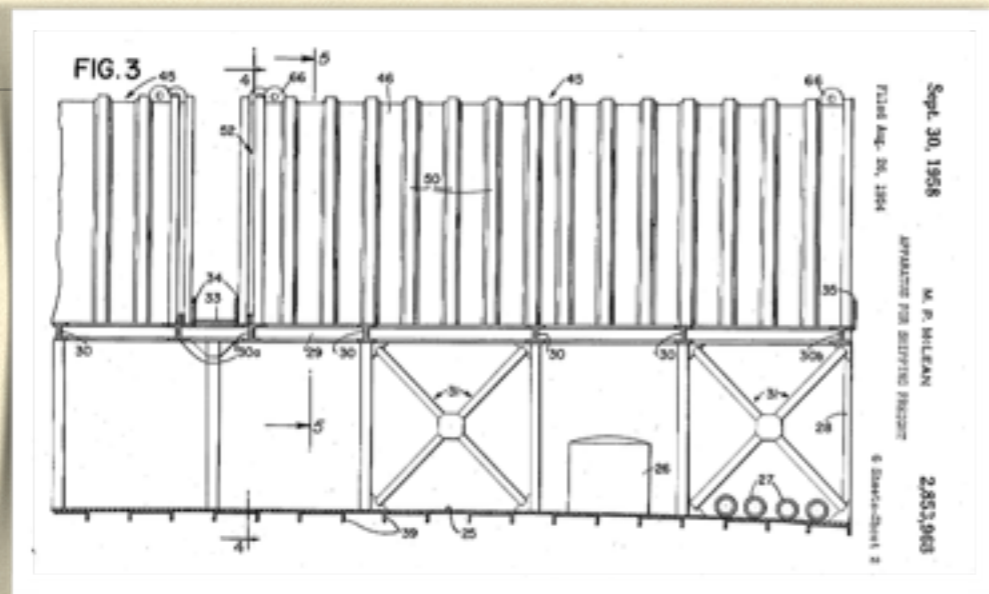




# Containerisation

Gareth Roy  
GridPP 32, Pitlochry

# Intermodal Containers



Apparatus for shipping freight (1958):  
US 2853968 A - Malcolm P McLean



Mærsk Mc-Kinney Møller (18270 TEU)

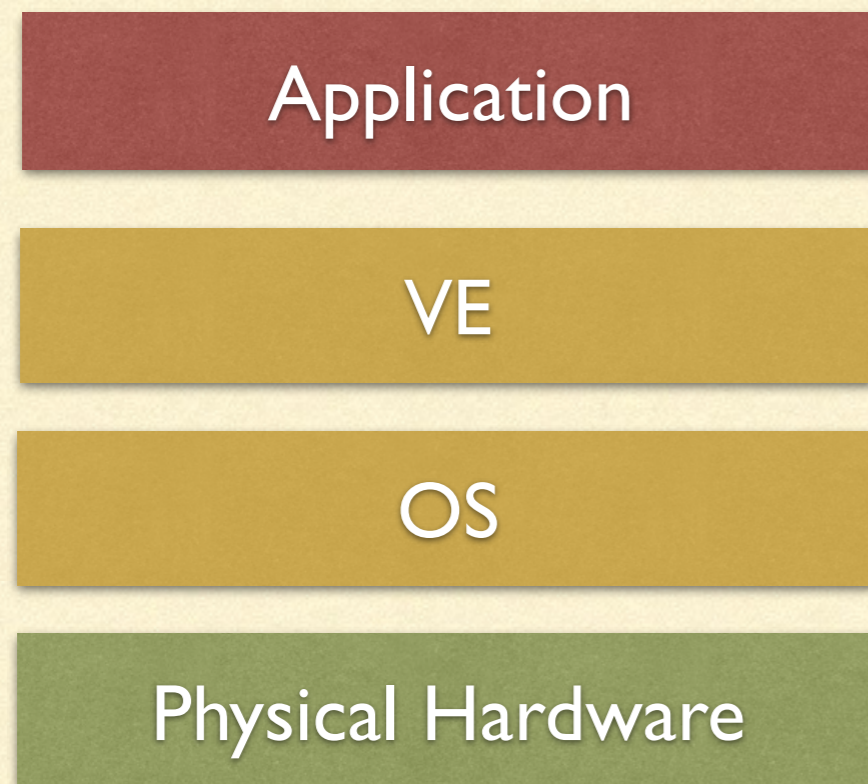
- Developed by Malcolm P. McLean & Keith W. Tantlinger.
- Reaction to slow loading times produced by using “break bulk cargo.”
- “In 1956, loose cargo cost \$5.86 per ton to load. Using an ISO shipping container, the cost was reduced to only .16 cents per ton.”

