Journal of Applied Informatics and Computing (JAIC)
Vol.8, No.1, July 2024, pp. 155~166
e-ISSN: 2548-6861

Comparative Study of Web Server Performance Testing with and without Docker Based on Virtual Machines

Fajar Kurnia Ramadhan [1]*, Garno [2]*, Arip Solehudin [3]*
* Teknik Informatika, Universitas Singaperbangsa Karawang
fajarkurniaramadhan@gmail.com [1], garno@staff.unsika.ac.id [2], arip.solehudin@staff.unsika.ac.id [3]

Article Info

Article history:

Received 2022-02-01
Revised 2024-06-28
Accepted 2024-07-01

Keyword:

Web Server Performance,
Docker,
Virtual Machines,
Performance Testing,
Load Test,
System Infrastructure Development Life Cycle (SIDLC).

ABSTRACT

Web server development is often hindered by the cost and resources required, as developing a web server typically requires a bare-metal server. Container technology, which allows for the development of multiple web servers on a single bare-metal server, has become popular. One of the most widely used containers is Docker. Docker reduces the need for costs and resources. Beyond the issues of cost and resource requirements, the performance of web servers also needs to be considered. The performance of web servers with and without Docker needs to be verified. This research aims to test the performance of two web servers, one using Docker and one not using Docker, utilizing the native hypervisor VMware ESXi. The web server performance test items in this study include CPU and RAM resource usage. The method for developing infrastructure systems uses SIDLC (System Infrastructure Development Life Cycle). Performance testing (Load Test) was conducted using Apache JMeter as a tool, with the manipulation of the number of threads predetermined. Resource usage information was monitored using Prometheus and Grafana. The research results show that with the same resources for each virtual machine, the CPU resource usage of Virtual Machine 2 (Undockerized) is less than that of Virtual Machine 1 (Dockerized). Meanwhile, RAM resource usage is not affected by the number of users on both virtual machines. Virtual Machine 2 (Undockerized) is better at handling HTTP requests. Virtual Machine 1 (Dockerized) can handle only 2,790 users, while Virtual Machine 2 (Undockerized) can handle more than 2,790 users without errors.

This is an open access article under the CC–BY-SA license.

I. PENDAHULUAN

Aplikasi berbasis web sangat populer karena kemudahan aksesnya, pengguna cukup membuka sebuah peramban seperti Mozila Firefox, Google Chrome, atau Microsoft Edge. Web Server menjadi wadah untuk mendapatkan laman web dan data yang berhubungan dengan aplikasi web yang dibuat sehingga bisa diakses pada perangkat yang berbeda [1]. Pada pengembangan yang konvensional satu server fisik hanya bisa menjalankan satu buah web server. Server fisik adalah wujud server yang memiliki berbagai komponen seperti komputer pada umumnya. Server fisik memiliki power supply, mainboard, memory, disk, dan sebagainya [2]. Hal ini mengakibatkan borosnya anggaran, sumberdaya, dan infrastruktur jika kita diharuskan membuat banyak web server [3,4].

Docker keluar sebagai solusi untuk menyelesaikan permasalahan tersebut pada proses *deployment* [5]. Docker bekerja dengan teknik kontainer, kontainer tersebut nantinya akan bertugas untuk memuat dan menjalankan kumpulan *image* yang berisi data konfigurasi dan juga *file* pendukung yang lainnya [6], dengan metode *file system as a layer* yang berarti Docker hanya akan menyalin lapisan perubahannya saja untuk dijalankan sebagai duplikasi *container* yang berbeda dengan *base image* yang sama [7].

Namun selain masalah biaya dan kebutuhan sumber daya, dalam pengembangan web server perlu diperhatikan juga segi performanya [8]. Performa web server dengan Docker dan tanpa Docker perlu dibuktikan. Penelitian ini bertujuan untuk menguji performa dua buah web server yang menggunakan Docker dan yang tidak menggunakan Docker dengan menggunakan teknik Virtualisasi.

Virtualisasi adalah sebuah teknik untuk menyembunyikan karakter fisik dari sumber daya komputer [9], atau bahkan menciptakan perangkat yang tidak ada menjadi ada [7]. Hypervisor dapat dijalankan dengan dua cara: dapat berjalan langsung pada perangkat keras yang disebut *native* Hypervisor atau Tipe-1, atau virtualisasi *bare-metal*, dan yang kedua dapat berjalan di atas sistem operasi mesin *host* yang dikenal sebagai Type-2 atau *hosted hypervisor* [7,8].

Butir uji performa yang dilakukan pada penelitian ini adalah penggunaan sumberdaya CPU dan RAM terhadap jumlah akses pengguna (*load condition*). Akan dibuat dua buah Mesin Virtual (VM), masing-masing server Mesin Virtual akan memiliki spesifikasi *hardware* dan *software* yang sama dan identik satu dengan yang lain, juga mendapatkan skema uji yang sama. Hal ini bertujuan agar *performance testing* dengan jenis *load testing* bersifat adil [12,13]. Pengujian dilakukan lebih dari 1 kali dengan tujuan mendapatkan nilai yang lebih relevan [14].

Uji performa *(load test)* menggunakan *tools* Apache Jmeter [15], dengan rekayasa *number of threads* dan *Ramp-up periode* yang telah ditentukan. Uji performa tidak lepas dari pengamatan dan pengambilan rekaman data [16]. Pengamatan dilakukan menggunakan *tools* dengan sistem terintegrasi yang akan menyediakan informasi dan data dengan bantuan visualisasi [17]. Untuk meraih informasi penggunaan sumberdaya dilakukan monitoring hasil menggunakan Grafana dan Prometheus [18,19]. Prometheus menampilkan data monitoring berupa metriks, diintegrasikan dengan Grafana yang mampu merubahnya menjadi grafik-grafik yang menarik untuk dilihat dan mudah dimengerti [20]. *Tools* ini dipilih karena bersifat *open source* dengan *observability*, dengan *dashboard* yang bisa dimodifikasi, dan mudah di konfigurasi [21].

Dibutuhkan sebuah metode pengembangan agar pengerjaan sistem infrastruktur berjalan secara terstruktur, efektif dan sesuai dengan tujuan yang diinginkan. Metode pengembangan infrastruktur sistem yang dirasa tepat untuk digunakan adalah *System Infrastructure Development Life Cycle [22]*. Metode ini sesuai dengan studi kasus yang akan diteliti. Metode ini mirip dengan model *Waterfall* pada pengembangan SDLC *software* dengan tumpang tindih (berurutan) dan umpan balik pada tiap prosesnya [23].

II. METODE PENELITIAN

Metode Penelitian yang digunakan pada penelitian ini adalah metodologi penelitian kuantitatif dengan SIDLC *(System Infrastructure Development Life Cycle)* sebagai model pengembangannya. Model ini dipilih karena dapat merepresentasikan kebutuhan pada pembuatan infrastruktur pengembangan beserta uji performanya.

A. Requirement Gathering

Pada tahap ini dibuat sebuah pemilihan kebutuhan alat dan bahan secara detail untuk membangun sebuah infrastruktur terkait. Pada tahapan ini juga dijelaskan bagaimana penulis mendapatkan informasi untuk menunjang penelitian ini.

1) Pengumpulan Informasi: Tahap pengumpulan informasi dan data dilakukan dengan metode studi literatur, yaitu penulis mencari referensi-referensi yang relevan dengan objek yang diteliti. Pencarian referensi dilakukan dengan mengumpulkan data dan informasi dari buku, website, dan jurnal yang terkait dengan pokok bahasan penelitian ini.

