PENERAPAN DOCKER CONTAINER SEBAGAI TEKNOLOGI RAMAH SKALABILITAS DIBANDING TEKNIK VIRTUALISASI UNTUK MEMBANGUN WEBSITE DI UBUNTU 18.04.4 LTS

F.S. Ariadi[1], C. Iswahyudi[2], E.K. Nurnawati[3]
[1, 2, 3]Jurusan Informatika, FTI, IST AKPRIND,
[1]adhieari80@gmail.com, [2]catur@akprind.ac.id, [3]ernakumala@akprind.ac.id

Abstract

The process of developing web applications using traditional techniques is done by installing services on the host server directly, so that they are not bound in an isolated environment. This can result in a dependency conflict when a developer is required to add a service with a different library version from an existing service. Hypervisor-based virtualization techniques are used to solve this problem, but the large amount of server resources used makes it less scalable friendly. The right solution is to use a containerization technique like Docker. Docker works by binding applications and libraries as needed in an isolated environment.This study aims to build a scalability-friendly website by implementing Docker Container. Then performed a comparison test between Containerization Techniques using Docker and Virtualization techniques using Virtual Box. The Docker feature used in this research is Docker Compose, which will simplify the configuration process and service running.The test results show that Docker can be used to produce website applications with scalability-friendly developing processes. Server resources can be used optimally, which results in stable server performance. Docker is also able to reduce or even eliminate the possibility of dependency conflicts, and Docker has an easy configuration.

***Keywords** : Website, Docker Container, Docker Compose, Virtualization*

Abstrak

Proses *developing* aplikasi webdengan teknik tradisional dilakukan dengan memasang service-service kedalam server host secara langsung, sehingga tidak terikat dalam lingkungan yang terisolasi. Hal ini dapat mengakibatkan konflik dependensi ketika *developer* diharuskan menambah service dengan versi *library* yang berbeda dari service yang telah ada sebelumnya. Teknik Virtualisasi berbasis Hypervisor digunakan untuk mengatasi permasalahan tersebut, namun besarnya sumber daya server yang dipakai membuatnya kurang ramah skalabilitas. Solusi yang tepat adalah dengan menggunakan teknik Containerisasi seperti Docker. Docker bekerja dengan cara mengikat aplikasi beserta *library* yang dibutuhkan didalam lingkungan yang terisolasi. Penelitian ini bertujuan untuk membangun website yang ramah skalabilitas dengan menerapkan Docker Container. Kemudian dilakukan pengujian perbandingan antara Teknik Containerisasi menggunakan Docker dengan teknik Virtualisasi menggunakan Virtual Box. Fitur dari Docker yang digunakan dalam penelitian ini yaitu Docker Compose, yang akan memudahkan proses konfigurasi dan *running* service. Hasil pengujian menunjukkan Docker dapat digunakan untuk menghasilkan aplikasi website dengan proses *developing* yang ramah skalabilitas. Sumber daya server dapat dipakai secara maksimal, yang berakibat pada stabilnya kinerja server. Docker juga mampu mengurangi atau bahkan menghilangkan kemungkinan konflik dependensi, serta Docker memiliki konfigurasi yang mudah.

