Docker Deep Dive, Part Two

Container Runtimes

Michael Irwin - December 14, 2016

What's a container again?

How's it different than an image?

Containers are...

- Simply an image with its own writeable layer
- Remember that image layers are immutable. New changes go into the containers layer


How's it *actually* work?

Image layers are "unioned" together

- Each image layer is mounted read-only, with only the top layer mounted as writable
- Uses COW (Copy-On-Write) to keep things efficient
- Default implementation is `AUFS`, but work is being done to swap that to `overlay2`

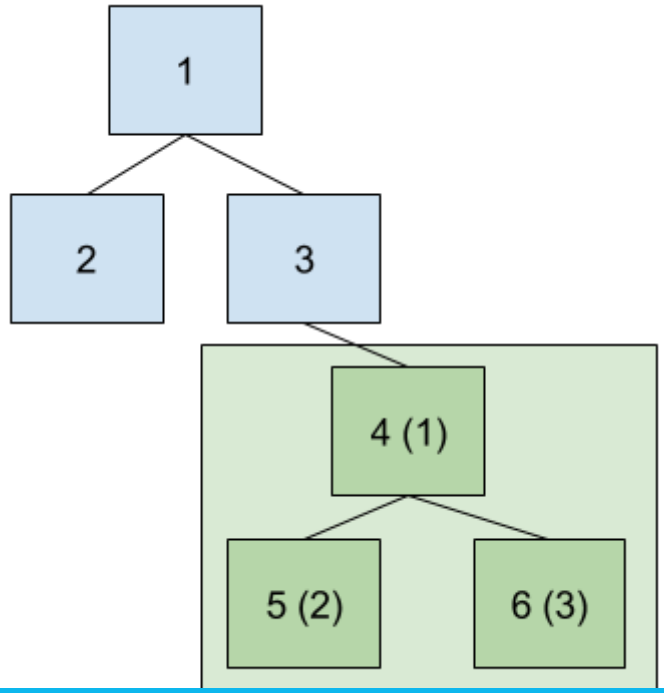
So, how's it look like a VM's isolation?

Uses the power of namespaces

- Provides kernel-level isolation of processes
- Was released in July 2008 with Linux 2.6.26
- Number of namespaces has increased over the years
- Think of it like `chroot`, but for other aspects of the operating system

Process namespace isolation

- Breaks "traditional" view of a single process tree
- Introduces the ability to have "nested" process trees, with each tree being isolated
- Prevents processes in one tree from affecting (and even knowing about) processes in other trees



Container process isolation

- When a container is started, it gets its own process tree
- The default command is run as the root (PID 1)
- When PID 1 exits, the tree dies and the container is done

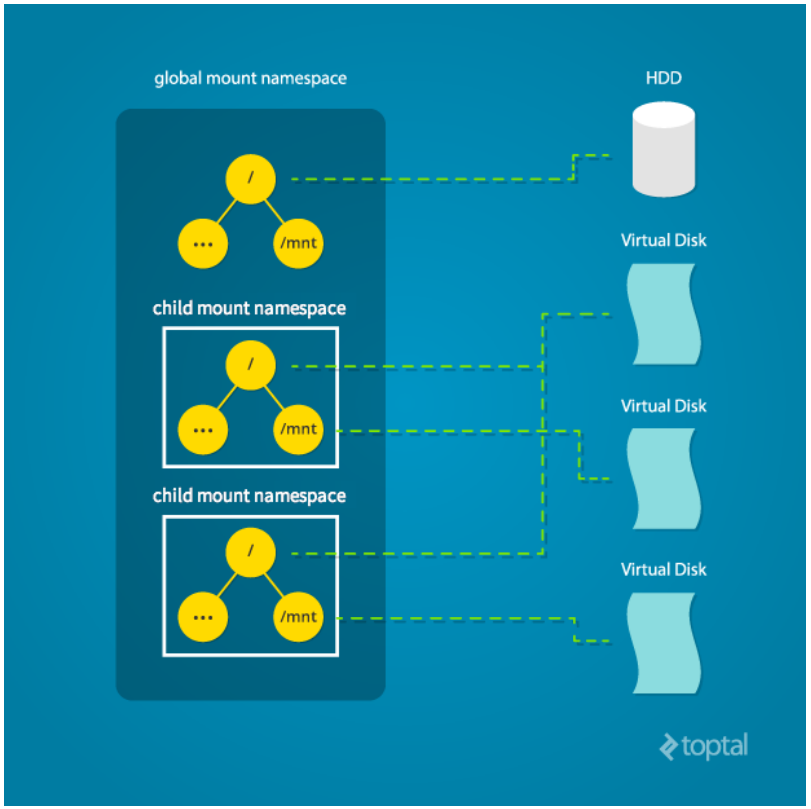
```
$ docker run --rm -ti ubuntu:16.04
root@a3c165bd5870:/# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.5  0.0  18240  3164 ?        Ss   21:34   0:00 /bin/bash
root         11  0.0  0.0  34424  2716 ?        R+   21:34   0:00 ps aux
root@a3c165bd5870:/#
```

Network namespace isolation

- Allows each process to see a completely different set of networking interfaces
  - Even the loopback interface is different for each namespace
- Each namespace gets virtual interfaces
- To make it work, a "routing process" has to route global network traffic to the correct interface

Mount namespace

- Isolates metadata about mounted disk partitions, where they're mounted, what's read-only, etc.
- Gives the feel that each namespace has its own root



User namespace

- Gives isolation of users and groups
- Allows users in one namespace to be completely isolated from another namespace
    - Reason why `root` in a container doesn't mean `root` on the host

Important: Still sharing same kernel (and resources)!

House vs Apartment

- Isolated vs Shared Infrastructure

Using cgroups

- Cgroups (Control Groups) provide resource controls
- Allows for soft and hard limits of CPU and RAM
- Does come with some overhead to keep up with the statistics
- Prevents one container from consuming all resources on the host, DOS'ing other containers

Thanks! Questions?