Jurnal Penelitian Ilmu Komputer, System Embedded & Logic
p-ISSN: 2303-3304, e-ISSN: 2620-3553
7 (2): 167 – 176 (September 2019)

Teknis Kerja Docker Container untuk Optimalisasi Penyebaran Aplikasi

Rakhmi Khalida [1,*], Adi Muhajirin [1], Siti Setiawati [1]

[1] Teknik Informatika; Universitas Bhayangkara Jakarta Raya; Jl.Raya Perjuangan Bekasi Utara, (021) 88955882; e-mail: rakhmi.khalida@dsn.ubharajaya.ac.id; e-mail: adi.muhajirin@dsn.ubharajaya.ac.id; e-mail: siti.setiawati@dsnl.ubharajaya.ac.id

*Korespondensi: e-mail: rakhmi.khalida@dsn.ubharajaya.ac.id

Diterima: 5 Agustus 2019; Review: 26 Agustus 2019; Disetujui: 16 September 2019

Abstract

Today, web-based applications more accessible because it can be used in various computer platform only by running the web browser. The developers of the website create some applications that are accessible to all people normally by installing the application in the server. Modern web hosting system, in each of the server, manage multiple web applications. Along with the increase in quality when the presence of high availability requirements of a web service then the server load will increase. Each web application has a dependency on one operating system specific versions of packages including intact as well as well as the provisioning system into application problems for the server. Software engineering with Docker container has been recognized by many developers because it is considered as a solution to current trends. In addition, Docker has a simple application deployment process. This research used Docker Container with the result that Docker container can provide solutions to web applications developers who need to perform a maximum spread of Docker application. The analysis of the information systems was used as a research method in developing a Docker container-based application. The result showed Docker Container could increase developer productivity, reduce server load and the storage.

Keywords: *Web, App, Docker, Container*

Abstrak

Aplikasi berbasis web semakin banyak digunakan karena dapat diakses di berbagai platform komputer hanya dengan menjalankan web browser. Para pengembang website agar aplikasi yang dibuat dapat diakses semua orang biasanya melakukan install aplikasi ke dalam server. Sistem web hosting modern, di dalam setiap servernya, mengelola banyak aplikasi web. Seiring peningkatan kualitas saat hadirnya kebutuhan high availability dari layanan web maka beban server akan meningkat. Masing-masing aplikasi web yang memiliki ketergantungan dengan satu sistem operasi utuh termasuk paket versi tertentu juga serta aplikasi bawaan sistem menjadi masalah untuk server. Software Docker dengan teknik container banyak dilirik para pengembang aplikasi karena dianggap sebagai solusi trend saat ini. sehingga Docker telah memudahkan proses penyebaran aplikasi. Pada penelitian ini akan dilakukan pembahasan teknis kerja Docker Container, melihat teknis kerja Docker Container dapat memberikan solusi pada pengembang aplikasi web yang membutuhkan Docker untuk maksimal melakukan penyebaran aplikasi. Metode penelitian yang digunakan adalah analisis sistem informasi yang telah dikembangkan dan dilakukan penyebaran menggunakan teknologi Docker container. Hasil penelitian yang didapatkan adalah Docker Container dapat meningkatkan produktivitas pengembang, membuat beban server berkurang, dan mengurangi jumlah penyimpanan.

Kata kunci- Web, Aplikasi, Docker, Container

1. Pendahuluan

Saat ini aplikasi berbasis website sangat diminati para pengembang software. Faktor pendukungnya adalah perkembangan komputer dan internet yang harganya semakin terjangkau. lalu tidak terlepas dari dukungan program dan aplikasi yang tersedia dan terus beragam. Pengembang dapat dengan mudah mengembangkan suatu aplikasi dengan berbagai tools yang sudah tersedia. Dalam membangun program, pengembang biasanya menjalankan virtualisasi pada server sehingga proses pembuatan program dapat berjalan pada berbagai platform maupun konfigurasi hardware. Masalah yang dihadapi dengan virtualisasi adalah perlunya menyiapkan satu sistem operasi secara utuh, termasuk berbagai aplikasi yang dibawa sistem tersebut. Banyaknya jumlah virtualisasi yang berjalan pada sebuah server akan memberatkan sistem tersebut.

