BAB II

LANDASAN TEORI

2.1 *Virtualisasi*

Virtualisasi adalah suatu metode menggabungkan dan saling berbagi sumber daya teknologi informasi yang bertujuan untuk mempermudah pengelolaan dan meningkatkan penggunaan aset sehingga sumberdaya teknologi informasi lebih maksimal digunakan serta dapat memenuhi permintaan bisnis, dengan virtualisasi suatu perangkat komputer server dapat digunakan berbagai aplikasi dan sistem operasi sehingga seolah-olah beroperasi dalam beberapa aset. Hal ini dapat meningkatkan utilitas aset dan mengurangi biaya kebutuhan aset fisik. Inti dari virtualisasi adalah membuat suatu simulasi dari perangkat keras, sistem operasi, jaringan dan lainya. virtualisasi digunakan sebagai sarana untuk improvisasi skalibilitas dari perangkat keras yang ada .[5]

“ Menurut Alan Murphy dalam papernya *Virtualization Defined Eight Different Ways*, menyebutkan setidaknya terdapat delapan istilah dalam penerapan virtualisasi. Diantaranya adalah *virtualisasi* sistem operasi, *virtualisasi* aplikasi server, *virtualisasi* aplikasi, manajemen *virtualisasi*, *virtualisasi* jaringan, *virtualisasi* perangkat keras, *virtualisasi* penyimpanan dan layanan *virtualisasi."*

2.1.1 Manfaat *Virtualisasi*

Virtualisasi memberikan banyak manfaat dibandingkan infrastruktur biasa. yaitu diantaranya dengan *virtualisasi* dapat mengurangi biaya fiskal dan kemudahan dalam pengelolaan dan pemasangan tambahan. layanan *virtualisasi* dapat memungkinkan *guest* tertentu dapat diduplikasi dengan mudah sehingga dapat digunakan dalam melakukan testing aplikasi yang akan dipasang meski dalam lingkungan operasi sistem yang berbeda.

Berikut adalah beberapa manfaat menggunakan *virtualisasi* :

- **Meningkatkan Efektifitas Komputer server.**

  Virtualalisasi memberikan manfaat meningkatkan keefektifan penggunaan sumberdaya komputer seperti RAM dan Prosesor. serta penghematan terhadap energi listrik dengan sedikit pembelian terhadap fisikal server. penerapan virtualisasi tersebut dengan menggunakan beberapa *virtual machine* dalam satu fisikal server /*Host*.

- **Memudahkan dalam Proses *Backup* dan *Recovery*.**

  Setiap server yang dijalankan di dalam sebuah mesin virtual dapat disimpan dalam sebuah *image* yang berisi seluruh konfigurasi system. Jika suatu saat terjadi masalah pada server yang berjalan, maka tidak perlu memasang dan konfigurasi ulang, cukup dengan mengambil salinan dari *image* yang telah disimpan.

- **Memudahkan dalam Proses Pemasangan.**

  Dengan *virtualisasi* server virtual dapat dikloning dan dapat dijalankan pada komputer lain dengan mengatur sedikit konfigurasi sehingga mengurangi beban kerja para staf teknologi informasi dan dapat mempercepat proses implementasi suatu system.

- **Memudahkan dalam Pengelolaan dan Pemeliharaan.**

  Jumlah perangkat komputer yang lebih sedikit secara otomatis akan mengurangi waktu dan biaya dalam pengelolaan, serta memberikan kemudahan dalam proses pemeliharaan perangkat yang lebih sedikit.

- **Melakukan Standardisasi Perangkat Keras.**

  Virtualisasi akan melakukan dan enkapsulasi perangkat keras sehingga proses pengenalan dan pemindahan suatu spesifikasi perangkat keras tertentu tidak menjadi suatu masalah karena akan secara otomatis terbaca oleh sistem.

2.1.2 Tipe *Virtualisasi*

A. *Virtualisasi* Perangkat Keras.

Virtualisasi perangkat keras adalah upaya menciptakan mesin virtual yang bekerja layaknya sebuah komputer beserta dengan sistem operasi. dalam virtualisasi perangkat keras, mesin *host* adalah mesin tempat virtualisasi itu berada, dan mesin *guest* itu sendiri.

