Introduction to Docker

Travis Cardwell

Tokyo Linux Users Group

2014-01-18 Technical Meeting

Presentation Motivation

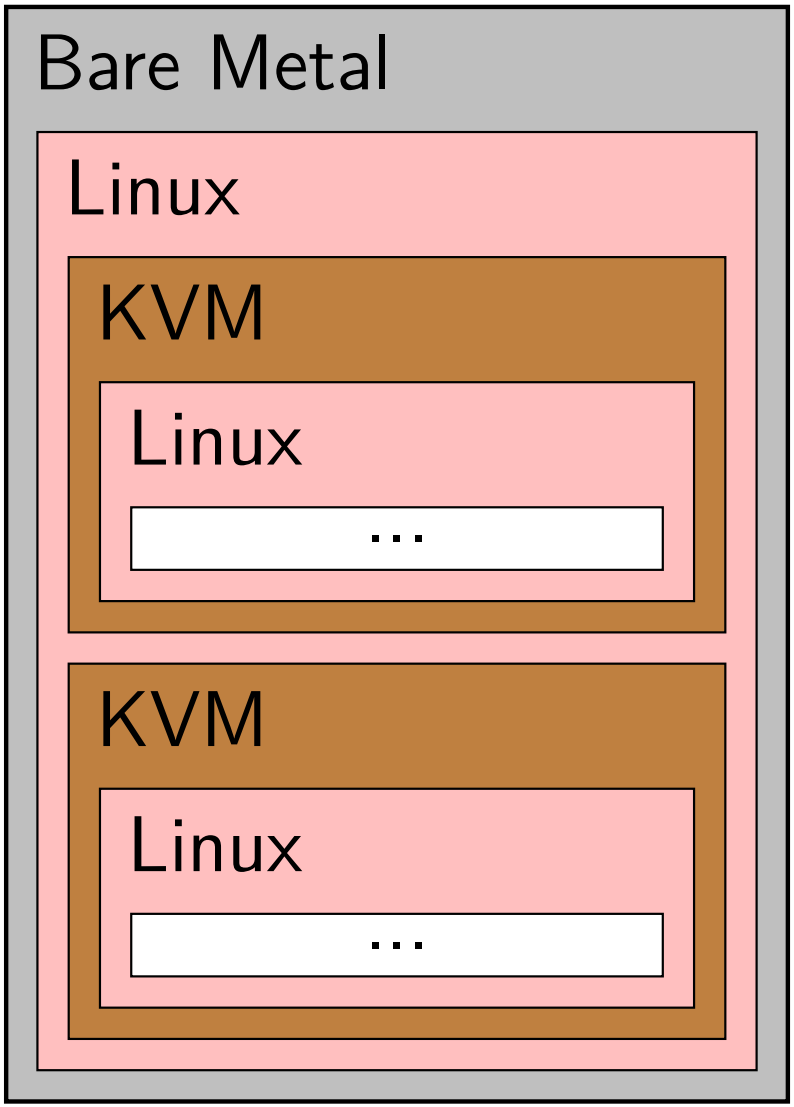
- OS-level virtualization is becoming accessible
- Docker makes it very easy to experiment with the technology
- If you have not already started learning about OS-level virtualization, now is the time!