2) Alat dan Bahan: Dibawah ini merupakan daftar alat dan bahan baik *hardware* maupun *software* yang dibutuhkan dalam penelitian.

a. Spesifikasi *Physical Server*

Untuk *physical server* penelitian ini menggunakan HP ProLiant ML10 sebagai wadah atau tempat untuk menginstal *Native Hypervisor*.

TABEL I
SPESIFIKASI PHYSICAL SERVER

No	Jenis	Nama	Keterangan
1	Server Fisik	HP ProLiant ML10	Processor : Intel Xeon E3-1220 v3 @ 3.10GHz, RAM : 2 x 8 GB, Storage : HDD 4TB
2	Hypervisor	*VMware ESXi*	VMware ESXi 7.0

b. Kebutuhan Mesin Virtual 1 (Dengan Docker)

Untuk Mesin Virtual 1 akan digunakan untuk membangun infrastruktur web server mengunakan Docker, alat dan bahan yang dibutuhkan diperlihatkan dalam Tabel II.

TABEL II
KEBUTUHAN MESIN VIRTUAL 1

No	Jenis	Nama	Keterangan
1	Mesin Virtual	VM1	4 Core Processor, RAM 8GB, Storage HDD 25GB
2	OS Server	Ubuntu	Ubuntu Server 18.04
3	Basis Data	MySQL	MySQL 5.7
4	Bahasa Pemrograman Web	PHP	PHP 8.0 fpm
5	Web Server Service	NginX	NginX 1.21.1
6	Source Code (CMS)	Joomla	Joomla 3.9
7	Container	Docker	Docker Engine, Docker Compose
8	SSH Server	OpenSSH	OpenSSH 8.6
9	Monitoring	Prometheus Grafana	Prometheus 2.32 Grafana 8.3

c. Kebutuhan Mesin Virtual 1 (Tanpa Docker)

Untuk Mesin Virtual 1 akan digunakan untuk membangun infrastruktur web server mengunakan Docker, alat dan bahan yang dibutuhkan diperlihatkan dalam Tabel III.

TABEL III
KEBUTUHAN MESIN VIRTUAL 2

No	Jenis	Nama	Keterangan
1	Mesin Virtual	VM2	4 Core Processor, RAM 8GB, Storage HDD 25GB
2	OS Server	Ubuntu	Ubuntu Server 18.04
3	Basis Data	MySQL	MySQL 5.7
4	Bahasa Pemrograman Web	PHP	PHP 8.0 fpm
5	Web Server Service	NginX	NginX 1.21.1
6	Source Code (CMS)	Joomla	Joomla 3.9
7	SSH Server	OpenSSH	OpenSSH 8.6
8	Monitoring	Prometheus Grafana	Prometheus 2.32 Grafana 8.3

d. Spesifikasi dan Kebutuhan *Client*
Dari sisi *client* digunakan *notebook* atau laptop untuk melakukan uji performa sekaligus pemantauan server yang ada. Spesifikasi dan kebutuhan *client* diperlihatkan dalam tabel IV.

TABEL IV
SPESIFIKASI DAN KEBUTUHAN CLIENT

No	Jenis	Nama	Keterangan
1	Notebook	Lenovo 80f6	Windows 10, Intel Core i5-5200U, RAM 12 GB, Storage SSD 256
2	Web Browser	Google Chrome	Versi 92.0.4
3	SSH Remote Tools	PowerShell	Versi 5.1
4	Databases Management Tools	DBeaver	Versi 21.1
5	Performance Testing Tools	Apache Jmeter	Versi 5.4.1

B. Analysis

Pada tahapan ini menjelaskan tentang analisis, cara kerja atau skenario di setiap pemilihan teknologi atau cara yang dipakai untuk menyelesaikan penelitian ini.

1) Analisis Mesin Virtual. VMware ESXi diinstalasi pada *physical server*, setelah itu VMware Esxi akan membuat dua Mesin Virtual baru, dimana masing masing Mesin Virtual akan memiliki spesifikasi yang sama dan identik satu dengan yang lain, baik Mesin Virtual 1 maupun Mesin Virtual 2 sama sama menggunakan Ubuntu Server 18.04 sebagai sistem operasinya.

2) Analisis Web Server. Pada Mesin Virtual 1 dan Mesin Virtual 2, aplikasi web yang dipakai adalah *Joomla Content Management Service* yang sudah diisikan beberapa konten penunjang didalamnya, *source code* Joomla akan dijalankan menggunakan *service* dari Nginx Web Server, yang membedakan Mesin Virtual 1 dengan Mesin Virtual 2 adalah teknik untuk mengembangkan aplikasi web, pada Mesin Virtual 1 aplikasi web akan berjalan menggunakan Docker, sedangkan pada Mesin Virtual 2 aplikasi web langsung berjalan di atas *host* sistem operasi.

3) Analisis Performance Test. Pada masing masing Mesin Virtual mendapatkan skema uji yang sama, hal ini bertujuan agar *performance testing* dengan jenis *load testing* bersifat adil. Sebelumnya penelitian ini telah dilakukan pengujian menggunakan *jumlah number of thread (user)* dibawah 1000 dengan kelipatan 100 data yang dihasikan pada server berjalan dengan baik. Oleh karena itu, agar data yang digunakan pada penelitian ini tidak terlalu banyak dan menghsilkan data yang mudah dipahami maka penelitian ini memulai jumlah *number of thread* (*user*) dengan kelipatan 500 dimulai dengan 1000 *user*.

Pada percobaan ke lima yang seharusnya jumlah *number of thread (user)* adalah 3000, salah satu server tidak sanggup memenuhi HTTP *request* yang ditandai dengan terjadinya *report error* pada hasil pengujian Apache Jmeter, sehingga penulis mencoba mencari nilai maksimal server untuk sanggup menghandle HTTP *request*. Setelah dilakukan beberapa uji coba diperoleh jumlah *user* maksimal sebesar 2790. Jadi, jumlah *number of thread (user)* yang digunakan adalah 1000, 1500, 2000, 2500, 2790, 2791, dan 3000. Dengan adanya hasil *error* membuktikan bahwa sampai titik itulah server diklaim mampu menangani banyaknya *user (number of thread)* secara maksimal dan *load test* akan dihentikan.

Penentuan *ramp-up periode* didasari oleh satuan waktu yang mudah untuk dianalisis. Untuk itu penelitian ini menggunakan *ramp-up periode* 60 detik atau satu menit, dalam contoh kasusnya penelitian akan dimulai pada pukul 20.01:45 dan berakhir pada 20.02:45. Sehingga mudah untuk dipantau, karena waktu awal dan akhir pengujian akan selalu berubah.

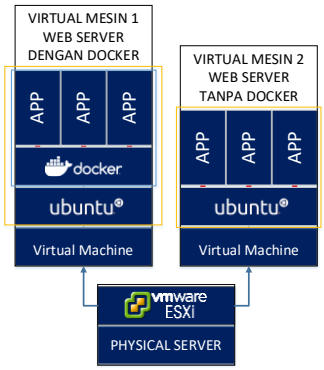
Selain itu pengujian juga dilakukan sebanyak tiga kali percobaan agar mendapatkan hasil yang lebih akurat dan melihat apakah performa server stabil di antara pengujian pertama, kedua dan ketiga dengan jumlah *user* dan waktu yang sama. Pengujian dilakukan saat CPU maupun RAM dalam kondisi *idle time*, dimana tidak ada aplikasi atau service lain yang bekerja ekstra sehingga uji performa didapatkan hasil yang baik.

4) Analisis Monitoring. Monitoring Server dilakukan untuk melihat sekaligus mengambil informasi penggunaan sumberdaya CPU dan RAM yang terpakai pada server, digunakan software Prometheus dan Grafana untuk memonitoring Mesin Virtual satu dan dua, Prometheus dan Grafana akan saling terkait satu sama lain, dan dibutuhkan node exporter dashboard untuk mempermudah *monitoring* dan pengambilan data sumberdaya tersebut.