Aplikasi web beserta software pendukung seperti web server, database server, environment pendukung lainya harus dipasang ke dalam server. Menurut Romadlon Bik dan Asmunin secara umum ada dua metode penyebaran aplikasi web kedalam server. Pertama memasang web aplikasi beserta environment yang dibutuhkan ke dalam server tunggal, kelebihannya adalah setup server mudah, simple dan cepat dalam proses penyebaran, tetapi metode tersebut memiliki kekurangan yaitu setiap aplikasi tidak tersiolasi, sehingga apabila ingin melakukan penyebaran beberapa aplikasi yang masing-masing memiliki ketergantungan dengan paket versi tertentu menimbulkan konflik dependensi (*dependecy hell*). Metode yang kedua yaitu dengan memanfaatkan teknologi virtualisasi berbasis hypervisor, jadi setiap aplikasi dan dependency yang dibutuhkan disebar kedalam Virtual Machine (VM) yang berbeda. Dengan metode ini dapat meningkatkan scalabilitas, karena setiap aplikasi berjalan pada resource (CPU, memory, I/O) yang berbeda sehingga dapat dengan mudah ditambahkan sesuai kebutuhan. masalah klasik yang ditimbulkan saat menjalankan virtualisasi berbasis hypervisor adalah membutuhkan resource yang besar [Bik, 2017].

Semakin lama, proses pengembangan dari awal hingga siap didistribusikan menjadi semakin mudah. Teknologi virtualisasi berbasis hypervisor dapat digunakan tetapi tidak tepat untuk ramah scalabilitas dan kemudahan dalam penyebaran aplikasi. Keberhasilan melakukan penyebaran aplikasi dengan penggunaan resources yang ringan, tidak memiliki ketergantungan dengan satu sistem operasi utuh termasuk paket versi tertentu juga serta aplikasi bawaan sistem merupakan impian bagi para pengembang, karena waktu kerja yang dimiliki dapat optimal melakukan pengembangan maupun operasional layanan aplikasinya. Teknologi yang tepat untuk mewujudkan sistem web hosting yang ramah scalabilitas dan memudahkan penyebaran aplikasi adalah dengan menggunakan Docker. Docker adalah teknologi virtualisasi sistem operasi berbasis Container untuk membangun, menguji, dan menyebarkan aplikasi terdistribusi dalam lingkungan yang terisolasi. Docker tidak membangun mesin virtual sendiri, sehingga lebih hemat memori, processor dan storage. Docker menyediakan virtualisasi ringan dengan overhead hampir nol. Tujuan penelitin ini adalah membahas teknis kerja Docker Container yang dapat meningkatkan produktivitas pengembang, memiliki aplikasi portable dan mengurangi jumlah penyimpanan karena guest OS atau Hypervisor serta biaya lisensinya dihilangkan.

Beberapa penelitian terkait dengan paper ini pernah dilakukan oleh Kurniawan mengimplementasikan Docker Container pada laboratorium. Docker memiliki potensial untuk membantu meningkatkan keamanan data yang dimiliki oleh mahasiswa dan mempermudah kinerja asisten lab. Dengan cara menyediakan virtual machine untuk masing-masing mahasiswa, tiap mahasiswa dapat dengan nyaman menyimpan data-data mereka, dan untuk asisten lab tidak harus menginstal ulang tiap-tiap komputer yang ada di labnya setiap semester [Kurniawan et al., 2016]. Penelitian oleh Sardi Docker menjadi sebuah solusi bagi web developer untuk bereksperimen atau mengembangkan aplikasi ke dalam Container tanpa mempengaruhi sistem host[Apridayanti et al., 2018]. Hung et al., fokus pada Docker Container tetapi diasumsikan bahwa aplikasi tidak digunakan dalam arsitektur monolitik. Keterbatasan arsitektur monolitik, seperti ketidakefisienan dalam memperbarui dan pengiriman, mendorong Hung et al., untuk mengeksplorasi alokasi sumber daya ulang untuk penyebaran aplikasi berbasis microservices [Hung et al., 2016].

2. Metode Penelitian

Berdasarkan analisis yang disebutkan diatas tahapan penelitian yang dilakukan adalah mengeksplorasi aplikasi cloud berbasis microservices menggunakan Docker Container, lalu pembahasan terminologi Docker Container, selanjutnya perbandingan teknologi virtualisasi dengan Docker Container. Tahapan selanjutnya membahas workflow pengembangan aplikasi dengan Docker Container. Data sumber daya yang digunakan menjadi akhir dari tahapan penelitian.

