chroot, lxc そしてdocker

樋口大輔

クリエーションライン株式会社

CL社内勉強会

2014/07/14

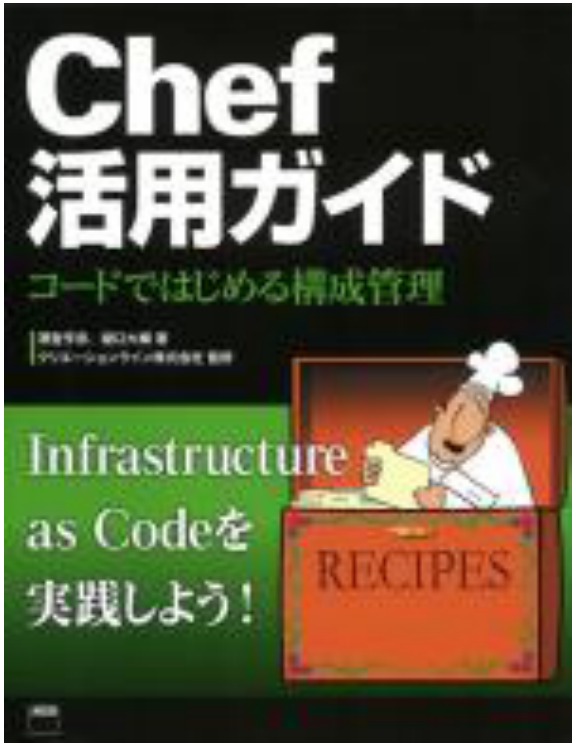
アジェンダ

- 自己紹介
- chrootとは: できること・できないこと
- lxcとは: コンテナ技術・できること・できないこと
- dockerとは: なぜ注目・できること・できないこと
- まとめ

自己紹介

- 樋口大輔（twitter: @dai_lxr）
- クリエーションライン株式会社
  - CL-LAB: http://www.creationline.com/lab
- Chef Approved Contributor (2012/06/27)
  - busser-serverspec, knife-sakura, ...
- Debian Project Official Developer (2012/01/01)
  - mikutter, uim, ...

Chef活用ガイド



クリエーションライン株式会社 監修
澤登亨彦、樋口大輔 著

chroot

- **ch**ange **root**directory
  - ルートディレクトリを変更するコマンド(システムコール)
- / ←ルートディレクトリ

```
% ls /
bin    dev   initrd.img  lost+found  proc  sbin  usr
boot   etc   lib         media       root  sys   var
cdrom  home  lib64       mnt         run   tmp   vmlinuz
```

ルートディレクトリを変更するってどういうこと？

- **/tmp/pseudo_root**にルートディレクトリを変更すると、それを実行したプロセスとその子プロセスは以降/tmp/pseudo_rootを**/**として取り扱う。

```
% ls /tmp/pseudo_root
bin
% ls /tmp/pseudo_root/bin
busybox  zsh
%

% sudo chroot /tmp/pseudo_root
# /bin/busybox ls /
bin
# /bin/busybox ls /bin
busybox  zsh
#
```

ルートディレクトリを変更するといいことあるの？(1/4)

- クリーンなビルド環境
  - chroot以下にビルド環境を構築してビルドすると、開発システムに存在するライブラリ等に影響されずに済む（ex. Debian pbuilder）

```
unstable%
unstable% sudo pbuilder --build uim_1.8.6-7.dsc --basetgz base-stable.tgz

amd64%
amd64% sudo pbuilder --build uim_1.8.6-7.dsc --basetgz base-i386.tgz
```

ルートディレクトリを変更するといいことあるの？(2/4)

- API/ABI互換のないソフトウェアの利用
  - chroot以下に現在のシステムと異なるAPI/ABIのライブラリを閉じ込めることで共存が可能になる（ex. 古いソフトウェア）

```
present% /tmp/potato-root/tmp/sl -l
/tmp/potato-root/tmp/sl: error while loading shared libraries: libncurses.so.4:
cannot open shared object file: No such file or directory
present% sudo chroot /tmp/potato-root
old# /tmp/sl -l
    ++      +------
    ||      |+-+ |
  /---------|| | |
 + ========  +-+ |
_|--/~\------/~\-+
//// O========O_/
```

ルートディレクトリを変更するといいことあるの？(3/4)

- システムのインストール、復旧
  - CD等で起動し、空HDDにchrootしてシステムのインストールを行う（ex. Debian debootstrap）
  - これの応用で、CD等で起動し、起動不可能になったHDDにchrootしての復旧も可能。

```
rescue# mkdir /tmp/mnt
rescue# mount /dev/sda1 /tmp/mnt
rescue# mount --bind /dev /tmp/mnt/dev
rescue# chroot /tmp/mnt
broken# /usr/sbin/grub-install /dev/sda
```