Kata kunci : Website, Docker Container, Docker Compose, Virtualisasi

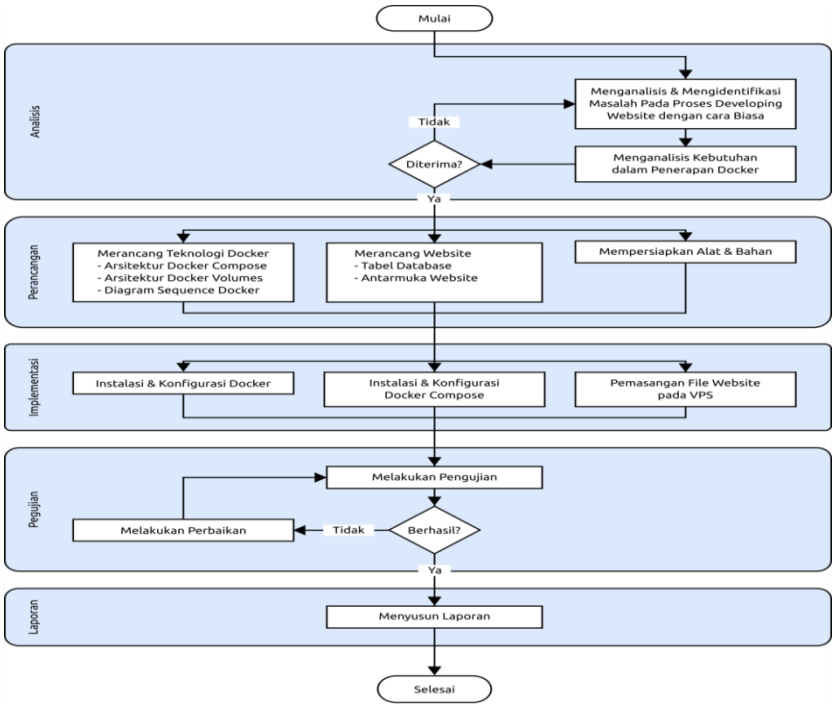
Pendahuluan

Website merupakan salah satu media informasi dan promosi yang sudah banyak digunakan oleh setiap orang/organisasi baik yang bergerak dibidang informasi maupun bisnis. Website dibangun oleh seorang *developer* menggunakan berbagai macam bahasa pemrograman. Sebelum website siap untuk dipublikasikan, ada beberapa langkah yang dilakukan oleh *developer*. Diantara langkah-langkahnya yaitu pemaketan file-file website sesuai dengan bahasa permrograman yang dipakai, misalnya berupa file php. Langkah selanjutnya yaitu penyiapan *environment* untuk menjalankan website tersebut, seperti server (baik berupa fisik maupun virtual), sistem operasi, serta layanan-layanan lain seperti server database, server mail dan lain-lain. Langkah selanjutnya yaitu *testing* atau melakukan percobaan/pengaksesan website yang telah dibangun. Langkah-langkah tersebut dilakukan oleh seorang *developer* dengan metode manual (tradisional) yang memiliki beberapa kendala/permasalahan.

Permasalahan tersebut yaitu kemungkinan terjadinya konflik dependensi terhadap proses penyebaran aplikasi, yaitu ketika *developer* harus melakukan penyebaran beberapa aplikasi yang masing-masing memiliki ketergantungan dengan paket versi tertentu. Permasalahan tersebut dapat diatasi dengan menggunakan teknik Virtualisasi berbasis Hypervisor. Namun teknik virtualisasi tersebut banyak mengkonsumsi sumber daya server yang dapat berpengaruh terhadap penurunan kinerja server. Berdasarkan beberapa permasalah tersebut, maka diterapkanlah teknologi Docker Container, sehingga aplikasi dipasang secara terisolasi. Selain itu penggunaannya pun sangat ringan sehingga dapat digunakan sebagai teknologi yang ramah skalabilitas untuk membangun aplikasi.

Penelitian ini bertujuan untuk membangun website publik yang ramah skalabilitas dengan memanfaatkan teknologi Docker Container, sehingga sumber daya (CPU, memori & swap) yang dimiliki dapat dimaksimalkan. Kemudian kinerja teknik Containerisasi pada Docker akan diuji dan dibandingkan dengan penggunaan mesin virtual yang boros sumber daya.

Metode pengumpulan data pada penelitian ini mengacu pada buku oleh (Suryana, 2010) yaitu berupa metode studi pustaka dan literatur untuk mengumpulkan data pustaka yang relevan serta teori-teori dan konsep-konsep yang relevan dalam implementasi teknologi Docker pada website, serta metode observasi untuk mengumpulkan, menentukan, memilah dan mengkaji ulang data-data yang dibutuhkan dalam implementasi teknologi Docker pada website yang dibangun. Gambar 1 berikut merupakan langkah-langkah dalam penelitian ini:



Gambar 1. Langkah-langkah Penelitian

Penelitian tentang bagaimana menerapkan Docker Container untuk mengatasi kendala pengembangan aplikasi telah dilakukan oleh beberapa penelitian.

Pada penelitian oleh (Adinta & Neforawati, 2017), Docker diterapkan untuk membangun Aplikasi Chatting Berbasis Web. Sistem informasi dalam sebuah perusahaan bersifat dinamis (sesuai perkembangan bisnisnya), sehingga menjadi masalah bagi *developer* dan administrator sebuah sistem aplikasi, terutama aplikasi berbasis web. *Developer* harus mempunyai aplikasi yang mampu berjalan dalam lingkungan yang berbeda untuk menjawab permasalahan seperti saat melakukan development dan testing pada environment yang berbeda atau migrasi server. Kendala yang sering terjadi adalah *conflicting dependencies*, *missing dependencies* dan perbedaan *platform* server itu sendiri. Maka dari itu digunakanlah teknologi Docker Container dalam penelitian tersebut, dimana cara kerja dari Docker Container mencakup penempatan aplikasi dalam sebuah Container di lingkungan operasinya sendiri, berbeda dengan virtual mesin yang diterapkan di server pada umumnya.

Pada penelitian oleh (Fihri, Negara & Sanjoyo, 2019) dilakukan implementasi &analisis performansi layanan web pada *platform* berbasis Docker, karena dalam proses pengembangan layanan web dibutuhkan beberapa aplikasi yang saling terintegrasi (*front-end* dan *back-end development*). Pada arsitektur monolitik, penyedia layanan membangun sebuah aplikasi yang seluruh komponennya terbuat menjadi sebuah kesatuan lalu disimpan dalam server. Jika penyedia layanan web ingin melakukan *scaling system* maka harus melakukan perombakan sistem secara menyeluruh karena penambahan *line coding* sekecil apapun akan memengaruhi keseluruhan aplikasi sehingga proses development sangat beresiko. Maka solusi dari arsitektur monolitik yang kurang efektif tersebut yaitu dengan melakukan transisi ke arsitektur *microservice*, aplikasi dipecah menjadi beberapa komponen yang dikembangkan dan berjalan secara independen. Untuk mengembangkan *microservice application* terdapat sebuah *platform* yang mendukung yaitu Docker, dimana satu server bisa digunakan untuk menjalankan banyak aplikasi tanpa mengintervensi kinerja aplikasi lainnya.