Ada beberapa jenis *virtualisasi* perangkat keras yaitu diantaranya:

- ***Virtualisasi* penuh** yaitu *virtualisasi* yang hampir menyerupai mesin asli dan mampu menjalankan sistem operasi tanpa perlu diubah .
- ***Virtualisasi* sebagian** yaitu *virtualisasi* di mana tidak semua aspek lingkungan disimulasikan dan tidak semua perangkat lunak dapat langsung berjalan, beberapa perlu disesuaikan agar dapat berjalan dalam lingkungan *virtual* tersebut.
- **Para *virtualisasi*** yaitu *virtualisasi* perangkat tidak disimulasikan tetapi perangkat lunak *guest* berjalan dalam domain nya sendiri seolah-olah dalam sistem yang berbeda. Dalam hal ini perangkat lunak tamu perlu disesuaikan untuk dapat berjalan.

B. *Virtualisasi* Perangkat Lunak

Virtualisasi perangkat lunak adalah *virtualisasi* yang memungkinkan satu komputer host untuk membuat dan menjalankan satu atau lebih lingkungan virtual. *virtualisasi* perangkat lunak banyak digunakan untuk mensimulasikan sebuah sistem komputer lengkap, dengan tujuan untuk memungkinkan sistem operasi *guest* berjalan.

2.2 Virtualisasi Kontainer

Di dalam konteks *virtualisasi Kontainer* dapat diartikan sebagai alat yang dapat dipergunakan untuk system yang terisolasi yang terletak pada level operasi sistem yang yang dijalankan pada satu induk *kernel* atau *Host*.

Terdapat 2 jenis *Kontainer* yang umum digunakan yaitu

1. *Kontainer* berbasis sistem operasi adalah suatu kontainer yang memberikan isolasi pada level sistem operasi dan memanfaatkan kernel yang sama dari suatu induk. contohnya: LXC, *openVZ* dan lainya.
2. *Kontainer* berbasis aplikasi adalah suatu kontainer yang memberikan isolasi pada level aplikasi dengan memanfaatkan beberapa komponen yang ada pada sistem operasi induk yang ditambah dengan beberapa komponen pada *kontainer* lain yang menjadi dasar dari berjalannya suatu aplikasi. Contohya: Docker,Rocket.

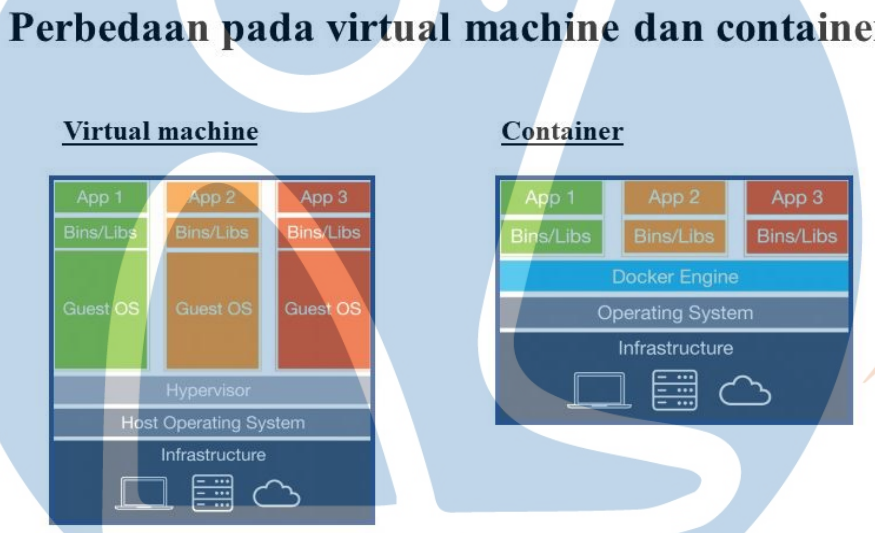
2.3 *Virtual Machine* dan *Kontainer*

Virtual machine adalah perangkat lunak yang dapat mengisolasi sebuah mesin komputer serta dapat menjalankan semua program seperti komputer aslinya atau duplikat dari komputer asli, sedangkan *kontainer* adalah suatu teknologi virtualisasi yang dapat mengisolasi sebuah proses dari proses yang lain nya yang akan mengisolasi *library* dan aplikasi yang digunakan saja tanpa mengisolasi seluruh komponen seperti perangkat keras kernel, serta sistem operasi.