C. Design

Pada tahapan ini akan dibuat design dari infrastruktur yang akan dibuat yaitu infrastruktur pengembangan web server dengan Docker, infrastruktur pengembangan web server tanpa Docker, *design* uji *performance* dengan jenis *load test*, dan topologi jaringan proses akses server-client.

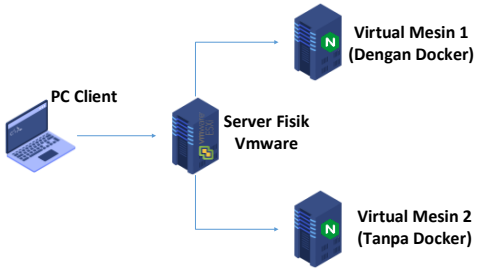
1) Design Infrastruktur Server dan Mesin Virtual :



Gambar 1 Design Infrastruktur Server dan Mesin Virtual

Gambar 1 menjelaskan pada Server Fisik akan terinstalasi VMware ESXi 7.0 sebagai Native Hypervisor, VMware ESXi akan membuat 2 buah VM (Mesin Virtual). Pada Mesin Virtual 1 akan terinstalasi segala kebutuhan untuk membangun sebuah web server sebagaimana telah di sampaikan pada Tabel I dan Pada Mesin Virtual 2 akan terinstal segala kebutuhan untuk membangun sebuah web server sebagaimana telah dijelaskan pada TABEL II. Perbedaannya terletak pada Mesin Virtual 1 diinstalasi Docker Container sedangkan pada Mesin Virtual 2 tidak diinstalasi Docker Container.

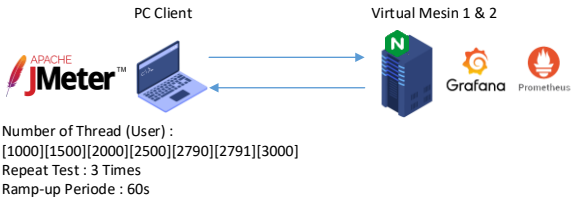
2) Topologi *Jaringan:*



Gambar 2 Topologi / Design Jaringan

Gambar 2 menjelaskan Topologi yang dipakai pada penelitian ini adalah Peer to Peer Conection, PC Client akan terkoneksi langsung dengan Physical Server menggunakan Kabel UTP.

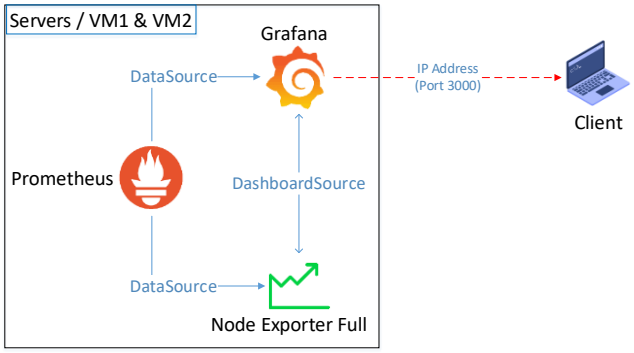
3) Design Performance Test:



Gambar 3 Skenario Uji Performa

Gambar 3 menjelaskan *Software Performance Tools* yang dipakai pada penelitian ini adalah Apache Jmeter, Apache Jmeter akan melakukan *Load testing* dengan mensimulasikan *number of threads* (jumlah *user*) dan *Ramp-up periode* (waktu yang dibutuhkan) untuk mengakses atau mendapatkan HTTP Request dari sebuah web server pada setiap Mesin Virtual.

4) Design Monitoring dan Pengambilan Data :



Gambar 4 Alur Pengamatan dan Pengambilan Data

Gambar 4 menjelaskan alur kerja dalam pengamatan dan pengambilan data, Grafana akan mengambil datasource dari Prometheus. Untuk mempermudah pengamatan dan pengambilan data, digunakan custom dashboard dari Node Exporter Full. Grafana yang sudah diinstall pada Mesin Virtual 1 dan 2 dapat diakses melalui client dengan mengunjungi IP Address VM1 dan VM2 dengan port 3000.

D. Testing

Pada tahap ini dilakukan simulasi pembangunan infrastruktur dengan menggunakan VirtualBox, simulasi ini bertujuan untuk meminimalisir terjadinya *error* pada saat di implementasikan di kondisi nyata. Setelah simulasi berjalan baik maka implementaasi terhadap keadanya nyata dapat di lakukan.

Pada tahapan ini dilakukan juga sebuah uji validasi fungsionalitas atau sering disebut dengan *blackbox testing* dari setiap proses infrastruktur yang akan dibuat, dari instalasi hypervisor sampai pelaksanaan *performance test.*

1) Uji Fungsionalitas Hypervisor VMware ESXi : Pada tahap ini dilakukan beberapa point uji fungsionalitas terhadap VMware ESXi, uji fungsionalitas ini berfungsi sebagai validasi bahwa system yang sedang dijalankan berjalan dengan baik. Hasil uji fungsionalitas Hypervisor VWWare ESXi dapat dilihat pada Tabel V.

TABEL V
UJI FUNGSIONALITAS HYPERVISOR VMWARE ESXI

No	Butir Uji Fungsionalitas	Validasi	
		Ya	Tidak
1	Hypervisor ESXi terinstal dengan baik	V	
2	Hypervisor ESXi mendapatkan Ip address	V	
3	Hypervisor ESXi terkoneksi dengan internet	V	
4	Ip address Hypervisor ESXi bisa diakses di web browser client	V	
5	Client dapat login ke Hypervisor ESXi	V	
6	Hypervisor ESXi dapat membuat Virtual machine	V	

2) Uji Fungsionalitas Mesin Virtual 1: Pada tahap ini dilakukan beberapa point uji fungsionalitas terhadap Mesin Virtual 1, uji fungsionalitas ini berfungsi sebagai validasi bahwa system yang sedang dijalankan berjalan dengan baik. Hasil uji fungsionalitas Mesin Virtual 1 dapat dilihat pada Tabel VI.

TABEL VI
UJI FUNGSIONALITAS MESIN VIRTUAL 1

No	Nama	Butir Uji Fungsionalitas	Validasi	
			Ya	Tidak
1	Ubuntu	Sistem Operasi Ubuntu berhasil terinstall dengan baik	V	
		Ubuntu mendapatkan Ip address	V	
		Ubuntu dapat terkoneksi dengan internet	V	
2	Docker	Nginx dapat terinstal dengan baik di Docker	V	
		Landing page Nginx bisa di akses pada browser client dengan memasukan IP Adrress Host OS	V	
		Dapat memeriksa versi Nginx pada Docker	V	
		Servis Nginx dalam keadaan berjalan di Systemctl	V	
		PHP dapat terinstal dengan baik di Docker	V	
		Dapat memeriksa versi PHP	V	
		Servis PHP dalam keadaan berjalan di Systemctl	V	
		Versi PHP bisa terlihat di landing page Nginx	V	
		Docker terinstall dengan baik	V	
		Dapat memeriksa versi docker di Host OS	V	
		Dapat mendownload docker image	V	
		Dapat membuat docker container	V	
		Docker compose terinstall dengan baik	V	
		Dapat melihat versi docker compose	V	
		Dapat menjalankan docker compose	V	
3	MySQL	Mysql dapat terinstall dengan baik di Ubuntu OS	V	
		Dapat memeriksa versi Mysql	V	
		Servis Mysql dalam keadaan berjalan di Systemctl	V	
		Dapat login ke Mysql	V	
		Dapat dilakukan remote database	V	
		Dapat berjalan dan terkoneksi dengan PHP	V	
4	Open SSH	SSH dapat terinstall dengan baik di Ubuntu OS	V	
		SSH dapat diremote melalui client	V	
5	Grafana	Grafana terinstall denganbaik	V	
		Dapat memeriksa versi Grafana	V	
		Servis Grafana dalam keadaan berjalan di Systemctl	V	
		Grafana dapat di akses dari client	V	
		Grafana dapat login	V	
		Grafana dapat menambahkan datasource prometheus	V	
		Grafana dapat terkoneksi dengan prometheus	V	
		Grafana dapat menampilkan dashboard penggunaan resource CPU dan RAM	V	
		Grafana dapat membuat report data resource	V	
6	Promet-heus	Prometheus terinstall dengan baik	V	
		Dapat memeriksa versi Prometheus	V	
		Service Prometheus dalam keadaan berjalan di systemctl	V	
		Prometheus dapat di akses dari client	V	