2.1. Aplikasi Menggunakan Docker Container

Salah satu manfaat terbesar Docker adalah portabilitas. Selama beberapa tahun terakhir, penyedia jasa IaaS terbesar seperti Microsoft Azure, OpenStack, Amazon Web Services (AWS), dan Google Compute Platform (GCP) telah tergabung dan mendukung Docker. Beberapa aplikasi berbasis microservices yang menggunakan Docker Container dapat dilihat pada Tabel 1.

Tabel 1. Pengguna Docker

Jenis	Nama Aplikasi
E-Commerce	Ebay
E-Commerce	Gilt
E-Commerce	Groupon
Media	Spotify
Life Sciences	Illumina
Life Sciences	Cambridge Healtcare
PaaS	Redhat
PaaS	Yandex
IT SaaS	New Relic
IT SaaS	AppDynamic

Sumber: [Sugianto, 2016]

Peran Docker dalam proses transformasi digital menjadi semakin penting. Para CIO berlomba-lomba memahami dan merangkul Docker, karena Docker berjalan dalam sebuah Container dan hal ini akan mengaktifkan satu aplikasi yang berjalan di atas Container, dan ini membuat aplikasi benar-benar portable diantara sistem operasi yang didukung [Bik, 2017].

Berbagai aplikasi dibangun dengan Docker karena dua alasan. Pertama, virtualisasi aplikasi melindungi dari lingkungan operasi sekitarnya. Hal ini dapat melindungi dari interaksi langsung dengan aplikasi lain dan dengan sistem operasi. Hal ini berarti pengujian akan menjadi sangat sederhana. Proses uji hanya dilakukan sekali, tidak dilakukan berkali-kali karenaa versi sistem operasi yang berbeda, dan dalam kombinasi dengan aplikasi yang berbeda. Kedua, Docker dilihat dari sudut pandang Sysadmin. Sysadmin mungkin sudah memiliki gambar mesin virtual yang mewakili berbagai lingkungan operasi yang berbeda untuk aplikasi. Jika aplikasi berjalan dalam Docker Container, Sysadmin dapat menggunakan container yang sama diseluruh konfigurasi sistem yang berbeda, daripada melakukan penyediaan server baru untuk setiap instance.

2.2. Terminologi Docker

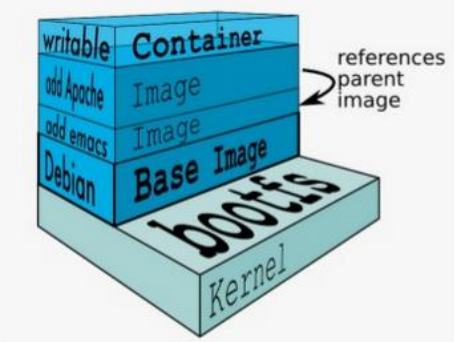
Arsitektur Docker menggunakan client dan server. Docker client mengirimkan request ke Docker Daemon untuk untuk membangun, mendistribusikan, dan menjalankan Docker Container. Keduanya Docker client dan Daemon dapat berjalan pada sistem yang sama. Antara Docker client dan Docker Daemon berkomunikasi via socket menggunakan restful API. Beberapa terminologi atau istilah menjadi komponen penting Docker:

Docker Images

Docker Image merupakan template yang bersifat read only. Docker image merupakan dasar template untuk docker container Docker Images dibutuhkan seperlu menjalankan Container. Dalam membangun image baru ataupun merubah image yang telah ada sebelumnya, Docker memberikan cara sederhana. Adapun di dalam Docker Registry bisa didapati banyak image yang dibuat oleh pengguna lain yang dapat dimanfaatkan sebagai base image. Misal untuk membuat aplikasi PHP menggunakan mysql dan apache web server, maka cukup mengunduh base image lalu tambahkan aplikasi PHP yang sudah dibuat sebelumnya.

Docker Container

Docker Container sendiri merupakan sebuah image yang dapat dikemas dan dibaca tulis, container berjalan diatas image. Pada setiap perubahan yang disimpan pada container akan menyebabkan terbentuknya layer baru di atas base image. Kita dapat melakukan instalasi aplikasi didalamnya dan melakukan penyimpanan.

