Seminar Nasional Inovasi Teknologi
UN PGRI Kediri, 22 Februari 2017

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

PEMANFAATAN DOCKER SWARM SEBAGAI KOLABORATOR *PRIVATE* DAN *PUBLIC CLOUD* UNTUK IMPLEMENTASI SCALABLE VIRTUALISASI

Toga Aldila Cinderatama[1], Hamdani Arif[2],
[1]Program Studi Teknik Informatika, Politeknik Kediri
[2]Program Studi Multimedia dan Jaringan, Politeknik Negeri Batam
E-mail: *[1] togaaldila@poltek-kediri.ac.id, *[2]hamdaniarif@polibatam.ac.id,

Abstrak **–** Berkembang pesatnya teknologi saat ini linier dengan semakin banyaknya aplikasi yang dibuat oleh software developer dimana diperlukan implementasi virtualisasi agar program yang dibuat dapat berjalan pada lintas platform dengan berbagai macam konfigurasi. Implementasi virtualisasi ini biasanya membutuhkan resource hardware yang banyak, yang tentunya akan membutuhkan biaya yang cukup besar. Teknologi docker sebagai salah satu platform yang dibangun untuk menjalankan virtualisasi dalam model container, dapat dimanfaatkan untuk membangun virtualisasi sistem operasi atau sebuah web server atau sebuah database server yang tentunya lebih ringan dibandingkan dengan teknik virtualisasi konvensional. Dalam perkembangannya, docker swarm hadir yang dapat berfungsi untuk membantu mengelompokkan container dari docker yang sudah ada untuk mewujudkan suatu private cloud. Scalability dan flexibility merupakan salah satu kelebihan dari cloud computing. Dengan mengkombinasikan docker yang bekerja di private cloud dan resource dari berbagai cloud service provider yang ada di *market*, diharapkan dapat mengimplementasikan suatu infrastructure hybrid cloud untuk mengatasi keterbatasan resource yang biasanya menjadi kendala pada implementasi virtualisasi di private cloud.

Kata Kunci *— docker, docker swarm, scalability, virtualisasi*

Abstract **–** *Utilizing Docker Swarm as private and public cloud collaborator for scalable virtualization implementations. The rapid growth of today's technology is linear with the increasing number of applications developed. It requires the implementation of virtualization so that the programs created can run cross-platform with a wide variety of configurations. The virtualization implementations typically require a lot of hardware resources, which would require considerable cost. Docker as one of the platforms that are built to run in a virtualized container models, can be used to build a virtualized operating system or a web server or a database server that is certainly lighter than the conventional virtualization technique. On its development phase, Docker swarm emerged as new way to help classify the existing docker containers to create a private cloud. Scalability and flexibility is one of the advantages of cloud computing. By combining docker who worked in private cloud resource and resources from various cloud service providers on market, the implementaion of a hybrid cloud infrastructure would become easier. Hence this hybrid infrqstructure is expected to address resource limitations that usually become an obstacle on the implementation of virtualization in a private cloud.*

Keywords *— docker, docker swarm, scalability, virtualization*

1. PENDAHULUAN

Munculnya Cloud Computing menyebabkan terjadinya suatu perubahan baru di domain pengembangan aplikasi. Dengan pemanfaatan potensi dari teknologi

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

virtualisasi dan service orchestration, Cloud Computing muncul sebagai paradigma baru yang menjanjikan. Tiga jenis model [6] layanan yang disediakan antara lain adalah Infrastructure as Service (IaaS), Platform as Service (PaaS) dan Software as a Service (SaaS). Paradigma ini tumbuh pesat menjadi ladang riset dan bisnis baru bagi dunia teknologi informasi.

Di satu sisi, server konvensional sebagai pusat dari pelayanan suatu sistem jaringan, baik penyimpanan data maupun pengelolaan informasi, telah menjadi kebutuhan dasar dari banyak instansi di Indonesia. Seiring berjalannya perkembangan dunia teknologi dan informasi muncullah permasalahan baru yakni semakin besarnya ukuran data yang perlu diolah dan semakin bervariasinya jenis data yang perlu diproses. Hal ini berimbas pada meningkatnya kebutuhan sumber daya untuk pengolahan. Disini, pengadaan dan perawatan infrastruktur IT yang membutuhkan biaya besar bukan menjadi pilihan solusi, karena dinamisnya kebutuhan yang tidak akan tepat bila dipasangkan dengan resource yang statis.