Presentation Outline

1. The Big Picture
2. Underlying Technology
3. Docker
4. Use Cases
5. Demonstration
6. How To Get Started

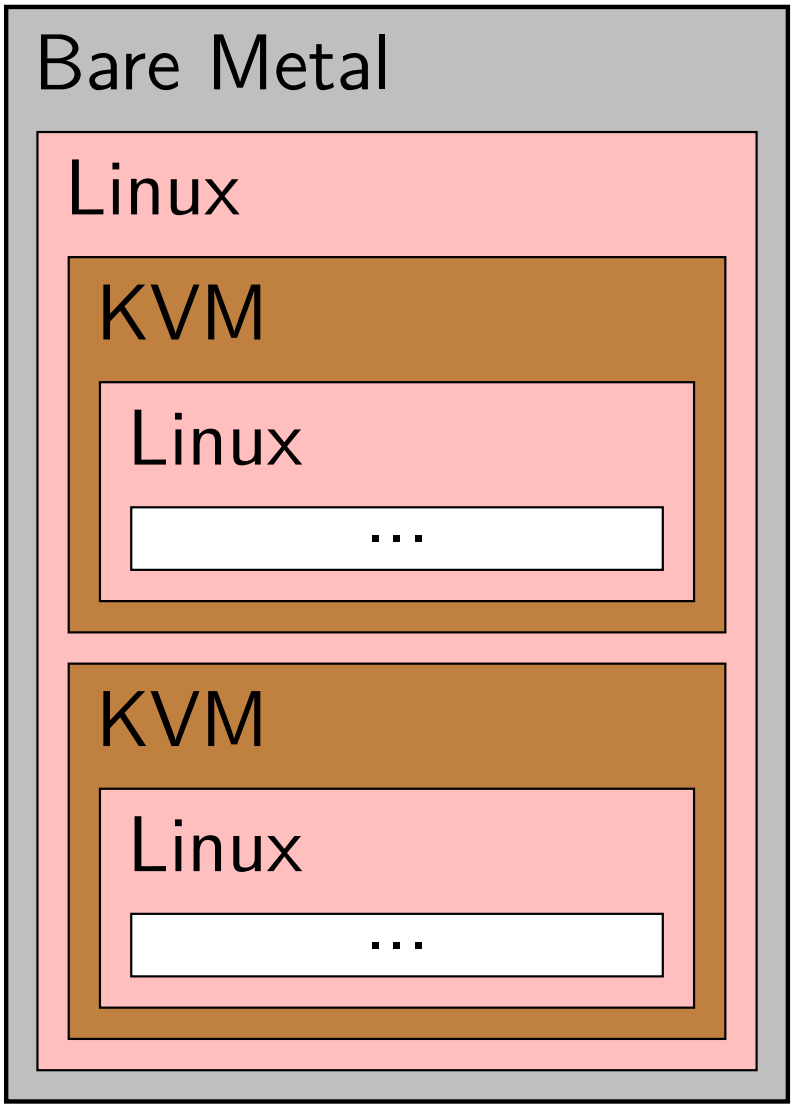
The Big Picture

Virtualization



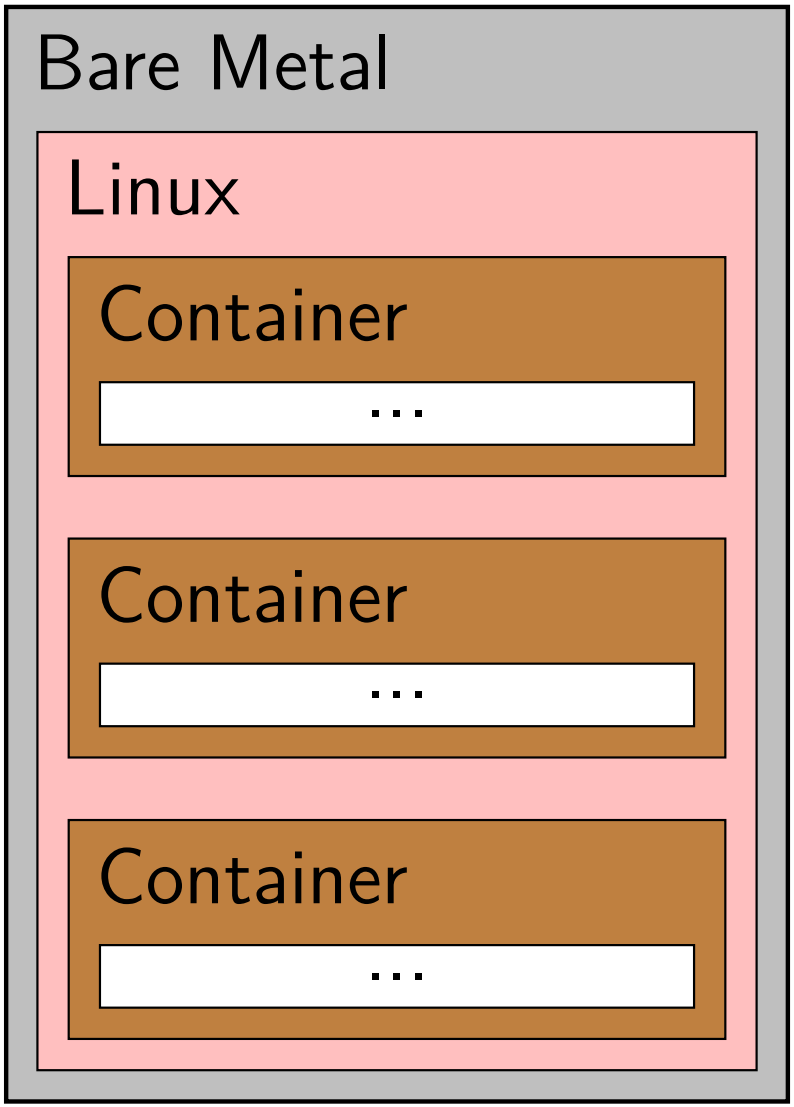
- Each virtual machine (VM) runs a full OS
- VMs require significant resources
- VMs take time to provision and boot

