Teknologi Virtualisasi: Virtualisasi Sistem Operasi

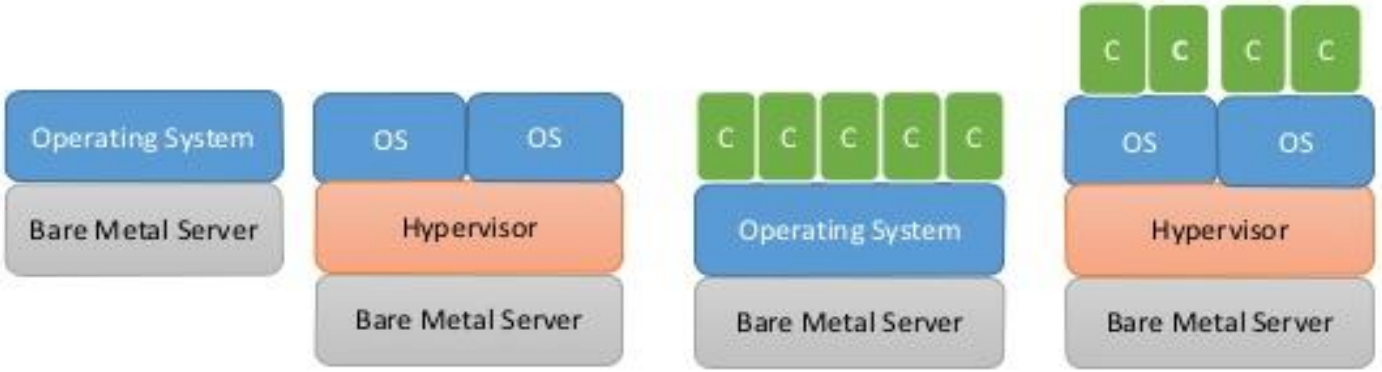
Henry Saptono,S.Si,M.Kom.
Sekolah Tinggi Teknologi Terpadu Nurul Fikri
Desember 2020

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**

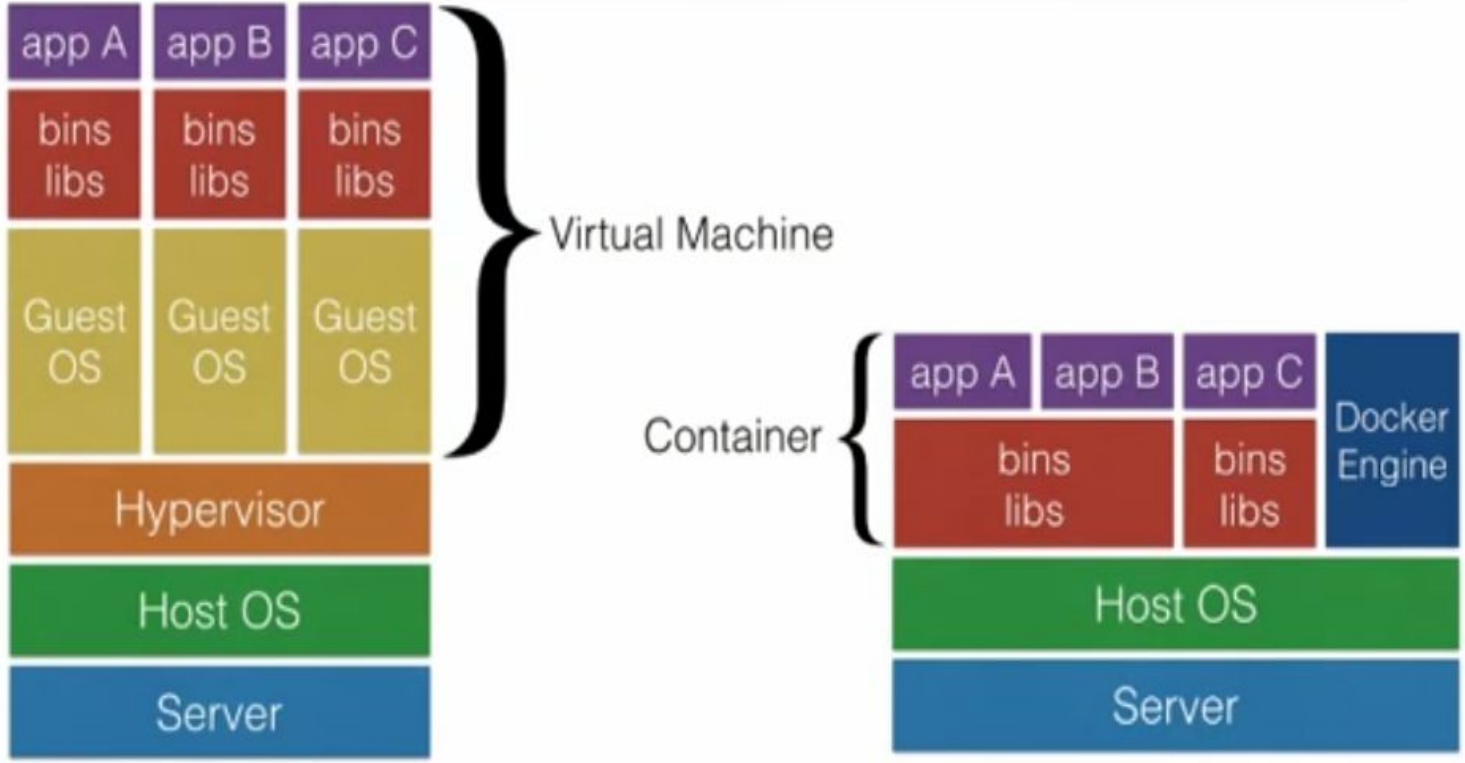
What are Containers?