Di sisi lain, Cloud Computing semakin mendapat perhatian lebih berkat kemampuan scalability-nya, dimana resource yang dipakai oleh suatu system dapat di tambahkan atau pun dikurangi sesuai dengan kebutuhan. Skema pembayaran pada cloud computing memungkinkan pengguna untuk membayar sesuai apa yang mereka pakai. Ini tentu menjadi suatu alasan baik untuk memakai teknologi Cloud computing dalam rangka pemenuhan QoS (Qualitiy of Service).

Namun, dibalik kemegahan yang nampak pada cloud computing, ada beberapa aspek yang memang tidak bisa diakomodasi secara menyeluruh. Salah satu aspek tersebut berkaitan dengan privacy data. Dalam dunia cloud computing, kontrol berkaitan informasi personal pelanggan secara tidak langsung dapat diketahui oleh cloud service provider (CSP). Merujuk pada paper Yunchuan Sun [5], ada tiga potensi besar ancaman pada cloud computing, antara lain, security, privacy, and trust. Singkatnya dalam pemanfaatan cloud computing akan lebih bijak bila tidak seluruh pekerjaan di ¬outsource keluar dari data center pribadi instansi, terutama apabila berkaitan dengan data konfidential.

Docker dengan virtualisasi model container belakangan ini hadir sebagai teknologi alternative dalam pengembangan system berbasis cloud. Docker menawarkan pengisolasian aplikasi dan dependensinya pada masing-masing container yang bertujuan untuk mempercepat proses development. Container mampu memberikan kekuatan System Administrator untuk dapat membangun, mengirimkan dan menjalankan aplikasinya pada multiple platform dalam waktu singkat. Docker kini juga mulai memperkenalkan tool untuk pengembangan sistem terdistribusi yang disebut Swarm. Docker Swarm merupakan native clustering pada Docker. Paper ini mengusulkan pengimplementasian docker swarm sebagai kolaborator private dan public cloud untuk mencapai dua keuntungan. Pertama, tercapainya cost effective dalam penggunaan resource sesuai kebutuhan organisasi. Kedua, memungkinkan adanya hak akses kontrol untuk memilah mana data kredensial yang perlu diolah pada *private* cloud dan mana data yang bisa di *outsource* ke public cloud seperti Digital Ocean, Amazon Web Service ataupun Cloud Provider lainnya

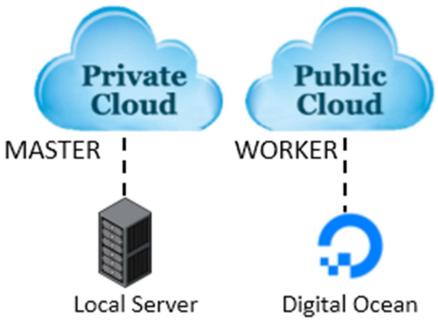
2. METODE PENELITIAN

Area riset bidang informatika di Indonesia saat ini masih didominasi pada domain pengembangan sistem informasi pada layer aplikasi. Tren tersebut terjadi seiring maraknya migrasi dari sistem lama yang manual dan tradisional menuju otomasi sistem dengan bantuan teknologi informasi dan komunikasi yang banyak terjadi di Indonesia. Hal ini lah yang menyebabkan penulis menemukan sebuah masalah patologis baru dalam tatanan dunia informatika tanah air.

Dengan dikemukakannya masalah penulis memiliki beberapa tahap dalam rangka menemukan pemecahan kasus tersebut. Beberapa tahapan yang dilakukan penulis adalah sebagai berikut; 1. Identifikasi masalah, 2. Perumusan masalah, 3. Penelusuran pustaka, 4. Rancangan Sistem Model, 5. Pengumpulan data dan informasi, 6. Pengolahan dan Implementasi, dan 7. Penyimpulan hasil. Dengan adanya penelitian ini, diharapkan munculnyua ide baru di area riset bidang teknologi infomasi dan komunikasi di Indonesia di masa mendatang.