3) Uji Fungsionalitas Mesin Virtual 2: Pada tahap ini dilakukan beberapa point uji fungsionalitas terhadap Mesin Virtual 2, uji fungsionalitas ini berfungsi sebagai validasi bahwa system yang sedang dijalankan berjalan dengan baik. Hasil uji fungsionalitas Mesin Virtual 2 dapat dilihat pada Tabel VII.

TABEL VII
UJI FUNGSIONALITAS MESIN VIRTUAL 2

No	Nama	Butir Uji Fungsionalitas	Validasi	
			Ya	Tidak
1	Ubuntu	Sistem Operasi Ubuntu berhasil terinstall dengan baik	V	
		Ubuntu mendapatkan Ip address	V	
		Ubuntu dapat terkoneksi dengan internet	V	
2	Nginx	Nginx dapat terinstal dengan baik di Ubuntu OS	V	
		Landing page Nginx bisa di akses pada browser client dengan memasukan IP Adrress Host OS	V	
		Dapat memeriksa versi Nginx pada Host OS	V	
		Servis Nginx dalam keadaan berjalan di Systemctl	V	
3	PHP	PHP dapat terinstal dengan baik di Ubuntu OS	V	
		Dapat memeriksa versi PHP	V	
		Servis PHP dalam keadaan berjalan di Systemctl	V	
		Versi PHP bisa terlihat di landing page Nginx	V	
4	MySQL	Mysql dapat terinstall dengan baik di Ubuntu OS	V	
		Dapat memeriksa versi Mysql	V	

		Servis Mysql dalam keadaan berjalan di Systemctl	V	
		Dapat login ke Mysql	V	
		Dapat dilakukan remote database	V	
		Dapat berjalan dan terkoneksi dengan PHP	V	
5	Open SSH	SSH dapat terinstall dengan baik di Ubuntu OS	V	
		SSH dapat diremote melalui client	V	
6	Grafana	Grafana terinstall dengan baik	V	
		Dapat memeriksa versi Grafana	V	
		Servis Grafana dalam keadaan berjalan di Systemctl	V	
		Grafana dapat di akses dari client	V	
		Grafana dapat login	V	
		Grafana dapat menambahkan datasource prometheus	V	
		Grafana dapat terkoneksi dengan prometheus	V	
		Grafana dapat menampilkan dashboard penggunaan resource CPU dan RAM	V	
		Grafana dapat membuat report data resource	V	
7	Promet-heus	Prometheus terinstall dengan baik	V	
		Dapat memeriksa versi prometheus	V	
		Service Prometheus dalam keadaan berjalan di systemctl	V	
		Prometheus dapat di akses dari client	V	

4) Uji Fungsionalitas Apache Jmeter: Pada tahap ini dilakukan beberapa point uji fungsionalitas terhadap Apache Jmeter, uji fungsionalitas ini berfungsi sebagai validasi bahwa aplikasi yang sedang dijalankan berjalan dengan baik. Hasil uji fungsionalitas Apache Jmeter dapat dilihat pada Tabel VIII.

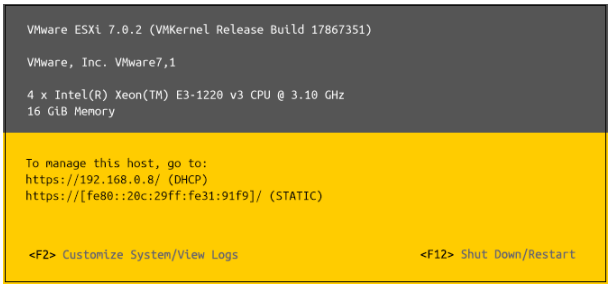
TABEL VIII
UJI FUNGSIONALITAS APACHE JMETER

No	Butir Uji Fungsionalitas	Validasi	
		Ya	Tidak
1	Apache Jmeter terinstall dan dapat dijalankan di Client	V	
2	Apache Jmeter dapat menambahkan Thread Group	V	
3	Apache Jmeter dapat menambahkan Sampler (http request)	V	
4	Apache Jmeter dapat menambahkan Listener (summary report dan result table)	V	
5	Apache Jmeter dapat terhubung dengan Mesin Virtual	V	

E. Implementation

Setelah mendapatkan informasi, analisis, *testing* simulasi yang cukup, maka dilanjutkan penerapan konsep konsep tersebut kedalam pengembangan infrastructure dan pelaksanaan uji performa pada kondisi nyata.

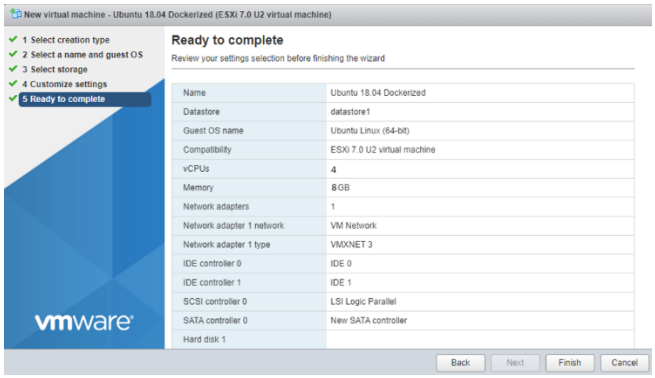
1) Installasi VMware ESXi : VMware ESXi 7.0.2 diinstal pada Physical Server, ISO Image File harus disiapkan terlebih dahulu dan dijadikan media bootable, mounting media bootable dan setting menjadi 1st boot dalam BIOS.



Gambar 5 VMware ESXi Berhasil di Instal

Gambar 5 menjelaskan tahap akhir dari proses installasi yang sudah selesai, tertera alamat IP Address VMware ESXi dimana kita dapat memanagenya melalui browser Client

2) Pembuatan Virual Mesin: Setelah proses installasi selesai VMware ESXi siap untuk digunakan, Client akan mengunjungi alamat IP VMware ESXi di browser untuk memanage Mesin Virtual, akan dibuat dua buah Mesin Virtual dimana masing masing Mesin Virtual akan memiliki spesifikasi 4 Core Processor, RAM 8GB, dan Storage HDD 25GB.



Gambar 6 Pembuatan Mesin Virtual

Gambar 6 menjelaskan tahap akhir dari proses pembuatan Mesin Virtual yang sudah selesai. Setelah pembuatan Mesin Virtual 1 dan Mesin Virtual 2 selesai, nyalakan Mesin Virtual.