	IMPERIAL	METRIC
Length	19' 10.5"	6.058 m
Width	8' 0"	2.438 m
Height	8' 6"	2.591 m
Empty Weight	4,850 lb	2,200 kg
Max Weight	66,139 lb	30,400 kg

---

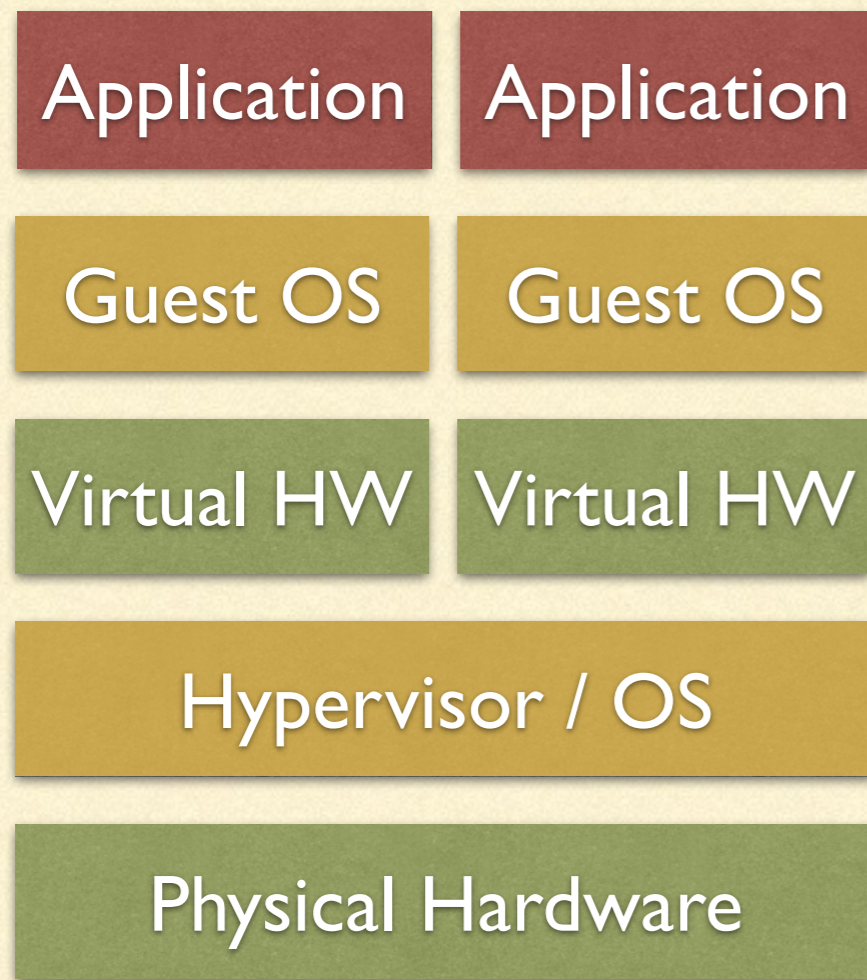
# Linux Containers

---

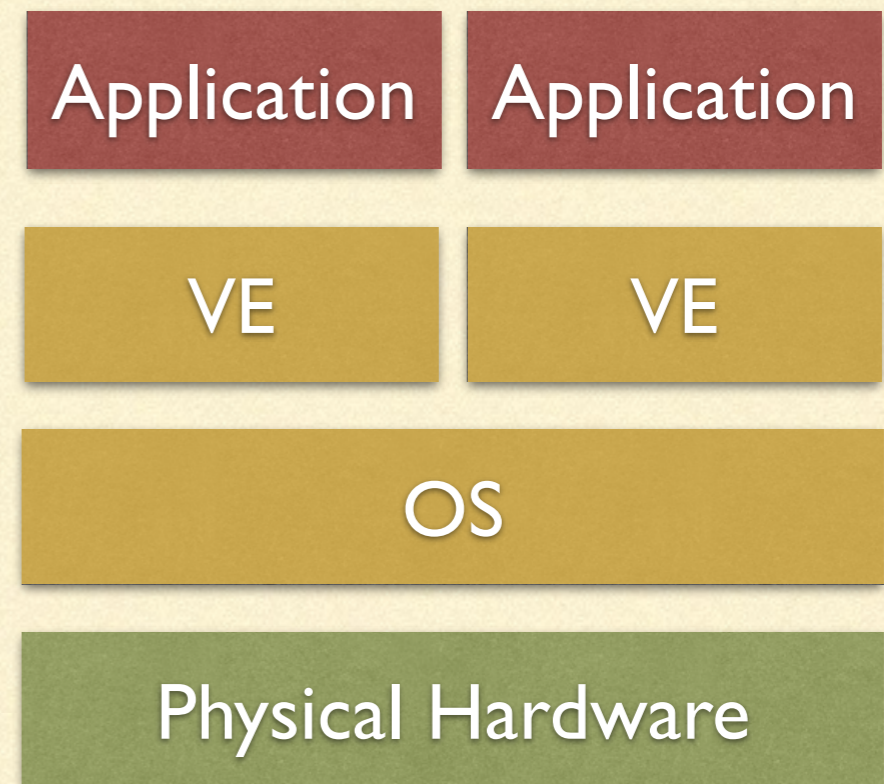


- Form of OS Level Virtualisation.
  - Kernel hosts multiple separated user-land instances (Virtual Environment/Engine).
  - Low overheads, elastic, multi-tenant.
  - Storage can be Copy-on-Write or use UnionFS
- Examples:
  - chroot (1982)
  - Solaris Containers (2005)
  - FreeBSD Jails (1988)
  - AIX WPARS (2007)
  - Virtuozzo (2001)
  - LXC (2008)
  - OpenVZ (2005)

# VM's vs Containers



Virtual Machine



Linux Container

---

# VM's vs Containers (Arguments)

---

- Pros:

- OS Independent
- Secure / Isolated
- Flexible
- Live Migration
- Mature Ecosystem

- Cons:

- Full System Image
- Slow Startup/Shutdown/Build
- Memory Consumption Overhead
- Opaque to System

Virtual Machine

- Pros:

