Systems Administration Miniconf

Creating Debian container images, the old and simple way

Hamish Coleman hamish@zot.org

LINUX.CONF.AU
21-25 January
2019
Christchurch, NZ

Creating Debian and Ubuntu container base images

The old and simple way

Sysadmin Miniconf Linux.Conf.Au 2019 Christchurch

Hamish Coleman - hamish@zot.org

What is a base image?

- A root filesystem
- Or a template for one
- eg: Docker, Kubernetes and LXC all will use base image templates

What is a base image?

- A root filesystem
- Or a template for one
- eg: Docker, Kubernetes and LXC all will use base image templates
- Not used for "single binary image" containers

What is a base image?

- A root filesystem
- Or a template for one
- eg: Docker, Kubernetes and LXC all will use base image templates
- Not used for "single binary image" containers

Where do *your* base images come from?

- Provided by your Operating System
- Downloaded by your framework

What is a base image?

- A root filesystem
- Or a template for one
- eg: Docker, Kubernetes and LXC all will use base image templates
- Not used for "single binary image" containers

Where do *your* base images come from?

- Provided by your Operating System
- Downloaded by your framework
- "Steve", around the back of the pub?

Why build your own?

- The three "R"s:
  - Repeatable
  - Reliable
  - Reproducible
- Customise the default config to match your needs
- Include special packages (private builds or specific versions)
- Use one build that can be used for Testing, QA, CI and Production
- Building for new architectures or hardware platforms

What options do I have?

- debootstrap
  - Used by the Ubuntu and Debian installers, implements all the package installation internally, so only needs shell, wget and binutils to run
- cdebootstrap
  - A "C implementation of debootstrap", with the same features
- multistrap
  - Uses the normal debian apt and dpkg tools. Supports installing the base system with packages from multiple repositories
- vmdebootstrap
  - Wrapper around debootstrap for building a virtual disk image for VMs
- Many more!

debootstrap

```
sudo debootstrap --arch=amd64 \
    SUITE \
    TARGETDIR \
    MIRROR
```

debootstrap

```
sudo debootstrap --arch=amd64 \
    SUITE \
    TARGETDIR \
    MIRROR
```

SUITE is one of the Debian/Ubuntu release code names:

- jessie, trusty, stretch, xenial, bionic, buster, etc

MIRROR is the package repository URL:

- http://httpredir.debian.org/debian
- http://archive.ubuntu.com/ubuntu

debootstrap

(See also the example1 files in the repo)

```
sudo debootstrap --arch=amd64 \
    stretch \
    /var/lib/container-example/example1 \
    http://httpredir.debian.org/debian
```

multistrap

```
sudo /usr/sbin/multistrap --arch amd64 \
    -f multistrap.conf \
    -d TARGETDIR
```

multistrap

```
sudo /usr/sbin/multistrap --arch amd64 \
    -f multistrap.conf \
    -d TARGETDIR
```

NOTE: it needs a config file!

multistrap

Config file (multistrap.conf)

```
[General]
bootstrap=NameOfSectionDefiningRepo
aptsources=NameOfSectionDefiningRepo

[NameOfSectionDefiningRepo]
source=MIRROR
suite=SUITE
keyring=debian-archive-keyring ubuntu-archive-keyring
packages= systemd udev kmod apt
```

multistrap

(See also the example2 files in the repo)

Install stage:

```
sudo /usr/sbin/multistrap --arch amd64 \
    -f multistrap.conf \
    -d /var/lib/container-example/example2
```

Configure stage: (sometimes optional)

```
chroot /var/lib/container-example/example2 \
    dpkg --configure -a
```

Compare the two tools:

	debootstrap	**multistrap**
Used by the distro installer	✔	
Can install from multiple package repos		✔
Has a Config file		✔
Installs packages with the normal tools		✔
Can install without runing binaries for the target architecture	✔	

Immediately Useful

- The outputs of these tools can be used as is:

```
chroot /var/lib/container-example/example2
dpkg -l
exit
```

Immediately Useful

- The outputs of these tools can be used as is:

```
chroot /var/lib/container-example/example2
dpkg -l
exit
```

Useful for

- Clean build environments
- Testing the installation of new packages

But not without issues

```
chroot /var/lib/container-example/example1
perl -e 'print "hello world\n"'
```

But not without issues

```
chroot /var/lib/container-example/example1
perl -e 'print "hello world\n"'

perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LANGUAGE = (unset),
        LC_ALL = (unset),
        LC_COLLATE = "C",
        LANG = "en_AU.UTF-8"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
hello world

exit
```

But not without issues

```
chroot /var/lib/container-example/example2
apt install less
```

