Building VM Images from Docker Containers

Max Rottenkolber <max@mr.gy>

Saturday, 19 September 2015

Dockerfiles (`https://docs.docker.com/reference/builder/`) are a convenient way to bootstrap user space file systems. The other day I was wondering how I could turn Docker containers into raw file system images that could, for instance, be run with *QEMU* (`http://qemu.org`).[1] Maybe, this post will come in handy for others attempting to do the same.

Basically you can make a copy of a Docker container's file system using "`docker export`", which you can then write to a loop device:

```
docker build -t <YOUR-IMAGE> ...
docker create --name=<YOUR-CONTAINER> <YOUR-IMAGE>
dd if=/dev/zero of=disk.img bs=1 count=0 seek=1G
mkfs.ext2 -F disk.img
sudo mount -o loop disk.img /mnt
docker export <YOUR-CONTAINER> | sudo tar x -C /mnt
sudo umount /mnt
```

Convert a Docker container to a raw file system image.

So far so good. You can now go ahead and attempt to run "`disk.img`" with a suitable kernel inside QEMU, but chances are that your image will not boot properly. This is because off-the-shelf Docker images usually come modified with some "Dockerisms" such as the respective OS's init system being prevented from running.[2] To run a container in QEMU we need to undo some of the Docker specific changes to get it to boot.

Taking the official "`ubuntu:14.04`" image as an example, you can see how it disables "`/sbin/initctl`" in its *Dockerfile* (`https://github.com/tianon/docker-brew-ubuntu-core/blob/6dba3ee12ff996640d1043139d5abf8c744862e2/trusty/Dockerfile#L4`). To get an image with a working init, we can undo the changes in a Dockefile based on the "`ubuntu:14.04`" image:

```
FROM ubuntu:14.04

RUN rm /usr/sbin/policy-rc.d \
    && rm /sbin/initctl \
    && dpkg-divert --local --rename --remove /sbin/initctl
```

Example: Re-enable init in `ubuntu:14.04` Docker image.

Now we can apply the steps illustrated above to get a Trusty Tahr user space that will run nicely in QEMU.

- 1. The VM images derived from Docker containers do not contain a kernel. A kernel can be built separately and specified using QEMU's "`-kernel`" option.
- 2. Because you usually do not need the init system when running a container using "`docker run`".