BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2. 1. Tinjauan pustaka

Tinjauan pustaka yang digunakan pada penelitian ini meliputi beberapa penelitian yang sudah dilakukan sebelumnya. Penelitian yang dilakukan oleh Sanjay P. Ahuja dan Niharika Deval, (2018) tentang Tentang Evaluasi Kinerja *IaaS Cloud Services* Dengan Benchmark Tingkat Sistem. Penelitian tersebut membandingkan beberapa *provider* IaaS: Amazon EC2, Google Compute Engine, Microsoft Azure dan IBM Softlayer, dengan menggunakan metode *benchmarking* untuk mengukur kinerja server, kinerja *file* I/O, kinerja jaringan, dan kinerja *variability* dari ke empat *provider*.

Penelitian yang dilakukan oleh Ida Sofiana, (2012) tentang perbandingan kinerja sistem virtualisasi dan layanan *Cloud Computing*. Penelitian tersebut membandingkan antara *mesin virtual* dan layanan *cloud computing* berdasarkan biaya, proses *setup* sistem, sumber daya yang dibutuhkan, serta kinerja (proses *login*, kirim dan baca *e-mail*, kirim dan baca *e-mail* dengan *attachment*). Berdasarkan penelitian tersebut, akan terlihat kelebihan dan kekurangan dari masing-masing sistem, sehingga membantu *system administrator* memilih solusi yang sesuai kebutuhan.

Penelitian yang di lakukan oleh Vishaka Balasubramanian Sekar, dkk (2017) tentang Perbandingan *Platform* Docker *Container* hosting AWS EC2 dengan

Joyent's Triton. Pada penelitian tersebut membandingan kinerja, keamanan, biaya, dan alat dari dua penyedia *cloud*, membandingkan docker yang di hosting di VM dan docker di tirton, dan Membandingkan Docker vs KVM dalam hal latensi dan penggunaan CPU untuk server web dan menyimpulkan bahwa Docker adalah alternatif yang ringan dan berkinerja tinggi untuk KVM.

Penelitian yang di lakukan oleh M. Agung Nugroho, M.Kom dan Rikie Katardi, (2018) tentang analisis kinerja penerapan *container* untuk load balancing web server pada raspberry pi. Pada penelitian tersebut menguji beban *request* terhadap singel *container* dan multi *container* untuk membandingkan kinerjanya. Pada uji coba multi *container* menerapkan load balancing dapat menunjukkan pembagian beban *resource* dari masing-masing *container* sehingga aplikasi dapat berjalan walaupun dengan beban *request* tinggi.

Penelitian yang dilakukan oleh Handik Yanwar Efendi, (2017) tentang uji performa linux *container* docker untuk *deployment* aplikasi berbasis web. Pada penelitian tersebut mengimplementasikan penggunaan docker pada *website* yang menggunakan *Content Management System* untuk melakukan pengujian kompatibilitas docker pada aplikasi berbasis *website* yang telah dibangun pada *Platform* dan lingkungan *host* yang berbeda dan melakukan pengujian terhadap efektifitas docker dalam penggunaan summber daya penyimpanan dan memori serta membandingannya dengan perangkat lunak virtualisasi berbasis *hypervisor*.

Penelitian yang dilakukan oleh Addri Pershance And Taga, (2017) tentang rancang bangun *web hosting* menggunakan docker *container* dan *clustering* pada CoreOS: docker *container*. Pada penelitian tersebut mengimplementasikan aplikasi berbasis *web* menggunakan docker dan melakukan pengujian terhadap efektifitas docker sebagai *software virtualisasi* modern dalam penggunaan sumber daya penyimpanan dan *memory* dan melakukan pengujian terhadap efektifitas CoreOS sebagai sistem operasi khusus *web* dalam penggunaan sumber daya penyimpanan dan memory.