Virtualization



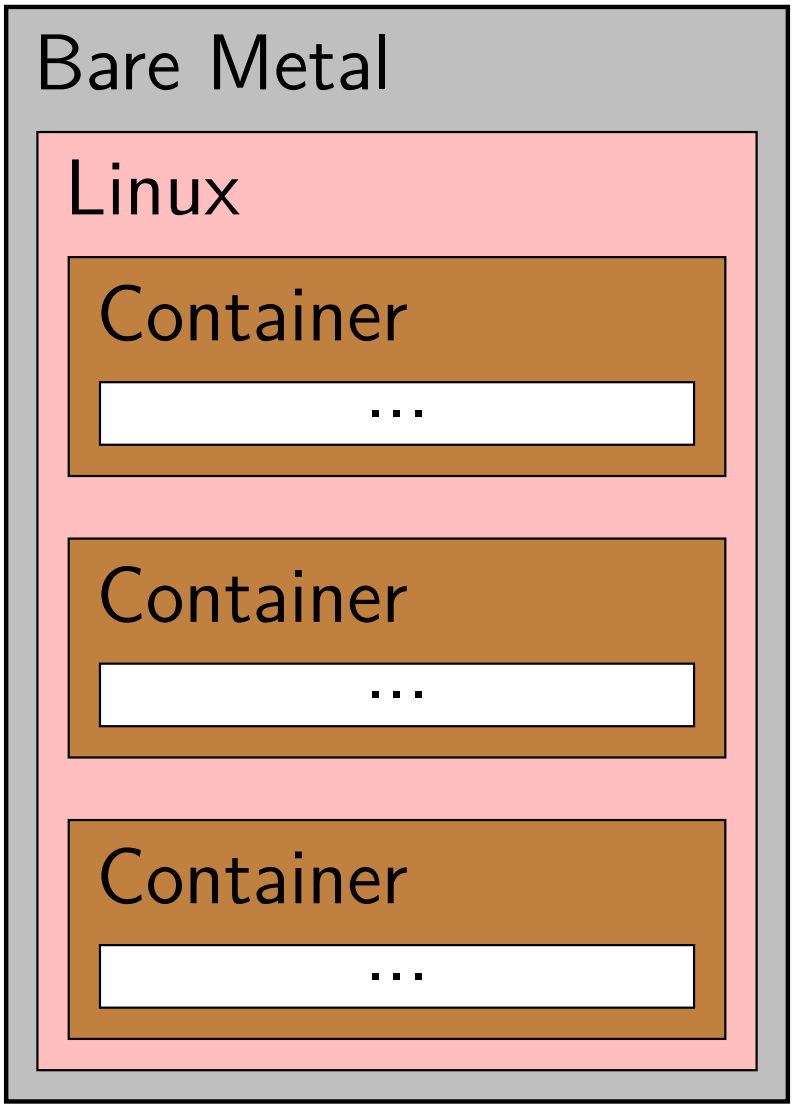
- 1967 first demo @IBM
- 1997 Virtual PC
- 1999 VMware
- 2003 Xen
  QEMU
- 2007 KVM
  VirtualBox

OS-Level Virtualization



- Containers share the host kernel
- Filesystem, network, etc. are virtualized
- Requires fewer resources
- A guest OS does not have to boot $\rightarrow$ starts fast

OS-Level Virtualization



- 1982 chroot
- 1998 FreeBSD jails
- 2001 Linux VServer
  Virtuozzo
- 2005 OpenVZ
  Solaris Containers
- 2007 AIX WPARS
  HP-UX Containers
- 2008 LXC

chroot

```
# export MY_CHROOT=/tmp/sid
# mkdir $MY_CHROOT
# debootstrap sid $MY_CHROOT \
  http://ftp.jp.debian.org/debian/
# mount proc $MY_CHROOT/proc -t proc
# mount sysfs $MY_CHROOT/sys -t sysfs
# chroot $MY_CHROOT /bin/bash
```