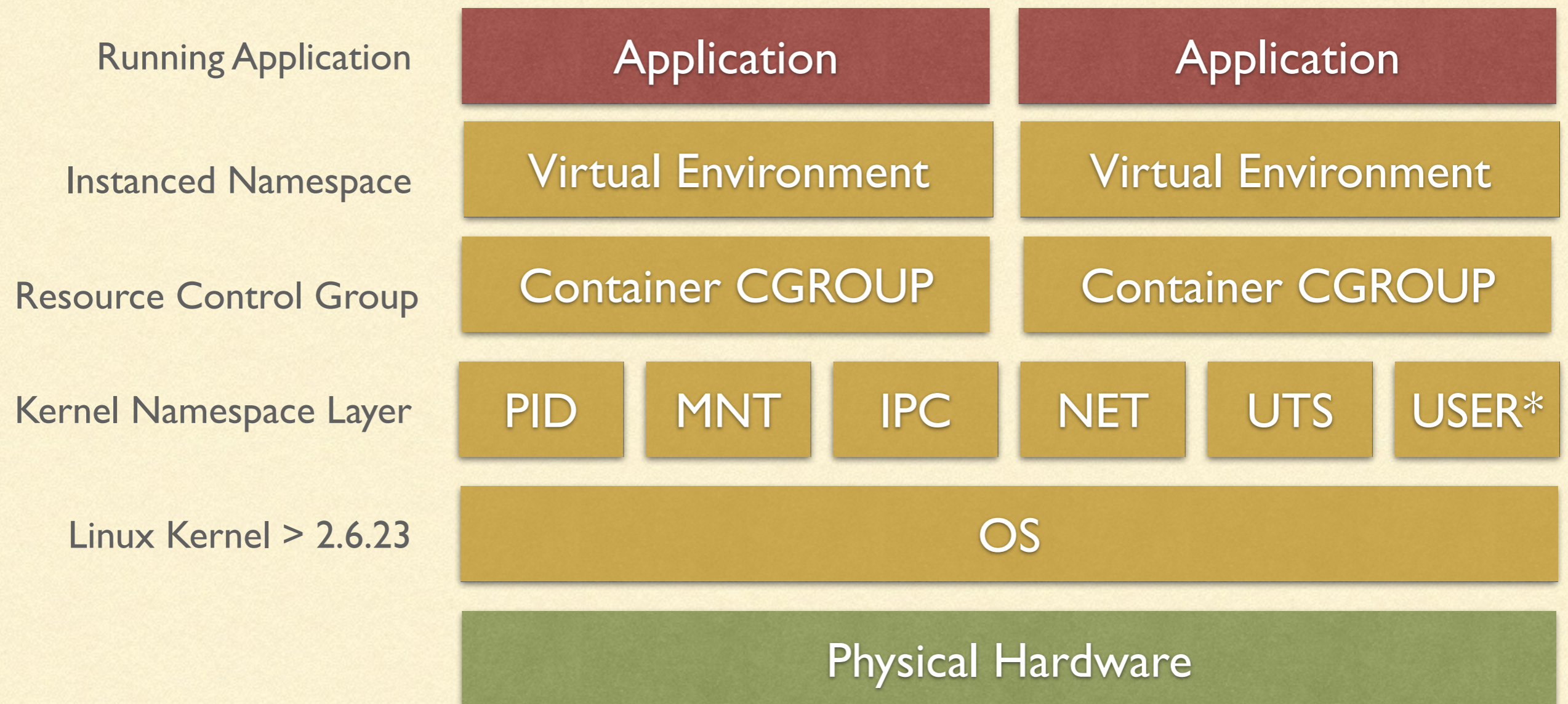
- Lightweight / Dense
- Fast Instantiation
- Elastic Resource
- Low Memory Consumption
- Native Performance

- Cons:

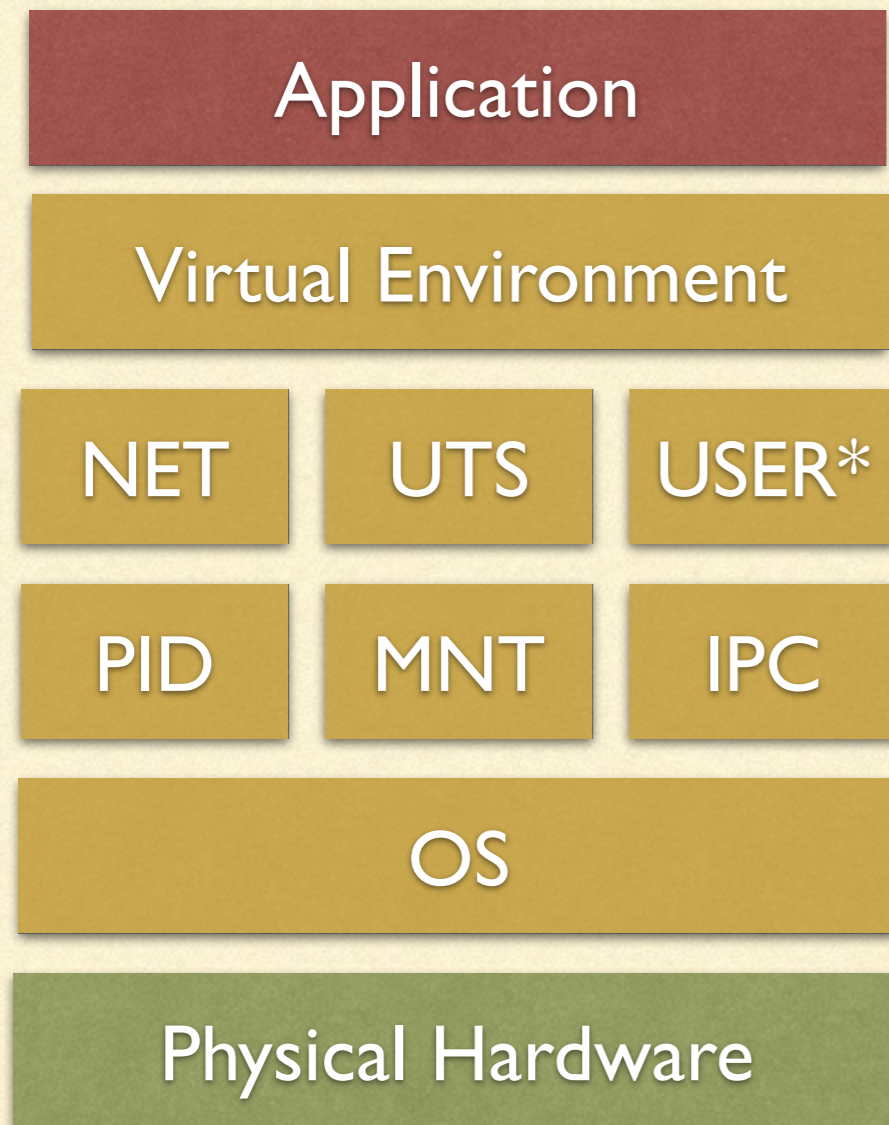
- Restricted / Linux Only
- Shared Kernel
- Security Model
- Young Ecosystem

Linux Container

# Containers in More Detail



# Namespaces



- A Namespace wraps a global resource and presents an isolated instance to running process.
  - MNT - Filesystem mounts
  - IPC - InterProcess Communication
  - PID - Process ID
  - NET - Network
  - UTS - nodename / domainname
  - USER - User & Group ID \*

# CGroups

```
[root@hv001 ~]# ls /cgroup
blkio  cpu  cpuacct  cpuset  devices  freezer  memory  net_cls
[root@hv001 ~]# █
```

Application

Virtual Environment

CGROUP

OS

Physical Hardware

- Control Groups are a set of features that allow monitoring and limiting resource usage.
- Initially developed by Google in 2007 and merged in kernel 2.6.24
- Not only applicable to Containerisation.
- As of RHEL 6x simple to set up:
  - `edit cgconfig.conf`
  - `service cgconfig start`



---

# Simple Usage Patterns

---



**ulimit++**

- Single application
- Resource Limits (Memory/Cores)
- Process Isolation



**quotas++**

- Multiple applications
- Resource Limits (Memory/Cores)
- Process Isolation
- Private or shared filesystem



**vm-lite**

- OS Services (c.f. initd)
- Resource Limits (Memory/Cores)
- Unique Identity (host/IP)

---

# Some User-land Implementations

---

## LXC – Linux Containers

Userspace tools for the Linux kernel containers



---

# Some User-land Implementations

---



- Linux implementation released 2005
- Comparable to VM Host
  - Provides tools to manage Containers
  - Usually starts `/sbin/init`
- Requires custom Kernel
- Provides improvements of namespaces & cgroups:
  - Live Migration
  - Check Pointing
  - Enhance Security