2. 2. Dasar Teori

2.2.1. Internet

Secara harfiah, internet (kependekan dari *interconnected-networking*) ialah rangkaian komputer yang terhubung secara global di dalam beberapa rangkaian dan menggunakan TCP/IP sebagai protokol pertukaran paket (*packet switching communication protocol*), dimana didalamnya terdapat berbagai sumber daya informasi dari mulai yang statis hingga yang dinamis dan interaktif. (Natalia, 2011)

2.2.2. Virtual Private Server (VPS)

Virtual Private Server (VPS) adalah virtual machine yang dijual sebagai layanan oleh hosting *provider*, dalam VPS user bisa mengakses dan mengelola seluruh aspek *software* dari server termasuk akses *administrator* di sistem oprasi server sampai aplikasi yang akan di implementasikan di server tersebut.

VPS dapat dibagi menjadi beberapa VM (*Virtual Machines*), dimana di setiap VM adalah berupa "*Virtual server*" yang dapat di *install* system operasi tersendiri. VPS terasa seperti sebuah Dedicated Server. Dibanding dengan shared hosting, menyewa VPS akan mendapatkan *resource* yang lebih baik sehingga tidak terganggu jika ada problem pada *website* yang dikelola. Selain itu VPS mendapatkan root akses sehingga lebih leluasa dalam mengkustomasi server sesuai kebutuhan(Hamida, 2017).

2.2.3. Cloud Computing

Cloud Computing (komputasi awan) merupakan gabungan pemanfaatan teknologi komputer (komputasi) dalam suatu jaringan dengan pengembangan berbasis internet (awan) yang mempunyai fungsi untuk menjalankan program atau aplikasi melalui komputer – komputer yang terkoneksi pada waktu yang sama, tetapi tak semua yang terkonekasi melalui internet menggunakan *cloud computing*.

Teknologi komputer berbasis sistem *cloud* ini merupakan sebuah teknologi yang menjadikan internet sebagai pusat *server* untuk mengelola data dan juga aplikasi pengguna.Teknologi ini mengizinkan para pengguna untuk menjalankan program tanpa instalasi dan mengizinkan pengguna untuk mengakses data pribadi mereka melalui komputer dengan akses internet.

Tiga model layanan dari *cloud computing*, yaitu :

1. *Cloud Software as a Service* (SaaS). Kemampuan yang diberikan kepada konsumen untuk menggunakan aplikasi penyedia dapat beroperasi pada

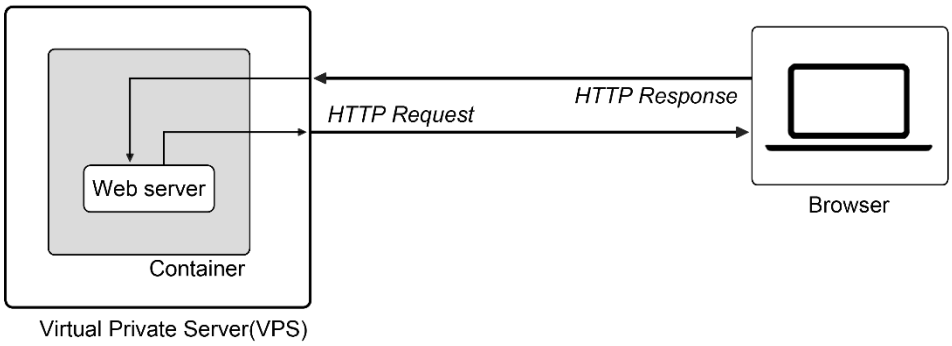
infrastruktur *cloud*. Aplikasi dapat diakses dari berbagai perangkat klien melalui antarmuka seperti *website* browser (misalnya, email berbasis *website*). Konsumen tidak mengelola atau mengendalikan infrastruktur *cloud* yang mendasar termasuk jaringan, *server*, sistem operasi, penyimpanan, atau bahkan kemampuan aplikasi individu, dengan kemungkinan pengecualian terbatas terhadap pengaturan konfigurasi aplikasi pengguna tertentu. Contohnya adalah Google Apps, SalesForce.com dan aplikasi jejaring sosial seperti FaceBook.

