Chroot on Steroids - Lightweight Containers with systemd-nspawn

NLUUG Spring 2017 Conference #nluug

Frank Scholten (@Frank_Scholten)

Bio

Frank Scholten

- Senior Software Engineer @ Container Solutions
- Focus on Cloud Native, Containers, DC/OS, Apache Mesos
- Creator of minimesos www.minimesos.org

Agenda

- What is systemd-nspawn?
- Application vs system containers
- `systemd-nspawn`
- `machinectl`
- `mkosi`
- Demo
- Limitations
- Conclusions
- Questions?

What is systemd-nspawn?

Like chroot

```
# debootstrap stable rootfs https://deb.debian.org/debian
# chroot /bin/bash rootfs
# pwd
/
```

...on steroids

```
# systemd-nspawn --boot --directory rootfs
Spawning container stable-rootfs on /home/frank/src/nluug-spri
Press ^] three times within 1s to kill container.
systemd 215 running in system mode. (+PAM +AUDIT +SELINUX +IMA
Detected virtualization 'systemd-nspawn'.
Detected architecture 'x86-64'.

Welcome to Debian GNU/Linux 8 (jessie)!

...

Debian GNU/Linux 8 frankcs console

frankcs login:
```

Application containers vs system containers

Application containers

- Popularized by Docker, rkt
- Run a single process: code + libraries
- Scheduled via orchestrator: Swarm, k8s, DC/OS
- Use specific image format (AppC, OCI)

Application containers

- Used in microservices architectures
- Typically used by developers
- Docker, rkt, runc, Apache Mesos containerizer

System containers

- Popularized by Solaris, OpenVZ, LXC
- Run a full system: OS distro + systemd
- Deployed as a systemd service
- Uses rootfs directory

System containers

- Used for testing and simulation of machines
- Repurpose rootfs for VMs or PXE boot
- Typically used by system operators

systemd-nspawn

systemd-nspawn

- Binary part of systemd
- Launching containers
- Lot's of flags and options
- Created to test systemd itself
  - https://kinvolk.io/blog/2015/12/testing-systemd-patches

systemd-nspawn

Installing systemd 233 on a Packet instance

```
$ git clone https://github.com/systemd/systemd
$ cd systemd
$ ./configure
$ ./autogen.sh c
$ make -j 4
# make install
# reboot
```

- https://github.com/systemd/systemd/blob/master/H

systemd-nspawn

Let's setup a proper container step by step

systemd-nspawn

Grab an Ubuntu Xenial 16.04.2 image from `cdimage.ubuntu.com`

```
# wget <...>/xenial-base-amd64.tar.gz
# mkdir /var/lib/machines/xenial
# tar zxvf xenial-base-amd64.tar.gz
            -C /var/lib/machines/xenial
```

systemd-nspawn

Running containers as services

```
[Unit]
Description=Ubuntu Xenial

[Service]
ExecStart=/usr/bin/systemd-nspawn \
        --quiet \
        --keep-unit \
        --boot \
        --link-journal=host \
        --directory=/var/lib/machines/xenial
Type=notify

[Install]
WantedBy=multi-user.target
```

Add to file `xenial.service`

systemd-nspawn

Running containers as services

```
# systemctl enable $PWD/xenial.service
Created symlink /etc/systemd/system/multi-user.target.wants/xe
Created symlink /etc/systemd/system/xenial.service → /root/xen
```

systemd-nspawn

Check the service

```
# systemctl status xenial
systemctl status xenial
● xenial.service - Ubuntu Xenial
   Loaded: loaded (/root/xenial.service; enabled; vendor prese
   Active: inactive (dead)
```

systemd-nspawn

Start the service

```
# systemctl start xenial
```

systemd-nspawn

Check the service

```
# systemctl status xenial
● xenial.service - Ubuntu Xenial
   Loaded: loaded (/root/xenial.service; enabled; vendor prese
   Active: active (running) since Mon 2017-05-15 20:34:54 UTC;
 Main PID: 16934 (systemd-nspawn)
...
```

systemd-nspawn

Logs are available also

```
# journalctl -u xenial
```

systemd-nspawn

But what about networking?

systemd-nspawn

Networking - Setup a bridge + veth device on the host

Set up networking

Networking - Enable daemons on the container

```
# systemctl enable systemd-networkd
Created symlink from /etc/systemd/system/multi-user.target.wan
Created symlink from /etc/systemd/system/sockets.target.wants/s
# systemctl enable systemd-resolved
Created symlink from /etc/systemd/system/multi-user.target.wan
```

systemd-nspawn

Networking - Pass bridge to container & enable veth device

```
# systemd-nspawn --boot xenial  \
                 --network-bridge=br0
```

systemd-nspawn

Networking - Setup DNS

On host:

- Install `nss-mymachines` plugin
- Add `nss-mymachines` to /etc/nsswitch.conf

```
hosts: files mymachines
```

systemd-nspawn

Networking - Setup DNS

```
# ping xenial
PING xenial (10.0.0.10) 56(84) bytes of data.
64 bytes from 10.0.0.10: icmp_seq=1 ttl=64 time=0.036 ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=64 time=0.031 ms
^C
--- xenial ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 999ms
rtt min/avg/max/mdev = 0.031/0.033/0.036/0.006 ms
```

systemd-nspawn

Advanced features

- btrfs volumes
- Templates
- /etc/systemd/nspawn/container.nspawn

machinectl

machinectl

- systemd machine manager
- Integrates with nspawn containers
- Manage containers and images

machinectl

List containers & VMs

```
$ machinectl list
MACHINE        CLASS     SERVICE
jessie-rootfs container systemd-nspawn


1 machines listed.
```

machinectl

Manage containers & VMs

```
$ machinectl start m1
$ machinectl status m1
$ machinectl shell m1 df -sh
$ machinectl terminate m1
```

machinectl

List images

```
machinectl list-images
NAME                TYPE      RO  USAGE CREATED MODIFIED
ubuntu-base-17.04 directory no  n/a   n/a     n/a

1 images listed.
```

machinectl

Import image (requries systemd-importd)

```
# machinectl pull-tar <tarball>
```

mkosi

mkosi

- Creating legacy-free OS images
- Wrapper around debootstrap, pacstrap, zypper, etc
- Building block for nspawn images

mkosi

```
$ mkosi --distribution ubuntu \
        --release xenial \
        --format directory \
        --output xenial \
        --package dbus
```

Create a Xenial image and install dbus

mkosi

- Add files from `mkosi.extra` folder
- Copied after OS is installed

mkosi

Customization

`mkosi.postinst` file with post install commands

mkosi

- Sharing images?
- Roll your own repository with minio
  - https://minio.io
  - AWS S3 Compatible object store

mkosi

Advanced

- Secure boot
- Partition management
- Validation with checksum

Demo

Limitations

- No access to devices
- Roll your own image management
- Requires root privileges

Conclusions

- Useful for experimenting, testing and simulation
- Requires manual networking and image management
- DNS works out of the box via resolved

Links & Resources

https://lists.freedesktop.org/mailman/listinfo/systemd-devel

https://www.freedesktop.org/wiki/Software/systemd/Vi

https://cloudnull.io/tag/nspawn

https://www.variantweb.net/blog/systemd-machinectl-vs-docker

Keep in touch!

- Twitter @Frank_Scholten
- LinkedIn https://www.linkedin.com/in/scholtenfrank

Thank you!

Questions?