Tabel 1. Layanan koreksi PPP yang sudah beroperasi

Layanan	Penyedia	Target GNSS	Jumlah Stasiun Referensi	Com. Link	Penerima	Akurasi	Negara
StarFire	Navcom	GPS, GLONASS	> 40	3 GEO (L-band), IP	NavCom	< 5 cm	Amerika Serikat
Apex/Ultra TerraStar	Veripos	GPS, GLONASS, GALILEIO, BEIDOU, QZSS (Apex)	∼ 80	7 GEO (L-band)	Veripos, Novatel, Septentrio, TOPCON, Hemisphere	Horizontal (H): < 5 cm Vertikal (V): 12 cm (95%)	Inggris
SeaStar	Fugro	GPS, GLONASS, GALILEO, BEIDOU	∼ 80	6 GEO (L-band), IP (NTRIP)	Fugro	H: 5 cm V: 15 cm (95%)	Belanda
CenterPoint RTX	Trimble	GPS, GLONASS, GALILEO, BEIDOU, QZSS	∼ 100	6 GEO (L-band), IP (NTRIP)	Trimble, Qualcomm	H: 2 cm V: 5 cm (RMS)	Amerika Serikat
magicGNSS	GMV	GPS, GLONASS, GALILEO, BEIDOU, QZSS	∼ 80	IP (NTRIP)	RTCM SSR	H: 5 cm V: 8 cm (RMS)	Spanyol

estimasi orbit dan jam satelit GNSS berdasarkan data pengamatan yang diakuisisi oleh jaringan CORS (*Continuously Operating Reference Station*) yang tersebar di regional Jepang. Untuk dapat mengakses produk tersebut, pengguna dapat mengaksesnya melalui perangkat lunak NTRIP (*Networked Transport of RTCM via Internet Protocol*) Caster melalui internet. Selain itu, karena MADOCA menggunakan format SSR (*State-Space Representation*) dalam mendsitribusikan produk koreksinya, pengguna harus menggunakan perangkat keras (penerima) dan perangkat lunak yang mendukung.

Pustaka

[1] Novatel, *An Introduction to GNSS Chapter 5 Resolving Errors*, url: `https://www.novatel.com/an-introduction-to-gnss/chapter-5-resolving-errors/precise-point-positioning-ppp/` (diakses pada tanggal 24 Februari 2019).

[2] Takasu, Tomoji, 2019, *PPP Correction Service–Current and Futuere, 24th GPS/GNSS Symposium 2019*, 16–18 October 2019, Tokyo, Japan.

[3] Gao, yang, 2006, *What is Precise Point Positioning (PPP), and What Its Requirements, Advantages and Challenges?*, GNSS Solutions.

[4] MADOCA: `ssl.tksc.jaxa.jp/madoca/public/public_index_en.html`

[5] PPP-WIZARD: `www.ppp-wizard.net`

■

TEKNOLOGI INFORMASI

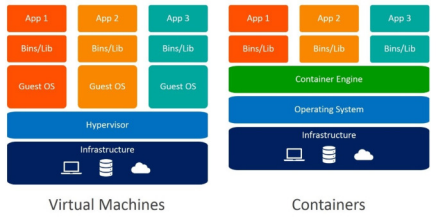
Container: Arsitektur Modern dalam Virtualisasi

Oleh
Y. Andrian | Pussainsa LAPAN

Setiap aplikasi atau perangkat lunak membutuhkan dukungan dari *system libraries, component, binaries* serta pengaturan konfigurasi sistem yang sesuai dengan kebutuhan. Semua dukungan tersebut merupakan *dependency* dan *environment* yang harus tersedia agar aplikasi dapat berjalan dengan baik di dalam sistem operasi. Kita juga tahu bahwa tidak semua aplikasi mempunyai kebutuhan yang sama, tergantung dari jenis dan bahasa pemrograman yang digunakan. Ketika seorang pembuat aplikasi (*programmer*) mengembangkan perangkat lunak, seringkali menemui kendala saat ingin melakukan *deploy* aplikasi ke

mesin yang berbeda untuk fase uji coba atau produksi karena adanya perbedaan *environment*. Hal tersebut mengakibatkan program atau aplikasi yang awalnya berhasil dijalankan pada mesin lokal milik *programmer*, namun mengalami *error* yang menyebabkan program gagal berjalan sempurna di *server* atau komputer lain. Selain itu, permasalahan yang sama dapat muncul pada sistem administrator yang bertanggung jawab pada *server*, ketika mereka hendak melakukan pembaharuan sistem operasi (kernel), ada kemungkinan terjadi masalah pada beberapa aplikasi di *server* karena belum mendukung kernel versi terbaru.
Permasalahan-permasalahan tersebut dapat diatasi dengan menggunakan teknologi kontainer. Kontainer akan mengemas aplikasi bersama dengan kode, *runtime*, dan dukungan lainnya sehingga aplikasi dapat berjalan dan kompatibel dengan berbagai *environment*.