2. *Cloud Platform as a Service* (PaaS). Kemampuan yang diberikan kepada konsumen untuk menyebarkan aplikasi yang dibuat konsumen atau diperoleh ke infrastruktur *cloud computing* menggunakan bahasa pemrograman dan peralatan yang didukung oleh *provider*. Konsumen tidak mengelola atau mengendalikan infrastruktur *cloud* yang mendasar termasuk jaringan, *server*, sistem operasi, atau penyimpanan, namun memiliki kontrol atas aplikasi yang disebarkan dan memungkinkan aplikasi melakukan hosting konfigurasi. Contohnya yang sudah mengimplementasikan ini adalah Force.com dan Microsoft Azure investment.
3. *Cloud Infrastructure as a Service* (IaaS). Kemampuan yang diberikan kepada konsumen untuk memproses, menyimpan, berjaringan, dan sumber komputasi penting yang lain, dimana konsumen dapat menyebarkan dan menjalankan perangkat lunak secara bebas, yang dapat mencakup sistem operasian

aplikasi.Konsumen tidak mengelola atau mengendalikan infrastruktur *cloud* yang mendasar tetapi memiliki kontrol atas sistem operasi, penyimpanan, aplikasi yang disebarkan, dan mungkin kontrol terbatas komponen jaringan yang pilih (misalnya, firewall *host*). Contohnya seperti *Amazon Elastic Compute Cloud* dan *Simple Storage Service*(Basari, 2014).

2.2.4. Web Server

Web server adalah perangkat lunak yang berfungsi sebagai penerima permintaan yang dikirimkan melalui browser kemudian memberikan tanggapan permintaan dalam bentuk halaman situs *web* atau lebih umumnya dalam dokumen HTML. Namun, *web* server dapat mempunyai dua pengertian berbeda, yaitu sebagai bagian dari perangkat keras (*hardware*) maupun sebagai bagian dari perangkat lunak (*software*).



Gambar 2. 1 *Request Server*

Saat mengambil halaman *websitesite*, browser mengirimkan permintaan ke server yang kemudian diproses oleh web server. HTTP *request* dikirimkan ke web server. Sebelum memproses HTTP *request* , web server juga melakukan pengecekan

terhadap keamanan. Pada web server, HTTP *request* diproses dengan bantuan HTTP server. HTTP server merupakan perangkat lunak yang bertugas menerjemahkan URL (alamat situs *website*) serta HTTP (protokol yang digunakan browser untuk menampilkan halaman *website*). Kemudian web server mengirimkan HTTP response ke browser dan memprosesnya menjadi halaman situs *website*.

Pada saat web server menerima HTTP *request* dari browser, jika diperlukan web server akan mengirimkan query ke database untuk memenuhi permintaan HTTP *request* yang dikirimkan oleh browser(Yasin, 2018).

2.2.5. Docker

Docker pertama kali diperkenalkan oleh Solomon Hykes, pendiri dan CEO ofdotCloud dalam presentasi singkatnya di Python *Developers Conference* di Santa Clara, California, pada tanggal 15 Maret 2013. Pada saat diperkenalkannya Docker, hanya sekitar 40 orang diberi kesempatan untuk mencoba Docker. Beberapa minggu kemudian proyek ini mengejutkan beberapa kalangan karena Docker adalah sebuah proyek open-source dan tersedia untuk umum pada GitHub, di mana orang bisa mengunduh dan berkontribusi dalam proyek ini.