---

# Some User-land Implementations

---



- Library front end to a range of Virtualisation technologies.
- Technology behind KVM
- Includes an “LXC” Container Driver
- Shares common features with other QEMU/KVM environments.
- Leverages CGroups to manage/control system resources
- Support multiple usage patterns (c.f. slide 9).
- Includes support for UID/GID mapping for security
- sVirt can also be used for security <sub>12</sub>

---

# Some User-land Implementations

---

## LXC – Linux Containers

Userspace tools for the Linux kernel containers

- Library, language bindings and CLI tools for managing containers.
- Popular on Debian base Linux distributions.
- Recently included in EPEL with support for RHEL based OS
- Works on Vanilla Kernels
- Younger library, lacking some features.
- As with LibVirt supports multiple usage patterns (c.f. slide 9).
- Backend for Docker
- With version 1.0.0 better security via User namespaces.

# Container: Hello World! with LibVirt

## SETUP:

service cgconfig start

ensure SELINUX is set to permissive

```
<domain type='lxc'>
  <name>helloworld</name>
  <memory>102400</memory>
  <os>
    <type>exe</type>
    <init>/bin/sh</init>
  </os>
  <devices>
    <console type='pty' />
  </devices>
</domain>
```

**helloworld.xml**

```
3. root@hv001:~ (ssh)
[root@hv001 ~]# export LIBVIRT_DEFAULT_URI=lxc:///
[root@hv001 ~]# virsh start helloworld
Domain helloworld started

[root@hv001 ~]# virsh console helloworld
Connected to domain helloworld
Escape character is ^]
sh-4.1# ps -elf
F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
4 S root           1     0  0  80   0 -  2868 wait  09:44 pts/2    00:00:00 /bin/
0 R root           3     1  0  80   0 -  3343 -    09:45 pts/2    00:00:00 ps -e
sh-4.1#
```

```
6. root@hv001:~ (ssh)
nobody    2142  0.0  0.0 12892  588 ?        S    09:43  0:00 /usr/sbin/dnsmasq --strict-order --local=// --domain-needed --pid-f
root      2243  0.0  0.0 72480 1660 ?        Ss   09:44  0:00 /usr/libexec/libvirt_lxc --name helloworld --console 21 --security=
root      2244  0.0  0.0 11472 1536 pts/2    Ss+  09:44  0:00 \_ /bin/sh
[root@hv001 ~]#
```

# Benchmarks - Testbed



- $T_1$  &  $T_2$ 
  - Supermicro Twin Chassis
  - Host - SL6.5
  - Dual Intel Xeon E5420@ 2.50GHz
  - 16GB RAM DDR2 667 MHz
  - Dual 500GB 7200rpm in Raid-0
  - Run 1 VM or 1 Container with all available resources
- $T_1$ : Virtual Machine Host
  - KVM using LVM image
  - VT-x and VT-d enabled
- $T_2$ : Container Host
  - LXC version 0.9.0

# Benchmarks - HEP SPEEC

32-Bit HEP SPEEC running on a 64-Bit OS (GRID Accounting)

```
./runspec.sh -d "HEP-SPEEC06 32-bit" -a 32 -b all_cpp
```

T	RUN 1	RUN 2	AVG	$\Delta$
Metal	73.11	72.54	72.83	-
VM	68.84	68.82	68.83	~5.4%

T	RUN 1	RUN 2	AVG	$\Delta$
Metal	72.48	72.51	72.50	-
Container	72.80	72.51	72.66	<1%



---

# Benchmarks - bonnie++

---

bonnie++ is a synthetic filesystem benchmark (here focusing on Bulk R/W)

`bonnie++ -d /disk/ -r 16384 -u benchmark`

T	R	$\Delta$	W	$\Delta$
Metal	134.7 MB/s	-	100 MB/s	-
VM	105.1 MB/s	20%	63 MB/s	37%

T	R	$\Delta$	W	$\Delta$
Metal	124.4 MB/s	-	97.7 MB/s	-
Container	121.6 MB/s	1%	97.5 MB/s	<1%

---

# Benchmarks - AthenaMP

Atlas production MC Simulation, multi-core payload.

