.85% , *single server* membutuhkan CPU 2.74% dan virtualisasi membutuhkan CPU 4.95%. Penggunaan memori dan CPU pada docker *container* lebih sedikit karena pada setiap virtualisasi server docker *container* tidak memuat kernel, melainkan berbagi kernel *host* dengan *container* yang lain. Berbeda dengan virtualisasi dan *single server* yang memiliki kernel sendiri untuk menjalankan aplikasi dalam hal ini menyebabkan kebutuhan *resource* memori dan CPU *container* lebih kecil. *Container* dan virtualisasi memiliki independensi yang terjaga, karena sumber daya dan aplikasi terisolasi secara terpisah, sehingga tidak menggangu konfigurasi aplikasi yang lain.

V. SARAN

Berdasarkan dari hasil penelitian yang didapatkan, peneliti memberikan masukkan untuk penelitian kedepannya mengembangkan kembali konfigurasi-konfigurasi docker yang nantinya dibutuhkan dalam menjalankan server web supaya mendapatkan hasil yang lebih efisien.

UCAPAN TERIMA KASIH

Rasa terima kasih dan ucapan rasa syukur peneliti sampaikan kepada :

1. Allah SWT yang Maha Esa atas segala berkah, rahmat dan hidayahnya, sehingga penulis dapat menyelesaikan penelitian ini.
2. Bapak Djoko Prastyo dan Ibu Rusmiwanti selaku orang tua saya yang selalu mendukung serta menyemangati.
3. Ghalib Agrizal, Suadibah, Naura dan Fine Prastika selaku keluarga yang selalu memberikan motivasi dan bimbingan.
4. Bapak Agus Prihanto selaku dosen pembimbing dengan sabar memberi arahan dan membantu penelitian ini dari awal hingga akhir.
5. Naufal Ammar, Wahyu Aldi, Alfian Fahrudi, Farry, Yanu Ade yang sering memberi wawasan dan refrensi untuk dapat menyelesaikan hambatan dalam penelitian.
6. Aditya Wahyu, Genta, Hafidh Guyen, Ubaidillah, Aditya Eka, Naufal Bahar, Arsy Akbar, Faisal Ilham dan teman-teman serta sahabat yang selalu memberikan energi positif.

REFERENSI

[1] D. Marta, M. A. E. Putra, and G. Barovih, "Analisis Perbandingan Performa Virtualisasi Server Sebagai Basis Layanan Infrastructure As A Service Pada Jaringan Cloud," *MATRIK J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 19, no. 1, pp. 1–8, 2019, doi: 10.30812/matrik.v19i1.433.

[2] B. Walidaini and A. M. Muhammad Arifin, "Pemanfaatan Internet Untuk Belajar Pada Mahasiswa," *J. Penelit. Bimbing. dan Konseling*, vol. 3, no. 1, 2018, doi: 10.30870/jpbk.v3i1.3200.

[3] M. Romadlon Bik and Asmunin, "Implementasi Docker Untuk Pengelolaan Banyak Aplikasi Web (Studi Kasus : Jurusan Teknik Informatika Unesa)," *J. Manaj. Inform.*, vol. 7, no. 2, pp. 46–50, 2017.

[4] H. N. A. Panjaitan, "Pengalokasian Resource Beberapa Conainer Pada Proxmox Virtual Environment Sebagai Server Cloud Computing," Sumatera Utara, 2018.

[5] Salman Farizy, "Implementasi Teknologi Virtualisasi Private Server Menggunakan Hyper-V Pada Stmik Pranata Indonesia," *J. Teknol. Inf. ESIT* , vol. 14, no. 1, pp. 31–40, 2019, [Online]. Available: http://www.jurnal-eresha.ac.id/index.php/esit/article/view/91.

[6] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *Prosisko*, vol. 7, no. 2, pp. 165–175, 2020, [Online]. Available: https://e-jurnal.lppmunsera.org/index.php/PROSISKO/article/view/2520/.

[7] Harfadzi and D. Irwan, "Perancangan dan Implementasi Virtualisasi Server Menggunakan Proxmox VE 3.4," vol. 4, no. 2, pp. 89–97, 2016.

[8] I. Hasan, M. Ali, and A. Muhammad Hatta, "Implementasi Teknologi Virtualisasi pada Scada & Control System untuk Efisiensi Energi & Biaya," 2018.

[9] M. Fihri, R. M. Negara, and D. D. Sanjoyo, "Implementasi & Analisis Performansi Layanan Web Pada Platform Berbasis Docker Implementation & Analysis of Web Service Performance Based on Docker Platform," vol. 6, no. 2, pp. 3996–4001, 2019, [Online]. Available: https://libraryeproceeding.telkomuniversity.ac.id/index.php/engineering/article/viewFile/10367/10222.

[10] F. Adiputra, "Container dan Docker : Teknik Virtualisasi dalam Pengelolaan Banyak," vol. 4, no. 3, 2015.

[11] O. A. Atriadi and B. Kristianto, "Perancangan dan Implementasi Virtual Server Sebagai Optimalisasi Penggunaan Hardware Resources (Studi Kasus : SMA Negri 3 Salatiga)," 2013.