Penelitian oleh (Khalida, Muhajirin & Setiawati, 2019) dilakukan karena teknologi Virtualisasi berbasis Hypervisor kurang ramah skalabilitas untuk melakukan penyebaran aplikasi. Hasil dari penelitian tersebut menunjukkan bahwa teknologi Docker Container tepat untuk mewujudkan sistem web hosting yang ramah skalabilitas dan memudahkan penyebaran aplikasi. Penelitian tersebut juga membahas keunggulan dari teknologi Docker Container yang memiliki teknis kerja yang dapat meningkatkan produktivitas pengembang, memiliki aplikasi *portable* dan mengurangi jumlah penyimpanan karena *guest* OS atau Hypervisor serta biaya lisensinya dihilangkan.

Penelitian oleh (List, 2017) dilakukan karena perangkat lunak sains seringkali tidak terdokumentasi dengan baik dan hanya diuji pada sejumlah konfigurasi sistem yang terbatas. Selain itu, para peneliti sering kekurangan sumber daya untuk memelihara perangkat lunak dengan benar dalam waktu yang lama. Menggunakan Docker, alat perangkat lunak dapat dijalankan dalam lingkungan yang terkendali. Dengan demikian, pengembang perangkat lunak menghemat dengan memeriksa hanya satu konfigurasi sistem tunggal yang dapat mereka harapkan berfungsi, terlepas dari konfigurasi sistem khusus dari calon pengguna. Selain itu, sistem *build* otomatis dapat digunakan untuk menjamin bahwa pengguna memiliki akses ke versi terbaru*tools* perangkat lunak. Di sisi lain, karena pengarsipan dependensi maka alat perangkat lunak yang dikirim melalui Docker tetap dapat digunakan untuk khalayak luas. Selain itu, semua komunikasi antara Docker Container berlangsung di jaringan virtual yang terisolasi yang dapat meningkatkan keamanan tanpa biaya. Docker juga merupakan teknologi yang *open source* sehingga layak untuk menetapkan Docker sebagai standar komunitas baru untuk distribusi perangkat lunak untuk bio informatika.

Website merupakan dokumen HTML *(Hyper Text Markup Language)* yang membuat informasi dalam web server yang menggabungkan informasi data teks, data gambar diam atau gerak, data animasi, suara dan video (Anggraini, 2019).

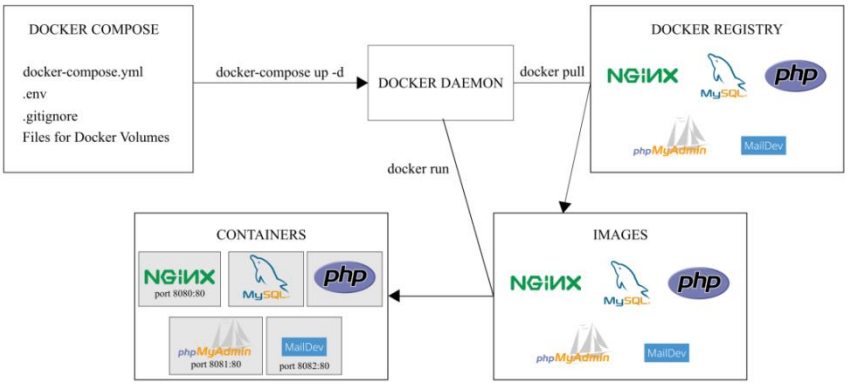
Docker merupakan aplikasi *open source* yang berfungsi untuk mengembangkan, mendistribusi, serta menjalankan software application. Desain arsitektur Docker memudahkan untuk mendistribusi serta mengembangkan aplikasi secara lebih cepat karena Docker memiliki sifat Lightweight Containerization yang dilengkapi dengan beragam komponen serta fitur

sehingga mampu memudahkan para developeruntuk mengembangkan dan memantau kinerja aplikasi yang diciptakan. (Fihri, Negara & Sanjoyo, 2019).

Docker Compose merupakan sebuah ekstensi Docker yang dirancang khusus untuk memfasilitasi interaksi beberapa Container Docker dalam konfigurasi perangkat lunak yang koheren. Docker Compose ini dibuat dengan nama docker-compose.yaml atau docker-compose.yml. (List, 2017).

