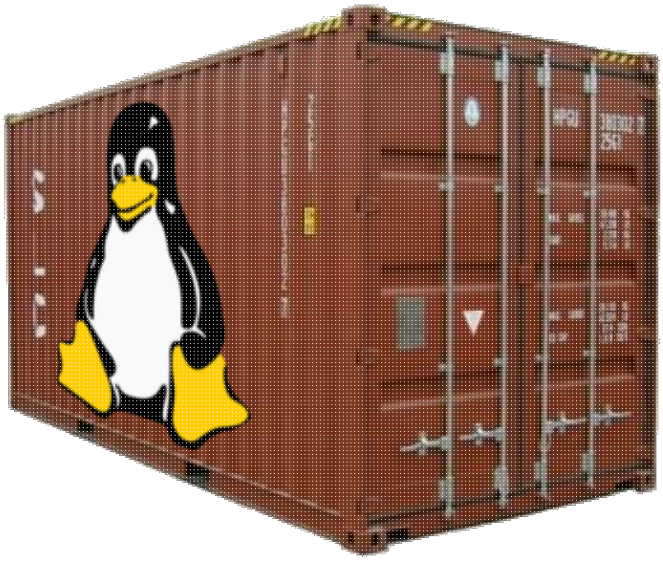

This are notes for my LXC workshop, in state of flux

t=0x87517680 [0,0]

Contents: [Dobrica PavlinuÅ¡iÄ 's random unstructured stuff]

        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (veth)
            - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (host-only bridge)
            - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (pseudo-random mac?)
            - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (slow network?)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (phys)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (limit container resources)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (cpuset.cpus)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (cpu.shares)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (memory)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (disk)
            - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (usage)
            - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (limit disk bandwith using cgroup blkio)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (network)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (LXC commands)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (lxc-create)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (lxc-execute)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (lxc-attach)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (devices)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (monitoring)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (htop)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (procfs)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (debugging)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (kernel patches)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (Are we in container?)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (32-bit guest on 64-bit kernel)
    - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (Container tweaks)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (udev)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (nfs)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (chromium)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (pam)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (X-server)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (Virtual PCI network cards)
        - Dobrica PavlinuÅ¡iÄ 's random unstructured stuff (don't delete files)

Cgroups

- http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt
- http://www.webupd8.org/2010/11/alternative-to-200-lines-kernel-patch.html
- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guid

- Resource allocation using cgroups http://blip.tv/file/4773168

Systemd

- systemd, beyond init http://www.youtube.com/watch?v=TyMLi8QF6sw

launchd alterntive, somilar to inetd, but for unix sockets (mostly)

LXC

Virtual Servers and Checkpoint/Restart in Mainstream Linux
http://lxc.sourceforge.net/doc/sigops/appcr.pdf

- you don't have hardware virtualization (netbooks, anyone?)
  - http://lxc.teegra.net/
  - http://en.gentoo-wiki.com/wiki/LXC
  - http://sysadvent.blogspot.com/2010/12/day-1-linux-containers-lxc.html
- Amazon EC2
  - http://www.phenona.com/blog/using-lxc-linux-containers-in-amazon-ec2/
- Running X
  - http://blog.ikibiki.org/2011/04/05/Running_X_from_LXC/
- LVM integration
  - http://s3hh.wordpress.com/2011/03/30/one-more-lxc-clone-update/

LXC inside KVM for testing

- http://sysadmin-cookbook.rot13.org/#lxc_kvm

setup KVM LXC test machine

- http://www.landley.net/lxc/01-setup.html

Step 1: Create a root filesystem for the KVM system.

- http://sysadmin-cookbook.rot13.org/#01_create_kvm_root_sh 3m12.426s

Step 2: Build a kernel for KVM, with container support.

