Teknologi Virtualisasi: Virtualisasi Sistem Operasi

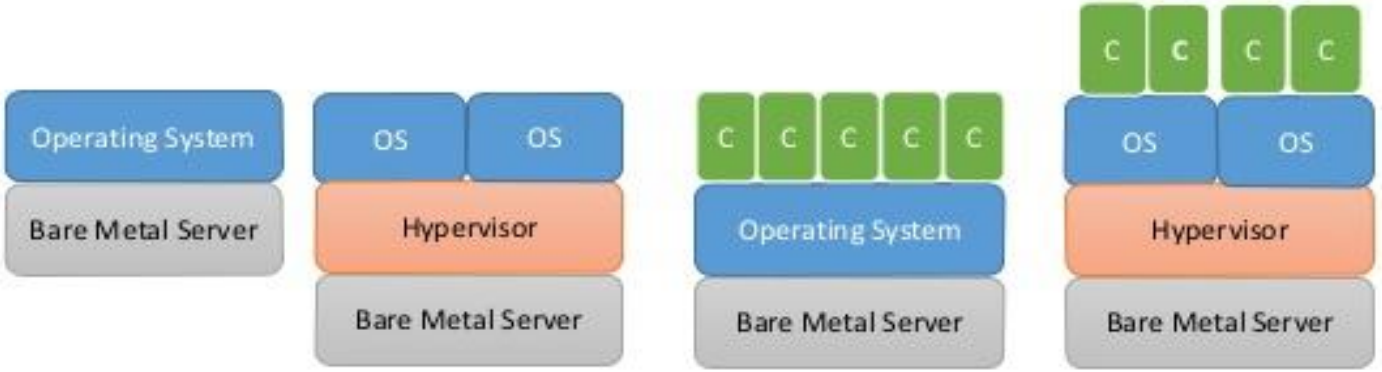
Henry Saptono,S.Si,M.Kom.
Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Desember 2020

Apa itu Virtualisasi Sistem Operasi ?

- **Virtualisasi sistem operasi** atau **Virtualisasi pada tingkat sistem operasi** (**OS-level Virtualization**) , adalah sebuah teknik atau teknologi **virtualisasi software** yang memanfaat **kernel sistem operasi host secara bersama sama** oleh **server virtual** atau **instance ruang pengguna (*user-space instances*)** untuk menjalankan satu atau beberapa aplikasi atau program yang masing masing server virtual atau instance ruang pengguna **terisolasi** satu dengan yang lainnya, bahkan masing masing tidak menyadari bahwa mereka menggunakan atau mengakses sistem operasi bersama.
- Virtualisasi tingkat sistem operasi ini juga dikenal atau sering disebut dengan istilah **Container based Virtualization**