Dalam teknologi virtualisasi *virtual machine* dan *docker* sama-sama memiliki konsep skema virtualisasi, akan tetapi ada yang membedakan antara *virtual machine* dan Kontainer, berikut adalah beberapa perbedaan mendasar antara *virtual machine* dan *kontainer*:

- *Virtual machine* menggunakan kernel tersendiri sehingga membuat beban pada *Host* menjadi lebih lebih berat sedangkan Kontainer membagi kernel nya kedalam kedalam kontainer yang sudah ada sehingga lebih efektif digunakan.

- *Virtual machine* menggunakan keseluruhan sumberdaya perangkat keras yang ada pada host sehingga *host* tersebut menjalankan operasi sistem secara ganda bersamaan, sedangkan kontainer bersifat seperti aplikasi dan hanya sedikit menggunakan sumberdaya pada *host.*
- *Virtual machine* tidak dapat mengalokasikan spesifikasi pada antar *virtual machine* sedangkan docker dapat mengalokasikan spesifikasi antar kontainer sehingga dapat melakukan efesiensi sumber daya dengan sebaik-baiknya pada sistem.



Gambar 2.1 perbedaan VM dan Docker

2.4 *Docker*

Docker adalah sebuah *platform* terbuka bagi para pengembang dan sistem administrator untuk membangun, menyebarkan, dan menjalankan aplikasi terdistribusi baik pada perangkat pribadi seperti laptop ataupun pada virtual data center atau *cloud* data senter.

Docker kontainer berjalan pada sebuah mesin tunggal yang berbagi pakai sistem operasi kernel yang sama, dan dapat dimulai secara langsung serta menggunakan lebih sedikit RAM. *Image* (sekumpulan berkas dan folder) dibangun dari *filesystem* berlapis dan berbagi berkas umum, membuat penggunaan penyimpanan dan *image* download jauh lebih efisien.

2.4.1 komponen-komponen Docker

- ***Docker image*** adalah dasar template untuk *docker kontainer*, sebuah *images* yang berisi operasi sistem maupun aplikasi yang akan diinstall dan telah jadi. image ini digunakan untuk menjalankan kontainer, docker index terdapat banyak *image* yang dapat di pilih dan dapat digunakan sebagai base *image*.
- ***Docker Kontainer*** adalah sebuah *image* yang dapat dikemas dan dibaca tulis, kontainer berjalan di atas nya. pada setiap perubahan yang dapat disimpan pada kontainer akan menyebabkan terbentuknya layer baru diatas base image.
- ***Docker registry*** adalah sebuah repositori distribusi kumpulan docker image yang terpusat baik bersifat public dan private repositori. registry publik disebut dengan dockerhub, didocker hub dapat dilakukan *push* dan *pull image* sendiri.
- ***Docker file*** adalah skrip yang berisi atau terdiri dari rangkaian perintah yang dieksekusi secara otomatis dan berurutan yang bertujuan membangun sebuah image. Dengan menggunakan perintah *docker build* dari terminal, maka kan terlihat *docker* membangun image secara bertahap berdasarkan eksekusi perintah dalam skrip.

- ***Docker Repository*** yaitu tempat penyimpanan *docker image* yang telah dibuat oleh pengembang,

2.4.2 Keunggulan Docker sebagai *Platform* Kontainer

Sistem Kontainer merupakan solusi terhadap masalah bagaimana untuk menjalankan perangkat lunak secara handal ketika pindah dari suatu lingkungan komputasi ke lingkungan yang lain. sehingga dapat berpindah dari pengembang untuk lingkungan tes, dari lingkungan rilis kedalam produksi dan mungkin dari perangkat fisik di data senter untuk mesin virtual atau *public cloud*.

Berikut adalah beberapa keunggulan platform docker:

- **Dapat Melakukan Pengujian dan Distribusi Secara Terus Menerus.**

  Docker mempunyai kemampuan dalam menghadirkan konsistensi diseluruh lingkungan sistem operasi. selalu ada perbedaan kecil antara lingkungan pengembangan dan aplikasi yang telah rilis. Kecuali jika anda memiliki lingkungan repositori sendiri sendiri dengan pemeriksaan yang lebih ketat. Perbedaan perbedaan karena versi paket yang berbeda atau dependensi tidak menjadi masalah, *docker* dapat mengatasi kesenjangan tersebut dengan memastikan lingkungan yang konsisten dari pembangunan untuk produksi. kontainer dikonfigurasi untuk memelihara semua konfigurasi dan dependensi internal, yang bertujuan agar anda dapat menggunakan kontainer yang sama pada tahap pengembangan hingga produksi perangkat lunak dan memastikan tidak ada perbedaan dan *intervensi* manual. dengan menggunakan *docker* kontainer anda juga dapat memastikan bahwa pengembang tidak perlu linkungan produksi yang mengatur secara identik. sebaliknya pengembang dapat menggunakan sistem mereka sendiri untuk menjalankan *docker kontainer* pada *virtualbox*.