3) Instalasi VMware 1: Pada Mesin Virtual ini Ubuntu Server 18.04 digunakan sebagai system operasi, dan akan terinstal Prometheus dan grafana selaku *monitoring tools*, pada Mesin Virtual 1 web server akan di deploy

menggunakan Docker. Berikut ini adalah langkah untuk menginstal Docker *engine* dan Docker Compose.

```
docker@vm1: ~$ sudo apt install
apt-transport-https ca-
certificates curl software-
properties-common
docker@vm1: ~$ curl -fsSL
https://download.docker.com/linux
/ubuntu/gpg | sudo apt-key add -
docker@vm1: ~$ sudo add-apt-
repository "deb [arch=amd64]
https://download.docker.com/linux
/ubuntu $(lsb_release -cs)
stable"
docker@vm1: ~$ apt-cache policy
docker-ce
docker@vm1: ~$ sudo systemctl
status docker
docker@vm1: ~$ sudo curl -L
https://github.com/docker/compose
/releases/download/1.21.2/docker-
compose-`uname -s`-`uname -m` -o
/usr/local/bin/docker-compose
docker@vm1: ~$ sudo chmod +x
/usr/local/bin/docker-compose
```

Gambar 7 Instalasi Docker pada Terminal Ubuntu

Gambar 7 menunjukan cara instalasi Docker *engine* dan Docker Compose pada terminal Ubuntu. Setelah selesai menginstal Docker dan Docker Compose, akan dibentuk struktur direktori dan *file* yang diperlihatkan Gambar 8.

```
home
└──user
    └──docker
        └──docker-joomla
            │   belajar.sql
            │   docker-compose.yml
            │   Dockerfile
            │
            ├──belajar (source code aplikasi web)
            ├──mysql
            │       my.cnf
            │
            ├──nginx
            │   └──conf.d
            │           app.conf
            │
            └──php
                    local.ini
```

Gambar 8 Struktur Folder dan File

Gambar 8 menunjukan struktur folder dan file yang dibuat pada penelitian kali ini untuk melakukan deployment web server dengan Docker dan Docker Compose

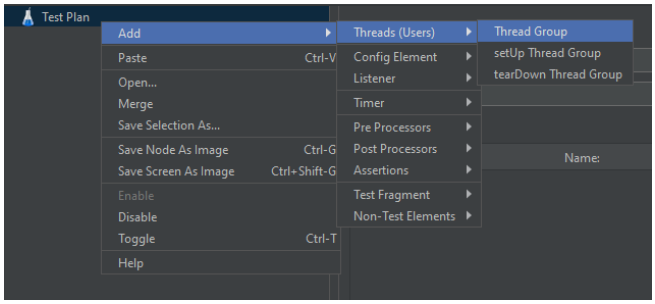
4) Installasi VMware 2: Pada Mesin Virtual ini Ubuntu Server 18.04 digunakan sebagai sistem operasi, dan akan terinstal Prometheus dan Grafana selaku *monitoring tools*, pada Mesin Virtual 2 web server akan dijalankan dengan menginstal aplikasi pendukungnya secara langsung pada sistem operasi. Beberapa aplikasi yang dibutuhkan seperti Nginx, PHP, dan MySQL

```
==============instalasi nginx==============
docker@vm2: ~$ sudo apt install nginx
docker@vm2: ~$ sudo ufw allow 'Nginx HTTP'
docker@vm2: ~$ systemctl status nginx
docker@vm2: ~$ sudo ln -s /etc/nginx/sites-
available/belajar /etc/nginx/sites-enabled/
==============instalasi php================
docker@vm2: ~$ sudo apt install php-fpm php-
mysql
==============instalasi MySQL==============
docker@vm2: ~$ sudo apt install mysql-server
```

Gambar 9 Installasi Nginx, PHP, dan MySQL

Gambar 9 menunjukan langkah langkah instalasi Nginx, PHP, dan MySQL.Yang harus diperhatikan dalam konfigurasi Nginx adalah file isi file dari sites-available, pastikan mengarah kepada alamat nameserver dan base directori file yang tepat. Sedangkan yang harus diperhatikan dalam konfigurasi PHP terletak pada *file* php.ini, sesuaikan dengan kebutuhan minimum dari aplikasi web yang akan dijalankan, pada penelitian ini aplikasi web yang digunakan adalah Joomla CMS versi 3.9, Joomla merekomendasikan *memory limit = 128M, upload max filesize = 30M, post max size = 30M, dan max_execution_time = 30*

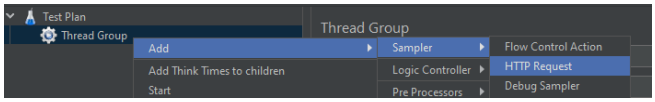
5) Performance Test: Apache Jmeter digunakan untuk melakukan Uji Performa dengan metode *Load test* pada penelitian kali ini. Pada Apache Jmeter akan dibuatkan sebuah *Thread group*.



Gambar 10 Penambahan Thread Group

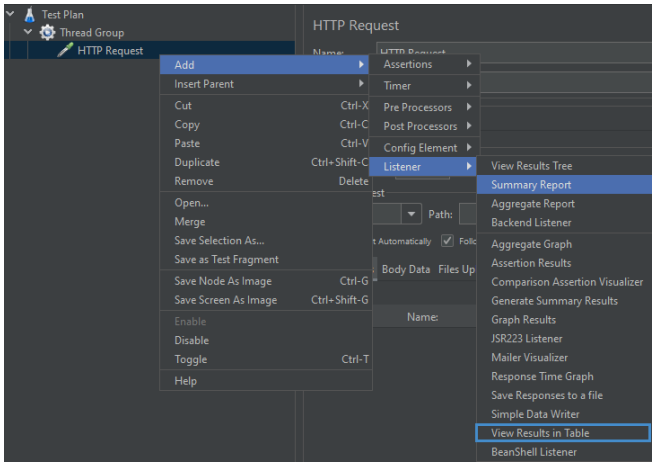
Gambar 10 menunjukan prosesi penambahan *thread group*. *Thread group* mendefinisikan kumpulan pengguna yang akan menjalankan kasus uji tertentu terhadap server. *Thread group* pada Apache Jmeter dapat memberi nama dari sebuah kumpulan utas, mengontrol jumlah pengguna yang disimulasikan *(number of thread)*, berapa lama waktu yang dibutuhkan untuk memulai sebuah utas *(ramp-up periode)*, berapa kali perulangan untuk melakukan *pengujian (loop count)*, dan secara opsional, mulai dan menghentikan waktu untuk ujian.

Setelah Penambahan *Thread Group* ditambahkan pula sampler HTTP *Request.*



Gambar 11 Penambahan Sampler HTTP Request

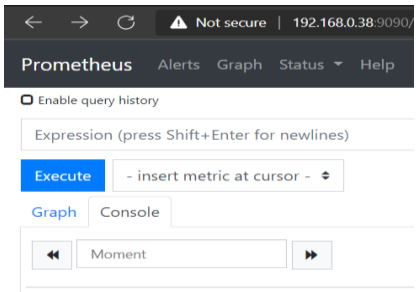
Gambar 11 menunjukan proses penambahan HTTP *request*. Pada HTTP/s *Request* kita diwajibkan untuk mengisi alamat domain atau Ip address yang akan diuji, serta pengisian port protocol atau path direktori website jika diperlukan. Selanjutnya menambahkan opsi Listener yang berfungsi sebagai fitur pemantauan yang berguna untuk melihat data laporan dari sebuah uji yang sedang berlangsung. butir listener yang digunakan untuk pengujian kali ini adalah *Summary Report* yang berfungsi untuk melihat *Number of thread*dan juga tingkat *error* dalam satuan persen yang sedang berjalan ketika sedang dilakukan pengujian, dan *View result in table* yang berguna untuk melihat jumlah *Number of thread* yang gagal maupun yang berhasil dalam melakukan pengujian.