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

3. DESAIN SISTEM



Gambar 3.1. Desain Sistem

Desain sistem yang digunakan adalah perpaduan dari private cloud dalam hal ini dapat menggunakan personal komputer maupun notebook, yang berperan sebagai Master node dalam arsitektur docker swarm. Master node ini berperan sebagai manager untuk mengkoordinasi worker node. Dalam implementasi sistem yang diterapkan, worker node merupakan host-host yang ada pada cloud, dalam hal ini memanfaatkan public cloud yaitu dari droplet-droplet pada public cloud DigitalOcean (www.digitalocean.com).

4. IMPLEMENTASI SISTEM

4.1 Instalasi Docker Toolbox

Tahap pertama implementasi adalah kita perlu menginstall docker toolbox. Docker toolbox ini dapat diunduh dari situs docker.com dan kita sesuaikan docker toolbox yang diunduh sesuai dengan sistem operasi yang kita gunakan. Pada implementasi ini, akan diinstall docker toolbox pada sistem operasi MacOS

Gambar 4.1. Docker toolbox

Setelah selesai melakukan isntalasi docker toolbox, kita akan mendapatkan Docker Quickstart Terminal yang merupakan command line interface (CLI) tempat melakukan konfigurasi. Jika menggunakan laptop yang berbasis macOS, setelah proses instalasi selesai dapat menuju ke Launchpad dan memilih Docker agar terbuka jendela CLI seperti yang terlihat pada gambar 3

Gambar 4.2. Docker CLI

Konfigurasi yang kita lakukan setelah docker siap adalah melakukan pengujian apakah docker engine sudah berjalan dengan sukses, pengujian yang dilakukan dapat melalui instalasi image hello-world

```
user$ docker run hello-world
```

Pada saat instalasi docker toolbox, selain docker engine yang telah terinstall pada komputer kita, maka docker machine adalah tools lain yang otomatis juga akan terinstall. Docker machine ini nantinya akan digunakan untuk virtualisasi komputer kita, dimana di dalamnya akan melakukan instalasi virtual box. Untuk melihat docker machine yang aktif pada computer kita dapat digunakan sintaks berikut

Pada tahap awal setelah instalasi docker toolbox, akan terdapat satu docker machine yang aktif yaitu host virtual box dengan IP yaitu 192.168.99.100:2376.

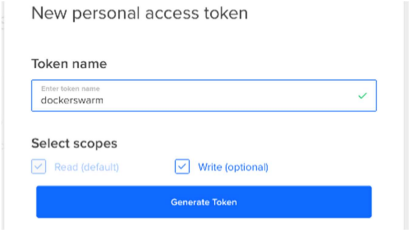
```
user$ docker-machine ls
NAME              ACTIVE       DRIVER
STATE                          URL
SWARM    DOCKER     ERRORS
default      *              virtualbox
Running    tcp://192.168.99.100:2376
v1.13.0
```

4.2 Menambahkan Host dari Public Cloud

Untuk menambahkan host baru, yaitu dengan memanfaatkan resource dari cloud provider (public cloud) langkah dilakukan adalah membuat akun di digital ocean. Selanjutnya setelah berhasil melakukan registrasi pada public cloud digitalocean.com kita perlu men-generate token baru melalui menu API dan kita generate token baru dan memasukkan nama token misalkan

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

dockerswarm, seperti yang terlihat pada gambar 4.3.