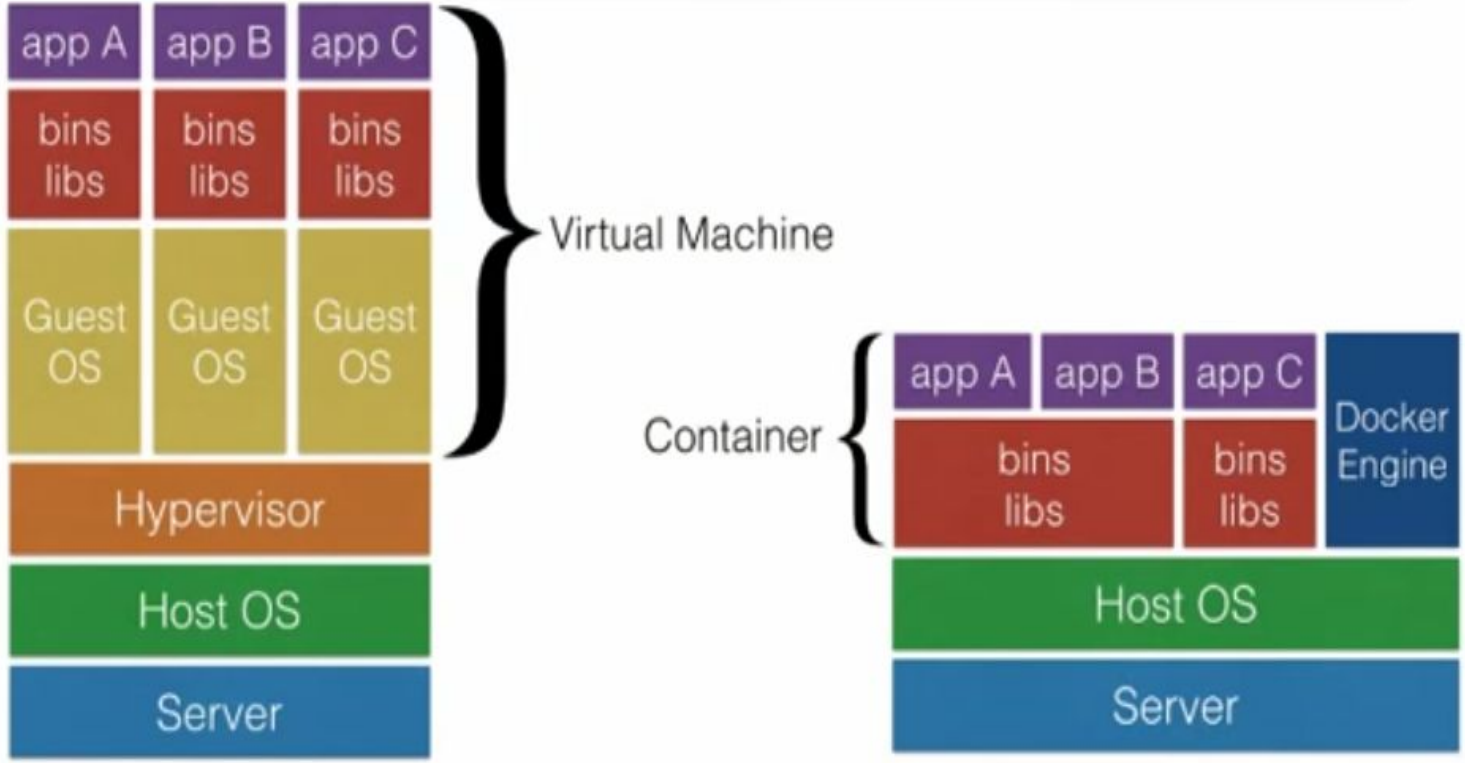
What are Containers?

- OS Level virtualization where kernel allows for multiple isolated user-space instances



https://www.teimouri.net/wp-content/uploads/what_are_containers-1.jpg

Virtual Machines vs Containers



Virtual Mesin v.s Container

VM membutuhkan banyak sumber daya sistem. Setiap VM tidak hanya menjalankan salinan lengkap sistem operasi, tetapi juga salinan virtual semua perangkat keras yang perlu dijalankan oleh sistem operasi. Ini dengan cepat menambah banyak RAM dan siklus CPU. Sebaliknya, semua yang dibutuhkan container adalah cukup sistem operasi, program dan pustaka pendukung, dan sumber daya sistem untuk menjalankan program tertentu.

Virtual Mesin v.s Container

Artinya dalam praktiknya adalah Anda dapat menempatkan dua hingga tiga kali lebih banyak aplikasi pada satu server dengan container daripada yang dapat Anda lakukan dengan VM.

Selain itu, dengan container, Anda dapat membuat lingkungan operasi yang portabel dan konsisten untuk pengembangan, pengujian, dan penerapan.

