Containers Anatomy

2018

Agenda

- Setting up a file system
- Namespaces
- CGroups

Chroot Jail

How do i use it?

```
chroot /path/to/new/root command
```

Running Bash

```
$ mkdir -p jail/bin
$ cp /bin/bash jail/bin/
$ sudo chroot jail/ /bin/bash
```

Errrrr....

Chroot Jail

```
$ mkdir -p jail/lib64/x86_64-linux-gnu jail/lib/x86_64-linux-gnu$ ldd $(which bash)
    linux-vdso.so.1 =>  (0x00007ffc75dd4000)
    libtinfo.so.5 => /lib/x86_64-linux-gnu/libtinfo.so.5 (0x00007f6577768000)
    libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f6577564000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f657719a000)
    /lib64/ld-linux-x86-64.so.2 (0x000055979f3fd000)
$ cp /lib/x86_64-linux-gnu/libtinfo.so.5  jail/lib/x86_64-linux-gnu/
$ cp /lib/x86_64-linux-gnu/libdl.so.2  jail/lib/x86_64-linux-gnu/
$ cp /lib/x86_64-linux-gnu/libc.so.6  jail/lib/x86_64-linux-gnu/
$ cp /lib64/ld-linux-x86-64.so.2 jail/lib64/

$ sudo chroot jail/ /bin/bash
```

Chroot Jail

Download rootfs

Link: wget https://github.com/ericchiang/containers-from-scratch/releases/download/v0.1.0/rootfs.tar.gz

```
$ tar -zxf rootfs.tar.gz
```

I can kill any process. Launch a process outside chroot in any other session (eg. top)

```
$ # from host not chroot
$ top &
$ # go inside chroot
$ sudo chroot rootfs /bin/bash
$ mount -t proc proc /proc
$ ps aux | grep top
$ pkill top
```

Namespaces

lightweight process virtualization through isolation

7 namespaces are available to rule them all:

- Mount - isolate filesystem mount points
- UTS - isolate hostname and domainname
- IPC - isolate interprocess communication (IPC) resources
- PID - isolate the PID number space
- Network - isolate network interfaces
- User - isolate UID/GID number spaces
- Cgroup - isolate cgroup root directory

Namespaces

user space additions:

IPROUTE package

- some additions like ip netns add/ip netns del and more.

util-linux package

- unshare util with support for all the 6 namespaces.
- nsenter – a wrapper around setns()

Namespaces

Lets Run an Example:

```
$ unshare -h
$ hostname
  dev-ubuntu
$ sudo unshare -u /bin/sh
$ hostname my-new-hostname
  my-new-hostname
$ hostname
$ exit
$ hostname
  dev-ubuntu
```

Namespaces

Continuing with our rootfs using unshare

```
$ sudo unshare -p -f --mount-proc=$PWD/rootfs/proc  chroot rootfs /bin/bash
$ ps aux

USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.0  20268  3240 ?        S    22:34   0:00 /bin/bash
root          2  0.0  0.0  17504  2096 ?        R+   22:34   0:00 ps aux
```

Namespaces

Entering namespaces with nsenter

```
$ # From the host, not the chroot.
$ ps aux | grep /bin/bash | grep root
$ sudo ls -l /proc/{YOUR_PID}/ns
$ sudo nsenter --pid=/proc/29840/ns/pid \
  unshare -f --mount-proc=$PWD/rootfs/proc \
  chroot rootfs /bin/bash
$ ps aux
```

Deploying an “immutable” container

```
$ # From the host, not the chroot.
$ sudo mkdir readonlyfiles
$ echo "hello" > readonlyfiles/hi.txt
$ sudo mkdir -p rootfs/var/readonlyfiles
$ sudo mount --bind -o ro $PWD/readonlyfiles $PWD/rootfs/var/readonlyfiles
$ sudo chroot rootfs /bin/bash
echo "bye" > /var/readonlyfiles/hi.txt
```

CGroups

cgroups, short for control groups, allow kernel imposed isolation on resources like memory and CPU.

- **Resource limiting** – groups can be set to not exceed a configured memory limit, which also includes the file system cache
- **Prioritization** – some groups may get a larger share of CPU utilization or disk I/O throughput
- **Accounting** – measures a group's resource usage, which may be used, for example, for billing purposes
- **Control** – freezing groups of processes, their checkpointing and restarting

The kernel exposes cgroups through the `/sys/fs/cgroup` directory.

CGroups

Example Creating Memory Group:

```
$ sudo mkdir /sys/fs/cgroup/memory/demo
$ ls /sys/fs/cgroup/memory/demo/
```

Lets Limit the cgroup to 100MB of Memory

```
$ echo "100000000" > /sys/fs/cgroup/memory/demo/memory.limit_in_bytes
$ echo "0" > /sys/fs/cgroup/memory/demo/memory.swappiness
```

The tasks file is special, it contains the list of processes which are assigned to the cgroup. To join the cgroup we can write our own PID.

Links

https://linux.die.net/man/7/capabilities

https://en.wikipedia.org/wiki/Seccomp

http://www.haifux.org/lectures/299/netLec7.pdf

http://man7.org/linux/man-pages/man7/namespaces.7.html

https://prefetch.net/blog/2018/02/22/making-sense-of-linux-namespaces/