Gambar 12 Penambahan Listener

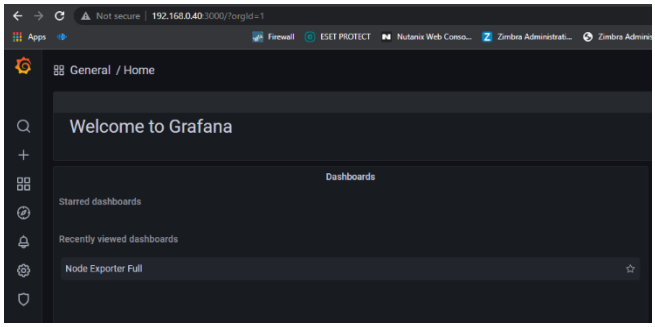
Gambar 12 menunjukan proses penambahan *Listener*, dengan butir *summary report* dan *view result in table*.

6) Pengamatan dan Pengambilan Data: Tools yang digunakan pada penelitian kali ini adalah Prometheus dan Grafana.



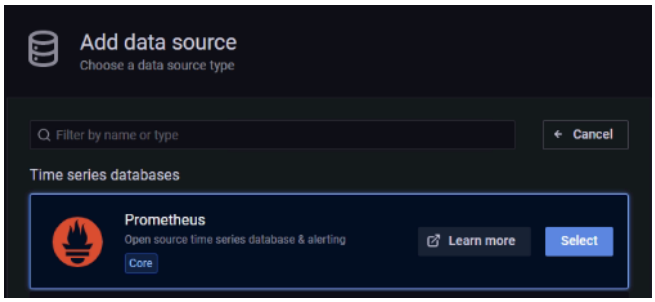
Gambar 13 Tampilan Awal Prometheus

Gambar 13 menjelaskan tampilan awal dari Prometheus yang diakses melalui web browser client dengan port 9090.



Gambar 14 Tampilan Awal Grafana

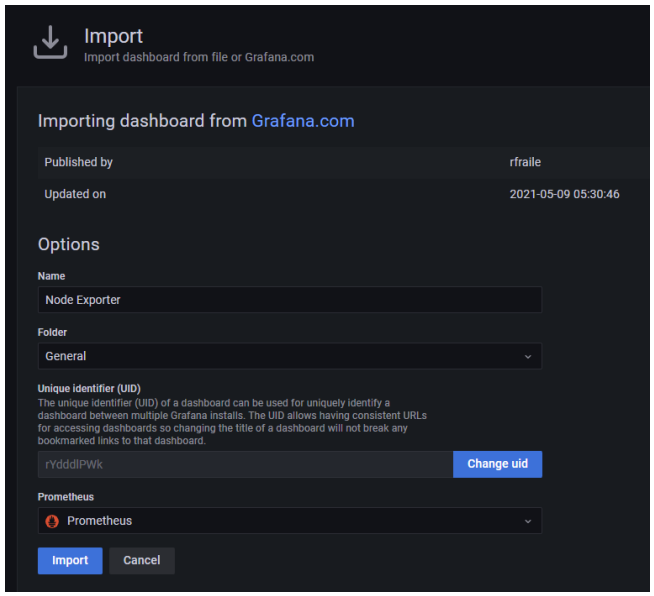
Gambar 14 menjelaskan tampilan awal dari Grafana yang diakses melalui web browser client dengan port 3000.



Gambar 15 Penambahan Data Source Prometheun kedalam Grafana

Gambar 15 menjelaskan proses penambahan *data source* Prometheus, Grafana perlu menambahkan *data source* Prometheus agar dapat membacanya kedalam sistem grafana.

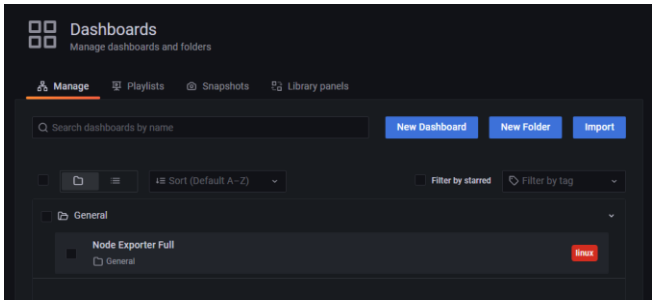
Untuk mempermudah pengamatan dan pengambilan data, digunakan custom dashboard dari Node Exporter Full.



Gambar 16 Proses Penambahan Node Exporter Full

Gambar 16 penambahan *custom dashboard Node Exporter Full by rfraile*, Dashboard ID = 1860. Setelah proses import

berhasil dilakukan, Grafana akan beralih ke halaman *manage dashboard*.



Gambar 17 Pemilihan Node Exporter Full sebagai Dashboar Monitoring

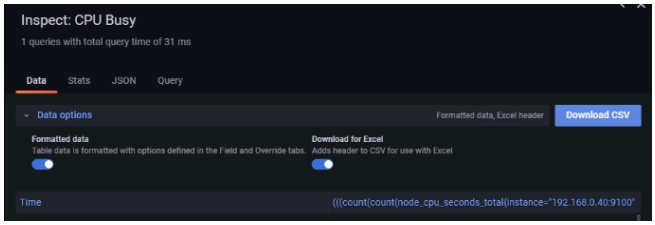
Gambar 17 menunjukan proses penggunaan dashboard dari Node Exporter Full untuk memonitoring server, berikut tampilan dari Node Exporter Full Dashboard

Gambar 18 Tampilan Dashboard Node Exporter Full

Gambar 18 menjelaskan tampilan dari Node Exporter Full, butir pemantauan yang dipakai adalah CPU *Busy* (penggunaan sumber daya CPU) dan RAM *Used* (penggunaan sumber daya RAM). Untuk mengambil data penggunaan sumber daya CPU maupun RAM, pilih salah satu butir monitoring, lalu pilih inspect data

Gambar 19 Memeriksa Data

Gambar 19 menunjukan cara untuk memeriksa data, Node Exporter Full dapat menarik informasi detail tentang konsumsi penggunaan sumberdaya CPU dengan interval update setiap 1 detik, sedangkan sumberdaya memory (RAM) setiap 2 detik, data inilah yang dipakai penulis untuk mengumpulkan informasi penggunaan sumberdaya CPU maupun RAM pada saat dilakukan uji performa. Untuk memudahkan dalam pembacaan data, node exporter full akan mengeksport data menjadi file CSV (Comma Separated Values).



Gambar 20 Eksport data ke CSV Files

Gambar 20 menunjukan proses export / unduh data yang berhasil didapatkan oleh *monitoring tools* ke *file* CSV dengan *template* yang bisa dibaca menggunakan Microsoft Excel.

III. HASIL DAN PEMBAHASAN

Banyaknya *Number of threads* (jumlah *user*) yang digunakan pada penelitian ini adalah 1000, 1500, 2000, 2500, 2790, 2791, dan 3000. *Ramp-up* Periode (waktu yang dibutuhkan) adalah 60 detik.

Hasil uji performa didapatkan dengan cara menarik data dari system monitoring yang sudah dibuat, terdapat 3 kali perulangan pada tiap-tiap kelompok *number of thread* (*user*) pada penelitian ini, dengan tujuan untuk mendapatkan validitas serta stabilitas uji. Pengujian dilakukan saat CPU maupun RAM dalam kondisi idle time, dimana tidak ada aplikasi atau service lain yang bekerja ekstra sehingga uji performa didapatkan hasil yang baik.,

A. Penggunaan Sumber Daya CPU di Mesin Virtual 1 (Dengan Docker)

Pada penggunaan sumberdaya CPU di Mesin Virtual 1 (dengan Docker) terdapat kenaikan nilai penggunaan

sumberdaya CPU yang konsisten saat dilakukan uji performa dengan *user* 1000 sampai dengan 2790, namun pada saat jumlah *user* diatas 2790 terdapat inkonsistensi dalam penggunaan sumberdaya CPU. Hasil penggunaan sumber daya CPU Mesin Virtual 1 dapat dilihat pada Tabel IX.

