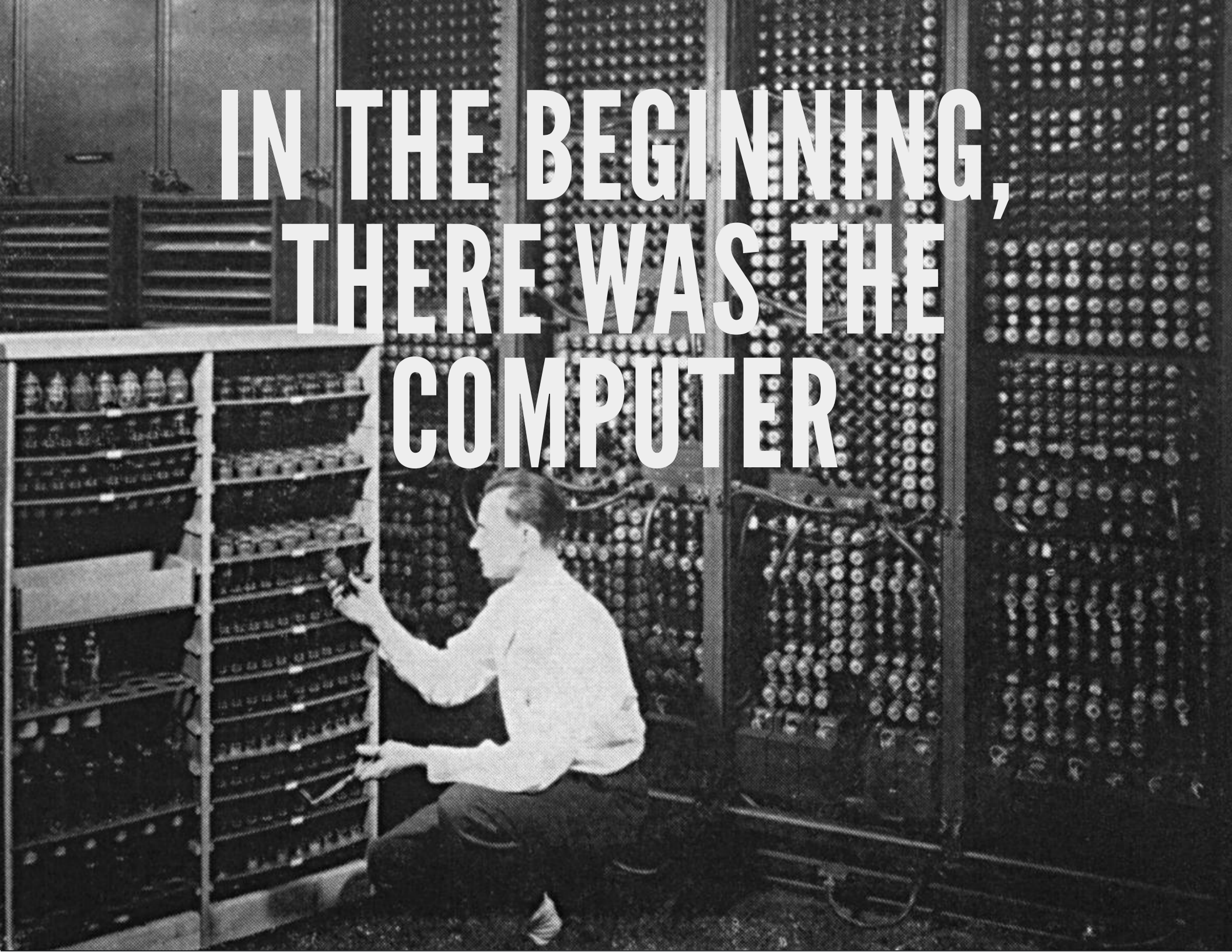
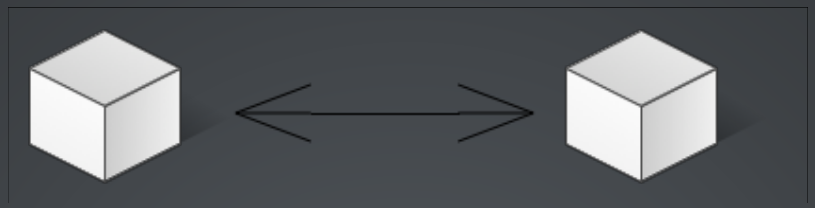
THE REVOLUTION WILL BE