ルートディレクトリを変更するといいことあるの？(4/4)

- プロセスの隔離、権限の分離
  - chroot以下にプロセスを隔離して本来のルートディレクトリに上がれなくすることで、他者のファイルへのアクセスを防ぐ（ex. ISC BIND, 各種ftpd, OpenSSH等）

```
# ps ax |grep named
4711 ?        Ssl    0:00 /usr/sbin/named -u named -t /var/named/chroot
# ls /var/named/chroot
etc  dev  proc  var
```

chrootの残念なところ(1/3)

- chrootする先に必要なファイルがすべて揃っていることが動作条件。
  - ライブラリ、デバイスファイル、procなどをいちいちコピーしたりマウントしないとうまく動かない。

```
# ls /var/named/chroot
etc  dev  proc  var
```

chrootの残念なところ(2/3)

- chrootはroot権限がないと動作しない。
  - さらにchroot後はroot権限を持ったままなので、即脱獄したり、不正なこともできる。
    - How to break out of a chroot() jail: http://www.bpfh.net/simes/computing/chroot-break.html

```
inside# ls /
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
inside# /tmp/jailbreak
outside# ls /
bin    dev   initrd.img  lost+found  proc  sbin  usr
boot   etc   lib         media       root  sys   var
cdrom  home  lib64       mnt         run   tmp   vmlinuz
outside#
```

chrootの残念なところ(3/3)

- ファイルシステムベースで隔離しているだけなので、外のプロセスに干渉できる。

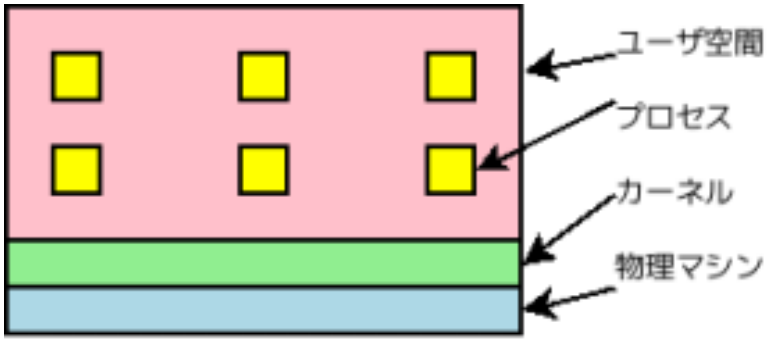
```
% ps auxwww | grep '[ t]op'
dai        9919  0.1  0.0  27580  1728 pts/8    S+   16:51   0:00 top
% sudo chroot /tmp/pseudo_root
# /bin/busybox kill 9919
# exit
% ps auxwww | grep '[ t]op'
%
```

- 当然、CPU・メモリ・I/O・ネットワークなどのリソースも共用なので制限されない。
  - このあたりを強化したのがFreeBSD jailやコンテナ技術。

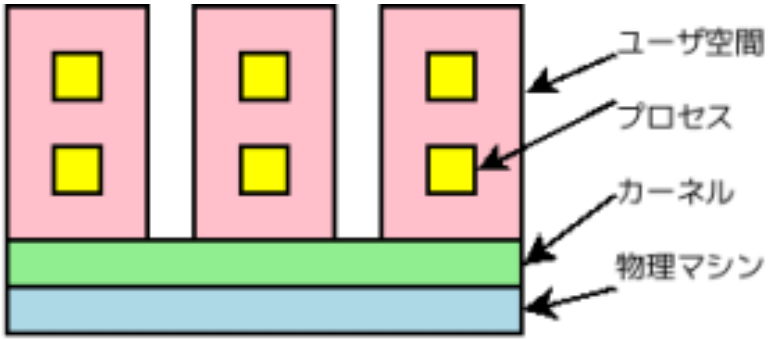
lxc

- linux container
  - Linux Kernelの機能で実現したコンテナ技術の1つ。
- そもそも「コンテナ技術」って何？
  - chrootではできなかった、プロセスやリソースの隔離を行う技術。
  - 隔離したプロセスやリソースのかたまりを「コンテナ」と呼ぶ。