- 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

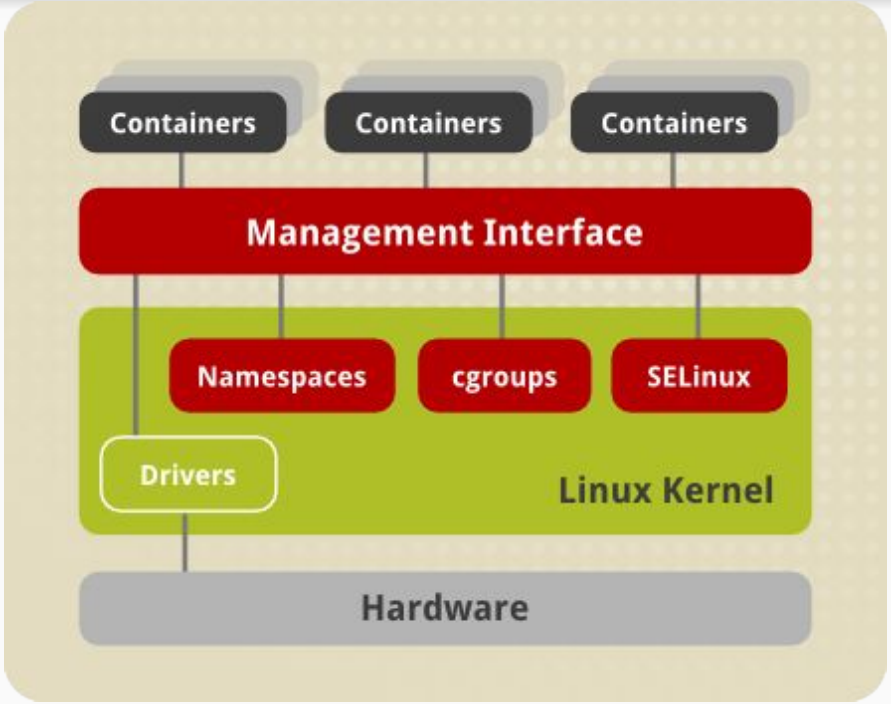
- Virtualisasi OS biasanya membebankan sedikit atau tidak ada biaya tambahan.
- Virtualisasi OS mampu melakukan migrasi langsung
- Dapat menggunakan load balancing dinamis dari kontainer antara node dan cluster.
- Mekanisme copy-on-write (CoW) tingkat file dimungkinkan pada virtualisasi OS yang membuat 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.