Container atau Kontainer menurut Kamus Besar Bahasa Indonesia (KBBI) mempunyai arti kotak kargo. Sedangkan arti kontainer secara harfiah dalam bahasa Inggris adalah suatu lingkungan terpisah yang diatur oleh sistem operasi yang dapat menjalankan satu atau lebih aplikasi dengan menetapkan sumber daya yang diperlukan. Sama halnya seperti kontainer pada pelabuhan yang berfungsi untuk mengisolasi benda agar tidak tercampur dengan yang lain, pada dunia virtualisasi pun kontainer memiliki fungsi mengisolasi aplikasi beserta *dependency* dan *environment*-nya dari aplikasi lain yang terdapat di dalam suatu komputer. Gambar 1 adalah bagan perbedaan lapisan antara



Gambar 1. Arsitektur Virtual Mesin dan Kontainer
(Sumber: `https://wave.works`)

arsitektur virtual mesin dan kontainer. Virtual mesin terdapat *Hypervisor* yang berfungsi sebagai *Virtual Machine Manager* (VMM) dan melakukan abstraksi dari perangkat keras fisik menjadi perangkat keras virtual untuk mendistribusikan beban kerja dari semua mesin virtual (VM) ke masing-masing perangkat keras secara proporsional seperti *hard drive*, RAM, *processor* yang akan dibuat pada mesin virtual. Dalam virtual mesin dapat membangun beberapa virtual mesin dengan sistem operasi yang berbeda-beda. Pada arsitektur kontainer terdapat *container engine* yang berfungsi untuk membangun *image* dan kontainer dengan memanfaatkan beberapa fitur pada kernel pada sistem operasi seperti cgroups, namespace, chroot dan SE Linux. Pada teknologi kontainer, tingkat abstraksi terjadi hanya pada kernel sebuah sistem operasi, sehingga kontainer yang terbentuk akan mempunya sistem operasi yang sama dan terisolasi berdasarkan *environment* dan *dependency*-nya.

Teknologi kontainer sudah berkembang pada era tahun 1960-1970an dan berfungsi sebagai bagian dari sistem operasi Linux yang mengisolasi *environment* hanya dibatasi pada *chroot* atau disebut metode *chroot jail*. Metode ini akan membuat direktori *root* virtual untuk pengguna yang masuk ke sistem dengan seolah-olah menjadikan *home* direktori pengguna sebagai *root* (/). Keuntungan dari metode ini adalah membatasi pengguna agar tidak dapat mengakses ke tingkat direktori yang lebih tinggi dan mengambil informasi penting dari sistem. Perkembangan era kontainer modern dimulai sekitar tahun 2005 ketika Sun Solaris 10 merilis fitur *Solaris Container* yang merupakan implementasi dari sistem operasi tingkat virtual pertama dengan menerapkan zona untuk menjalankan aplikasi tertentu. Kemudian istilah kontainer menjadi populer untuk para *developer* pada setiap konfigurasi *environment* sistem operasi dengan membatasi lingkup *file system* dan mengisolasi sebuah proses dari semua *resources* kecuali yang diijinkan secara eksplisit. Saat ini, ada beberapa platform kontainer terkenal yang sering digunakan yaitu Docker, RedHat Openshift, Hyper-V rkt, dan LMCTFY. Diantara semua platform kontainer yang tersedia, para *developer* lebih memilih Docker sebagai platform mereka. Hal ini ditunjukkan oleh hasil survei yang dilakukan oleh situs Statistica 2 tahun lalu bahwa pada tahun 2018 Docker mempunyai persentase tertinggi untuk penggunaan teknologi kontainer yang sering digunakan. Pesatnya perkembangan teknologi kontainer tidak terlepas karena beberapa keuntungan yang didapat dari penerapan teknologi ini. Berdasarkan laporan dari situs Gartner, pada tahun 2023, diprediksi lebih dari 70% organisasi global akan menjalankan lebih dari dua aplikasi dalam satu kontainer pada fase *production*. Kontainer menyediakan fleksibilitas secara keseluruhan jika dibandingkan dengan *server* secara fisik dan virtual mesin (VM). Para