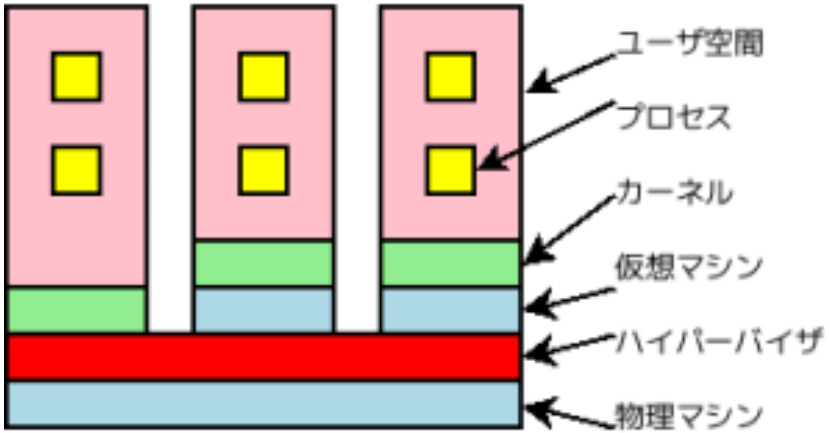
コンテナ型とハイパーバイザ型



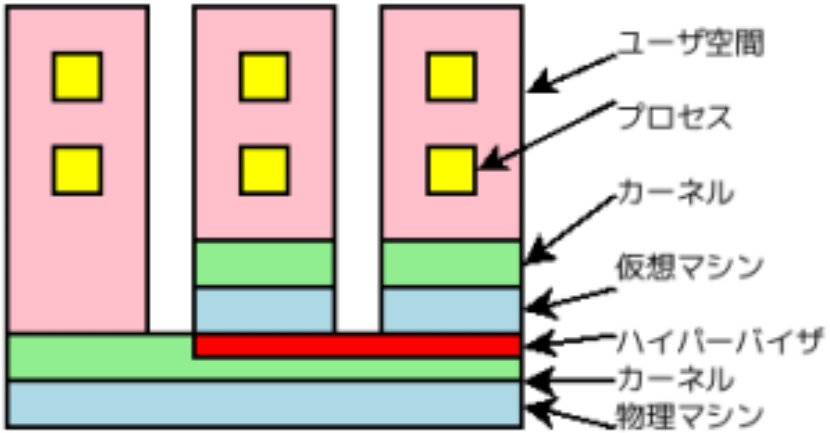
一般的な環境



コンテナ型



ハイパーバイザ型(Xen)



ハイパーバイザ型(KVM)

Linuxにおけるコンテナ技術

- Virtuozzo (ばーちゅおっぞ)
  - PleskやParallels Desktopで有名なParallels, Inc. (旧SWsoft)による商用製品。
- OpenVZ (おーぷんう゛ぃーずぃー)
  - Virtuozzoのオープンソース版。
- LXC (えるえっくすしー)
  - cgroup (**c**ontrol **group**)と名前空間 (**Namespace**)でプロセスやリソースを、chrootでファイルシステムを隔離して作ったコンテナを扱うための仕組み。

lxcのリソース隔離(1/4)：ファイルシステム

- chrootと同じ。

```
outside% ls /var/cache/lxc/precise/rootfs-amd64
bin   dev  home  lib64  mnt  proc  run   selinux  sys  usr
boot  etc  lib   media  opt  root  sbin  srv      tmp  var
outside%

inside$ ls /
bin   dev  home  lib64  mnt  proc  run   selinux  sys  usr
boot  etc  lib   media  opt  root  sbin  srv      tmp  var
inside$
```

lxcのリソース隔離(2/4)：プロセス

- lxcのコンテナ内からは外のプロセスに干渉できない。
  - 名前空間で実現している。
  - そもそもPIDがまったく異なる。
  - もちろん、lxcのコンテナ同士も干渉できない。

```
outside% ps auxwww | grep '[ /]sbin/init'
root         1  0.0  0.0  45972  4532 ?        Ss   15:20   0:00 /sbin/init
root      2650  0.0  0.0  24968  2036 ?        Ss   18:00   0:00 /sbin/init
outside%

inside$ ps auxwww | grep '[ /]sbin/init'
root         1  0.0  0.0  24968  2036 ?        Ss   18:00   0:00 /sbin/init
inside$
```