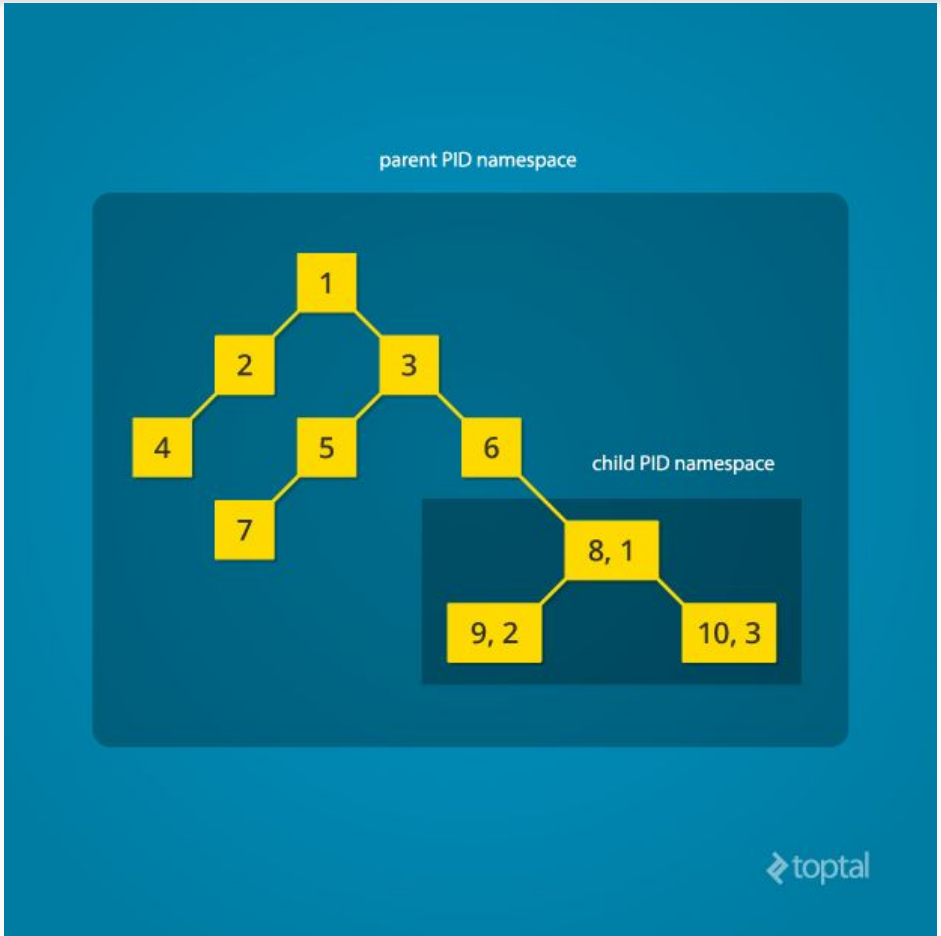
System call **setns** memungkinkan proses pemanggil untuk bergabung dengan namespace berbeda yang sudah ada, dengan menentukan deskriptor file yang terkait dengan namespace yang dimaksud.