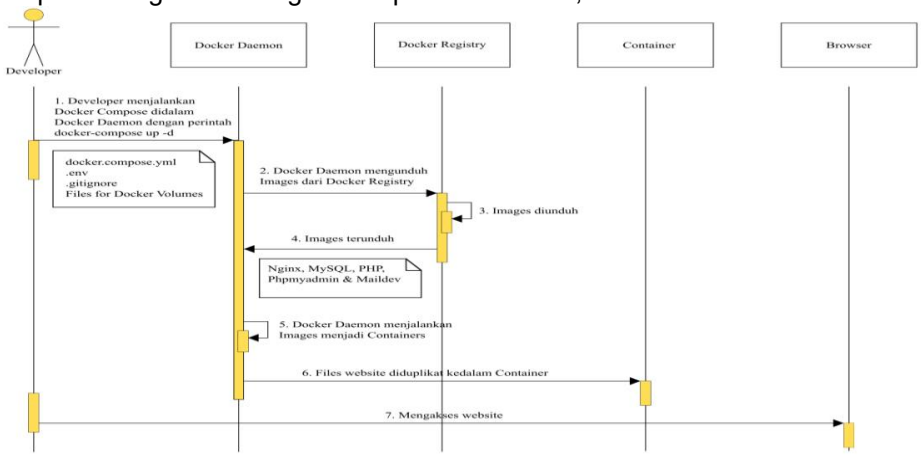
Hasil Dan Pembahasan

Pada penelitian ini digunakan Doker Compose untuk mengkonfigurasi service-service yang dibutuhkan untuk membangun website. Gambar 2 dibawah merupakan detail perancangan arsitektur pada Docker Compose tersebut. File-file konfigurasi yang digunakan dalam Docker Compose ini berupa file docker-compose.yml yang berisi konfigurasi service nginx, mysql, php, phpmyadmin & maildev, file .env yang berisi deklarasi user database, password database & port, file .gitignore yang berisi konfigurasi untuk mengecualikan file & folder terentu agar tidak ditambahkan kedalam Image. Docker Compose dijalankan didalam Docker Daemon, sehingga seluruh Image yang telah dikonfigurasi didalam file docker-compose.yml akan diunduh dari Docker Registry. Setelah semua Image terunduh maka Docker Daemon akan menjalankannya menjadi Container.



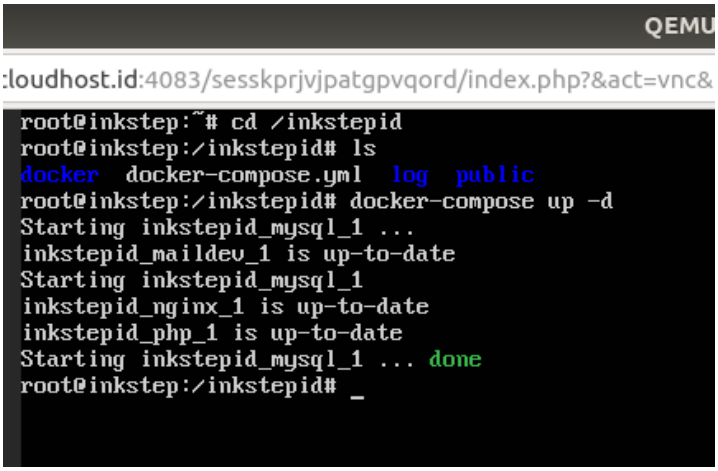
Gambar 2. Arsitektur Docker Compose

Perancangan Diagram Sequence digunakan untuk menunjukkan alur kerja dari teknologi Docker Container untuk membangun website secara detail dan berurutan. Gambar 3 berikut ini merupakan perancangan dari diagram sequence tersebut,



