CS 447/647

Containers

Containers

- Standardize software packaging
- Modern applications have a lot of components and dependencies.
    - Code
    - Libraries
        - NCR ReactJS application has > 500 dependencies
    - Interpreter (Python or Ruby), Java Runtime Environment, and unique services
        - Versions matter
            - python36, python37, python38
            - OpenJDK 8 - 10
    - Localizations, accounts, and environment settings
- Large organizations have dozens of these applications
    - Conversion from VMs to containers
        - download.cse.unr.edu, ncr-remote.cse.unr.edu, git.cse.unr.edu, ph.engr.unr.edu, mx0.engr.unr.edu

Container Images

- Packages application(s) into a file.
    - Similar to a chroot
    - Repositories of common OS (singularity URL)
        - docker://centos:8
        - docker://debian:buster
- Fat - Contains entire OS and “boots”
- Thin - runs just an application
- Copy and Paste Portability

Chapter 25

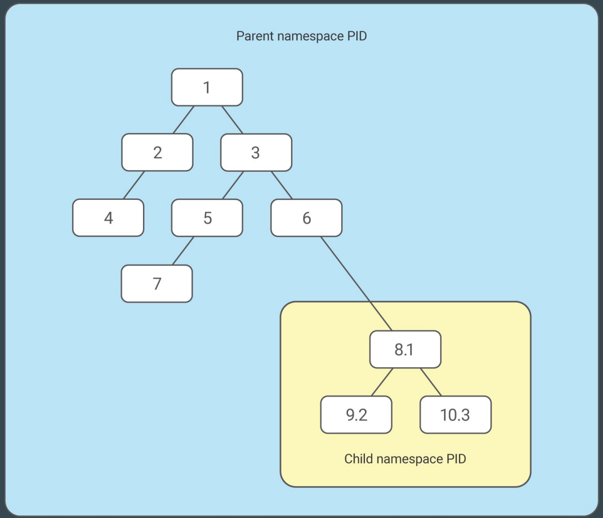
- The book focuses on:
  - Docker
- Other common containerization technologies:
  - Podman
  - systemd-nspawn
    - Lowest level container system
    - Effectively a robust chroot solution
  - LXC - FOSS container system
    - Incus - Community developed
    - LXD - Developed by Canonical (Ubuntu)
  - Apptainer (Singularity) - HPC containers
    - Focuses on portability
    - Developed at LLNL

Core Concepts

- Fusion of existing kernel features
- Container systems tie them all together
- Makes the features easy to use
- A container system should
    - Create and configure virtual Ethernet devices
    - Isolate processes
    - chroot the filesystem
    - Manage the storage volumes
    - Restrict resources (CPU, RAM, Networking)
    - Handle creation, starting, stopping and destruction
        - Maybe backups

Kernel Support - Features

1. Namespaces (2002) - partitions resources by process
2. Control Groups (2007) - restrict resources like RAM
3. Capabilities(1999) - Independent kernel capabilities
    a. CAP_SYS_CHROOT - Allows chroot(2) to be used.
    b. CAP_NET_ADMIN - Allows changes to network system.
    c. man 7 capabilities
4. Secure Computing Mode (2005) - restricts syscall access
    a. /proc/PID/seccomp = 1
        i. Process can only: read, write, exit and sigreturn



namespaces

- man 2 clone
  - used by fork
- CLONE_NEWNS (since Linux 2.4.19)
  - Start the child in a new mount namespace.
- CLONE_NEWNET
  - Start child in a new network namespace
- CLONE_NEWPID
  - create the process in a new PID namespace

```
cmd.SysProcAttr = &syscall.SysProcAttr{
        Cloneflags: syscall.CLONE_NEWPID | syscall.CLONE_NEWNET,
}
```