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

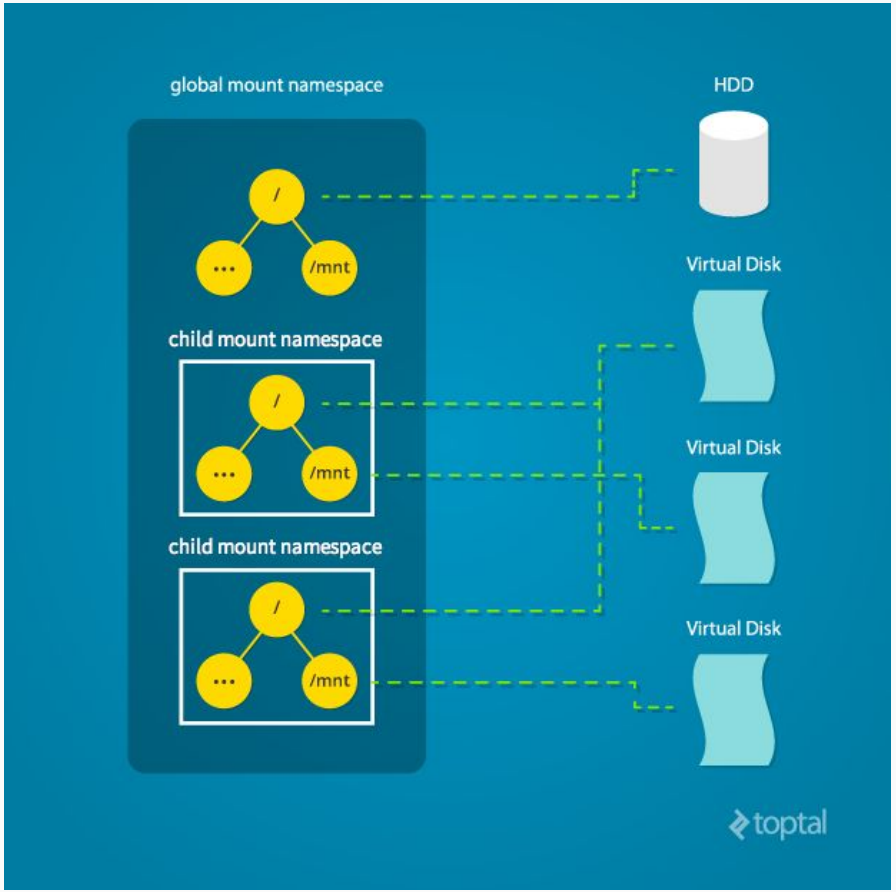
PID Namespaces



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.

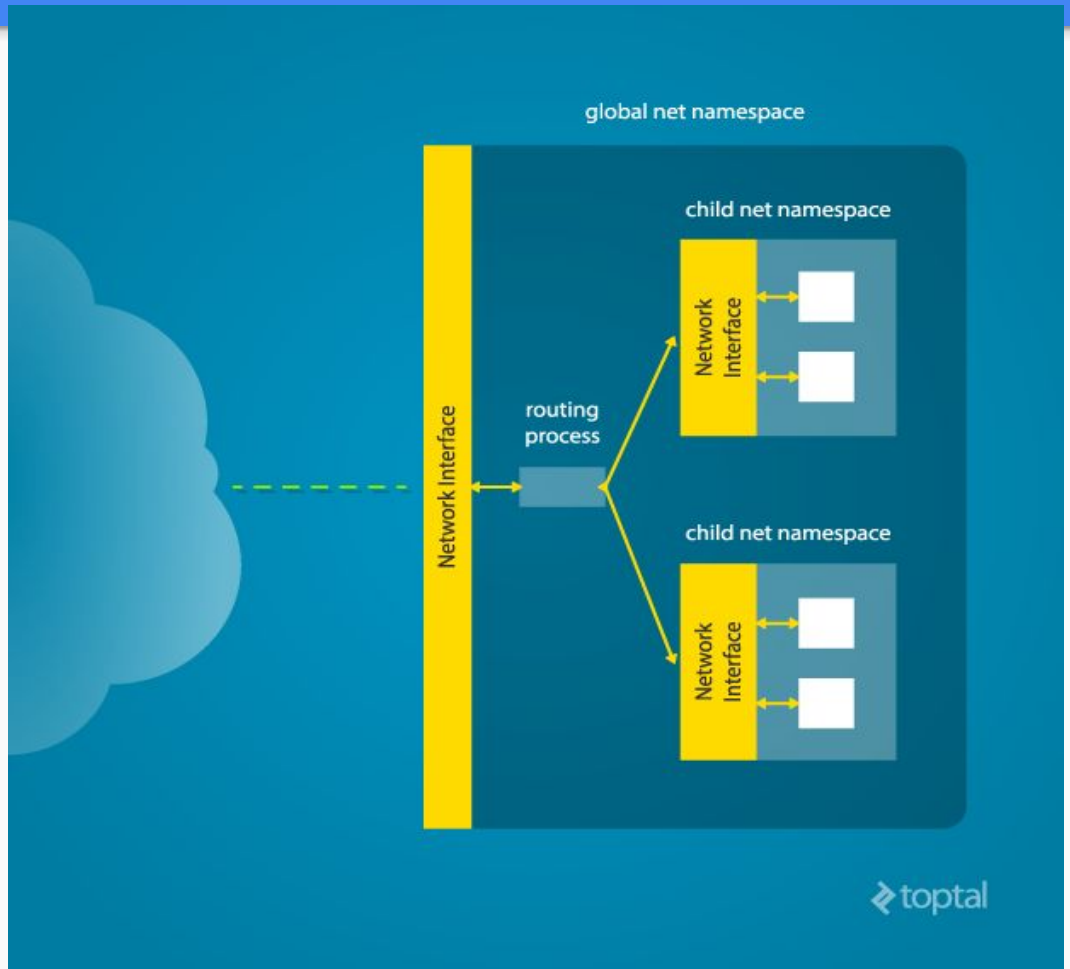
Jenis jenis namespaces

- **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

Studi kasus:
Implementasi virtualisasi sistem operasi menggunakan LXC (Linux Container)