- **Docker Dapat Menjadi *Platfom Multi-Cloud*.**

  Manfaat *docker kontainer* lainya adalah portabilitas, *Docker* kontainer dapat dijalankan didalam *Amazon EC2*, *Google compute engine*, *Server Rackspace* atau *virtualbox* asalkan *host* mendukung platform *docker*. selain AWS dan GCP *docker* dapat bekerja sangat dengan berbegai penyedia IaaS lain seperti *Microsoft Azure* dan *Openstack* dan dapat digunakan berbagai manajemen konfigurasi seperti *chef, bunnet, Ansible*.

- **Standarisasi Lingkungan dan Kontrol Versi.**

  Kontainer docker memastikan konsistensi di beberapa siklus pengembangan, rilis dan standardisasi lingkungan. *docker* kontainer bekerja seperti repositori GIT yang dapat melakukan perubahan *Image docker* dan dapat mengendalikan versi.

- **Docker Dapat Melakukan Isolasi.**

  Docker memastikan pada setiap kontainer memiliki sumberdaya sendiri yang terisolasi dengan kontainer lainnya yang berada dalam satu lingkungan, sehingga dapat memiliki dapat memiliki berbagai kontainer untuk aplikasi yang terpisah yang berjalan sepenuhnya pada tumpukan yang sama.[5]

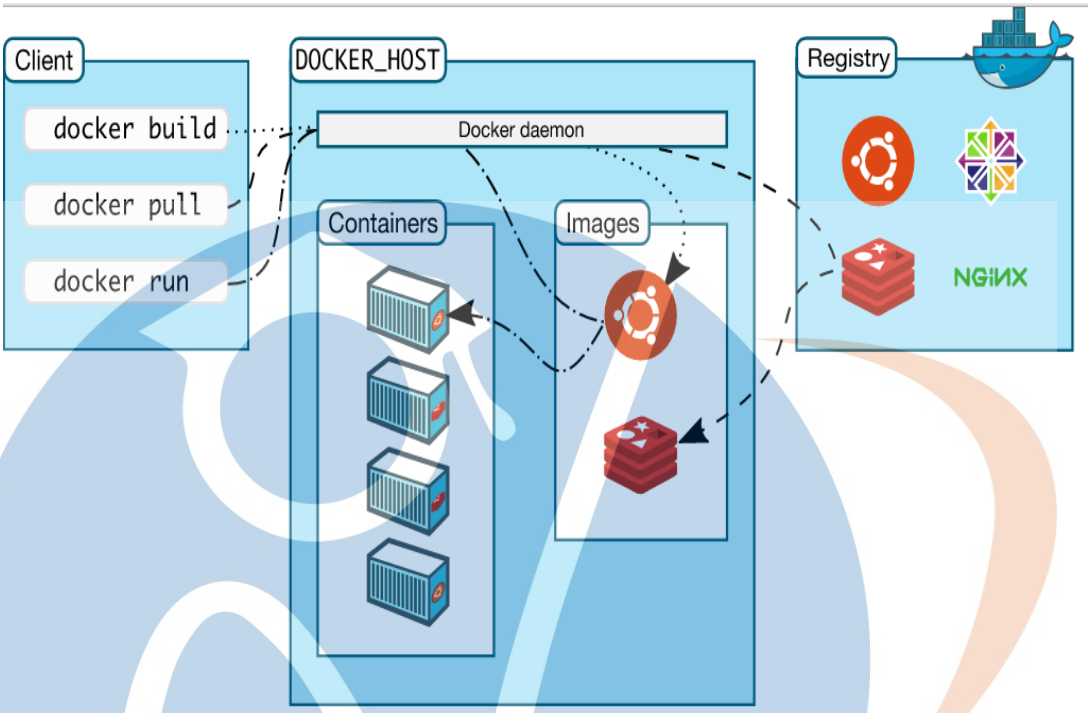
2.4.3 Kelebihan Docker dari Mesin Virtual (*virtual machine*)