Tabel 1. Perbandingan Kontainer dan Virtual Mesin

Fitur	Kontainer	Virtual Mesin
Isolasi	Isolasi ringan pada level sistem operasi	Isolasi penuh pada level *hardware*
Sistem Operasi	Menjalankan pada *user space* di sistem operasi, setiap kontainer menyesuaikan kebutuhan minimal dari aplikasi	Menjalankan secara utuh pada sistem operasi termasuk kernel dan sumber daya lain (CPU, *memory*, *drive*, dst)
Waktu *Start-up*	Hitungan detik	Hitungan menit
Performa	Maksimal tergantung pada hardware fisiknya	Terbatas pada konfigurasi VM

developer dapat menjalankan aplikasi mereka pada sistem operasi manapun tanpa mempengaruhi aplikasi lain yang terdapat di sistem operasi. Selain itu ada beberapa keuntungan lainnya jika menerapkan teknologi kontainer:

- *Overhead* kecil
  Kontainer membutuhkan sumber daya yang lebih sedikit dibanding *environment server* tradisional dan perangkat keras VM karena kontainer tidak menyertakan *images* dari sistem operasi.
- Ringan dan Portabel
  Kontainer merupakan virtualisasi yang ringan karena hanya menyediakan kebutuhan minimum dari aplikasi agar dapat berjalan dan tidak berjalan di atas *hypervisor* layaknya VM. Selain itu, ukuran kontainer yang terbilang kecil, sehingga memudahkan dalam penggunaannya di platform lain walaupun dengan sistem operasi yang berbeda.
- Efisien
  Karena butuh sumber daya minimal, kontainer memungkinkan aplikasi untuk lebih cepat diterapkan, *patched* dan penggunaan pada skala yang lebih besar.
- Meningkatkan keamanan
  Keamanan aplikasi lebih tinggi karena diisolasi dari *host* sistem dan kontainer lainnya.
- Operasional yang lebih simpel
  Manajemen aplikasi menjadi lebih mudah dikarenakan hanya butuh sebuah sistem operasi yang dapat menjalankan berbagai kontainer.

Virtualisasi sudah menjadi tren di era IT modern saat ini, dari beberapa keuntungan teknologi kontainer di atas, kita dapat mengetahui alasan kenapa saat ini *developer* lebih memilih kontainer dibandingkan dengan teknologi VM.

Penggunaan teknologi kontainer ini diprediksi akan terus meningkat di masa yang akan datang karena sangat menunjang kinerja para *developer* dalam memenuhi kebutuhan teknologi informasi yang semakin besar serta cocok dengan *agile process* yang mempercepat proses pengerjaan perangkat lunak mulai dari proses *developing*, *staging*, *testing*, hingga *production*. Teknologi kontainer bermanfaat untuk dukungan pengembangan perangkat lunak di Pussainsa. Saat ini, pengembangan prototype aplikasi untuk sistem pengambil keputusan (DSS) yang dibangun di Pussainsa mulai menerapkan teknologi kontainer sebagai arsitektur sistemnya. Beberapa DSS yang dikembangkan diantaranya adalah DSS Cuaca Antariksa, DSS Navigasi, DSS Benda Jatuh Antariksa, dan DSS Komunikasi Radio. Sistem berbasis teknolgi kontainer juga sangat mendukung dalam pendekatan arsitektur *microservices* dimana sistem terbagi menjadi beberapa aplikasi kecil (*service*) sehingga memudahkan *developer* dalam pengembangan sistem berkelanjutan.

Pustaka

[1] kbbi.web.id/kontainer

[2] https://www.lexico.com/definition/container

[3] http://masrifqi.staff.ugm.ac.id/index.php/2007/10/chroot-jail-menggunakan-openssh/

[4] https://openlibrary.telkomuniversity.ac.id/pustaka/files/93271/resume/implementasi-dan-analisis-sun-containers-pada-solaris-10.pdf

[5] www.biznetgio.net/en/news/mengenal-docker-kubernetes

[6] www.cio.com/article/3434010/more-enterprises-are-using-containers-here-s-why.html

[7] https://docs.docker.com/engine/faq/

[8] docs.microsoft.com/en-us/virtualization/windowscontainers/about/containers-vs-vm ■