Pendahuluan

- Linux Container adalah **teknologi virtualisasi** yang **ringan**.
- Linux Container lebih mirip dengan **chroot** yang ditingkatkan **daripada virtualisasi penuh** seperti Qemu atau VMware, **karena container tidak mengemulasi** perangkat keras dan karena kontainer **berbagi sistem operasi** yang sama dengan host. Linux Container mirip dengan Solaris Zones atau BSD jails.
- Linux-vserver dan OpenVZ adalah dua implementasi fungsionalitas mirip container yang sudah ada dan dikembangkan secara independen untuk Linux.

Pendahuluan

- Ada dua implementasi container ruang-pengguna pada sistem linux, masing-masing memanfaatkan fitur kernel yang sama.
  - **Libvirt**
  - **LXC**

Apa itu LXC ?

LXC adalah antarmuka ruang pengguna untuk fitur container kernel Linux. Melalui API yang powerfull dan alat sederhana, ini memungkinkan pengguna Linux dengan mudah membuat dan mengelola container sistem atau aplikasi.

Fitur LXC

LXC saat ini menggunakan fitur kernel berikut:

- Kernel namespaces (ipc, uts, mount, pid, network and user)
- Apparmor and SELinux profiles
- Seccomp policies
- Chroots (using pivot_root)
- Kernel capabilities
- CGroups (control groups)

Tujuan LXC

Container LXC sering dianggap sebagai sesuatu di antara **chroot** dan **mesin virtual** yang lengkap.

Tujuan dari LXC adalah menciptakan lingkungan yang sedekat mungkin dengan instalasi Linux standar tetapi tanpa memerlukan kernel terpisah.

Komponen LXC

LXC saat ini dibuat dari beberapa komponen terpisah:

- Pustaka liblxc
- Beberapa binding bahasa pemrograman untuk API: Python, Lua, Go, Ruby, Haskell
- Satu set alat standar untuk mengontrol container
- Template container distribusi

Perijinan (licensing)

- LXC adalah perangkat lunak Free, sebagian besar kodenya dirilis di bawah persyaratan lisensi GNU LGPLv2.1 +, beberapa kompatibilitas Android dirilis di bawah lisensi standar 2-Clause BSD dan beberapa binari dan template dirilis di bawah lisensi GNU GPLv2.
- Lisensi default untuk proyek ini adalah GNU LGPLv2.1 +.

Instalasi

Paket lxc dapat diinstal menggunakan:

```
$ sudo apt install lxc
```

Penggunaan dasar

Untuk membuat container , Anda cukup melakukan:

```
$ sudo lxc-create --template download --name con1
```

Atau

```
sudo lxc-create -t download -n con1
```

Ini akan secara interaktif meminta jenis sistem file root container untuk diunduh - khususnya distribusi, rilis, dan arsitektur.

Penggunaan dasar

Untuk membuat container secara non-interaktif, Anda dapat menentukan nilai-nilai ini di baris perintah:

```
$ sudo lxc-create -t download -n con1 -- --dist ubuntu
--release DISTRO-SHORT-CODENAME --arch amd64
```

Atau

```
$ sudo lxc-create -t download -n con1 -- -d ubuntu -r
DISTRO-SHORT-CODENAME -a amd64
```

Penggunaan dasar

Anda dapat menggunakan **lxc-ls** untuk menampilkan daftar container, **lxc-info** untuk mendapatkan informasi container yang mendetail, **lxc-start** untuk memulai dan **lxc-stop** untuk menghentikan container. **lxc-attach** dan **lxc-console** memungkinkan Anda untuk memasuki sebuah container, jika ssh bukan merupakan pilihan. **lxc-destroy** menghapus container, termasuk rootf-nya.