Metode mesin virtual yang biasa digunakan untuk mendistribusikan aplikasi dan mengisolasi eksekusi nya adalah menggunkan virtual mesin atau VMs. Format virtual mesin biasanya adalah VMWare's dalam *vmdk, Oracle Virtualbox'x vdi, dan Amazon EC2's ami*. Secara teori format tersebut akan mengijinkan setiap developer secara otomatis membungkus aplikasinya kedalam sebuah "mesin" untuk kemudahan distribusi dan pemasangan. Namun pada praktiknya, hal itu hampir tidak terjadi karena beberapa alasan yaitu diantaranya::

- ***Size***: mesin virtual sangat besar yang membuatnya tidak praktis untuk disimpan dan dipindahkan .
- ***Performance***: menjalankan mesin virtual mengkonsumsi CPU dan memory secara signifikan yang membuatnya kurang praktis.
- ***Portability***: lingkungan virtual mesin kurang berjalan baik satu sama lain meskipun terdapat *tool* konversi ,akan tetapi terbatas dan menambah lebih banyak *overhead.*

Sebaliknya docker lebih ringan karena menggunakan metode sandboxing berbeda yang dikenal sebagai *containerization*. *docker* kontainer mengambil tempat pada level kernel. banayak sistem kernel sistem operasi modern saat ini mendukung kebutuhan untuk containerazation, seperti linux dengan *OpenVZ , LXC* dan lainya. *Docker* dibangun atas *primitive low-level* tersebut menawarkan developer sebuah format portable dan lingkungan runtime yang memecahkan 4 layer, Doker pada dasarnya memiliki nol overhead memory dan CPU , Docker benar-benar portable dan didesain dengan *application centric* .

2.4.4 Aritektur Docker



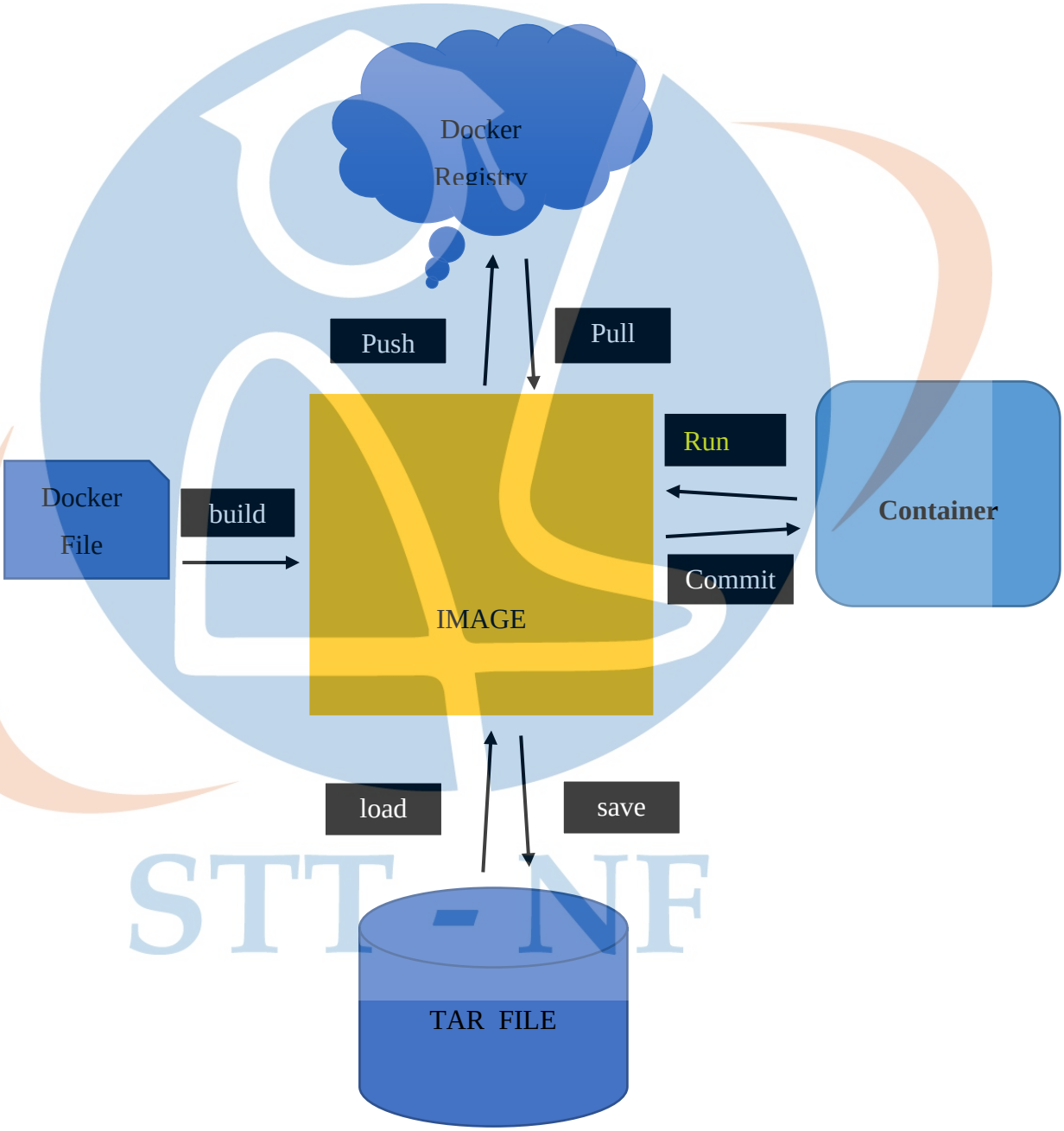
Gambar 2.2 Arsitektur Docker [3]