- http://sysadmin-cookbook.rot13.org/#02_build_kvm_kernel_sh 8m22.248s

Step 3: Boot the result under QEMU or KVM

- http://sysadmin-cookbook.rot13.org/#03_boot_kvm_sh

Step 4: ssh into the KVM instance.

```
ssh root@127.0.0.1 -p 9876
```

Step 5: Set up a simple busybox-based container under the KVM system.

```
wget http://busybox.net/downloads/binaries/latest/busybox-i686 -O busybox
chmod +x busybox
echo -e "lxc.utsname = container\nlxc.network.type = empty" > container.conf
PATH=$(pwd):$PATH lxc-create -f container.conf -t busybox -n container
```

Step 6: Launch the container

```
lxc-start -n container

# console is broken, so start another

lxc-console -n container
```

Step 7: Stop the container, and the KVM system.

```
lxc-stop -n container

# remove container
lxc-destroy -n container
```

Setup networking

- http://www.landley.net/lxc/02-networking.html

Step 1: Add a TAP interface to the Laptop.

```
# FIXME change username
tunctl -u dpavlin -t kvm0
ifconfig kvm0 192.168.254.1 netmask 255.255.255.0
echo 1 > /proc/sys/net/ipv4/ip_forward

# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Step 2: Launch KVM with two ethernet interfaces.

```
kvm -m 1024 -kernel ../01-setup/linux-2.6.*/arch/x86/boot/bzImage -no-reboot \
    -hda ../01-setup/squeeze.ext3 -append "root=/dev/hda rw panic=1" \
    -net nic,model=e1000 -net user -redir tcp:9876::22 \
    -net nic,model=e1000 -net tap,ifname=kvm0,script=no
```

Step 3: Set up a new container in the KVM system.

```
root@kvm:~# cat > busybox.conf << EOF
lxc.utsname = busybox
lxc.network.type = phys
lxc.network.flags = up
lxc.network.link = eth1
#lxc.network.name = eth0
EOF

PATH=$(pwd):$PATH lxc-create -f busybox.conf -t busybox -n busybox
lxc-start -n busybox
```

```
root@kvm:~# lxc-console -n busybox

ifconfig eth1 192.168.254.2 netmask 255.255.255.0
route add default gw 192.168.254.1
```

Step 4: Fun with routing.

On host, bring up loopback alias in KVM network:

```
dpavlin@x200:~$ sudo ifconfig lo:1 10.0.2.200 netmask 255.255.255.0
```

busybox container can reach it, while KVM can't !

network hints

prepare host machine

macvlan

- aliased IP at eth devel
- new device with own mac **with** offloading
- can't communicate with other containers or host (< 2.6.33)

lxc.network.type=macvlan
lxc.network.link=eth0
lxc.network.flags=up

ip link add link <phys> name <vif> address <mac address> type macvlan mode (bridge|vepa|private)

ip link add link bond200 name bond200:0 address 00:aa:bb:cc:dd:ee type macvlan mode bridge

ip -d show link bond200:0

lxc.network.type = macvlan
lxc.network.macvlan.mode = bridge
lxc.network.flags = up
lxc.network.link = bond200
lxc.network.name = eth7
lxc.network.mtu = 1500
lxc.network.ipv4 = 192.168.90.11/24
lxc.network.hwaddr = 4a:49:43:49:79:0B

veth

```
sudo apt-get install bridge-utils dnsmasq
```

```
# setup hints
```

```
sysctl -w net.ipv4.ip_forward=1

ptables -t nat -A POSTROUTING -o wlan0 -j SNAT --to-source=WLAN0_IP

# or for nat
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up

1. name inside container

lxc.network.name = eth0.12
lxc.network.mtu = 1500
lxc.network.ipv4 = 10.60.0.12/23
lxc.network.hwaddr = AC:DE:48:00:00:0C

1. name host interface for bridge

lxc.network.veth.pair = veth12

host-only bridge

```
$ cat /etc/network/interfaces

auto br0
iface br0 inet static
        bridge_ports dummy0
        bridge_maxwait 0
        address 172.16.16.1
        netmask 255.255.255.0
```

pseudo-random mac?

http://en.wikipedia.org/wiki/Mac_address

x2:xx:xx:xx:xx:xx
x6:xx:xx:xx:xx:xx
xA:xx:xx:xx:xx:xx
xE:xx:xx:xx:xx:xx

IP=192.168.0.50 # container nic IP
HA=`printf "02:00:%x:%x:%x:%x" ${IP//./ }` # generate a MAC from the IP

slow network?