TABEL IX
PENGGUNAAN SUMBER DAYA CPU VM1

No	Perlakuan (Jumlah User)	Perulangan (%)			Rata-Rata (%)
		1	2	3	
1	1000	21,71	27,94	20,36	23,34
2	1500	30,52	29,09	32,34	30,65
3	2000	40,89	39,23	42,86	40,99
4	2500	52,43	51,97	53,83	52,74
5	2790	54,83	51,40	54,07	53,43
6	2791	54,99	48,50	48,98	50,82
7	3000	49,14	53,12	51,56	51,27

B. Penggunaan Sumber Daya RAM di Mesin Virtual 1 (Dengan Docker)

Pada penggunaan sumberdaya RAM di Mesin Virtual 1 (dengan Docker) terdapat kenaikan nilai penggunaan sumberdaya RAM saat dilakukan uji performa dengan *user* 1000 sampai dengan 2790, namun pada saat jumlah *user* diatas 2790 terjadi penurunan dalam penggunaan sumberdaya RAM. Dengan melihat nilai penggunaan RAM dari hasil pengujian tersebut maka dapat disimpulkan bahwa banyaknya jumlah *user* tidak mempengaruhi penggunaan RAM pada Mesin Virtual 1. Hasil penggunaan sumber daya RAM pada Mesin Virtual 1 dapat dilihat pada Tabel X.

TABEL X
PENGGUNAAN SUMBER DAYA RAM VM1

No	Perlak-uan (Jumlah *User*)	Perulangan (MiB)			Rata-Rata (MiB)
		1	2	3	
1	1000	931,43	941,30	930,53	934,42
2	1500	949,70	954,97	957,77	954,14
3	2000	970,53	982,07	989,37	980,66
4	2500	973,57	989,73	1000,07	987,79
5	2790	1124,00	1142,00	1155,00	1140,33
6	2791	841,17	840,83	844,97	842,32
7	3000	806,90	815,00	837,50	819,80

C. Penggunaan Sumber Daya CPU di Mesin Virtual 2 (Tanpa Docker)

Pada penggunaan sumberdaya CPU di Mesin Virtual 2 (tanpa Docker) terdapat kenaikan nilai penggunaan sumberdaya CPU yang konsisten saat dilakukan uji performa dengan *user* 1000 sampai dengan 3000. Hasil penggunaan sumber daya CPU pada Mesin Virtual 2 dapat dilihat pada Tabel IX.

TABEL XI
PENGGUNAAN SUMBER DAYA CPU VM2

No	Perlakuan (Jumlah *User*)	Perulangan (%)			Rata-Rata (%)
		1	2	3	
1	1000	7,38	6,87	6,99	7,08
2	1500	10,27	10,48	10,24	10,33
3	2000	13,58	13,50	14,35	13,81
4	2500	18,29	17,32	18,85	18,15
5	2790	20,15	19,38	21,24	20,26
6	2791	21,26	20,65	21,87	21,26
7	3000	21,28	21,43	21,27	21,33

D. Penggunaan Sumber Daya RAM di Mesin Virtual 2 (Tanpa Docker)

Pada penggunaan sumberdaya RAM di Mesin Virtual 2 (tanpa Docker) terjadi ketidakstabilan penggunaan sumber daya RAM yang digunakan pada setiap perlakuan jumlah *user*. Hal ini dapat disimpulkan bahawa jumlah user tidak berpengaruh terhadap penggunaan RAM pada Mesin Virtual 2. Hasil penggunaan sumber daya RAM pada Mesin Virtual 2 dapat dilihat pada Tabel IX.

TABEL XII
PENGGUNAAN SUMBER DAYA RAM VM2

No	Perlakuan (Jumlah User)	Perulangan (MiB)			Rata-Rata (MiB)
		1	2	3	
1	1000	755,83	754,57	754,00	754,80
2	1500	554,17	565,40	567,33	562,30
3	2000	751,60	759,23	758,07	756,30
4	2500	626,17	759,23	633,50	672,97
5	2790	632,00	633,77	637,33	634,37
6	2791	576,97	576,47	583,73	579,06
7	3000	756,33	756,40	755,00	755,91

E. Hasil Error pada Uji Performa Web Server Mengunakan Docker

Terjadi nilai *error* yang ditampilkan oleh Apache Jmeter pada saat dilakukan uji performa pada Mesin Virtual 1 (dengan Docker), pada pengujian 2791 *user* hingga 3000 *user* terjadi *error* dan sejalan dengan meningkatnya jumlah *user*, *presentase error* juga meningkat hal ini dapat diartikan jika Mesin Virtual 1 (dengan Docker) tidak dapat menerima HTTP *request* dengan jumlah *user* lebih dari 2790 *user*. Tabel XIII menunjukan hasil pengujian *error* pada Mesin Virtual 1.

TABEL XIII
HASIL ERROR DENGAN DOCKER

No	Perlakuan (Jumlah User)	Nilai Error Perulangan (%)			Rata Rata Error (%)
		1	2	3	
1	1000	0,00	0,00	0,00	0,00
2	1500	0,00	0,00	0,00	0,00

3	2000	0,00	0,00	0,00	0,00
4	2500	0,00	0,00	0,00	0,00
5	2790	0,00	0,00	0,00	0,00
6	2791	0,18	0,29	0,21	0,23
7	3000	6,57	7,13	6,40	6,70

F. Hasil Error pada Uji Performa Web Server Tanpa Mengunakan Docker

Tidak ada *error* yang ditampilkan oleh Apache Jmeter pada pengujian jumlah *user* lebih dari 2790 hingga 3000 *user* pada Mesin Virtual 2 (tanpa Docker). Hal ini dapat diartikan bahwa Mesin Virtual 2 mengirim dan menerima HTTP *request* lebih dari 2790 *user* dengan baik. Tabel XIV menunjukan hasil pengujian *error* pada Mesin Virtual 2.

TABEL XIV
HASIL ERROR TANPA DOCKER

No	Perlakuan (Jumlah *User*)	Nilai Error Perulangan (%)			Rata Rata Error (%)
		1	2	3	
1	1000	0,00	0,00	0,00	0,00
2	1500	0,00	0,00	0,00	0,00
3	2000	0,00	0,00	0,00	0,00
4	2500	0,00	0,00	0,00	0,00
5	2790	0,00	0,00	0,00	0,00
6	2791	0,00	0,00	0,00	0,00
7	3000	0,00	0,00	0,00	0,00

IV. KESIMPULAN

Dengan perlakuan jumlah *user*, waktu, dan perulangan yang sama, terjadi kenaikan penggunaan sumberdaya CPU yang bertahap pada masing-masing Mesin Virtual, namun pada saat jumlah *user* diatas 2790 penggunaan CPU Mesin Virtual 1 menurun namun hasil pengujian mengembalikan nilai *error* yang sejalan dengan kenaikan jumlah *user*. Dapat disimpulkan bahwa batas *user* yang dapat ditangani oleh Virtual Mesin 1 tidak lebih dari 2790. Dan disisi lain penggunaan sumber daya CPU pada Mesin Virtual 1 (dengan Docker) jauh lebih tinggi dibandingkan dengan penggunaan sumber daya CPU pada Mesin Virtual 2 (tanpa Docker).