7 namespaces

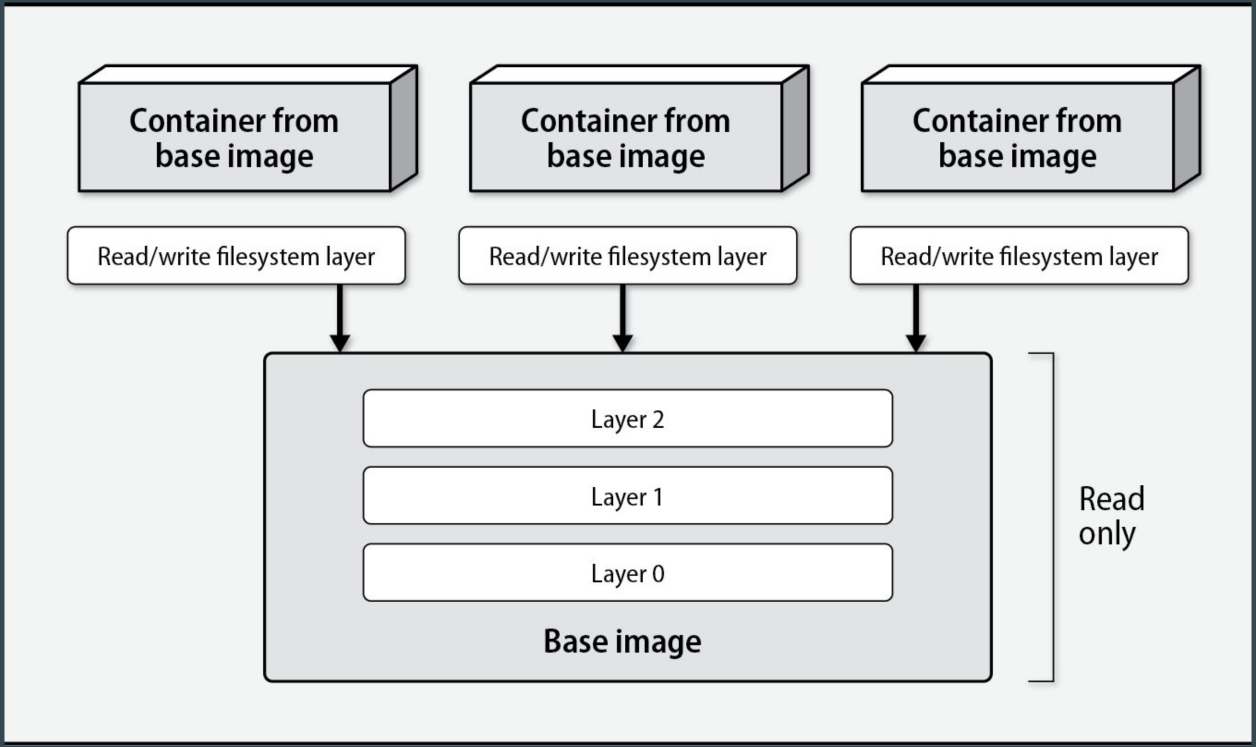
- Mount - isolate filesystem mount points
- UTS - isolate hostname and domainname
- IPC - isolate interprocess communication (IPC) resources
- PID - isolate the PID number space
- Network - isolate network interfaces
- User - isolate UID/GID number spaces
  - https://manpages.debian.org/buster/passwd/subuid.5.en.html
- Cgroup - isolate cgroup root directory

control groups

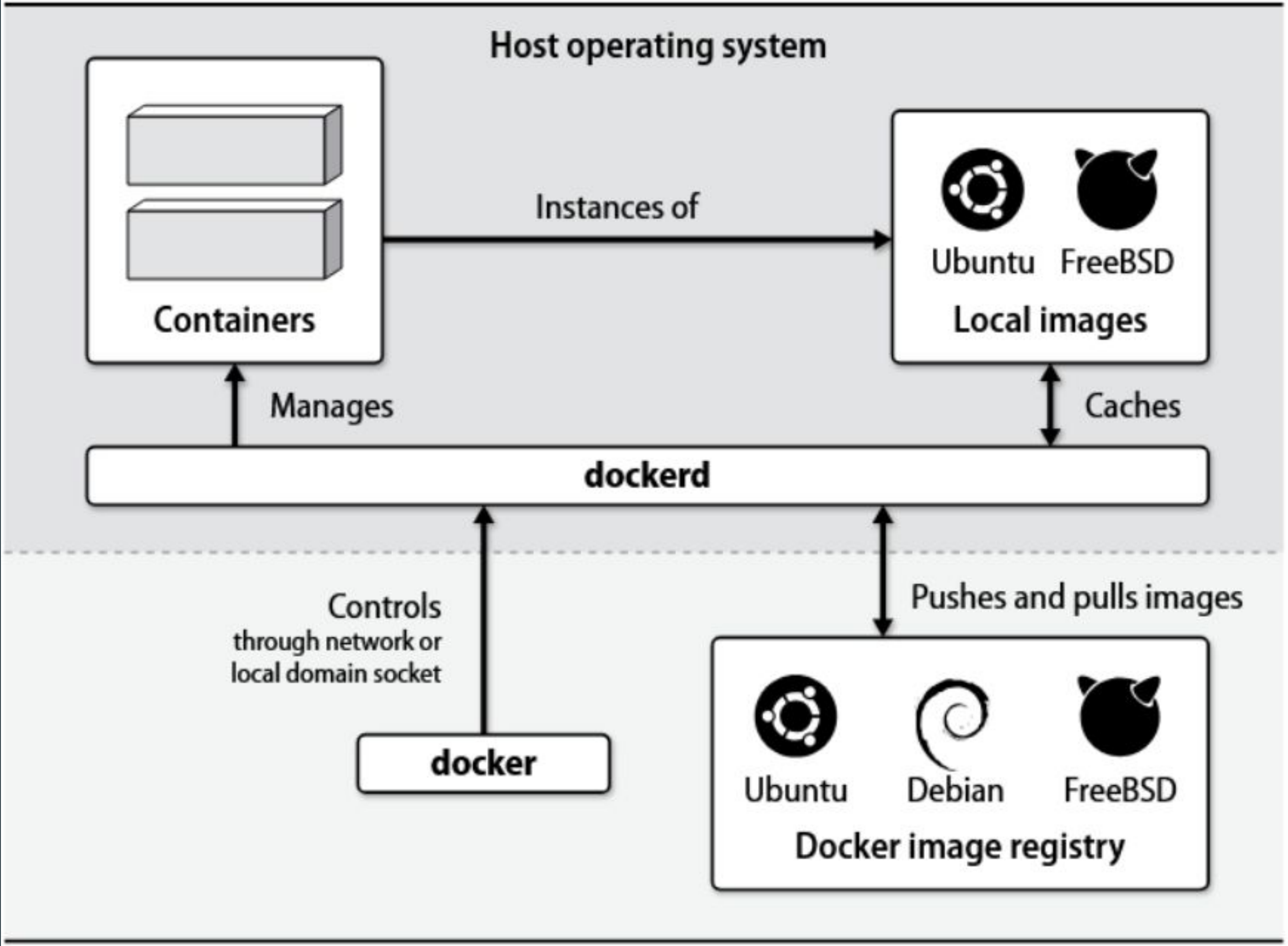
- Restrict resources of a process
    - Cores [0,1,2,3]
    - CPU time in microseconds
        - 100000 is the default
    - Memory
    - Network
    - Block IO
- ad-hoc commands