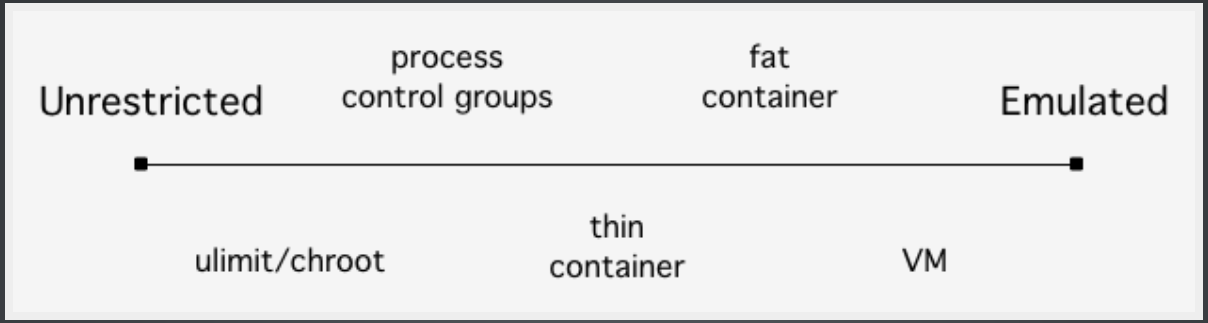

IN THE BEGINNING,
THERE WAS THE
COMPUTER

WHAT DO WE NEED TO SHARE?