Docker menggunakan arsitektur berbasis klien server. Dalam hal ini, *docker* klien mengirimkan permintaan berupa sebuah perintah kepada *docker* daemon untuk membangun, mendistribusikan, dan menjalankan Docker Kontainer. Docker klien dan server berkomunikasi via *socket* menggunakan *RestfulSPI*. [2]

Terlihat pada gambar, Docker *daemon* berjalan pada *host*, sehingga pengguna tidak dapat berinteraksi langsung. Untuk mengaksesnya, menggunakan docker klien yang merupakan tampilan utama untuk pengguna, sehingga pengguna dapat berkomunikasi dengan docker *daemon*.

2.4.5 Mengenal Alur Kerja Docker

Dalam implemetasi nya ada beberapa perintah *docker* yang sering digunakan yaitu diantaranya *build, push, pull, run, commit*, dibawah ini adalah ilustrasi sederhana dari alur kerja *docker*.



2.4.6 Perkembangan Docker

Saat ini virtualisasi dengan *docker* merupakan standar alat berbasis kontainer yang banyak digunakan serta diintegrasikan oleh projek PaaS (*platform as servis*) seperti *Flynn, Deis,Dokku* dan *vragrant*.

Docker memiliki kelebihan dalam melakukan isolasi, banyak perusahaan besar maupun personal yang telah membuktikan bahwa docker sangat efektif dan ringan dalam menjalankan aplikasi sehingga menjadi lebih cepat dibandingkan dengan *virtual machine*. virtualisasi kontainer sangat cocok jika diterapkan pada *cloud computing* karena dapat berjalan baik pada aplikasi ataupun servis yang bergantung pada server.

2.5 *Dependency Hell* (Masalah ketergantungan)

Permasalahan ketergantungan timbul karena *shared package* dan *library* dimana beberapa paket lain memiliki dependensi tetapi tergantung pada perbedaan dan versi yang tidak kompatibel dari sebuah *shared packages*. Jika *shared package* atau *library* hanya bisa diinstall menjadi satu versi, pengguna atau administrator akan perlu untuk mengatasi permasalahan yang didapat dari versi lama/baru dari ketergantungan paket.

Permasalahan umum bagi developer adalah sulitnya mengelola semua *dependency* aplikasi dalam sebuah cara otomasi yang sederhana. Hal ini merupakan bentuk dari *dependency hell* tersebut diatas.

Contoh permasalahan dependency hell yang disebutkan oleh developer Docker ada 3 yaitu *Cross-platform dependencies*, *conflicting dependencies* dan *Custom dependencies*.

2.5.1 *Cross-Platform Dependencies*

Aplikasi modern seringkali bergantung pada kombinasi sistem *library* dan *binary*, paket bahasa yang spesifik, modul framework spesifik, komponen internal yang dikembangkan untuk projek lain, dan lain-lain. *Dependencies* tersebut hidup pada dunia yang berbeda dan memerlukan alat yang berbeda, alat tersebut biasanya tidak bekerja dengan baik antara satu dan lainnya, membutuhkan integrasi perubahan yang janggal.