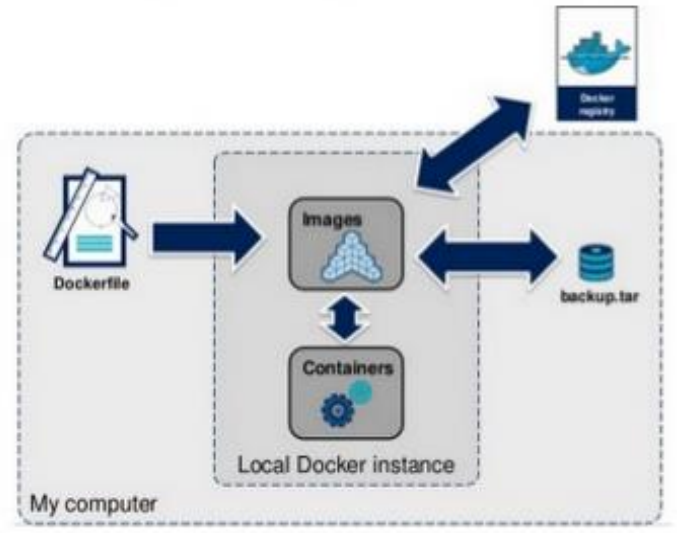


Sumber: [Krochmalski, 2017]

Gambar 1. Docker Container dan Image

Docker Registry

Docker registry merupakan repositori distribusi kumpulan docker image yang terpusat baik bersifat public dan private repositori. Registry public Docker disebut dengan Docker Hub atau Docker Index. Disini dapat melakukan push image kita sendiri maupun pull image.



Sumber: [Krochmalski, 2017]

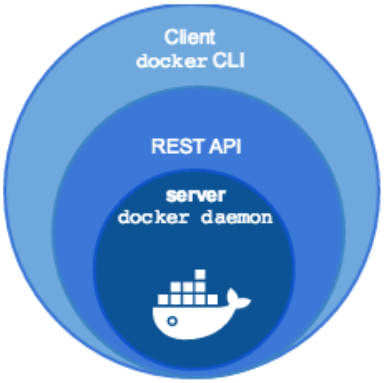
Gambar 2. Ekosistem Docker

Dockerfile

Dockerfile merupakan skrip yang yang berisi atau terdiri dari serangkaian perintah atau intruksi yang akan dieksekusi secara otomatisasi dan berurutan untuk membangun sebuah image.

Docker Daemon: adalah aplikasi yang berjalan di host machine. Docker server berjalan di background (sebagai daemon) dan menunggu perintah dari docker client. Begitu mendapatkan perintah, docker server bekerja membuat container, menjalankan/mematikan container, dan sebagainya.

Docker Engine: adalah gabungan aplikasi yang menjalankan docker container, docker client, dan docker daemon. Dia menyediakan layanan umum agar container dapat berjalan dengan baik, misalnya: networking, storage, alokasi memori, CPU, dan sebagainya.



Sumber: [Krochmalski, 2017]

Gambar 3. Docker Engine

Docker Machine: adalah sistem untuk mengelola docker engine. Dia bisa menginstal engine baru di berbagai target. Saat ini dia mendukung VirtualBox, Digital Ocean, dan Amazon. Dengan Docker Machine, kita bisa membuat dan menjalankan container dengan docker client, dan hasilnya container kita berjalan di local (dengan VirtualBox) atau di cloud (dengan DigitalOcean atau Amazon).

3. Hasil dan Pembahasan

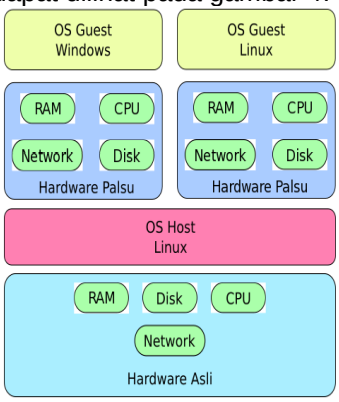
Pada era transformasi digital sudah banyak perusahaan yang menyediakan jasa layanan cloud. Sysadmin sudah difasilitasi sehingga tidak repot lagi. Konsep yang disenangi adalah virtualisasi dan Docker.

3.1. Teknologi Virtualisasi

Secara sederhana, virtualisasi adalah lawan kata dari physical *machine* atau mesin fisik. *Physical machine* adalah wujud server yang memiliki berbagai komponen seperti power supply, mainboard, memory, disk, dan sebagainya. Virtualisasi adalah aplikasi seolah-olah berjalan sendirian dalam satu mesin, tetapi sebenarnya virtualisasi berjalan di atas mesin lain, bersama-sama dengan aplikasi lain.

Manfaat virtualisasi diantaranya: 1). Keinginan menjalankan sistem operasi yang berbeda dengan sistem operasi host. 2). Memaksimalkan penggunaan mesin fisik. Banyak aplikasi dengan sistem operasi berbeda, versi-versi paketan berbeda dan software pendukung lainnya dapat dijalankan pada satu mesin fisilk. 3). Terbatasnya sumber daya dan ruangan penyimpanan.