Gambar 4.3. Input token name pada Digital Ocean

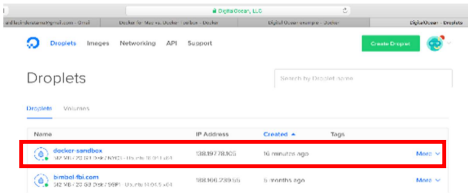
Dari proses generate diatas kita akan mendapatkan suatu token yang nantinya akan digunakan untuk membuat suatu cloud host. Maka kita perlu mencatat token yang digenerate, pada contoh implementasi cloud host di digitalocean.com, token yang dibuat adalah:

0e09af3a7e378e577584e886024910aacc772cf2f2b4c05f82c63983b7708626

Penulis akan menggunakan token tersebut untuk membuat cloud host agar ditambahkan di docker-machine pada komputer master. Pada komputer master, tampilkan kembali docker-machine yang aktif, kemudian tambahkan 1 cloud host dari digital ocean tersebut dengan perintah dibawah ini.

```
user$ docker-machine create --driver digitalocean --digitalocean-access-token 0e09af3a7e378e577584e886024910aacc772cf2f2b4c05f82c63983b7708626 docker-sandbox
.....
Setting Docker configuration on the remote daemon...
Checking connection to Docker...
Docker is up and running!
To see how to connect your Docker Client to the Docker Engine running on this virtual machine, run: docker-machine env docker-sandbox
```

Jika cloud host yang kita buat berhasil maka pada akun digitalocean kita akan terdapat satu droplet baru yang menandakan bahwa cloud host kita berhasil dibuat seperti pada gambar 4.4.



Gambar 4.4. Cloud host pada DigitalOcean

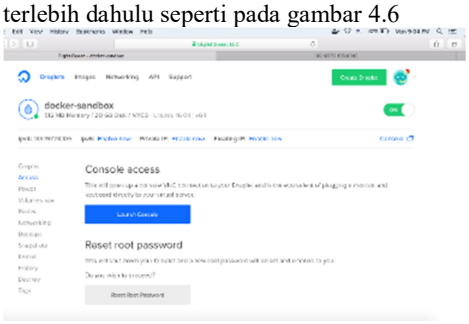
Untuk melihat list dari docker-machine yang telah dibuat, dapat diperiksa dengan mengetikan perintah

Akan muncul satu host baru yaitu docker-sandbox. Pada point ini, kita telah berhasil membuat 2 host yaitu host pada komputer kita sebagai private cloud dan satu cloud host yang dibuat di cloud provider digitalocean sebagai representasi dari public cloud.

```
user$ docker-machine ls
NAME          ACTIVE   DRIVER       STATE     URL
SWARM   DOCKER    ERRORS
default       *        virtualbox   Running
tcp://192.168.99.100:2376          v1.13.0
docker-sandbox   -        digitalocean   Running
tcp://138.197.78.105:2376          v1.13.0
```

Gambar 4.5. List docker machine

Pada gambar 4.5 terlihat bahwa machine yang aktif adalah pada komputer local kita. Selanjutnya untuk menguji apakah docker engine telah terinstall dengan benar di public cloud, dapat dilakukan pengetesan dengan menjalankan docker command basic, misalkan seperti yang dilakukan penulis adalah menjalankan container webserver dalam contoh ini nginx dan menampilkan halaman hello world nginx. Langkah-langkah yang dilakukan adalah kita terhubung ke remote machine di public cloud dengan protokol ssh, jika kita tidak mengetahui password dari root pada remote machine, maka perlu dilakukan reset root password terlebih dahulu seperti pada gambar 4.6



Gambar 4.6. Reset root password pada droplet DigitalOcean

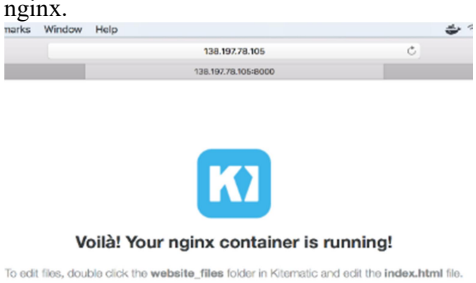
Selanjutnya kita menjalankan pengujian yaitu menampilkan hello world dari webserver nginx dimana kita menggunakan command seperti pada gambar 4.7 yang terdapat parameter –p yaitu akan mengekspos port 80 dari nginx container dan membuat agar bisa diakses pada port 8000 pada docker-sandbox host. Command selengkapnya terlihat seperti pada gambar 4.7