/usr/sbin/ethtool -K br0 sg off
/usr/sbin/ethtool -K br0 tso off

phys

kernel > 2.6.35

lxc.network.type=phys
lxc.network.link=eth1
lxc.network.name=eth1

limit container resources

cpuset.cpus

echo 1 > /cgroup/<name>/cpuset.cpus # 2nd CPU!

echo 1,2,3 > /cgroup/<name>/cpuset.cpus

echo 0-7 > /cgroup/<name>/cpuset.cpus

lxc-execute -n foo -s lxc.cgroup.cpuset.cpus="1,2,3" myforks

cpu.shares

lxc-execute -n foo -s lxc.cgroup.cpu.shares=1 /bin/bash

lxc-execute -n bar -s /bin/bash

while $(true); do echo -n . ; done

lxc-cgroup -n foo cpu.shares=1024

memory

lxc.cgroup.memory.limit_in_bytes = 256M
lxc.cgroup.memory.memsw.limit_in_bytes = 1G

disk

usage

standard Linux tools:

- LVM
- quota (it can be bypassed if the container runs with CAP_SYS_ADMIN and/or CAP_SYS_RESOURCE capabilities)

limit disk bandwith using cgroup blkio

- http://www.mjmwired.net/kernel/Documentation/cgroups/blkio-controller.txt

Required kernel configuration

CONFIG_BLK_CGROUP=y
CONFIG_CFQ_GROUP_IOSCHED=y
CONFIG_BLK_DEV_THROTTLING=y

create containers for test

```
#!/bin/sh -xe

lxc-ls | xargs -i sh -c "lxc-stop -n {} ; lxc-destroy -n {}"

echo "lxc.network.type = empty" > blkio.conf

PATH=$(pwd):$PATH lxc-create -f blkio.conf -t busybox -n disk1
PATH=$(pwd):$PATH lxc-create -f blkio.conf -t busybox -n disk2
PATH=$(pwd):$PATH lxc-create -f blkio.conf -t busybox -n disk3

lxc-ls | xargs -i dd if=/dev/zero of=/var/lib/lxc/{}/rootfs/tmp/zero bs=1M count=100

cat > /tmp/speed.sh <<EOF
#!/bin/sh
while true ; do
        sync ; echo 3 > /proc/sys/vm/drop_caches
        dd if=/tmp/zero of=/dev/null 2>&1
done | grep MB
EOF

chmod +x /tmp/speed.sh

lxc-ls | xargs -i cp /tmp/speed.sh /var/lib/lxc/{}/rootfs/tmp/speed.sh

lxc-ls | xargs -i lxc-start -d -n {}
```

login into each container and run test

```
root@kvm:~# lxc-console -n disk1

Type <Ctrl+a q> to exit the console

disk1 login: root
~ # /tmp/speed.sh
104857600 bytes (100.0MB) copied, 0.958453 seconds, 104.3MB/s
```

Test limits (be careful not to enter 1000, you might oops kernel!)

```
root@kvm:~# echo 100 > /mnt/cgroup/disk1/blkio.weight
root@kvm:~# echo 200 > /mnt/cgroup/disk2/blkio.weight
root@kvm:~# echo 500 > /mnt/cgroup/disk3/blkio.weight

root@kvm:~# cat /mnt/cgroup/disk?/blkio.weight
100
200
500
```