Secara garis besar, ada dua strategi yang umum digunakan dalam membuat virtualisasi: **Pertama**, **Hypervisor atau full virtualization, yaitu:** Pada hypervisor, semua hardware dibuatkan virtualisasinya. Masing-masing sistem guest mendapatkan satu set hardware virtualisasi. Arsitektur hypervisor dapat dilihat pada gambar 4.

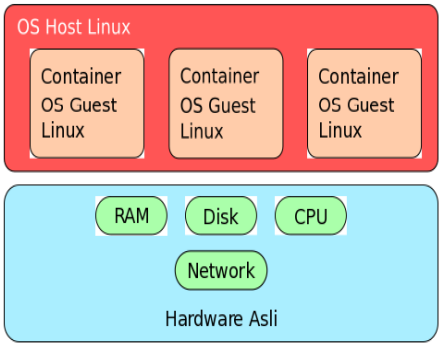


Sumber: [Krochmalski, 2017]

Gambar 4. Arsitektur hypervisor

Produk virtualisasi yang berbasis hypervisor:
a. Oracle Virtualbox
b. VMWare
c. KVM
d. Microsoft Hyper-V

Container atau partial virtualization
Strategi Container, semua *guest* berbagi pakai sebagian besar hardware. Host cuma mengatur pemisahan folder dan ijin akses antar guest sehingga tidak bisa saling melihat. Arsitektur container dapat dilihat pada gambar 5.



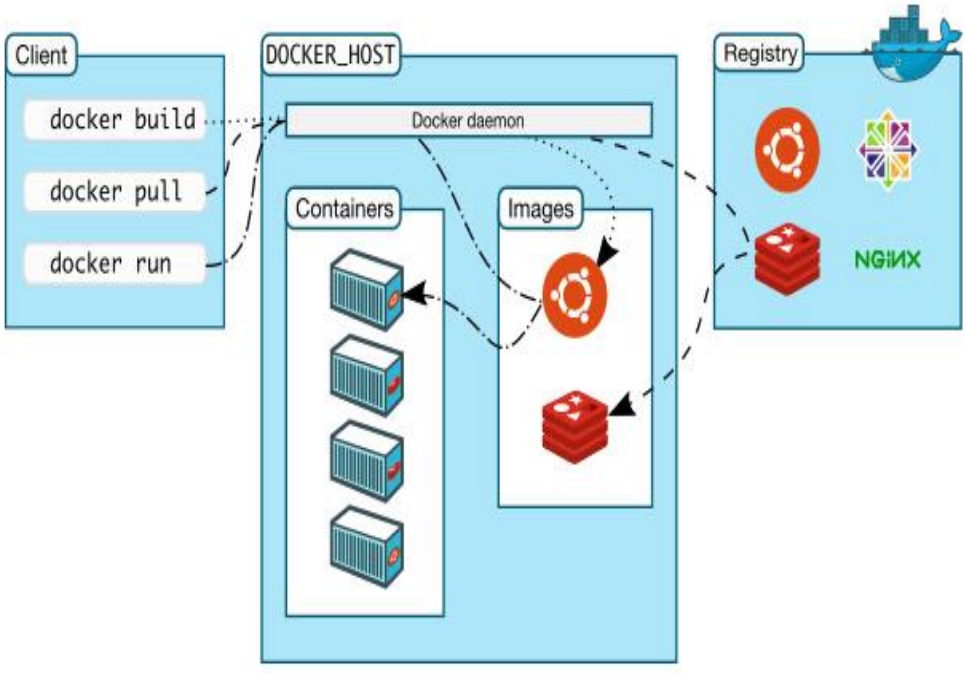
Sumber: [Krochmalski, 2017]

Gambar 5. Arsitektur hypervisor [Krochmalski, 2017]

Produk virtualisasi yang berbasis container:
1. OpenVZ
2. LXC
3. Docker

3.2. Konsep Docker Container

Docker terdiri dari beberapa komponen, dapat dilihat pada gambar 6.



