Running Systemd within a Docker image

NOTE: This is not for general purpose use – CAP_SYS_ADMIN grants the container a large number of dangerous privileges – this should be used only by a sysadmin (Ops) – not a developer – for the purpose of containerizing infrastructure.

Some of my insights gained on running systemd within docker – I'm aware the general idea is to run a single process, but that's for developers. I'm a sysadmin, so I know that underlying "docker" as a management system is a full featured Linux namespace/cgroup facility allowing me to run a fully containerized Linux userland.

This is cobbled together from some of the example Dockerfiles provided by Red Hat's **Project Atomic** as well as personal research into systemd, unprivileged container limitations, and the Linux capability system. This isn't guaranteed to work on anything but the latest version of Docker at the time of this post – 1.4.1. May work on a previous version if it has support for –add-cap.

```
FROM centos:centos7
MAINTAINER "Brad Laue" <brad@brad-x.com>
RUN yum -y update; yum clean all \
yum -y swap -- remove fakesystemd -- install systemd systemd-libs \
(cd /lib/systemd/system/sysinit.target.wants/; for i in *; do [ $i == systemd-tmpfiles-setup.service ] || rm -f $i; done);
\
rm -f /lib/systemd/system/multi-user.target.wants/*;\
rm -f /etc/systemd/system/*.wants/*;\
rm -f /lib/systemd/system/local-fs.target.wants/*; \
rm -f /lib/systemd/system/sockets.target.wants/*udev*; \
rm -f /lib/systemd/system/sockets.target.wants/*initctl*; \
rm -f /lib/systemd/system/basic.target.wants/*;\
rm -f /lib/systemd/system/anaconda.target.wants/*;
RUN systemctl mask getty.target swap.target
RUN systemctl disable graphical.target; systemctl enable multi-user.target
VOLUME ["/sys/fs/cgroup"]
VOLUME ["/run"]
VOLUME ["/export"]
ENV container=lxc
CMD ["/usr/lib/systemd/systemd"]
```

Build with:

```
docker build --rm -t bradlaue/centos-systemd .
```

Run with:

```
docker run -ti --cap-add=SYS_ADMIN -e container=lxc bradlaue/centos-systemd
```

Note this does not require a fully privileged container instance as many seem to indicate – systemd requires only CAP_SYS_ADMIN in order to avoid a segfault.

From here you can build out a container by installing/running standard Linux services in the normal systemd way, including process monitoring.