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

```
root@docker-sandbox:~# docker run -d -p 8000:80 --name webserver kitematic/hello-world-nginx
Unable to find image 'kitematic/hello-world-nginx:latest' locally
latest: Pulling from kitematic/hello-world-nginx
……
Digest: sha256:ec0ca6dcb034916784c988b4f2432716e2e92
Status: Downloaded newer image for kitematic/hello-world-nginx:latest
6a42f8521190ba553fd8f05cbd03ad16a02757228e3d597b932c7dba30547655
```

Gambar 4.7. Pengujian docker engine pada remote host.

Hasil pengujian ini dapat dilihat pada browser akan menampilkan hello-world nginx.



Gambar 4.8. Hasil pengujian docker engine pada remote host

Sampai pada tahap ini kita telah berhasil membuat dan menguji remote machine pada public cloud. Jika kita menginginkan lebih banyak remote host machine di public cloud kita dapat menambahkan host yang lain dan tahapan yang dilakukan adalah sama seperti saat membuat docker-sandbox host. Kita generate token baru kemudian dari token tersebut kita jalankan kembali docker command untuk menghasilkan remote machine pada public cloud. Token yang degenerate pada tahap ini adalah : 5c1928f196b0fd94706b2eba926cc255174bdf4d36fde5340f41d8eb6afab711.

Kemudian kita buat host baru dengan nama docker-sandbox2

```
~ user$ docker-machine create --driver digitalocean --digitalocean-access-token 5c1928f196b0fd94706b2eba926cc255174bdf4d36fde5340f41d8eb6afab711 docker-sandbox2
```

Dan kita periksa apakah remote machine kedua telah berhasil. Dari langkah diatas kita telah memiliki satu host local dan dua remote host di DigitalOcean. Langkah selanjutnya adalah membuat infrastruktur clustering hybrid cloud dimana menggabungkan resource lokal yaitu komputer kita dengan resource dari cloud provider. Maka tahapan yang dilakukan adalah kita menginstall docker swarm pada host lokal yang berperan sebagai master node, dan cloud host pada digitalocean akan berperan sebagai worker node.

4.3 Instalasi Docker Swarm

4.3.1 Konfigurasi Master Node

Sebelumnya kita telah menampilkan docker machine yang aktif pada computer kita yaitu dengan IP 192.168.99.100, untuk membangun suatu clustering pada virtualisasi docker ini, maka dibutuhkan instalasi docker swarm. Perintah untuk melakukan instalasi docker swarm adalah sebagai berikut

```
user$ docker swarm init --advertise-addr 192.168.99.100
```

Dengan perintah diatas akan menginisialisai node pada suatu clustering yang akan dibuat dimana host yang telah aktif akan menjadi Master node yang bertugas untuk mengkoordinasi worker-worker node yang akan kita buat pada proses selanjutnya.

4.3.2 Konfigurasi Worker Node

Tahap selanjutnya dalam implementasi ini adalah menambahkan worker node yang dimana resource yang digunakan adalah public cloud dari digitalocean.com. Setelah kita mengeksekusi perintah pada bagian inisialisasi master node, akan dihasilkan output juga berupa perintah untuk menambahkan worker node. Contoh perintahnya adalah sebagai berikut.

```
To add a worker to this swarm, run the following command:
docker swarm join \
--token SWMTKN-1-58s4xnffbpvuf9r7gtqnvl8vjxwh00zi8exws0cbupsq3mkglt-3lpqj4kcmpzyidyikjgd5n5v7 \
    192.168.99.100:2377
```

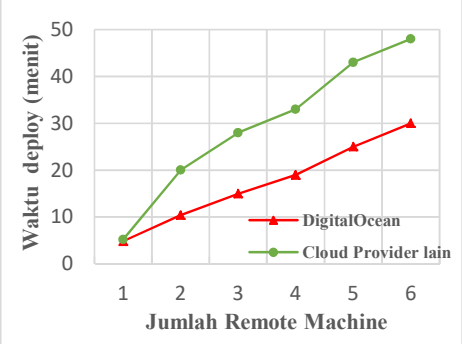
Maka tahapan selanjutnya yang perlu dilakukan adalah melakukan koneksi remote ke worker-worker node dalam hal ini docker-sandbox dan docker-sandbox2 machine, kemudian langkah selanjutnya kita perlu menjalankan perintah tersebut agar dapat