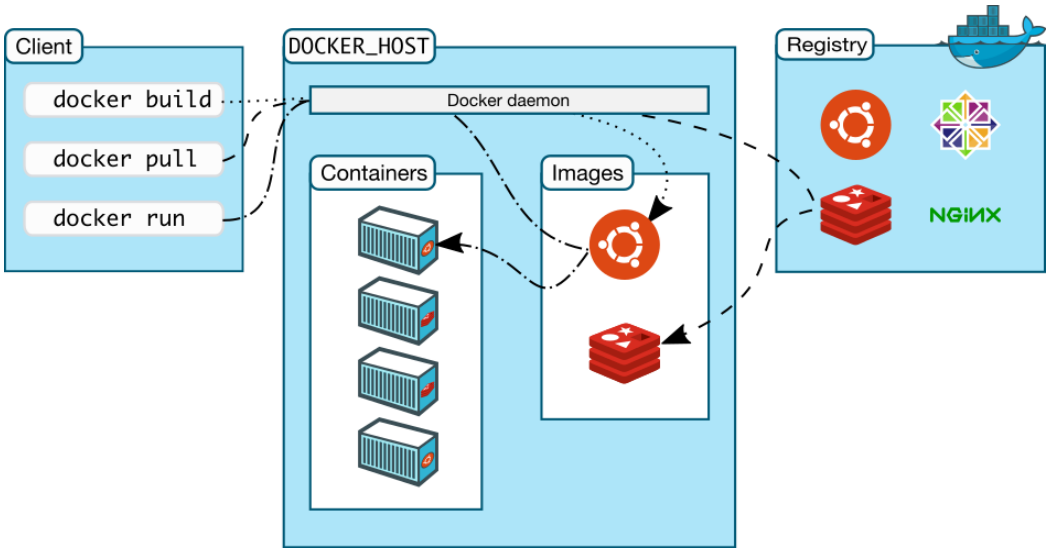
Docker merupakan project open-source yang menyediakan *Platform* terbuka dalam bentuk teknologi virtualisasi berbasis *container*, ditujukan bagi para *developer* maupun *sysadmin* untuk dapat membangun, membundel dan menjalankan aplikasi dimanapun dalam satu *container* yang ringan. Mirip seperti *virtual machine* namun

lebih ringan karena Docker tidak membawa keseluruhan sistem operasi, melainkan berbagi sistem dengan *host* induknya.

Docker menyatukan perangkat lunak dalam *filesystem* lengkap yang berisi semua yang diperlukan menjalankannya: *source code*, paket sistem untuk runtime, perangkat sistem, sistem pustaka *software* apapun yang dapat diinstal pada server. Hal ini menjamin bahwa perangkat lunak akan selalu berjalan sama, tidak tergantung pada lingkungannya. Pada perkembangan terkini, Docker bahkan bisa dijalankan diatas sistem Windows maupun MacOS(Nugroho, 2018).

2.2.6. Arsitektur Docker

Arsitektur Docker menggunakan *client* dan server. Docker *client* mengirimkan *request* ke docker daemon untuk untuk membangun, mendistribusikan, dan menjalankan *container* docker. Keduanya docker *client* dan daemon dapat berjalan pada sistem yang sama. Antara docker *client* dan docker daemon berkomunikasi via *socket* menggunakan RESTful API.



Gambar 2. 2 Arsitektur Docker

1. Docker daemon berfungsi untuk membangun, mendistribusikan dan menjalankan *container* docker. User tidak dapat langsung menggunakan docker daemon, akan tetapi untuk menggunakan docker daemon maka user menggunakan docker *client* sebagai perantara atau cli.
2. Docker *images* adalah sebuah template yang bersifat *read only*. Template ini sebenarnya adalah sebuah OS atau OS yang telah di *install* berbagai aplikasi. Docker *images* berfungsi untuk membuat docker *container*, dengan hanya 1 docker *images* kita dapat membuat banyak docker *container*.
3. Docker *container* bisa dikatakan sebagai sebuah *folder*, dimana docker *container* ini dibuat dengan menggunakan docker daemon. Setiap docker *container* disimpan maka akan terbentuk *layer* baru tepat diatas docker *images* atau base *image* diatasnya.