Sedangkan pada penggunaan sumber daya RAM baik Mesin Virtual 1 dan Mesin Virtual 2 terjadi kenaikan dan penurunan nilai secara tidak stabil dimana dapat diartikan bahwa nilai penggunaan sumber daya RAM tidak dipengaruhi oleh jumlah kenaikan *user*. Dan hasil pengujian juga menunjukan terjadinya *error* yang ditunjukan oleh Apache Jmeter saat dilakukan uji performa di Mesin Virtual 1 (dengan Docker) pada saat dilakukan penambahan jumlah *user* diatas 2790, sedangkan pada Mesin Virtual 2 (tanpa Docker) dengan jumlah *user* yang sama berhasil menyelesaikan uji performa tanpa terjadi *error*.

Berdasarkan hasil pengujian tersebut dapat disimpulkan bahwa dengan spesifikasi sumber daya yang sama antara masing-masing mesin virtual, Mesin Virtual 2 (tanpa Docker) lebih baik dalam menangani HTTP *Request* karena dapat menangani lebih dari 2790 *user* dengan nilai penggunaan CPU lebih kecil dibandingkan Mesin Virtual 1 serta tidak mengembalikan nilai *error*.

DAFTAR PUSTAKA

[1] I. Muakhori, Sunardi, and A. Fadlil, "Membangun Web Server Menggunakan Dynamic Domain Name System (DNS) Berbasis Berkeley Internet Name Domain (BIND9) pada IP Dinamis," *Prosiding JSI*, vol. 5, no. 2, Aug. 2018.

[2] R. Khalida, A. Muhajirin, and S. Setiawati, "Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi," *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic*, vol. 7, no. 2, pp. 167–176, Sep. 2019.

[3] Abdurrahman, Soni, and A. Hafid, "Optimalisasi Sumber Daya Komputer Dengan Virtualisasi Server Menggunakan Proxmox Ve," *JURNAL FASILKOM*, vol. 9, no. 2, pp. 369–376, Aug. 2019.

[4] Sriyanta, W. W. Winarno, and Sudarmawan, "Optimalisasi Penggunaan Hardware Server Mempergunakan Virtualisasi Server Di Sman 1 Wonosari," *Jurnal INFORMA Politeknik Indonusa Surakarta*, vol. 4, no. 2, pp. 35–42, 2018.

[5] I. Miell and A. H. Sayers, *Docker in Practice*, Second Edition. Shelter Island: Manning Publications Co., 2019.

[6] M. Dimas Erlangga and A. Prihanto, "Analisis Reliabilitas Multiserver Menggunakan Load Balancing Dengan Metode Denial Of Service," *Journal of Informatics and Computer Science*, vol. 03, no. 03, pp. 258–266, 2022.

[7] S. Dwiyatno, E. Rakhmat, and O. Gustiawan, "Implementasi Virtualisasi Server Berbasis Docker Container," *Jurnal PROSISKO*, vol. 7, no. 2, pp. 165–175, Sep. 2020.

[8] O. Jader, S. Zeebaree, and R. Zebari, "A State Of Art Survey For Web Server Performance Measurement And Load Balancing Mechanisms," *International Journal of Scientific & Technology Research*, vol. 8, pp. 535–543, Jan. 2019.

[9] S. Apridayanti, Isnawaty, and R. A. Saputra, "Desain dan Implementasi Virtualisasi Berbasis Docker untuk Deployment Aplkasi Web," *semanTIK*, vol. 4, no. 2, pp. 37–46, Jul. 2018, doi: 10.5281/zenodo.1407862.

[10] G. C. Obasuyi and A. Sari, "Security Challenges of Virtualization Hypervisors in Virtualized Hardware Environment," *International Journal of Communications, Network and System Sciences*, vol. 08, no. 07, pp. 260–273, 2015, doi: 10.4236/ijcns.2015.87026.

[11] Z. Gu and Q. Zhao, "A State-of-the-Art Survey on Real-Time Issues in Embedded Systems Virtualization," *Journal of Software Engineering and Applications*, vol. 05, no. 04, pp. 277–290, 2012, doi: 10.4236/jsea.2012.54033.

[12] A. Junaidi, "Studi Perbandingan Performansi Antara MongoDB dan MySQL Dalam Lingkungan Big Data," in *Annual Research Seminar UNSRi 2016*, 2016, vol. 2, no. 1, pp. 460–465. [Online]. Available: http://ars.ilkom.unsri.ac.id460

[13] Y. Irawan, "Pengujian Sistem Informasi Pengelolaan Pelatihan Kerja UPT BLK Kabupaten Kudus dengan Metode Whitebox Testing," *Journal Speed-Sentra Penelitian Engineering dan Edukasi*, vol. 9, no. 3, pp. 59–63, 2017.

[14] A. D. Putra, W. Yahya, and A. Bhawiyuga, "Analisis Kinerja Dan Konsumsi Sumber Daya Aplikasi Web Server Pada Platform Raspberry Pi," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, vol. 3, no. 4, pp. 3513–3521, Apr. 2019.

[15] C. P. Agustika, W. S. Saputra, and M. Idhom, "Pengujian Aplikasi Greenwallet dengan Metode Load Testing dan Apache Jmeter," *Jurnal Informatika dan Sistem Informasi (JIFoSI)*, vol. 2, no. 2, pp. 190–195, Jul. 2021.

[16] S. S. Abed and al-H. H. Omar, "Implementing Web Testing System Depending on Performance Testing Using Load Testing Method," in *Research in Intelligent and Computing in Engineering*, 2021, pp. 475–490.

[17] H. Lokawati and Y. Widyani, "Monitoring System of Multi-Tenant Software as a Service (SaaS)," in *2019 International Conference on Data and Software Engineering (ICoDSE)*, Nov. 2019, pp. 1–5. doi: 10.1109/ICoDSE48700.2019.9092741.

[18] M. Brattstrom and P. Morreale, "Scalable Agentless Cloud Network Monitoring," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*, Jun. 2017, pp. 171–176. doi: 10.1109/CSCloud.2017.11.

[19] L. Chen, M. Xian, and J. Liu, "Monitoring System of OpenStack Cloud Platform Based on Prometheus," in *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, 2020, pp. 206–209. doi: 10.1109/CVIDL51233.2020.0-100.

[20] P. Barca, B. Vujanić, and N. Maček, "Monitoring and Predicting Linux Server Performance With Linear Regression," in *Sinteza 2018 - International Scientific Conference on Information Technology and Data Related Research*, 2018, pp. 68–73. doi: 10.15308/Sinteza-2018-68-73.

[21] Ramadoni, M. Z. Amirudin, R. Fahmi, E. Utami, and M. S. Mustafa, "Evaluasi Penggunaan Prometheus dan Grafana Untuk Monitoring Database Mongodb," *JIP (Jurnal Informatika Polinema)*, vol. 7, no. 2, pp. 43–50, Feb. 2021.

[22] D. Kardha, A. R. Pamungkas, and H. Setiawan, "Pengembangan Virtual Server dengan Proxmox VE 6.2 sebagai Cloud Computing berbasis Free/Open Source Software," *Go Infotech: Jurnal Ilmiah STMIK AUB*, vol. 26, no. 1, p. 85, Jun. 2020, doi: 10.36309/goi.v26i1.126.

[23] M. Jiang, C. J. Jong, P. Poppell, K. Budhathoky, and R. Hull, "System Infrastructure Development Life Cycle for Enterprise Computing Systems," in *2009 International Conference on Computational Intelligence and Software Engineering*, Dec. 2009, pp. 1–6. doi: 10.1109/CISE.2009.5363878.