Sumber: [Krochmalski, 2017]

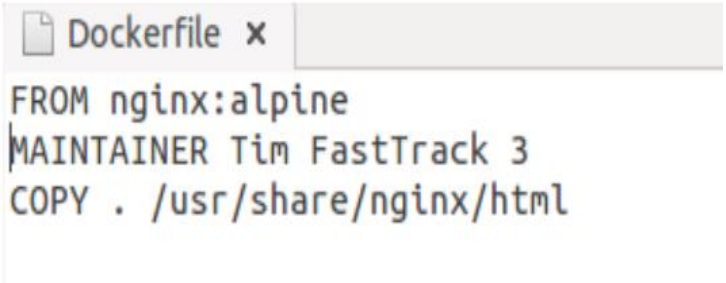
Gambar 6. Komponen Docker

Container secara istilah merupakan alat untuk mempermudah, mengemas, dan mendistribusikan suatu hal dari suatu tempat ke tempat yang lain.

Dalam konteks lingkungan Linux, Docker Container diartikan sebagai alat yang dapat dipergunakan untuk memberikan sistem yang terisolasi pada level OS yang dijalankan pada induk Linux Kernel.

Untuk dapat melihat teknis kerja Docker Container, harus dilakukan simulasi deploy aplikasi ke dalam Docker Image.

1. Mencari image dasar yang sudah lengkap terdapat database MySQL pada Docker Registry.
2. Melakukan koding pada Dockerfile seperti pada gambar 7.



Sumber: Hasil Penelitian (2019)

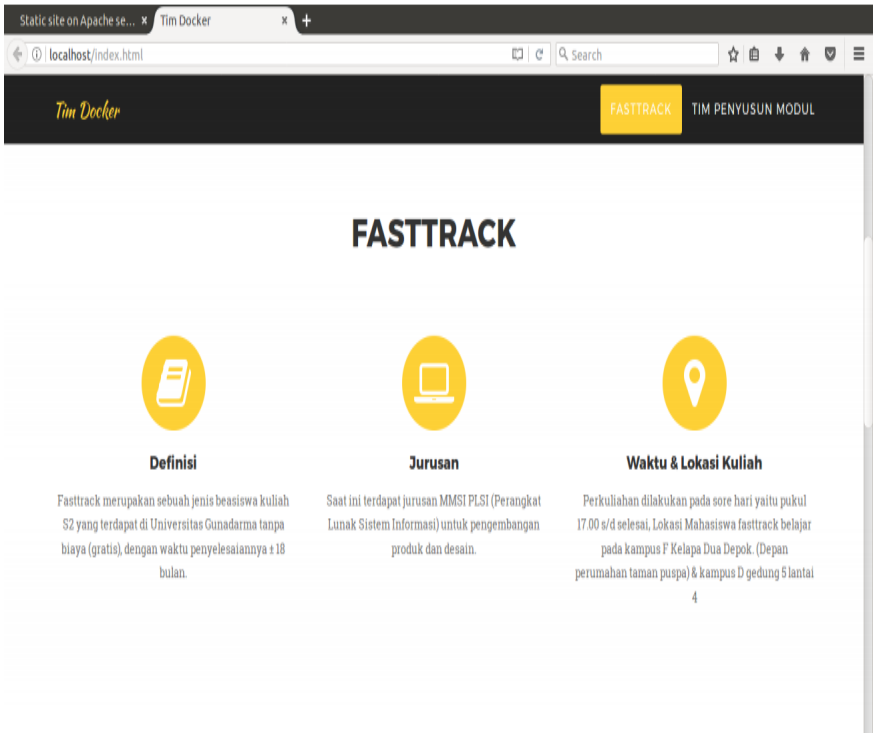
Gambar 7. Isi Dockerfile

3. Melakukan build Dockerfile hingga sukses.
4. Docker Image dapat dijalankan seperti gambar 8.

Sumber: Hasil Penelitian (2019)

Gambar 8. Running Docker Image

Perintah-perintah dalam gambar 8 dapat dijelaskan seperti ini, $ Docker run --name container-web-ft3 -p 80:80 -d web-ft3 adalah perintah menjalankan images baru yang telah dibangun dengan meluncurkan sebuah container ke daemon, yang memetakan ke perkumpulan container internal 5000 ke host port 32775 sehingga kita dapat melihat output aplikasi. Docker ps adalah perintah untuk melihat daftar list container yang terdapat dalam sebuah image.