But not without issues

```
chroot /var/lib/container-example/example2
apt install less

Reading package lists... Done
[..omitted..]
Need to get 126 kB of archives.
After this operation, 284 kB of additional disk space will be
Err:1 http://httpredir.debian.org/debian stretch/main amd64 le
  Temporary failure resolving 'httpredir.debian.org'
E: Failed to fetch http://httpredir.debian.org/debian/pool/mai
E: Unable to fetch some archives, maybe run apt-get update or

exit
```

Improvements

(See the example3 files in the repo for scripts implementing these)

Building a reusable template:

- fixup
- customise
- minimise

Turning a template into a specific machine:

- instantiate

fixup

These are changes that help the image to boot properly or remove (incorrect) identifying information - anything that could be considered "broken"

```
shred -fu TARGETDIR/etc/hostname
shred -fu TARGETDIR/etc/machine-id
shred -fu TARGETDIR/etc/ssh/*_key
echo "nameserver 1.1.1.1" > TARGETDIR/etc/resolv.conf
```

customise

The changes needed for your local configuration

- anything that is applicable to all instances of the template.

```
echo root:hunter2 | sudo chpasswd -c SHA256 -R TARGETDIR
chroot TARGETDIR systemctl enable systemd-networkd
chroot TARGETDIR systemctl enable systemd-resolved
echo "nameserver 127.0.0.53" > TARGETDIR/etc/resolv.conf
```

minimise

Size reductions on the image

```
rm -rf TARGETDIR/usr/share/locale/*
rm -rf TARGETDIR/usr/share/doc/*
rm -rf TARGETDIR/lib/udev/hwdb.bin
```

instantiate

- Some container systems take care of instantiation for you
- If the end result is not a container infrastructure, you probably need to do it yourself.

```
echo realhostname >TARGETDIR/etc/hostname
echo 127.0.1.1 realhostname >>TARGETDIR/etc/hosts
chroot TARGETDIR dpkg-reconfigure openssh-server
```

Simple boot

The systemd-nspawn tool provides a quick and no-fuss way to startup and test containers.

```
sudo systemd-nspawn --boot --ephemeral \
    --directory=/var/lib/container-example/example2
```

`--ephemeral` means that nspawn will copy the template dir and all changes will be lost when the container stops.

Simple boot

The systemd-nspawn tool provides a quick and no-fuss way to startup and test containers.

```
sudo systemd-nspawn --boot --ephemeral \
    --directory=/var/lib/container-example/example2
```

`--ephemeral` means that nspawn will copy the template dir and all changes will be lost when the container stops.

- login using the root password you set earlier
- stop the container from inside with a `systemctl poweroff` or from the host with `machinectl stop NAME`

Using libvirt + LXC

(See example4 in the repo)

```
<domain type='lxc'>
  <name>newhost</name>
  <memory unit='KiB'>200000</memory>
  <os>
    <type arch='x86_64'>exe</type>
    <init>/sbin/init</init>
  </os>
...
```

- `virsh -c lxc:/// define newhost.xml`
- `virsh start newhost`

Can also be used to build VM disk images

- guestfish
- uefi booting
- packing for qemu

Questions?

Example files are in the repository

- github project:
  - https://github.com/hamishcoleman/talk-containers1/
- talk slides:
  - https://hamishcoleman.github.io/talk-containers1/slides.html

Hamish Coleman - hamish@zot.org

Can I see that libvirt config again?

```
<domain type='lxc'>
  <name>newhost</name>
  <memory unit='KiB'>200000</memory>
  <os>
    <type arch='x86_64'>exe</type>
    <init>/sbin/init</init>
  </os>
  <devices>
    <emulator>/usr/lib/libvirt/libvirt_lxc</emulator>
    <filesystem type='mount' accessmode='mapped'>
      <source dir='/var/lib/container-example/newhost'/>
      <target dir='/'/>
    </filesystem>
    <interface type='network'>
      <source network='default'/>
      <guest dev='host0'/>
    </interface>
```

How do I do this for RPM based distros?

From Debian or Ubuntu, this is a starting point:

```
apt install rpm yum
HOME= rpm --root=/var/lib/container-example/centos \
    --initdb
mkdir -p /var/lib/container-example/centos/var/lib
mv /var/lib/container-example/centos/.rpmdb \
    /var/lib/container-example/centos/var/lib/rpm
wget http://mirror.centos.org/centos/7/os/x86_64/Packages/cent
rpm --root=/var/lib/container-example/centos \
    -i --nodeps \
    centos-release-7-6.1810.2.el7.centos.x86_64.rpm
yum --installroot=/var/lib/container-example/centos \
    install -y rpm-build yum --nogpgcheck
```