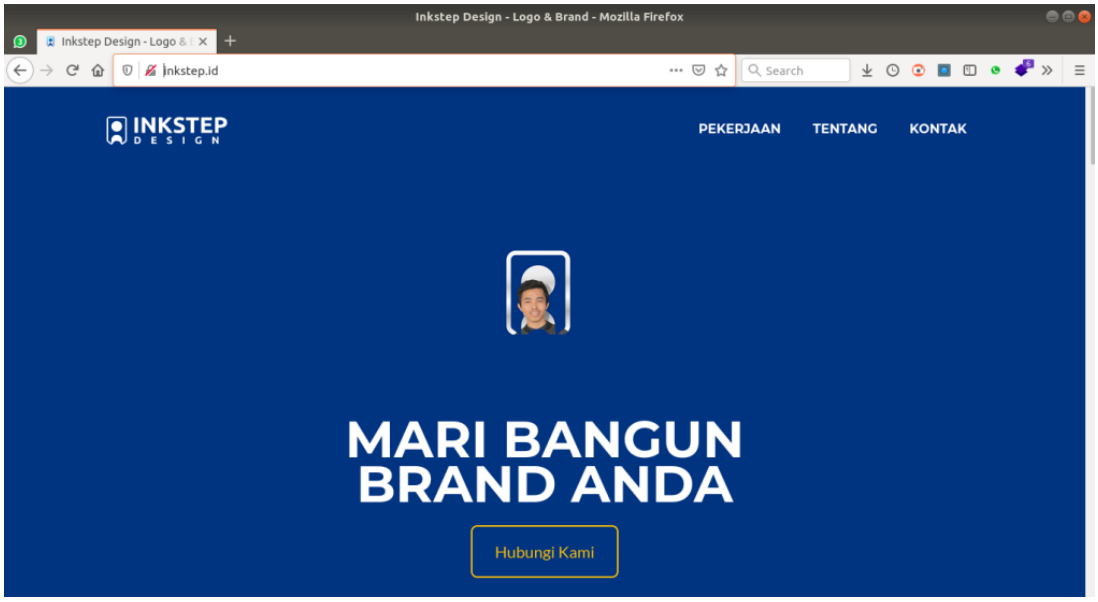
Gambar 3. Diagram Sequence Docker

File Docker dan file-file website diletakkan pada VPS agar website dapat diakses secara publik melalui koneksi internet. Gambar 4 berikut ini merupakan proses *running* Docker Compose didalam VPS, sehingga dalam keadaan tersebut website sudah bisa diakses secara publik.



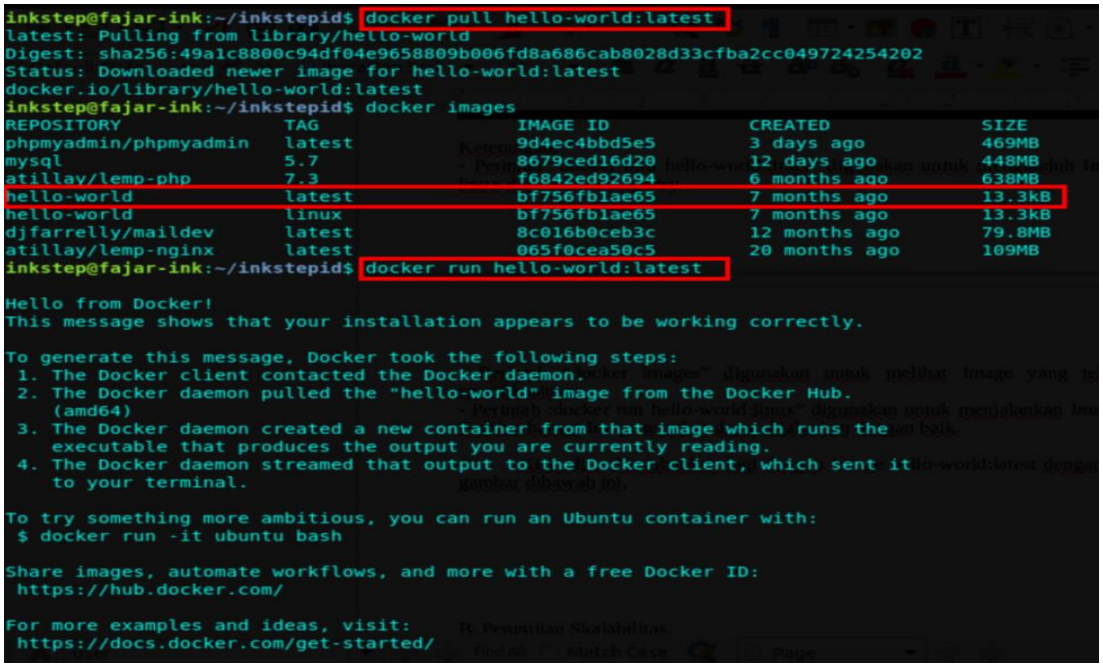
Gambar 4. *Running* Docker Compose

Tampilan website yang diakses dapat dilihat pada Gambar 5 dibawah ini



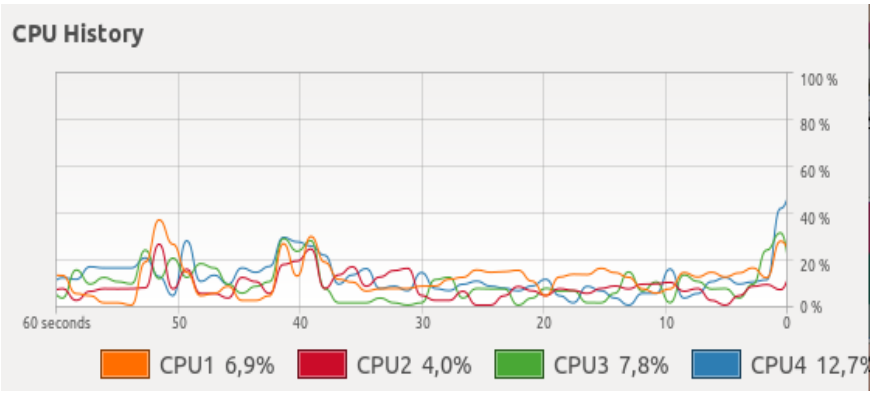
Gambar 5. *Screenshot* tampilan website

Gambar 6 dibawah ini menunjukkan bahwa dengan penerapan Docker Container dua Image yang sama dengan versi berbeda dapat dipasang secara bersamaan. Dua Image tersebut adalah hello-world versi latest dan hello-world versi linux. Keduanya bisa dijalankan tanpa harus menghapus salah satu Image. Artinya Docker Container mampu mengatasi masalah konflik dependensi.