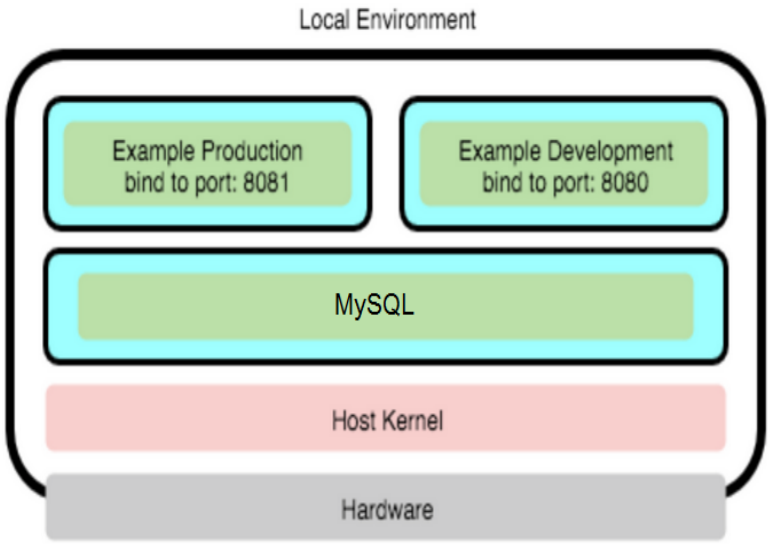
Sumber: Hasil Penelitian (2019)

Gambar 9. Hasil Deploy Aplikasi ke Docker Image

Dari menjalankan perintah http://localhost/index.html pada browser, hasil deploy aplikasi ke dalam container dan Docker image dapat terlihat.

5. Melakukan deploy aplikasi yang berasal dari base image nginx:alpine dengan ketentuan seperti berikut ini:
   a. Service example fase production pada port 8081.
   b. Service example fase development pada port 8080.

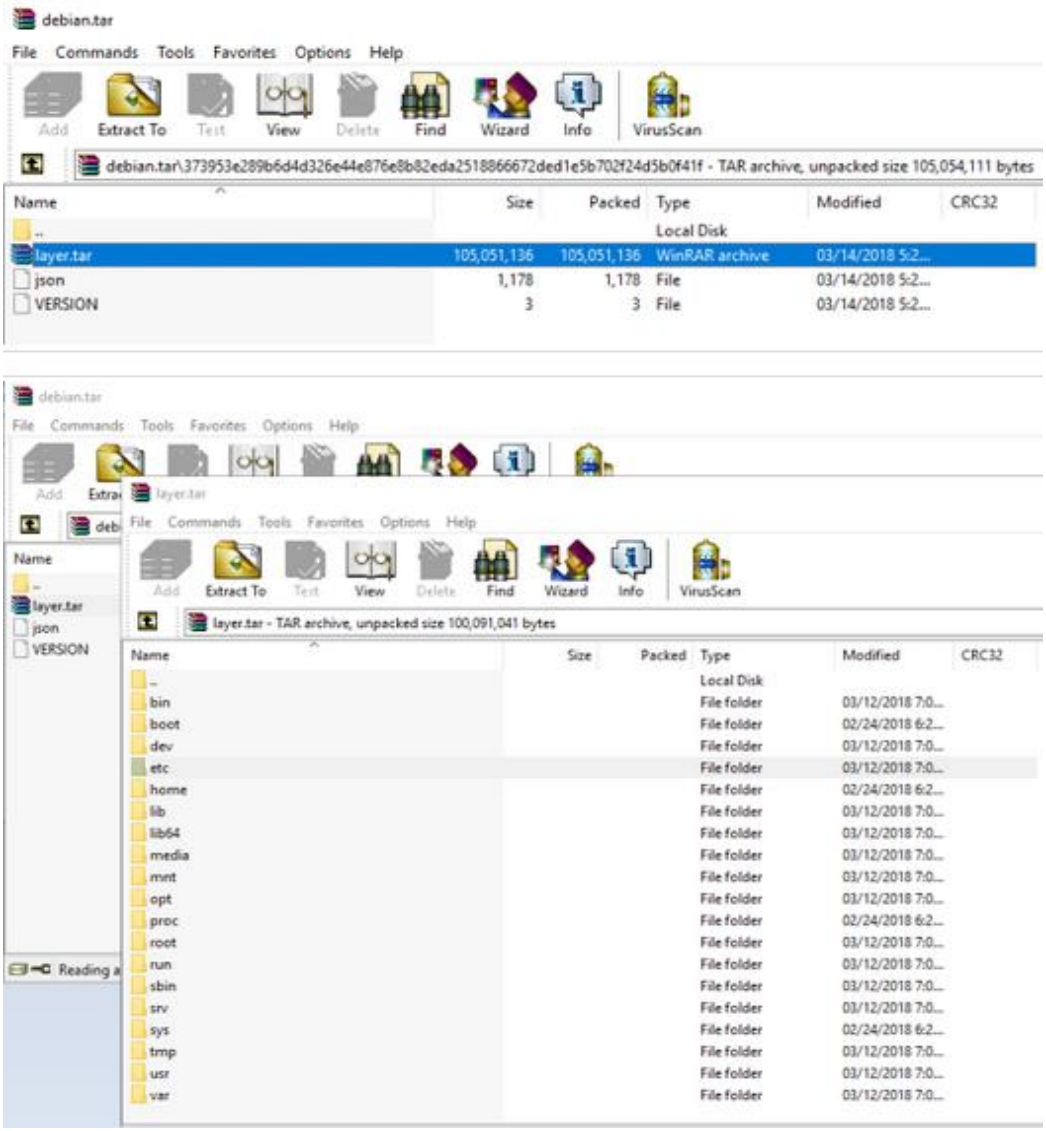
Dari simulasi yang dibuat akan menghasilkan lingkungan Container seperti gambar 10.