CPU bound with gather step to create HITS file.

8 cores \* 5 Events per core = 40 events

T	RUN 1	RUN 2	AVG	$\Delta$
Metal	80.1	80.3	80.2	-
VM	85.5	85.2	85.3	~6.3%

T	RUN 1	RUN 2	AVG	$\Delta$
Metal	81.3	81.4	81.3	-
Container	81.0	80.6	80.8	<1%

\*Thanks to Graeme for supplying payload\*

---

# Benchmarks - AthenaMP

---

Atlas production MC Simulation, multi-core payload.

CPU bound with gather step to create HITS file.

8 cores \* 50 Events per core = 400 events

T	RUN I	$\Delta$
Metal	63	-
VM	67.2	~6.6%

T	RUN I	$\Delta$
Metal	62.9	-
Container	62	<1%

---

---

# Conclusions

---

- *Containers are an interesting alternative to whole system Virtualisation.*
- *Key components are “baked” into the standard Linux Kernel with a number of user-land tools available.*
- *Containers can be used in a variety of different ways, depending on requirements.*
- *Performance characteristics of containers is similar to bare metal.*
- *One more thing...*

---

# docker.io

---



*“Docker containers can encapsulate any payload, and will run consistently on and between virtually any server. The same container that a developer builds and tests on a laptop will run at scale, in production\*, on VMs, bare-metal servers, OpenStack clusters, public instances, or combinations of the above”*



821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

821197 0  
2/G1

MGW.  
TARE.

NET.  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

MAX. GROSS  
TARE.

MAX. NET GROSS  
CU. CAP.

September 19, 2013

## Red Hat and dotCloud Collaborate on Docker to Bring Next Generation Linux Container Enhancements to OpenShift Platform-as-a-Service

Raleigh

NC, Global, September 19, 2013

Collaboration aims to bring developers simpler, more secure, lightweight, and portable environments for applications

**RALEIGH, N.C. and SAN FRANCISCO- September 19, 2013** – Red Hat, Inc. (NYSE: RHT), the world's leading provider of open source solutions, and dotCloud, the company behind Docker, an open source project to pack, ship, and run any application as a lightweight container, today announced a technical collaboration based on next-generation Linux Containers technology to help drive the next evolution of [OpenShift by Red Hat](#), Red Hat's Platform-as-a-Service (PaaS) offering. The collaboration between the OpenShift, Red Hat Enterprise Linux, and Docker teams aims to combine the versatile capabilities of Docker with the security and stability of Red Hat Enterprise Linux Gears in OpenShift. OpenShift developers will benefit from a simpler, more secure, lightweight, and portable environment for applications.

September 19, 2013

## RED HAT AND DOCKER COLLABORATE

We are thrilled to [announce](#) the collaboration between

Collaboration with Red Hat is important for a number

- Driving compatibility with the most widely deployed Linux
- Enabling integration with one of the most prominent application solutions
- Collaborating with the most prominent, pure open source

**First, it is critically important for us to make Docker work seamlessly with Red Hat Enterprise Linux and related Linux distributions, such as Fedora.**

This is the #1 requested enhancement for Docker, and is obviously a major concern for people who want to deploy Docker in mainstream production environments. Our teams have been working together to package Docker for Fedora in time for the next release of Docker (0.7). Red Hat and dotCloud are planning to make Docker available for all Fedora users with upcoming releases, and are doing the initial packaging work that will ultimately enable Docker to more easily build and deploy on Red Hat Enterprise Linux.