Level virtualisasi berbasis container

Virtualisasi berbasis container dapat diterapkan pada level:

- **OS**: Inilah yang disebut virtualisasi sistem operasi. Contoh software: Linux Container/LXC, OpenVZ, Windows Container, Solaris Container, FreeBSD Jails
- **Aplikasi**: Ini disebut virtualisasi aplikasi. Contoh software: Docker, Rkt

Keunggulan virtualisasi OS

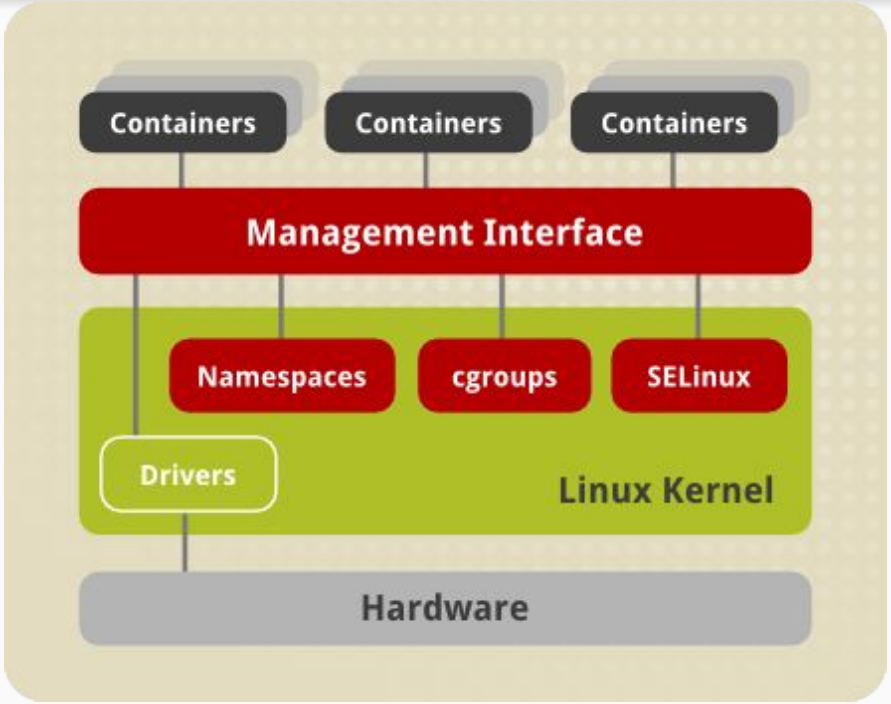
- Virtualisasi OS biasanya membebankan sedikit atau tidak ada biaya tambahan.
- Virtualisasi OS mampu melakukan migrasi langsung
- Dapat menggunakan load balancing dinamis dari kontainer antara node dan cluster.
- Mekanisme copy-on-write (CoW) tingkat file dimungkinkan pada virtualisasi OS yang membuat lebih mudah untuk membuat cadangan file, lebih hemat ruang dan lebih sederhana untuk cache daripada skema copy-on-write tingkat blok.

Penggunaan virtualisasi os

- Digunakan untuk lingkungan hosting virtual.
- Digunakan untuk alokasi yang aman dari sumber daya perangkat keras yang terbatas di antara sejumlah besar pengguna
- Administrator sistem menggunakannya untuk mengintegrasikan perangkat keras server dengan memindahkan layanan pada host terpisah.
- Untuk meningkatkan keamanan dengan memisahkan beberapa aplikasi ke beberapa wadah (container).
- Bentuk virtualisasi ini tidak memerlukan perangkat keras untuk bekerja secara efisien.

Virtualisasi OS: Linux Container

Arsitektur Linux Container



Komponen Linux Container

Kernel linux dilengkapi dengan beberapa sub sistem kernel yang memungkinkannya dilakukan **containerization** di linux, yaitu terdiri dari:

- **Namespaces** - Namespace memastikan isolasi proses
- **Cgroups** - cgroup digunakan untuk mengontrol sumber daya sistem.
- **SElinux** - SElinux digunakan untuk memastikan pemisahan antara host dan container (wadah) dan juga antar individu container.