Lintas platform (*cross platform*) adalah istilah teknologi informasi mengenai sebuah pernagkat lunak yang dapat digunakan di beberapa sistem informasi yang berbeda seperti mikrosoft windows, Linux, Mac dan lain sebagainya. Umumnya perangkat lunak yang memiliki kemampuan lintas *platform* adalah perangkat lunak bebas.

Contoh perangkat lunak yang memiliki kemampuan lintas *platform* ini antara lain:

- Apache HTTP server
- MySQL
- PHP

2.5.2 *Conflicting Dependencies*

Aplikasi yang berbeda akan bergantung pada versi yang berbeda dalam satu *dependency*. mempaketkan alat mengatasi situasi tersebut dengan berbagai tingkat kemudahan, namun ke semua itu mengatasinya dengan berbeda dan cara yang tidak kompatibel, yang lagi-lagi memaksa pengembang melakukan kerja ekstra.

2.5.3 *Custom Dependencies*

Seorang developer mungkin perlu menyiapkan sebuah versi kostum dari ketergantungan aplikasinya. Beberapa sistem pemaketan bisa mengatasi versi kostum dari sebuah *dependency*, namun yang lainnya tidak bisa dan mengatasi secara berbeda.

2.6 Penelitian Terkait

Dalam penyusunan skripsi ini, penulis sedikit banyak terinspirasi dan mereferensi dari penelitian-penelitian sebelumnya yang berhubungan dengan masalah pada skripsi ini.Berikut ini penelitian terdahulu yang berhubungan dengan skripsi ini antara lain :

2.1 Table Penelitian Terkait

NO	JUDUL	NAMA,TAHUN	TUJUAN PENELITIAN	KESIMPULAN
1	Kontainer *dan* Docker : Teknik Virtualisasi Dalam Pengelolaan Banyak Aplikasi	Firmansyah Adiputra,2015	Penelitian ini bertujuan untuk melakukan literature terhadap teknologi virtualisasi dalam pengelolaan banyak aplikasi web yang biasanya dikelola dalam suatu webhosting. Dimulai dengan melihat konsep hosting aplikasi web tradisional dan *virtual machine* (VM).	Teknik kontainer merupakan solusi tepat dalam pengelolaan banyak aplikasi web pada suatu sistem hosting. Docker terbukti dapat menhadirkan layanan aplikasi web yang unggul dari sisi kinerja, dan hemat sumberdaya.
2	Implementasi docker untuk Pengelolaan	M. Fadlullah romadhon , 2017	Peneletian ini bertujuan untuk merancang dan	Telah berhasil dirancang dan

	Banyak Aplikasi web		melakukan implementasi leight virtualization dengan menggunakan linux kontainer (LXC) dan docker deployment untuk banyak aplikasi web.	diimplementasi *Lightweight Virtualization* dengan menggunakan Linux *Containers* (LXC) dan *Docker deployment* aplikasi *web* setelah dilakukan beberapa pengujian.
3	Implementasi dan Analisis Computer Clustering System dengan Menggunakan Virtualisasi Docker	Tanjung P Kusuma, Dr. Ir. Rendy Munadi, M.T, Danu Dwi Sanjoyo, S.T,M.T, 2017	Penelitian ini betujuan untuk melakukan implementasi layanan clustering server dengan menggunakan docker dan melakukan analisa serta memberikan kesimpulan dari performance system clustering meliputi parameter (throughput , latency, dan CPU utilization Menggunakan Docker.	Dapat mengoperasikan computer cluster menggunakan docker.

Dari hasil ketiga penilitian yang disebutkan dalam tabel diatas, posisi peneliti berdekatan dengan Firmansyah adiputra dengan judul " Kontainer dan Docker: Teknik Virtualisasi Dalam Pengelolaan Banyak Aplikasi" namun penelitian yang dilakukan oleh firmansyah adiputra lebih menjelaskan pada solusi pengelolaan banyak aplikasi

web sedangkan peniliti mengacu pada hasil analisis rancangan efektifitas arsitektur docker yang akan dirancang serta kompatibilitas docker dalam mengatasi perbedaan platform sistem operasi.