ISBN : 978-602-61393-0-6
e-ISSN : 2549-7952

menambahkan worker node ke system clustering kita. Setelah proses tersebut selesai maka kita telah mempunyai sebuah system virtualisasi yang menggabungkan resource local dari server kita dengan resource-resource dari publik cloud provider.

5. PENGUJIAN DAN EVALUASI SISTEM

Dalam pengujian ini, penulis akan mengukur variasi waktu deploy dari remote host machine yang dibuat pada public cloud. Penulis membandingkan beberapa percobaan waktu deploy untuk remote host machine di DigitalOcean dengan cloud provider lain yaitu AWS EC2



Gambar 5.1. Deployment time

Pada gambar 5.1 terlihat bahwa waktu generate host machine pada cloud provider DigitalOcean relatif stabil dengan rata-rata waktu deploy adalah 5 menit, sedangkan pada cloud provider lain penulis mencoba membandingkan dengan waktu deploy instance pada Amazon AWS EC2 dimana waktu untuk membuat machine adalah bervariasi. Hal ini dikarenakan pengaruh perbedaan penempatan lokasi server saat pembuatan virtual machine. Pada DigitalOcean rata-rata machine akan ditempatkan pada server New York, sedangkan pada Amazon AWS EC2 ada beberapa pengaturan region untuk penempatan machine seperti East USA region atau West USA region.

6. SIMPULAN

Dengan pengimplementasian aplikasi Docker banyak keuntungan yang dapat diperoleh terutama saat menerapkan teknik virtualisasi pada server lokal, karena Docker merupakan aplikasi yang ringan. Penggunaan docker juga dapat dijadikan solusi saat mengalami keterbatasan resource pada server dengan mengkombinasikan resource lokal dan resource dari public cloud provider.

7. SARAN

Penelitian ini dapat dikembangkan dengan implementasi clustering pada beberapa public cloud provider yang berbeda. Kemudian dapat diteliti bagaimana konfigurasi penggunaan resources dari cloud yang optimal seperti pemilihan hardware resource, lokasi server dan lain-lain. Sehingga infrasturktur clustering yang dibangun dapat digunakan untuk implementasi aplikasi-aplikasi skala besar seperti map reduce atau aplikasi berskala besar lainnya.

DAFTAR PUSTAKA

[1] Kurniawan A., Palit H.N., Adjarwirawan J. 2016. Eksplorasi Pemanfaatan Docker untuk Mempermudah Pengelolaan Instalasi Komputer di Laboratorium Komputer Teknik Informatika Universitas Kristen Petra. *Jurnal Infra*. Vol 4 No 2.

[2] Adiputra, F. 2015. Container Dan Docker: Teknik Virtualisasi Dalam Pengelolaan Banyak Aplikasi Web. *Jurnal Simantec Vol 4, No 3.*

[3] Docker. 2017. Install Docker Machine. https://docs.docker.com/machine/install-machine/ diakses pada tanggal 10 Januari 2017.

[4] Docker. 2017. Create a Swarm in Docker. https://docs.docker.com/engine/swarm/swarm-tutorial/create-swarm/ diakses pada tanggal 15 Januari 2017.

[5] Yunchuan Sun et.al. 2014. Data Security and Privacy in Cloud Computing. International Journal of Distributed Sensor Networks, Volume: 10 issue: 7.

[6] P. Melland, T. Grance. 2011. The NIST definition of cloud computing Recommendations of the National Institute of Standards and Technology. Special Publication 800-145.

[7] Zainal A. Hasibuan. 2007 Metodologi Penelitian pada Bidang Ilmu Komputer dan Teknologi Informasi. Fakultas Ilmu Komputer Universitas Indonesia.