Gambar 6. Daftar Image yang terpasang

Perbandingan penggunaan sumber daya oleh Docker dan Virtual Box dilakukan diuji menggunakan aplikasi Gnome System Monitor. Hasil menunjukkan bahwa penggunaan sumber daya (CPU, Memori dan Swap) oleh Docker lebih hemat dibanding dengan penggunaan sumber daya oleh Virtual Box. Gambar 7 berikut ini merupakan informasi penggunaan sumber daya CPU ketika menjalankan Docker. Dengan berjalannya Docker tersebut berarti website pun sedang berjalan dan bisa diakses,



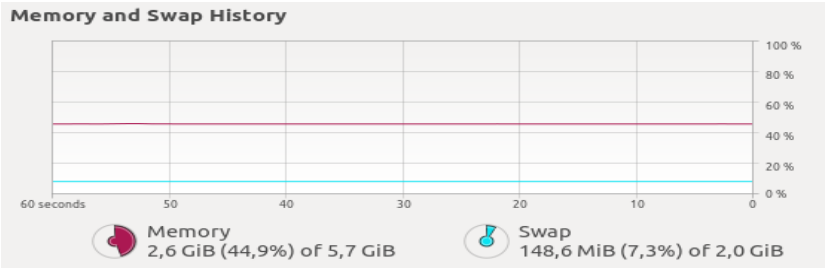
Gambar 7. Informasi penggunaan sumber daya CPU oleh Docker

Pada Gambar 7 diatas terdapat 4 (empat) kurva dengan warna yang berbeda. Kurva berwarna jingga menunjukkan penggunaan CPU 1, yaitu sebesar 6,9 %. Kurva berwarna merah menunjukkan penggunaan CPU 2, yaitu sebesar 4,0 &. Kurva berwarna hijau menunjukkan penggunaan CPU 3, yaitu sebesar 7,8 %. Kurva berwarna biru menunjukkan penggunaan CPU 4, yaitu sebesar 12,7 %. Perubahan kurva pada grafik ini ditampilkan setiap 60 detik.

Tabel 1. Informasi penggunaan CPU oleh Docker

CPU (%)			
Parameter	**Kapasitas**	**Penggunaan**	**Sisa**
CPU 1	100	6,9	99,1
CPU 2	100	4,0	96,0
CPU 3	100	7,8	92,2
CPU 4	100	12,7	83,7

Sedangkan Gambar 8 di bawah ini merupakan informasi penggunaan memori dan swap pada saat menjalankan Docker,



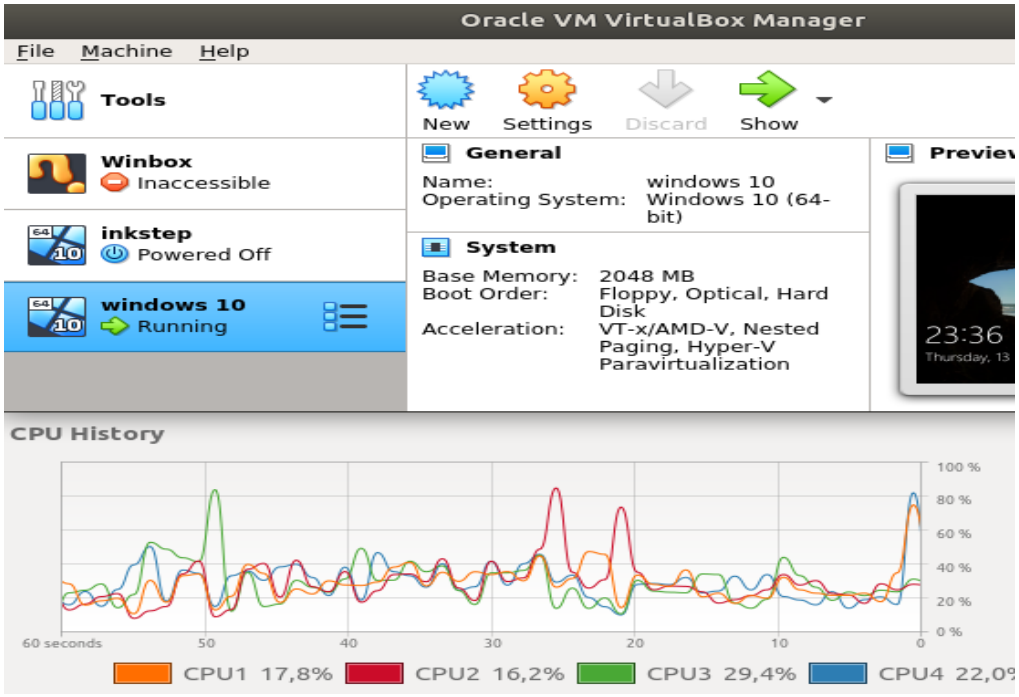
Gambar 8. Informasi penggunaan Memori & Swap oleh Docker