mount | grep cgroup #Created for use by systemd, Memory=2G

Images


Container from base image
Container from base image
Container from base image
Read/write filesystem layer
Read/write filesystem layer
Read/write filesystem layer
Layer 2
Layer 1
Layer 0
Base image
Read only

Docker architecture



Frequently used docker subcommands

Subcommand	What it does
docker info	Displays summary information about the daemon
docker ps	Displays running containers
docker version	Displays extensive version info about the server and client
docker rm	Removes a container
docker rmi	Removes an image
docker images	Displays local images
docker inspect	Displays the configuration of a container (JSON output)
docker logs	Displays the standard output from a container
docker exec	Executes a command in an existing container
docker run	Runs a new container
docker pull/push	Downloads images from or uploads images to a remote registry
docker start/stop	Starts or stops an existing container
docker top	Displays containerized process status

Networking

- Virtual Ethernet Device (veth)
  - Tunnel between network namespaces
  - Pairs

```
ip link add <p1-name> type veth peer name <p2-name>

ip link set <p2-name> netns <p2-namespace>
```

p1-name

p2-name

Setting up Apptainer/Singularity

- Why?
  - Unique Security
    - Untrusted users
    - Untrusted containers
    - Runs as user account
  - Less isolation
    - Focuses on portability and integration
    - By default it only isolated the mount points

https://sylabs.io/guides/3.7/user-guide/quick_start.html

Using singularity(1)

```
#Build a simple Ubuntu images
singularity build ubuntu.sif docker://ubuntu #Not terribly useful

#Sandbox
singularity build --sandbox ubuntu docker://ubuntu

#Customize
singularity shell --writable ubuntu/
apt update -y && apt install -y emacs

#Build into a portable .sif
singularity build ubuntu.sif ubuntu/
```

Singularity Features

- Encrypted containers
  - Encrypted filesystem
    - Keys or Passphrase
- GPU Support
- Persistent Overlays
  - Easy to transport modifications
  - immutable container image (default)
  - preserves environment
- Bind Mounting
  - Mount a directory from the host in a container
  - -B /src:/dst
- Services
  - singularity instance start <some.sif>

systemd-nspawn

- No frills containers, similar to chroot
  - Create your own network bridge
  - Configure your own chroot or raw image
  - Create nspawn config
  - Create systemd service file
  - machinectl(1) to manage lifecycle

```
apt install systemd-container
```

Getting started

```
brctl addbr br0

#chroot
debootstrap --arch=amd64 --include=systemd-container sid postfix

#.nspawn conf
/etc/systemd/nspawn/postfix.nspawn

#Run it
systemd-nspawn -D postfix -b -M c1
machinectl login # or machinectl shell
```