Namespaces

- Namespace secara efektif merupakan **sarana untuk mengisolasi** sekumpulan **nama** atau **pengenal (identifier),** dan sangat dikenal oleh pengembang perangkat lunak, karena konsep ini banyak digunakan, baik secara eksplisit maupun implisit, dalam banyak bahasa pemrograman yang berbeda.
- **Linux namespace mengisolasi proses**, sehingga proses yang berbeda memiliki tampilan yang berbeda dari berbagai sumber daya sistem, sesuai dengan ruang nama (namespace) tempat mereka berada

Namespaces

- Kernel menyediakan **isolasi proses** dengan membuat **namespace terpisah** untuk container.
- Namespaces memungkinkan pembuatan **abstraksi** dari sumber daya **sistem global** tertentu dan membuatnya muncul sebagai instance terpisah untuk **proses dalam namespace**. Akibatnya, beberapa container dapat menggunakan resource yang sama secara bersamaan tanpa menimbulkan konflik.

Namespaces

- Sejak namespace pertama kali hadir di kernel Linux dalam versi 2.4.19 pada tahun 2002, sebanyak **sepuluh** namespace berbeda telah diusulkan, tetapi hingga saat ini hanya **enam** yang berhasil **masuk ke kernel** mainstream.
- Namespace yang telah diterapkan adalah namespace MNT, namespace UTS, namespace PID, namespace NET, namespace IPC, dan namespace USER. Yang hilang dari daftar asli adalah namespace security, namespace key security, namespace devices, dan time namespace.

Namespaces

Jadi, **bagaimana kita menempatkan proses ke dalam namespace ?**

Ini dicapai dengan tiga panggilan sistem (system calls) yang berbeda: **clone**, **unshare**, dan **setns**.

System call **clone** adalah sebuah fungsi tujuan umum untuk membuat proses baru ('child') berdasarkan konteks proses pemanggilan '('parent'), yang dapat dimodifikasi berdasarkan flag yang diteruskan, dan dapat digunakan untuk menetapkan namespace yang baru, tempat child tersebut berada.

Namespaces

System call **unshare** juga merupakan sebuah fungsi tujuan umum, yang dapat digunakan oleh proses pemanggil untuk **memisahkan konteks bersama** dari proses lain, dan untuk kepentingan kita menempatkan dirinya di namespace baru. Intinya, system call **clone membuat proses baru**, sedangkan unshare tidak.