Underlying Technology

Kernel Namespaces

- Namespaces are used for isolation of:
  - filesystem - like `chroot` but more secure
  - UTS (host and domain names)
  - IPC (interprocess communication resources)
  - PIDs (process ID number space)
  - network stack (devices, addresses, routing, ports, etc.)
  - users (user and group IDs)

Kernel Control Groups

- cgroups partition sets of tasks into hierarchical groups
- Allows control over system resources:
  - resource limits (CPU, memory)
  - bandwidth limits (block I/O)
  - prioritization
  - access control (devices)
- Provides accounting/metrics
- Allows management of tasks:
  - suspend/resume

LXC (LinuX Containers)

- Userspace interface for kernel containment features

```
lxc-create -t ubuntu -n p1
lxc-start -n p1 -d
lxc-ls
lxc-stop -n p1
```

aufs

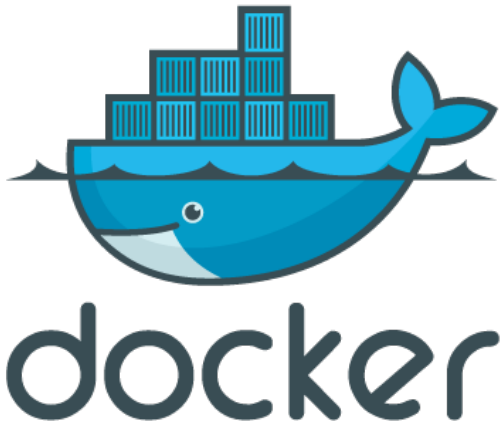
- Implements a *union mount*
- Overlays filesystems, creating a unified hierarchy
- Smaller size (diffs) allow for faster deployment

```
# cd /tmp
# mkdir aufs-{orig,diff,mount}
# debootstrap sid aufs-orig \
  http://ftp.jp.debian.org/debian/
# mount -t aufs \
  -o br=/tmp/aufs-diff:/tmp/aufs-orig \
  none /tmp/aufs-mount
```

Docker

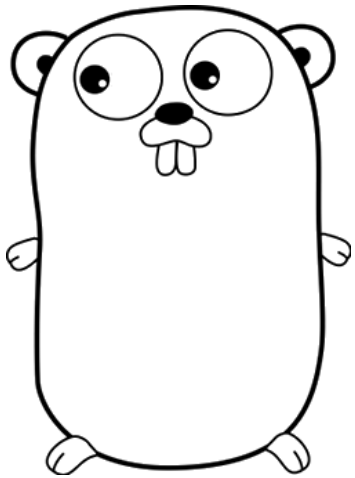
Docker

- Created by dotCloud (now Docker, Inc.), a Platform-as-a-Service company
- Created to automate the deployment of any application
- Open source, on GitHub, active community
- License: Apache 2.0