Sumber: Hasil Penelitian (2019)

Gambar 10. Lingkungan Container yang dihasilkan

Apabila ada perubahan di masing-masing Container seperti melakukan upgrade, maka tidak diperlukan melakukan restart seluruh sistem, melainkan cukup 1 Container saja. Suatu Container tidak harus membutuhkan config resource virtual mesin yang seperti di PC yang ada OS sepenuhnya. Sebuah Container diciptakan agar menggunakan resource serendahnya dan seperlunya saja, yang tentunya sudah terpangkas tak menggunakan sistem OS sepenuhnya.
Jika pada PC atau VM pada umumnya sebelum ada Container, setiap VM dalam membangun OS nya harus ada file .ISO sebagai installer OS nya dan master aplikasi untuk di install. namun Container tidak sepenuhnya seperti itu, dalam menjalankannya ini hanya me-read sistem OS (image.tar) dan terbentuklah Container bedasarkan image yang di load di dalam sistem Docker. sebenarnya image.tar ini seperti me-remastering linux sesuai kebutuhan khusus yang diinginkan dan mengemas ulang ke .ISO misal Linux Mint yang basisnya dari ubuntu gambar 11 adalah melihat isi image.tar.



Sumber: Hasil Penelitian (2019)

Gambar 11. Isi Image.tar

Container ini sebenarnya ada sistem OS pendukung yang disertakan walau tak selengkap dan power full dengan OS installer sepenuhnya pada desktop user dan perlu diperhatikan bahwa image.tar debian di atas hanya sebesar sekitar 100 MB saja.

4. Kesimpulan

Kesimpulan dari penelitian ini adalah Docker Container sangat cocok untuk desain arsitektur sistem dengan pendekatan microservice, karena masing-masing service memiliki lingkungan yang terisolasi namun tetap dapat berkomunikasi satu sama lain. Microservice sendiri merupakan suatu pendekatan desain arsitektur sistem informasi yang menspesifikasikan dan memecah fungsi dari sistem yang besar menjadi sistem atau service yang kecil dan spesifik. Adapun saran untuk penelitian selanjutnya adalah perlu banyak referensi tentang konfigurasi web server desain dan pengembangan aplikasi berbasis Container ini karena pada proses pengembangan aplikasi hal ini dapat mempengaruhi desain arsitektur suatu sistem.

Daftar Pustaka

Apridayanti S, Saputra RA, Informatika JT, Teknik F, Oleo UH, Kunci K, Web A. 2018. Desain dan implementasi virtualisasi berbasis. SEMANTIK 4: 37–46.

Bik MFR. 2017. Implementasi Docker Untuk Pengelolaan Banyak Aplikasi Web (Studi Kasus : Jurusan Teknik Informatika UNESA) Asmunin Abstrak. J. Manaj. Inform. 7: 46–50.

Hung L, Kristiyanto D, Lee SB, Yeung KY. 2016. GUIdock : Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research. 1–14.

Krochmalski J. 2017. Docker and Kubernetes for Java Developers, 1ste. Lennart Martens, UGent / VIB B, editor. Birmingham: Packt Publishing Ltd. 318 p.

Kurniawan A, Palit HN, Adjarwirawan J. 2016. Eksplorasi Pemanfaatan Docker untuk Mempermudah Pengelolaan Instalasi Komputer di Laboratorium Komputer Teknik Informatika Universitas Kristen Petra. INFRA 4: 2–7.

Sugianto, Masim Vavai., Marsan Susanto dkk. (2016). Virtualisasi Modern Berbasis Docker. Bekasi Timur: PT. Excellent Infotama Kreasindo.