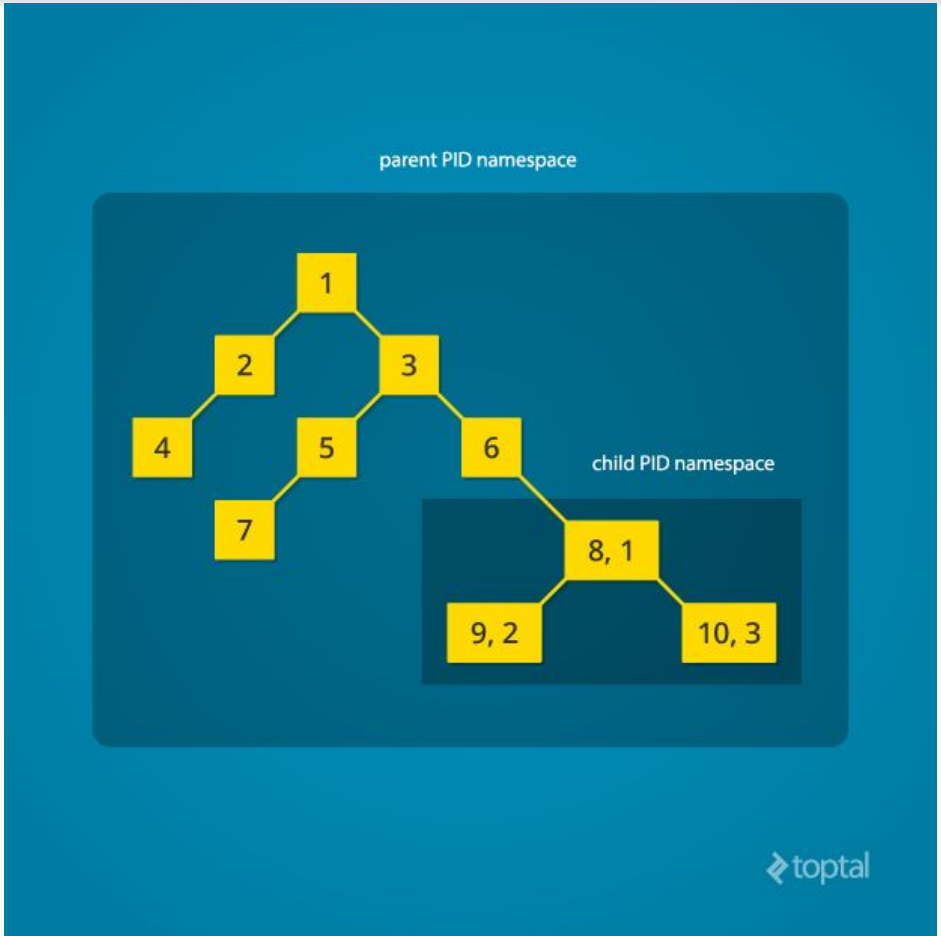
System call **setns** memungkinkan proses pemanggil untuk bergabung dengan namespace berbeda yang sudah ada, dengan menentukan deskriptor file yang terkait dengan namespace yang dimaksud.

Jenis jenis namespaces

Ada beberapa jenis namespace, yaitu:

- **PID Namespaces** memungkinkan proses dalam container berbeda untuk memiliki PID yang sama, sehingga setiap container dapat memiliki proses init (PID1) sendiri yang mengelola berbagai tugas inisialisasi sistem serta siklus hidup container. Selain itu, setiap container memiliki direktori /proc yang unik. Perhatikan bahwa dari dalam container Anda hanya dapat memantau proses yang berjalan di dalam container ini. Dengan kata lain, container hanya mengetahui proses aslinya dan tidak dapat "melihat" proses yang berjalan di berbagai bagian sistem. Di sisi lain, sistem operasi host mengetahui proses yang berjalan di dalam container, tetapi memberikan nomor PID yang berbeda