Penggunaan dasar

Contoh mungkin terlihat seperti ini:

```
$ sudo lxc-ls --fancy
$ sudo lxc-start --name con1 --daemon
$ sudo lxc-info --name con1
$ sudo lxc-stop --name con1
$ sudo lxc-destroy --name con1
```

Konfigurasi global

File konfigurasi berikut dikonsultasikan oleh LXC. Untuk penggunaan dengan hak istimewa, mereka dapat ditemukan di /etc/lxc, sedangkan untuk penggunaan yang tidak memiliki hak istimewa, mereka ada di ~ /.config/lxc. (Kita hanya membahas penggunaan LXC dengan hak istimewa)

- **lxc.conf** dapat secara opsional menetapkan nilai alternatif untuk beberapa pengaturan lxc, termasuk **lxcpath**, konfigurasi default, **cgroup** yang akan digunakan, pola pembuatan cgroup, dan pengaturan backend penyimpanan untuk **lvm** dan **zfs**.

Konfigurasi global

- **default.conf** menentukan konfigurasi yang harus dimuat di setiap container yang baru dibuat. Ini biasanya berisi setidaknya satu bagian jaringan, dan, untuk pengguna yang tidak memiliki hak istimewa, bagian pemetaan id
- **lxc-usernet.conf** menentukan bagaimana pengguna yang tidak memiliki hak istimewa dapat menghubungkan kontainer mereka ke jaringan yang dimiliki host.

lxc.conf dan **default.conf** keduanya berada di bawah **/etc/lxc** dan **$HOME/.config/lxc**, sedangkan **lxc-usernet.conf** hanya untuk seluruh host.

Secara default, kontainer ditempatkan di bawah /var/lib/lxc untuk pengguna root.

Networking

- Secara default, LXC membuat namespace jaringan pribadi untuk setiap container, yang meliputi network stack layer 2. Container biasanya terhubung ke dunia luar dengan memiliki NIC fisik atau titik akhir terowongan **veth** yang dilewatkan ke dalam container.
- LXC membuat bridge NAT, **lxcbr0**, saat startup host. Container yang dibuat menggunakan konfigurasi default akan memiliki satu NIC dengan ujung jarak jauh dicolokkan ke bridge **lxcbr0**.
- NIC hanya bisa ada di satu namespace dalam satu waktu, jadi NIC fisik yang diteruskan ke container tidak dapat digunakan di host.

Networking

- Dimungkinkan untuk membuat container tanpa namespace jaringan pribadi. Dalam kasus ini, container akan memiliki akses ke jaringan host seperti aplikasi lainnya. Perhatikan bahwa ini sangat berbahaya.
- Untuk memberi container di **lxcbr0** alamat ip yang tetap berdasarkan nama domain, Anda dapat menulis entri ke **/etc/lxc/dnsmasq.conf** seperti:

```
dhcp-host=lxcweb,10.0.3.100

dhcp-host=lxcmysql,10.0.3.101
```

Networking

- Jika diinginkan agar container dapat diakses publik, ada beberapa cara untuk melakukannya. Salah satunya adalah menggunakan iptables untuk meneruskan port host ke container, misalnya

```
$ sudo iptables -t nat -A PREROUTING -p tcp -i eth0 --dport 8080 -j DNAT --to-destination 10.0.3.100:80
```

Networking

- Kemudian, tentukan host bridge di file konfigurasi container sebagai ganti **lxcbr0**, misalnya

```
lxc.network.type = veth
```

```
lxc.network.link = br0
```

Backing Stores

- LXC mendukung beberapa penyimpan (Backing Stores) untuk sistem file root kontainer. Standarnya adalah penyimpanan direktori sederhana, karena tidak memerlukan kustomisasi host sebelumnya, selama sistem file yang mendasarinya cukup besar. Ini juga tidak memerlukan hak akses root untuk membuat penyimpanan, sehingga itu mulus untuk penggunaan tanpa hak istimewa.

Backing Stores

- **Rootfs** untuk container yang didukung direktori dengan hak istimewa terletak (secara default) di bawah /var/lib/lxc/con1/rootfs, sedangkan rootfs untuk contianer yang tidak memiliki hak istimewa ada di ~/.local/share/lxc/con1/rootfs. Jika **lxcpath** kustom ditentukan di **lxc.system.com,** maka container rootfs akan berada di bawah $lxcpath/con1/rootfs.