Pada Gambar 8 diatas terdapat 2 (dua) kurva dengan warna yang berbeda. Kurva berwarna merah menunjukkan penggunaan Memori, yaitu sebesar 44,9 % atau 2,6 GiB dari total kapasitas sebesar 5,7 GiB. Kurva berwarna biru menunjukkan penggunaan Swap, yaitu sebesar 7,3 % atau 148,6 MiB dari total kapasitas sebesar 2,0 GiB.

Tabel 2. Informasi penggunaan Memori & Swap oleh Docker

Memori & Swap (GiB)			
Parameter	**Kapasitas**	**Penggunaan**	**Sisa**
Memori	5,7	2,6	3,1
Swap	2,0	0,15	1,85

Berikut ini merupakan informasi penggunaan sumber daya CPU ketika menjalankan Virtual Box. Didalam Virtual Box tersebut dilakukan instalasi sistem operasi tambahan yaitu Windows 10 Pro, lihat Gambar 9 dibawah ini,



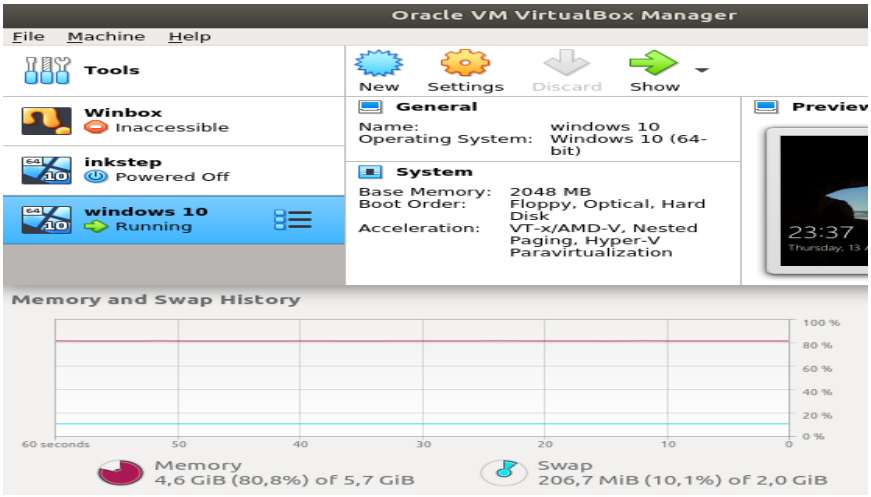
Gambar 9. Informasi penggunaan CPU oleh Virtual Box

Pada Gambar 9 diatas terdapat 4 (empat) kurva dengan warna yang berbeda. Kurva berwarna jingga menunjukkan penggunaan CPU 1, yaitu sebesar 17,8 %. Kurva berwarna merah menunjukkan penggunaan CPU 2, yaitu sebesar 16,2 &. Kurva berwarna hijau menunjukkan penggunaan CPU 3, yaitu sebesar 29,4 %. Kurva berwarna biru menunjukkan penggunaan CPU 4, yaitu sebesar 22,0 %. Perubahan kurva pada grafik ini ditampilkan setiap 60 detik.

Tabel 3. Informasi penggunaan CPU oleh Virtual Box

CPU (%)			
Parameter	**Kapasitas**	**Penggunaan**	**Sisa**
CPU 1	100	17,8	82,2
CPU 2	100	16,2	83,8
CPU 3	100	29,4	70,6
CPU 4	100	22,0	78,0

Sedangkan Gambar 10 dibawah ini merupakan informasi penggunaan sumber daya memori dan swap pada saat menjalankan Virtual Box,



Gambar 10. Informasi penggunaan Memori & Swap oleh Virtual Box

Pada Gambar 10 diatas terdapat 2 (dua) kurva dengan warna yang berbeda. Kurva berwarna merah menunjukkan penggunaan Memori, yaitu sebesar 80,8 % atau 4,6 GiB dari total kapasitas sebesar 5,7 GiB. Kurva berwarna biru menunjukkan penggunaan Swap, yaitu sebesar 10,1 % atau 206,7 MiB dari total kapasitas sebesar 2,0 GiB.

Tabel 4. Informasi penggunaan Memori & Swap oleh Virtual Box

Memori & Swap (GiB)			
Parameter	**Kapasitas**	**Penggunaan**	**Sisa**
Memori	5,7	4,6	1,1
Swap	2,0	0,21	1,79

Berdasarkan pengujian diatas dapat diketahui perbandingan penggunaan sumber daya CPU, memori dan swap antara Docker dan Virtual Box, lihat Tabel 5 berikut ini.