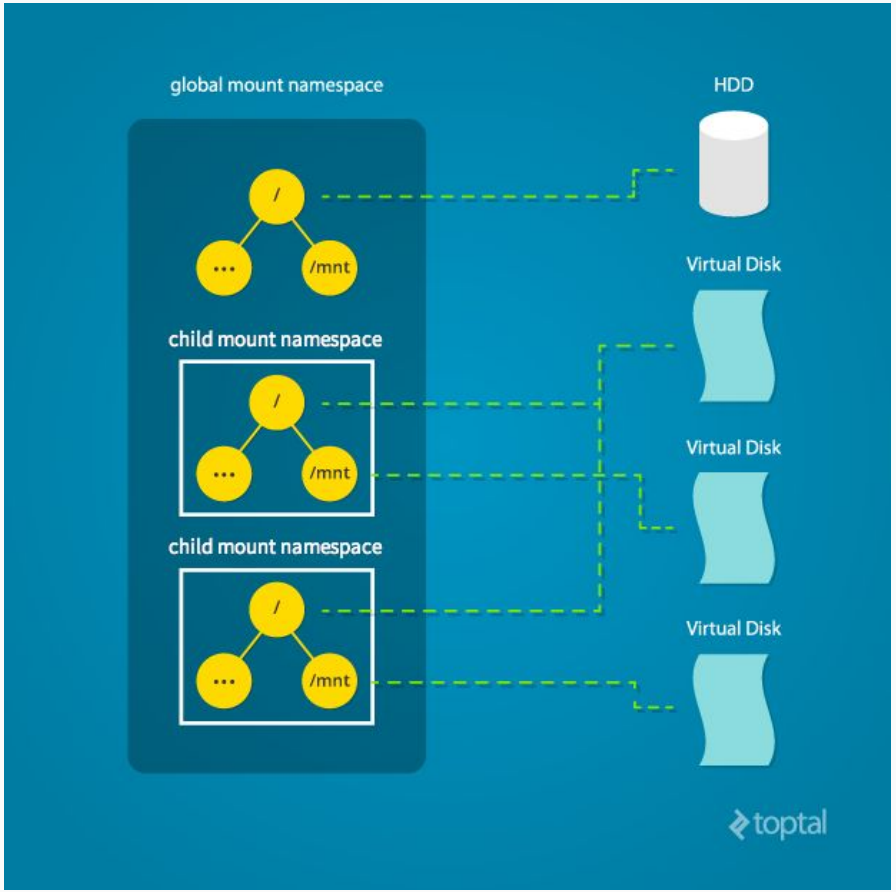
PID Namespaces



Jenis jenis namespaces

- **Mount namespaces**. Mount namespaces mengisolasi kumpulan **mount point** file system yang dilihat oleh sekelompok proses sehingga proses di mount namespace yang berbeda dapat memiliki tampilan berbeda dari hierarki sistem file. Dengan mount namespace, pemanggilan sistem mount() dan umount() berhenti beroperasi pada set global mount point (terlihat oleh semua proses) dan sebagai gantinya melakukan operasi yang hanya memengaruhi mount namespace yang terkait dengan proses container. Misalnya, setiap container dapat memiliki direktori /tmp atau /var sendiri atau bahkan memiliki ruang pengguna (*userspace*) yang sama sekali berbeda.

Mount Namespace



Jenis jenis namespaces

- **UTS Namespaces .** Memisahkan dua pengenal sistem - nama nodename dan nama domain, yang ditampilkan oleh pemanggilan sistem uname(). Hal ini memungkinkan setiap container memiliki **nama host** dan **nama domain NIS** sendiri, yang berguna untuk inisialisasi dan skrip konfigurasi berdasarkan nama-nama ini. Anda dapat menguji isolasi ini dengan menjalankan perintah nama host pada sistem host dan container - hasilnya akan berbeda.