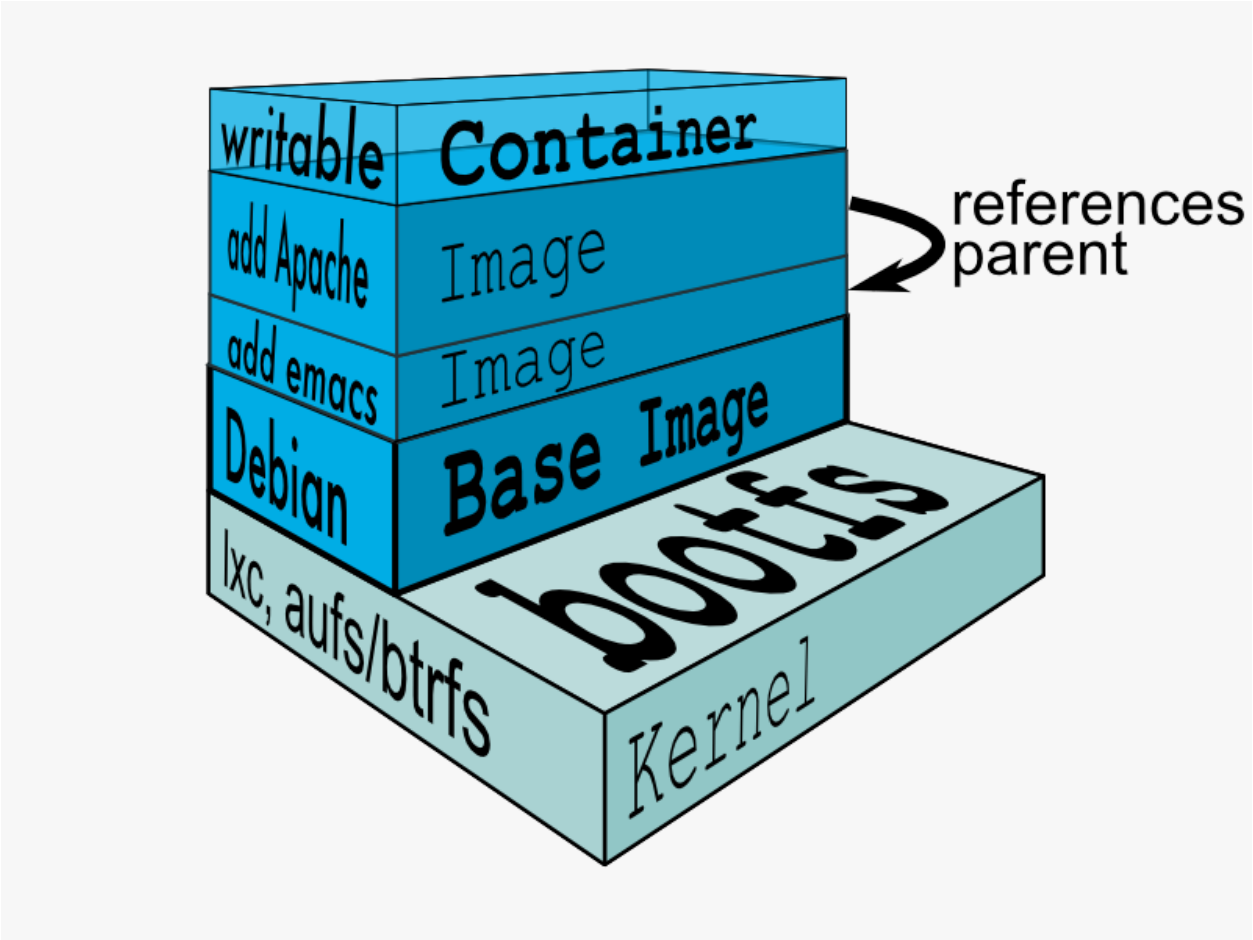
Feature: Compiled

- Docker is written in Go
- The executable is statically compiled



Feature: Layers

- The filesystem is layered using `aufs`
- Changes are committed, similar to `git` commits



Feature: Dockerfiles

- Configuration files that define how to build containers from images
- Use configuration tools, build tools, packages, etc.

```
FROM ubuntu
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y build-essential
```

Feature: Registry

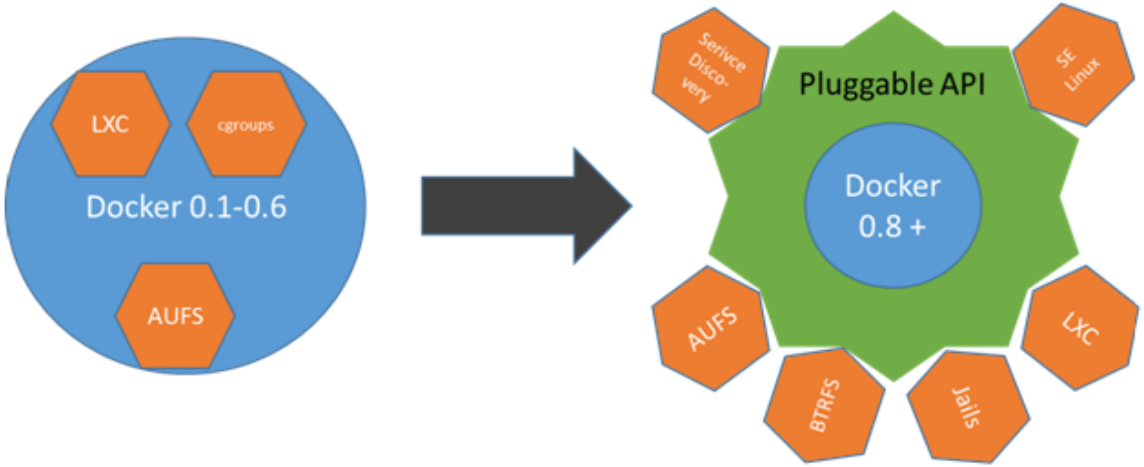
- A server that stores *repositories*
- Provides an API for uploading/downloading them
- There is a public registry called *the index*
- Open source, so you can host your own

```
# docker search ghc
# docker pull afriel/ghc-head

# docker login localhost:8080
# docker push tcard/gitit
```