4. Docker *registry* adalah kumpulan docker *image* yang bersifat private maupun public yang dapat anda akses di docker hub. Dengan menggunakan docker *registry*, anda dapat menggunakan docker *image* yang telah dibuat oleh developer yang lain, sehingga mempermudahkan kita dalam pengembangan aplikasi.
5. Docker *client* adalah seperangkat perintah *command line* untuk mengoperasikan docker *container*, misalnya membuat *container*, *start/stop container*, menghapus (*destroy*), dan sebagainya. Docker *client* hanya bertugas mengirim perintah saja. Pekerjaan sesungguhnya dilakukan oleh docker daemon(Nugroho, 2018).

2.2.7. Benchmarking

Benchmarking adalah proses evaluasi *performance* dari suatu sistem pada kondisi tertentu. *Benchmarking* dilakukan dengan menjalankan sebuah atau kumpulan program yang dinamakan *benchmarking tools*. Program tersebut akan menjalankan operasi-operasi tertentu untuk menguji kemampuan dari suatu sistem. *Benchmarking tools* dalam dunia komputasi secara umum menguji beberapa aspek dalam komputer seperti kemampuan CPU, *storage*, *graphics*, maupun sistem secara keseluruhan(Tjandra dkk, 2016).

Cara lain untuk mengukur kinerja adalah untuk mengidentifikasi tugas-tugas khusus dan mengukur kinerja dalam microbenchmarks. Microbenchmarks dalam hal ini berguna bukan untuk memutuskan semua sistem yang digunakan, tetapi untuk

menunjukkan sejumlah prosesor yang menarik. Sehingga investigasi menyeluruh tentang prosesor ini dapat dilakukan. Kelemahan utama dari benchmark ini tentu saja sulit untuk menemukan tugas-tugas yang kecil dan representatif. Kelemahan lainnya adalah jika aplikasi yang dimaksud tidak dipahami dengan baik maka hal ini akan menarik kesimpulan yang salah dari microbenchmarks. Keuntungan utama adalah bahwa hal ini masuk akal untuk membandingkan berbagai macam sistem karena benchmark sendiri tidak terlalu sulit untuk dibuat(Ehliar, 2004).

2.2.8. Apache Benchmark

Apache Benchmark(AB) adalah alat untuk mengukur kinerja apache *Hypertext Transfer Protocol* (HTTP) server, dengan ab kita dapat melihat kapabilitas apache untuk melayani request dari client. Aplikasi apache benchmark ini bisa digunakan tidak hanya untuk Apache, tapi untuk web server lain juga seperti Lighthttpd, Nginx atau Miscrosoft IIS. AB menggunakan tampilan cli untuk pengujiannya dan output yang dihasilkan berupa teks cli.

2.2.9. Sysbench

Sysbench adalah sebuah *benchmark tool* yang digunakan untuk menghitung kemampuan CPU, *memory* dan *file* I/O. Pengujian dilakukan dengan membuat sebuah *instance* yang paling mendekati dengan sebuah *node*, yaitu sebuah *instance* dengan menggunakan *flavor* terbesar.

Pengujian ditujukan untuk mengetahui seberapa dekat kemampuan yang dicapai oleh sebuah *instance* dengan kapabilitas sebuah *instance* dibandingkan

dengan *host* yang sesungguhnya. Oleh karena itu, pengujian dilakukan terhadap sebuah *host* dan terhadap sebuah *instance*(Kuntani dkk, 2015).