---

# Demo



---

# Backup

```
# docker search -t centos
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker search -t centos
```

NAME	DESCRIPTION	STARS	OFFICIAL	TRUSTED
kalefranz/centos		2		[OK]
russmckendrick/docker-centos	My CentOS Base build	1		[OK]
hzhang/centos-ssh		1		[OK]
longsight/centos-dspace		1		[OK]
tutum/centos-6.4	Centos 6.4 image with SSH access. For the ...	1		[OK]
hzhang/centos-nginx		1		[OK]
dockerfiles/centos-lamp		1		[OK]
saltstack/centos-6-minimal		1		[OK]

```
# docker images
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
centos	6.4	539c0211cd76	11 months ago	300.6 MB
centos	latest	539c0211cd76	11 months ago	300.6 MB

```
[root@hv002 ~]#
```

```
# docker run -i -t centos /bin/bash
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker run -i -t centos /bin/bash
bash-4.1#
```

```
# docker ps
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
6b307fba5e41   centos:6.4 /bin/bash               About a minute ago Up 59 seconds          jolly_davinci
[root@hv002 ~]#
```

```
# docker ps -a
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS          NAMES
6b307fba5e41   centos:6.4 /bin/bash               About a minute ago Up About a minute          jolly_davinci
b9184facc9ab   centos:6.4 /bin/bash               2 weeks ago     Exit 0
ac46c7299f25   centos:6.4 /bin/bash               2 weeks ago     Exit 0
naughty_tesla
[root@hv002 ~]#
```

```
# docker run -i -t -v /cvmfs/atlas.cern.ch:/cvmfs/atlas.cern.ch:ro centos /bin/bash
```

```
2. root@hv002:~/Atlas/Templates (ssh)
root      6767   0.0   0.0   515040 15112 pts/0    Sl   09:12   0:01 /usr/bin/docker -d
root      8491   0.0   0.0   18908  1132 pts/2    Ss   10:22   0:00 \_ lxc-start -n 9a992d20f541c5291f40483e4a71f19e4f80f054b337e70b87643720644982e1 -f /var/lib/docker/containers/9a992d20f541c5291f40483e4a71f19e4f80f054b337e70b87
root      8496   0.0   0.0   11956  2400 pts/2    S    10:22   0:00 \_ /bin/bash
root     10991   0.3   0.0   122712 16204 pts/2    S+   10:41   0:01 \_ python /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/AtlasProduction/17.7.3.4/InstallArea/share/bin/Sim_tf.py --DBRelease=current
root     11126   0.0   0.0   11472  1496 pts/2    S+   10:42   0:00 \_ /bin/sh ./runwrapper.EVNTtoHITS.sh
root     11127  34.1   6.3  1435620 1030552 pts/2    Sl+  10:42   2:05 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cvmfs/
root     11278  96.8   5.9  1456104 979256 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11279  96.6   6.0  1459404 982308 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11280  96.8   6.0  1460704 982688 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11281  96.0   6.0  1466228 988016 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11282  97.0   6.0  1465480 986592 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11283  96.2   6.0  1464732 985156 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11284  96.1   6.0  1464880 986336 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     11285  98.8   6.0  1465284 987216 pts/2    RL   10:47   0:36 \_ /cvmfs/atlas.cern.ch/repo/sw/software/x86_64-slc6-gcc46-opt/17.7.3/sw/lcg/external/Python/2.6.5p2/x86_64-slc6-gcc46-opt/bin/python -tt /cv
root     10992   0.0   0.0   4104   572 pts/2    S+   10:41   0:00 \_ tee o
```

```
# docker logs <id>
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker logs 6b307fba5e41
bash-4.1#
bash-4.1# ls
bin boot dev etc home lib lib64 media mnt opt proc root sbin selinux srv sys tmp usr var
bash-4.1# ps -elf
F S UID          PID  PPID  C  PRI  NI ADDR  SZ  WCHAN  STIME TTY          TIME CMD
4 S root           1     0  0  80   0 - 2825 wait  04:55 ?           00:00:00 /bin/bash
0 R root           7     1  0  80   0 - 3340 -    09:41 ?           00:00:00 ps -elf
bash-4.1#
[root@hv002 ~]#
```

```
# docker diff <id>
```

```
1. root@hv002:~ (ssh)
[root@hv002 ~]# docker diff 6b307fba5e41
C /dev
A /dev/kmsg
A /dev/tty1
[root@hv002 ~]#
```