Features Coming Soon

- Container wiring and service discovery
- Plugin API
- Broader kernel support
- Cross-architecture support



Development Status

- Development is moving quite quickly
- A production ready version is coming soon

0.1.0	2013-03-23	8	31 days
0.2.0	2013-04-23	2	13 days
0.3.0	2013-05-06	4	28 days
0.4.0	2013-06-03	8	44 days
0.5.0	2013-07-17	3	36 days
0.6.0	2013-08-22	7+7	95 days
0.7.0	2013-11-25	(6)	(54 days)

Usage Status

- Currently requires `x86_64`
- Currently requires Linux 3.8 or higher
- Currently not production ready
- Containers are not considered secure
  - Advice: Avoid root access in containers
  - Advice: Use SELinux if you need more security

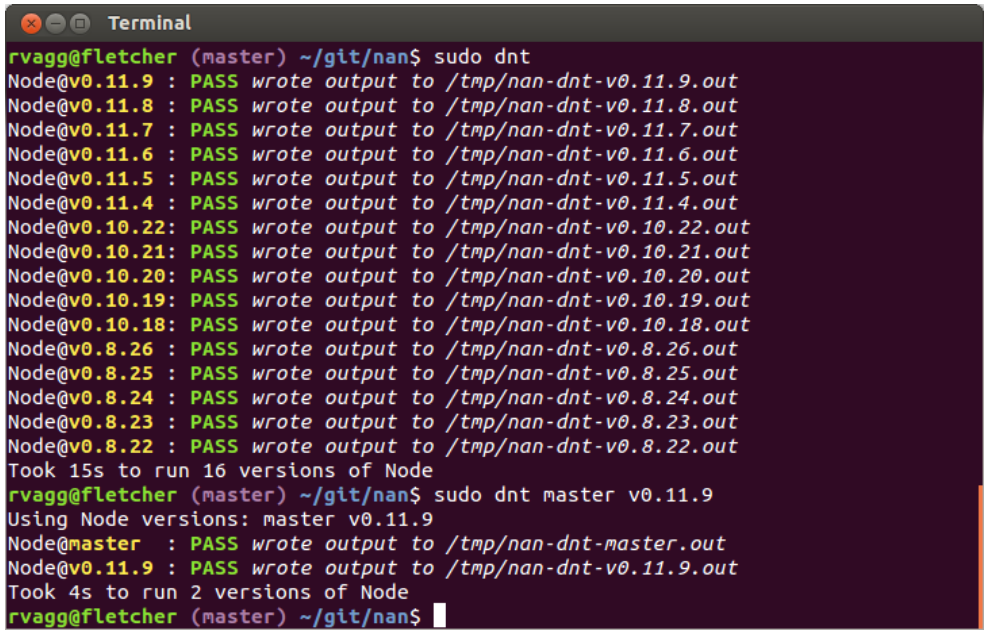
Use Cases

Deployment

- OS is included, so there are fewer parts to break
- Same way for development, staging, and production
- Can have fast transfer and boot times
- Scale applications and services
- Examples:
  - *CoreOS* is a distro for distributed platforms
  - *Flynn* is an open source Platform-as-a-Service
  - *Dokku* is a mini-Heroku in 100 lines of BASH

Test Automation

- Test across different distributions and library versions
- Perform fast unit and integration testing
- Examples:
  - *DNT* tests code against multiple versions of Node.js simultaneously
  - *NodeChecker* is a website that tests *all* NPM modules