Templates

Membuat container umumnya melibatkan pembuatan sistem file root untuk container tersebut. **lxc-create** mendelegasikan pekerjaan ini ke template, yang umumnya per-distribusi. Template-template **lxc** yang disertakan bersama **lxc** dapat ditemukan di bawah **/usr/share/lxc/templates**, dan termasuk template untuk membuat container Ubuntu, Debian, Fedora, Oracle, centos, dan gentoo.

Templates

- Membuat image distribusi dalam banyak kasus memerlukan kemampuan untuk membuat node perangkat, seringkali membutuhkan alat yang tidak tersedia di distribusi lain, dan biasanya cukup memakan waktu.
- Oleh karena itu, lxc dilengkapi dengan **template unduhan khusus**, yang **mengunduh image** container yang telah dibuat sebelumnya dari server lxc pusat.

Templates

- Saat menjalankan **lxc-create**, semua opsi yang muncul setelah -- diteruskan ke **template**. Dalam perintah berikut, --name, --template, dan --bdev diteruskan ke lxc-create, sementara --release diteruskan ke template:

```
lxc-create --template ubuntu --name con1 --bdev loop -- --release DISTRO-SHORT-CODENAME
```

Templates

Anda bisa mendapatkan bantuan untuk opsi yang didukung oleh container tertentu dengan meneruskan --help dan nama template ke **lxc-create**. Misalnya, untuk bantuan dengan templat unduhan,

```
lxc-create --template download --help
```

Autostart

LXC mendukung penandaan container untuk dimulai saat boot sistem. Sebelum Ubuntu 14.04, ini dilakukan dengan menggunakan tautan simbolik di bawah direktori /etc/lxc/auto. Dimulai dengan Ubuntu 14.04, ini dilakukan melalui file konfigurasi container. Sebuah entri

```
lxc.start.auto = 1

lxc.start.delay = 5
```

berarti contianer harus dimulai saat boot, dan sistem harus menunggu 5 detik sebelum memulai container berikutnya.

Consoles

Container memiliki jumlah konsol yang dapat dikonfigurasi. Satu selalu ada di **/dev/console** container. Ini ditampilkan di terminal tempat Anda menjalankan lxc-start, kecuali opsi **-d** ditentukan. Output pada /dev/console dapat diarahkan ke file menggunakan opsi **-c console-file** ke **lxc-start**. Jumlah konsol tambahan ditentukan oleh variabel **lxc.tty**, dan biasanya diatur **4.** Konsol tersebut ditampilkan di /dev/ttyN (untuk 1 <= N <= 4). Untuk masuk ke konsol 3 dari host, gunakan:

```
sudo lxc-console -n container -t 3
```

Consoles

atau jika opsi **-t N** tidak ditentukan, konsol yang tidak digunakan akan dipilih secara otomatis. Untuk keluar dari konsol, gunakan escape sequence **Ctrl-a q**. Perhatikan bahwa escape sequence tidak bekerja di konsol yang dihasilkan dari **lxc-start** tanpa opsi **-d**.

Logging

Jika terjadi kesalahan saat memulai container, langkah pertama yang harus dilakukan adalah mendapatkan logging penuh dari LXC:

```
sudo lxc-start -n con1 -l trace -o debug.out
```

Ini akan menyebabkan lxc mencatat pada tingkat paling panjang, melacak, dan mengeluarkan informasi log ke file yang disebut '**debug.out**'. Jika file debug.out sudah ada, informasi log baru akan ditambahkan.

Memantau status container

Dua perintah tersedia untuk memantau perubahan status container.

```
sudo lxc-monitor -n con[0-5]*
```

akan mencetak semua perubahan status ke penampung apa pun yang cocok dengan ekspresi reguler yang tercantum, sedangkan

```
sudo lxc-wait -n con1 -s 'STOPPED|FROZEN'
```

akan menunggu sampai container con1 memasuki status STOPPED atau status FROZEN dan kemudian keluar.

Attach

Mulai Ubuntu 14.04, dimungkinkan untuk attach ke namespace container. Kasus paling sederhana adalah melakukannya

```
sudo lxc-attach -n con1
```

yang akan memulai shell yang di-attach ke namespace **con1**, atau, secara efektif di dalam container.