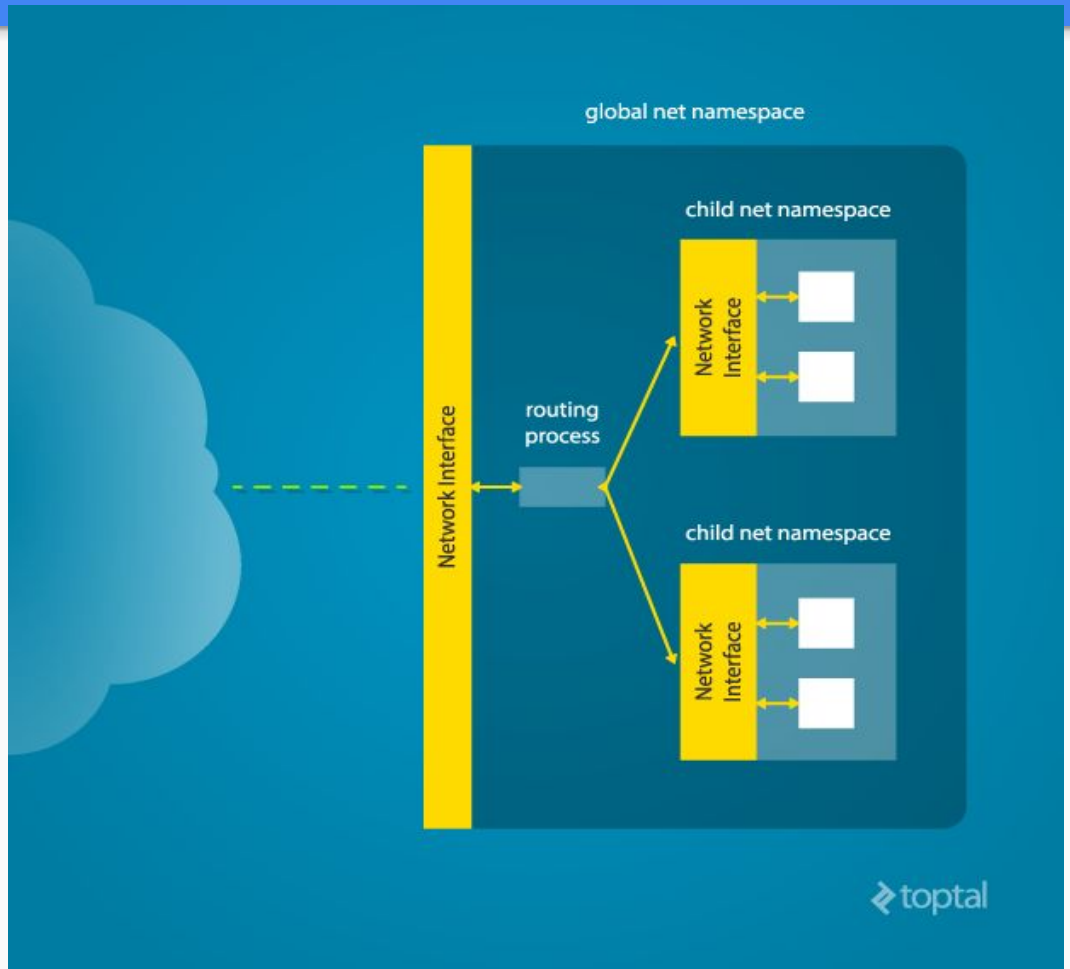
Jenis jenis namespaces

- **IPC Namespaces** mengisolasi **sumber daya komunikasi antarproses** (IPC) tertentu, seperti objek Sistem V IPC dan antrean pesan POSIX. Ini berarti bahwa dua container dapat membuat segmen memori bersama dan semaphore dengan nama yang sama, tetapi tidak dapat berinteraksi dengan segmen memori container lain atau memori bersama.

Jenis jenis namespaces

- **Network Namespaces** menyediakan isolasi network controllers, sumber daya sistem yang terkait dengan jaringan, firewall, dan tabel routing. Ini memungkinkan container untuk menggunakan tumpukan jaringan virtual, perangkat loopback, dan ruang proses yang terpisah. Anda dapat menambahkan perangkat virtual atau nyata ke container, menetapkan Alamat IP mereka sendiri dan bahkan aturan iptables lengkap. Anda dapat melihat pengaturan jaringan yang berbeda dengan menjalankan perintah ip addr pada host dan di dalam container.

Network Namespaces



Jenis jenis namespaces

- **User Namespaces** mirip dengan PID Namespaces, memungkinkan Anda untuk menentukan berbagai UID host yang didedikasikan untuk container. Akibatnya, sebuah proses dapat memiliki hak akses root penuh untuk operasi di dalam container, dan pada saat yang sama menjadi tidak memiliki hak istimewa untuk operasi di luar container

Control Groups (cgroups)

Kernel menggunakan cgroups untuk mengelompokkan proses untuk tujuan manajemen sumber daya sistem. Cgroup mengalokasikan waktu CPU, memori sistem, bandwidth jaringan, atau kombinasi dari ini di antara kelompok-kelompok tugas yang ditentukan pengguna.

Studi kasus:
Membuat container sederhana pada sistem linux

Alat & Referensi

https://github.com/nbrownuk/Namespaces