Isolation

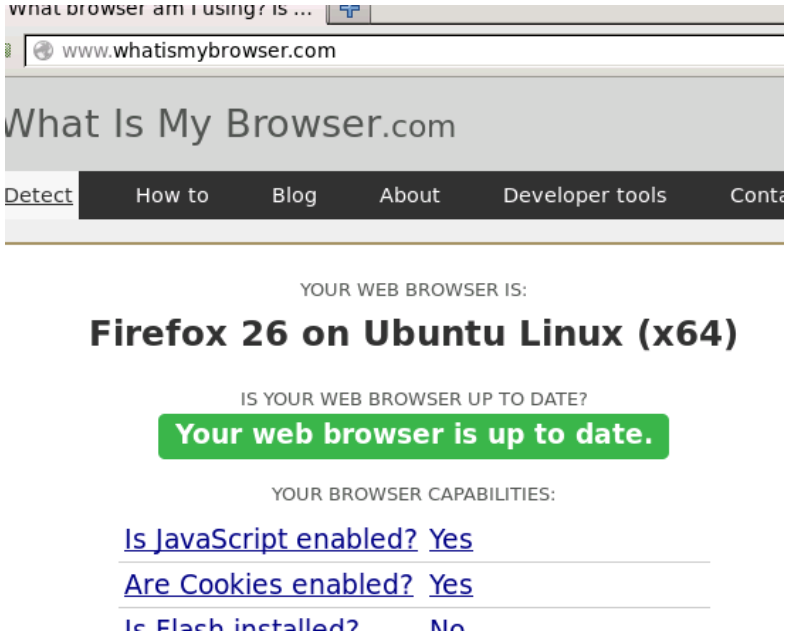
- Run some services on battle-tested RHEL and others on bleeding-edge Arch
- Sandbox web applications; example:
  - *JiffyLab* is a Python/Unix web-based teaching environment
- Sandbox local applications; example:
  - Run Mozilla Firefox in an ephemeral container

Lightweight Virtualization

- Launch virtualized environments quickly
- Reduce resource requirements
- Use Xpra ("`screen` for X") to manage sessions

Share Builds

- Provide quick access to difficult builds
- Provide easy access to new users
- Examples:
  - `ghc-head` repository provides latest builds of GHC
  - `docker-selenium-firefox-chrome` repository provides Selenium testing of specific browser buttons



Demonstration

Containers

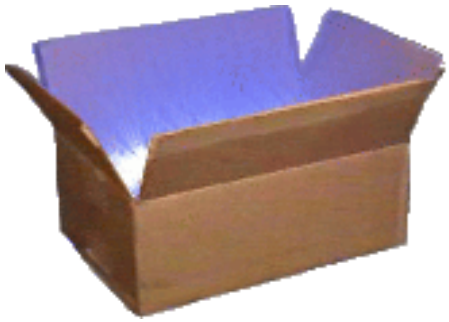
- Separate process space
- Separate filesystems
- Separate networking

Committing

- The service listens to virtual port 8888
- It can be routed that to any port
- The service is run in detached mode

BusyBox

- The image is small: `<5MB`
- Run with the `-rm` option to automatically remove the container



IPython Service

- The service listens to virtual port 8888
- It can be routed that to any port
- Bind mount a directory for data

Sandboxed Firefox

- Bind mount `/tmp/.X11-unix` (X11 unix socket)
- Bind mount `/dev/snd`
- Give access to `c 116:*` (ALSA)
- Pass the `$DISPLAY` environment variable
- Choose what to do with data:
  - Ephemeral: delete on close
  - Data on host: bind mount a host directory
  - Data container: use a volume

How To Get Started

Linux Beginners: Vagrant

1. Install VirtualBox
2. Install Vagrant
3. Install git
4. Deploy a Docker VM:

```
git clone https://github.com/dotcloud/docker.git
cd docker
vagrant up
```

5. Connect to the VM: `vagrant ssh`
6. Run Docker in the VM: `sudo docker`

Linux Veterans: Debian

- Works painlessly on Jessie (testing)
- Dependencies are listed in `/hack/PACKAGERS.md`
- You will need to:
  - Add some parameters to `/etc/default/grub`
  - Add a `cgroup` mount to `/etc/fstab`
  - Enable forwarding in `/etc/sysctl.conf`
- All output of `lxc-checkconfig` should be green
- To install Docker:
  - `wget` the binary from the Docker website
  - `wget` the SysVinit script from the GitHub repo

Tokyo Docker Meetup

- `http://www.meetup.com/Docker-Tokyo/`
- First meeting has not been scheduled yet