lxcのリソース隔離(3/4)：ネットワーク

- 個々のコンテナがネットワークの口を持つ。
  - 名前空間で実現している。
  - lxcのコンテナごとに仮想NIC(vethX)を割り当てて、物理NIC(eth0)とブリッジ(br0)で接続する。

```
outside% brctl show
bridge name        bridge id               STP enabled        interfaces
br0                8000.8c89a56f525d       yes                eth0
                                                              veth1TEH9P
outside% ifconfig
veth1TEH9P Link encap:Ethernet  HWaddr fe:84:c5:17:a0:9d

inside$ ifconfig
eth0      Link encap:Ethernet  HWaddr 4a:49:43:49:79:bf
          inet addr:192.168.24.72  Bcast:192.168.24.255  Mask:255.255.255.0
```

lxcのリソース隔離(4/4)：CPU・メモリ・I/O・帯域制御など

- cgroupで制限ができる

```
inside$ cat /proc/cgroups
#subsys_name        hierarchy       num_cgroups       enabled
cpuset          3           3           1
cpu             4           25          1
cpuacct         4           25          1
memory          0           1           0
devices         5           3           1
freezer         6           3           1
net_cls         7           3           1
blkio           8           3           1
perf_event      9           3           1
```

lxcのいいところ・残念なところ

- chrootに比べて隔離できる範囲が広い。
- ハイパーバイザ型の仮想化基盤よりお手軽に利用できてオーバーヘッドも少ない。
  - 結局は同一kernel上で動いているので、異なるkernel (OS)は動作できない。
    - けどこれは一概に残念なところとは言えないよね。
- lxcの設定は人それぞれで、ポータビリティに欠ける。

docker

- 簡単に言えば、既存の技術を組み合わせて使いやすくしたコンテナ技術。
  - libcontainerによるkernel API呼び出し（0.9 以前はlxcを直接利用）
  - AUFS (Another Union File System)によるコンテナの世代・差分管理
- 新しい技術でもないのになぜこんなに騒がれるのか？

なぜdockerなのか？(1/4)

- 小さいことはよいことだ
  - lxcが「OS」のコンテナであれば、dockerは「アプリケーション」のコンテナ。

```
└─zsh───sudo───lxc-start───init─┬─cron
                               ├─dhclient3
                               ├─5*[getty]
                               ├─ondemand───sleep
                               ├─rsyslogd─┬─{rs:main Q:Reg}
                               │          └─2*[{rsyslogd}]
                               ├─sshd
                               ├─udevd
                               ├─upstart-socket-
                               └─upstart-udev-br

├─docker─┬─nginx───4*[nginx]
│        └─17*[{docker}]
```

なぜdockerなのか？(2/4)

- 簡単なビルド

```
debian% cat Dockerfile
FROM ubuntu
MAINTAINER d-higuchi <d-higuchi@creationline.com>
RUN apt-get install -y nginx
debian%

debian% sudo docker build -t nginx-ubuntu .
       :
Successfully built 53a44d8c6ca9
debian%
```

なぜdockerなのか？(3/4)

- 高いポータビリティ
  - dockerで作ったコンテナはどのdockerでも動く。

```
debian% sudo docker export 800c967d6cd9 > nginx-ubuntu.tar

centos$ cat nginx-ubuntu.tar | sudo docker import - nginx-ubuntu
```

- Debianのdockerで作ったUbuntuのNginxコンテナがCentOSのdockerでも動く。

なぜdockerなのか？(4/4)

- gitライクな世代・差分管理
  - コンテナをあたかもgitで管理するかのように履歴を追ったり分岐したりできる。

```
% sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                 CREATED ...
800c967d6cd9        nginx-test:latest   /usr/sbin/nginx -g '    38 minutes ago ...
%
% sudo docker commit 800c967d6cd9 tag-test
e0ad2d2f909272f2d9966aa163bedca1644ac0b440a4f4a3d59303fbdf51a371
%
% sudo docker images | head
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
tag-test            latest              e0ad2d2f9092        14 seconds ago      294.9 MB
nginx-test          latest              53a44d8c6ca9        41 minutes ago      294.9 MB
%
```

dockerでやってくれないこと

- 動的なデータの取り扱い
  - ログやデータファイルはコンテナの中に置けない。
    - commitしないと再起動で消えちゃう。
    - コンテナの外に置くか？ その方法は？
- セキュリティ
  - 考えなしにやると穴だらけになるんじゃ…？
    - 不正侵入はもちろんデータ漏洩や機密保持、監視などなど。

まとめ

- chroot、lxc、dockerと環境の隔離やパッケージ化は進化してきた。
- ハイパーバイザ型の仮想化技術とはまた異なるコンテナ技術。
- dockerの周辺技術も徐々に充実しつつある。
  - イメージ管理：DockerHub, Shipyard
  - メタ管理：libswarm, Kubernetes
  - PaaS: Dokku, Flynn, Deis

Any Questions?