Tabel 5. Perbandingan penggunaan sumber daya oleh Docker & Virtual Box

Parameter	**CPU (%)**				**Memory dan Swap (GiB)**	
	CPU 1	CPU 2	CPU 3	CPU 4	Memory	Swap
Docker	6,9	4,0	7,8	12,7	2,6	0,15
Virtual Box	17,8	16,2	29,4	22,0	4,6	0,21
Selisih	10,9	12,2	21,6	9,3	2,0	0,06

Dari Tabel 5 diatas dapat disimpulkan bahwa penggunaan sumber daya server pada saat menjalankan Docker adalah sangat kecil. Semakin kecil sumber daya yang dipakai maka semakin bagus, menunjukkan bahwa Docker sangat hemat sumber daya. Sedangkan penggunaan sumber daya pada saat menjalankan Virtual Box adalah boros. Terlihat selisih yang cukup signifikan antara keduanya. Borosnya Virtual Box ini disebabkan karena adanya pemasangan sistem operasi tambahan secara penuh diatas sistem operasi utama. Semakin

banyak sistem operasi tambahan yang dipasang, maka sumber daya server pun akan semakin terkuras sehingga menurunkan kinerja server tersebut.

Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan berdasarkan rumusan masalah yang telah dijabarkan adalah sebagai berikut:

1. Dalam membangun website, Docker Container bekerja dengan cara mengikat service, *library* dan dependensinya bersamaan dengan aplikasi tersebut dalam lingkungan yang terisolasi. Sehingga penggunaan Docker Container ini tidak memerlukan pemasangan sistem operasi tambahan pada sistem operasi utama untuk mengisolasi aplikasi. Hal inilah yang menyebabkan proses *developing* aplikasi web menjadi ramah skalabilitas, yaitu tidak boros dalam pemakaian sumber daya.
2. Cara membangun website publik menggunakan teknologi Docker ini yaitu dengan memanfaatkan fitur dari Docker yang disebut Docker Compose. Dimana setiap service yang dibutuhkan dikonfigurasi dan dijalankan dengan Docker Compose tersebut didalam Virtual Privat Server (VPS).
3. Penggunaan teknologi Docker Container jauh lebih unggul dibandingkan dengan penggunaan teknologi Virtualisasi berbasis Hypervisor. Dalam pengujian yang dilakukan pada penelitian ini diperoleh data bahwa Docker Container hanya memerlukan sumber daya CPU sekitar 4,0 – 12,7 % saja dari kapasitas penuh CPU, sedangkan Virtualisasi Hypervisor dengan Virtual Box memerlukan sumber daya CPU sekitar 16,2 – 29,4 %. Pada penggunaan sumber daya memori Docker hanya memerlukan 2,6 GiB saja dari total memori 5,7 GiB, sedangkan Virtual Box memerlukan memori sebesar 4,6 GiB. Begitu pula pada penggunaan swap Docker unggul karena hanya memerlukan swap sebesar 0,15 dibanding Virtual Box yang memerlukan swap 0,21.

Saran untuk penelitian lanjutan:

1. Perlu dilakukan penambahan *environment* website selain dari environment yang sudah dibuat sebelumnya, untuk menguji kemampuan Docker untuk menjalankan website dengan kebutuhan service yang berbeda
2. Penerapan Docker perlu dilakukan pada sistem operasi yang lain sepert MacOS dan Windows.
3. Pembuatan website tidak hanya menggunakan *script* yang sederhana, namun perlu dikembangkan dengan *script* yang lebih kompleks agar dapat menghasilkan website yang lebih sempurna.

Daftar Pustaka

Adinta, F., & Neforawati, I. (2017). Rancang Bangun Aplikasi Chatting Berbasis Web Menggunakan Docker. *JOISIE Journal Of Information System And Informatics Engineering* , 1(1), 1-68.

Anggraini, I. (2019). Perancangan Website Penerimaan Siswa Baru dengan Menggunakan Metode Waterfall. *Jurnal Ilmiah Binary STMIK Bina Nusantara Jaya, Vol. 01 No. 02. ISBN: 2657-2117.*

Fihri, M., Negara, R. M., & Sanjoyo, D. D. (2019). Implementasi dan Analisis Performansi Layanan Web Pada platform Berbasis Docker. *Jurnal e-Proceeding of Engineering*, 6(2), 3996-4001.

Khalida, R., Muhajirin, A., & Setiawati, S. (2019). Teknik Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi. *Jurnal Penelitian Ilmu Komputer, System Embedded & Logic*, 7(2), 167-176.

List, M. (2017). Using Docker Compose for the Simple Deployment of an Integrated Drug Target Screening Platform. *Jurnal of Integrative Bioinformatics*, DOI: 10.1515/jib-2017-0016.

Suryana. (2010). *Metodologi Penelitian Model Praktis Penelitian Kuantitatif dan Kualitatif.* Bandung: Universitas Pendidikan Indonesia