2.2.10. Throughput

Throughput adalah jumlah *bit* yang diterima dengan sukses perdetik melalui sebuah sistem atau media komunikasi dalam selang waktu pengamatan tertentu. Umumnya *throughput* direpresentasikan dalam satuan *bit* per *second* (bps). Aspek utama *throughput* yaitu berkisar pada ketersediaan *bandwitch* yang cukup untuk menjalankan aplikasi. Hal ini menentukan besarnya trafik yang dapat diperoleh suatu aplikasi saat melewati jaringan. (Muhammad Syaiful Adnan, 2017)

2.2.11. Response Time

Waktu Tanggap (*Response Time*) adalah waktu tanggap yg diberikan oleh antar muka/*interface* ketika user me*request*/ mengirim permintaaan ke komputer. Secara umum, pengguna menginginkan bahwa program aplikasinya dapat memberikan waktu tanggap yang sependek - pendeknya. Tetapi waktu tanggap yang baik memang tidak dapat ditentukan, karena ada beberapa aspek yang mempengaruhi, antara lain yakni ragam interaksi yang diinginkan dan kefasihan pengguna dalam menjalankan program aplikasi tersebut(Linda, 2015).

2.2.12. Pengertian Website

Website adalah halaman informasi yang disediakan melalui jalur internet sehingga bisa diakses di seluruh dunia selama terkoneksi dengan jaringan internet. *Website* merupakan komponent atau kumpulan komponen yang terdiri dari teks,

gambar, suara animasi sehingga lebih merupakan media informasi yang menarik untuk dikunjungi.

Secara teknologi, *website* adalah kumpulan dari halaman-halaman situs, yang biasanya terangkum dalam sebuah domain atau subdomain, yang tempatnya berada didalam *World Wide Web* (WWW) di internet. Sebuah halaman *website* adalah document yang ditulis dalam format HTML(*Hyper Text Markup Language*), yang hamper selalu bisa diakses melalui HTTP, yaitu protokol yang menyampaikan informasi dari server *website* untuk ditampilkan kepada para pemakai melalui web browser. Semua publikasi dari *website-website* tersebut dapat membentuk sebuah jaringan informasi yang sangat besar(Aryadi, 2017).

2.2.13. CMS

Content Management System atau CMS, pertama kali muncul sebagai jawaban atau solusi dari kebutuhan manusia akan penyediaan informasi yang sangat cepat. Jika melihat kebelakang, kita tahu bahwa betapa sederhananya sebuah *website* di era tahun 90-an. Dengan hanya mengandalkan bahasa pemrograman HTML dan beberapa gambar serta informasi yang statis, sebuah perusahaan berusaha sebaik mungkin menampilkan informasi secukupnya kepada para pengunjung.

CMS (Content Management System) atau Sistem Manajemen Konten merupakan sebuah sistem yang memberikan kemudahan kepada para pengguna dalam mengelola dan mengadakan perubahan isi sebuah *website* dinamis tanpa sebelumnya dibekali pengetahuan tentang pemrograman web. Setelah pengguna menginstall dan

mengkonfigurasi aplikasi CMS, selanjutnya pengguna dapat memulai “blogging” dengan konten-konten yang dapat diatur sendiri. Dengan begitu para pengguna baru akan semakin mudah dalam mengelola *website*-nya sendiri sesuai dengan keinginannya tanpa harus paham akan kode-kode pemrograman web.

2.2.14. WorpdPress

WordPress pertamakali didirikan pada tahun 2003 oleh Mike Little dan Matt Mullenweg saat membuat b2/cafelog. WordPress dibangun dengan menggunakan bahasa pemrograman PHP dan MySQL dan dilisensikan di bawah GPLv2. WordPress merupakan salah satu CMS (*Content Management System*) bersifat *open source* yang digunakan untuk membuat *website* dan lebih menekankan terhadap *accessibility, performance, security,* dan memudahkan bagi pengguna. Sifatnya yang *open source* membuat WordPress memiliki banyak komunitas dan dukungan. Karenanya wordpress menjadi *software* paling mudah untuk membuat *website* atau blog yang kuat. Karena cara penggunaanya yang mudah, WordPress juga dapat digunakan oleh orang yang kurang memahi pemrograman.