Limit /dev/hda to 1Mb/s read

```
root@kvm:~# ls -al /dev/hda
```

```
brw-rw---- 1 root disk 3, 0 May 15 00:10 /dev/hda

root@kvm:~# echo "3:0 1048576" > /mnt/cgroup/disk1/blkio.throttle.read_bps_device
```

network

- http://vger.kernel.org/netconf2009_slides/Network%20Control%20Group%20Whitepaper.odt

```
# mkdir -p /dev/cgroup
# mount -t cgroup net_cls -o net_cls /dev/cgroup
# mkdir /dev/cgroup/A
# mkdir /dev/cgroup/B

# cd /dev/cgroup
# echo 0x1001 > A/net_cls.classid # 10:1
# echo 0x1002 > B/net_cls.classid # 10:2

# tc qdisc add dev eth0 root handle 10: htb

# tc class add dev eth0 parent 10: classid 10:1 htb rate 40mbit
# tc class add dev eth0 parent 10: classid 10:2 htb rate 30mbit

# tc filter add dev eth0 parent 10: protocol ip prio 10 handle 1: cgroup
```

LXC commands

lxc-create

/usr/lib/lxc/templates/

export MIRROR=http://192.168.1.20:3142/ftp.debian.org
export SUITE=lenny

cat > /tmp/lenny.conf
lxc.network.type=veth
lxc.network.link=br0
lxc.network.flags=up

1. <ctrl+d>

t61p:~# lxc-create -n lenny -t debian -f /tmp/lenny.conf

lxc-execute

application container (shares filesystem!)

lxc-ssh

lxc-execute -n foo -s lxc.utsname=foo /bin/bash
lxc-execute -n bar -s lxc.utsname=bar /bin/bash

lxc-attach

Needs kernel patch

lxc-attach ~~n n0 –~~ /usr/sbin/tcpdump -i eth0

devices

http://lwn.net/Articles/273208/

lxc.cgroup.devices.allow = <type> <major>:<minor> <perm>

<type> : b (block), c (char), etc ...
<major> : major number
<minor> : minor number (wildcard is accepted)
<perms> : r (read), w (write), m (mapping)

monitoring

htop

htop - cgroups > r192

t61p:/tmp# apt-get source htop
t61p:/tmp# apt-get build-dep htop
t61p:/tmp# dpkg-source -x htop_0.9-2.dsc
t61p:/tmp# cd htop-0.9/

t61p:/tmp/htop-0.9# DEB_BUILD_OPTIONS="--enable-cgroup" fakeroot debian/rules binary

1. sigh, no work, patch debian/rules to add --enable-cgroup

t61p:/tmp/htop-0.9# fakeroot debian/rules binary
t61p:/tmp/htop-0.9# dpkg -i ../htop_0.9-2_i386.deb

procfs

http://lxc.sourceforge.net/download/procfs/procfs.tar.gz (fuse, defunct)
http://www.tinola.com/lxc/ (somewhat newer)

debugging

lxc-start --logpriority=TRACE -o /tmp/trace.log --name my_container

(must have redirect to file!)

kernel patches

http://lxc.sourceforge.net/patches/linux/

Are we in container?

on host:

dpavlin@stage:~$ cat /proc/$$/cgroup
1:net_cls,freezer,devices,cpuacct,cpu,ns,cpuset:/

inside container:

dpavlin@narada:~$ cat /proc/$$/cgroup
1:net_cls,freezer,devices,cpuacct,cpu,ns,cpuset:/narada

32-bit guest on 64-bit kernel

(lxc >= 0.7.3)

lxc.arch=x86

Container tweaks

udev

echo udev hold | dpkg --set-selections

nfs

kernel doesn't have nfs namespaces yet, use user-space nfs servers:

- http://unfs3.sourceforge.net
- http://sourceforge.net/apps/trac/nfs-ganesha

chromium

- http://www.chromium.org/chromium-os/chromiumos-design-docs/system-hardening
- http://git.chromium.org/gitweb/?p=chromiumos/platform/minijail.git;a=summary

pam

- http://pam-netns.sourceforge.net/

pam_netns allows to setup a private network namespace for every user
session (comparable with pam_namespace for filesystem namespaces). This
is especially useful on multiseat environments.

X-server

- http://box.matto.nl/lxcxserver.html (Xnest example)

- https://launchpad.net/arkose - Arkose - Desktop Application Sandboxing (using aufs2)

Virtual PCI network cards

- http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization/sect-Para-virtual

don't delete files

dpkg-divert --rename /etc/init/theinitfile.conf

.pre