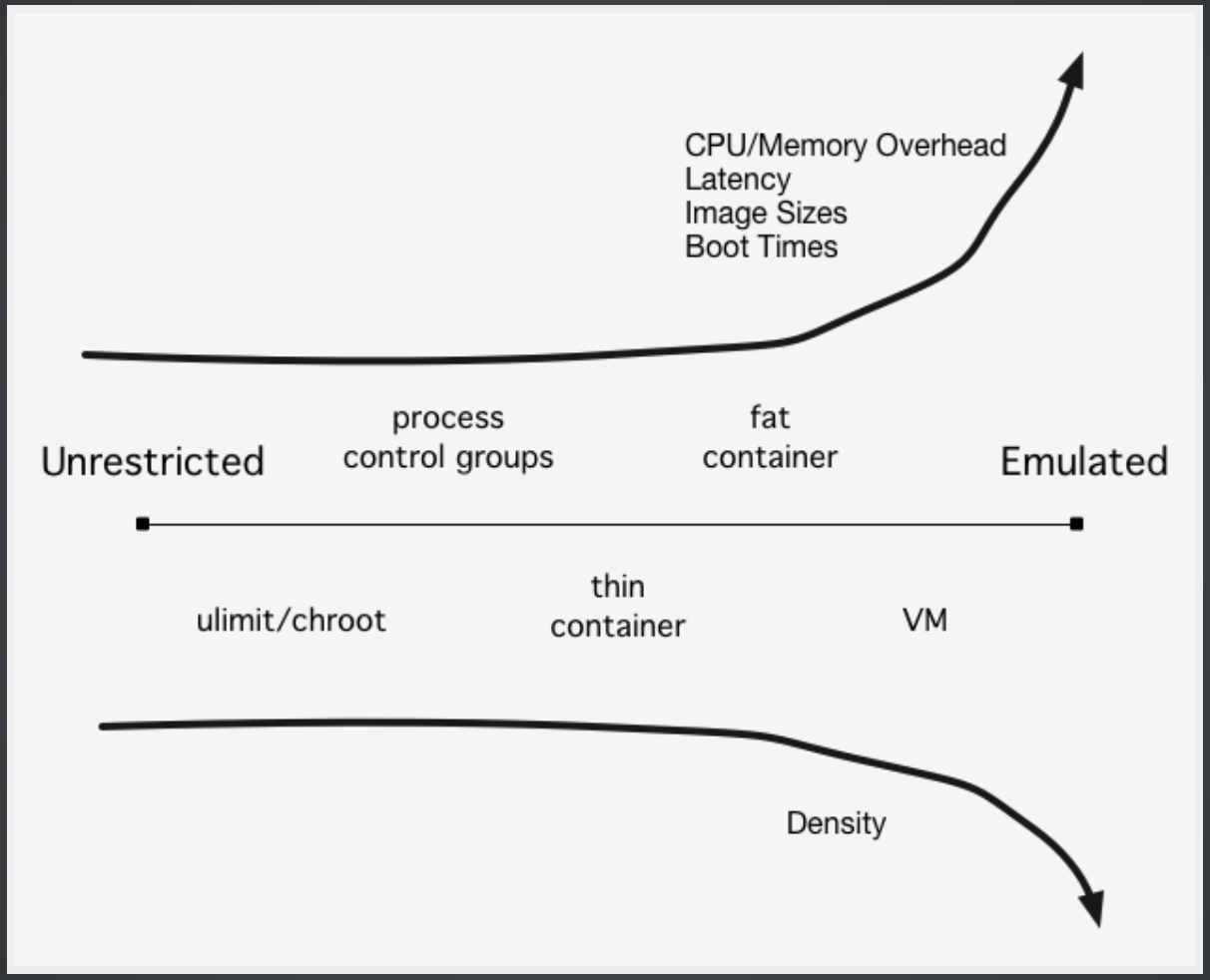


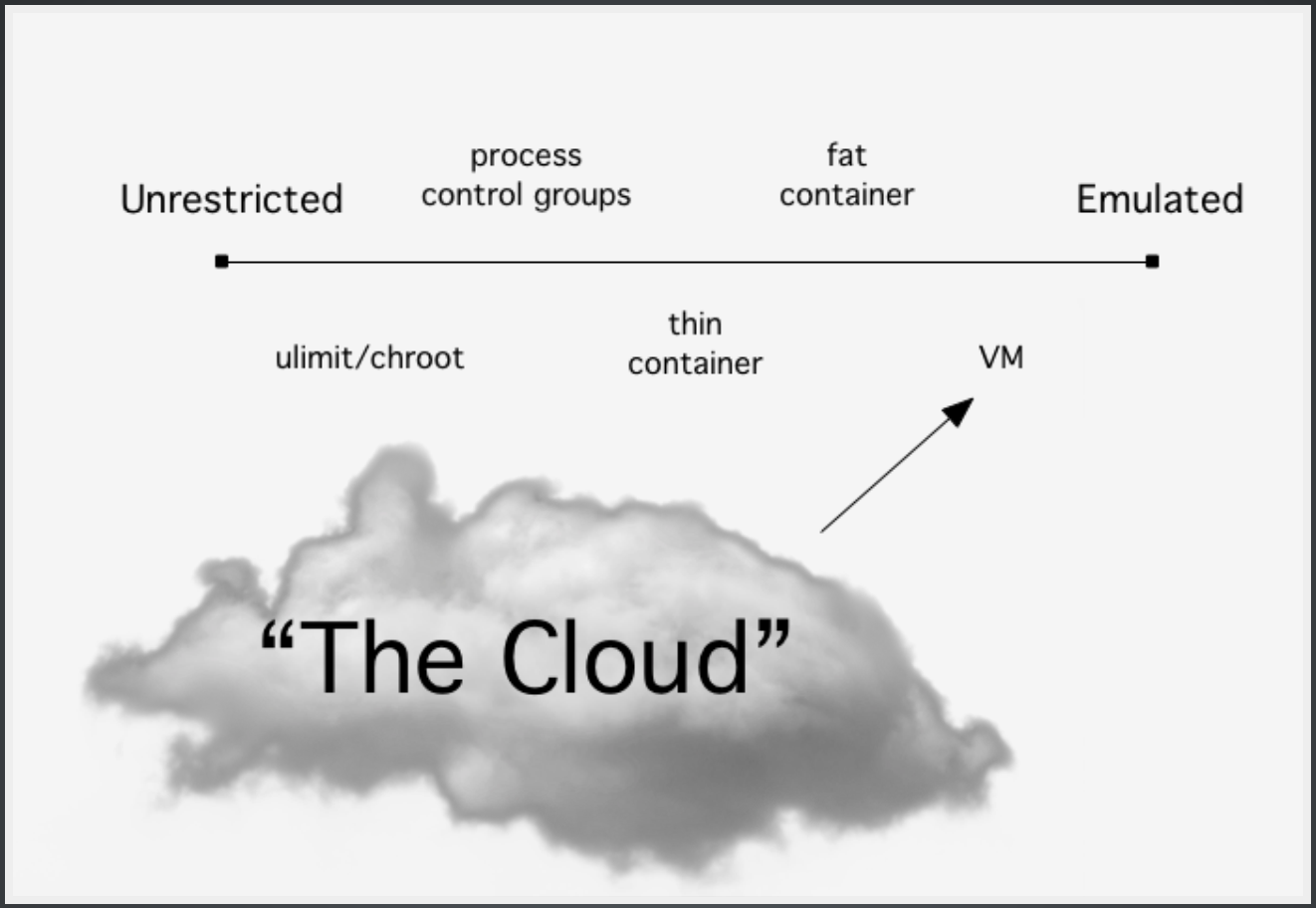
1. **Resources:** We all get enough (but not too much)
2. **Security/Privacy:** We stay out of each other's stuff
3. **Dependencies:** Apps & Libs I need

MANY STRATEGIES

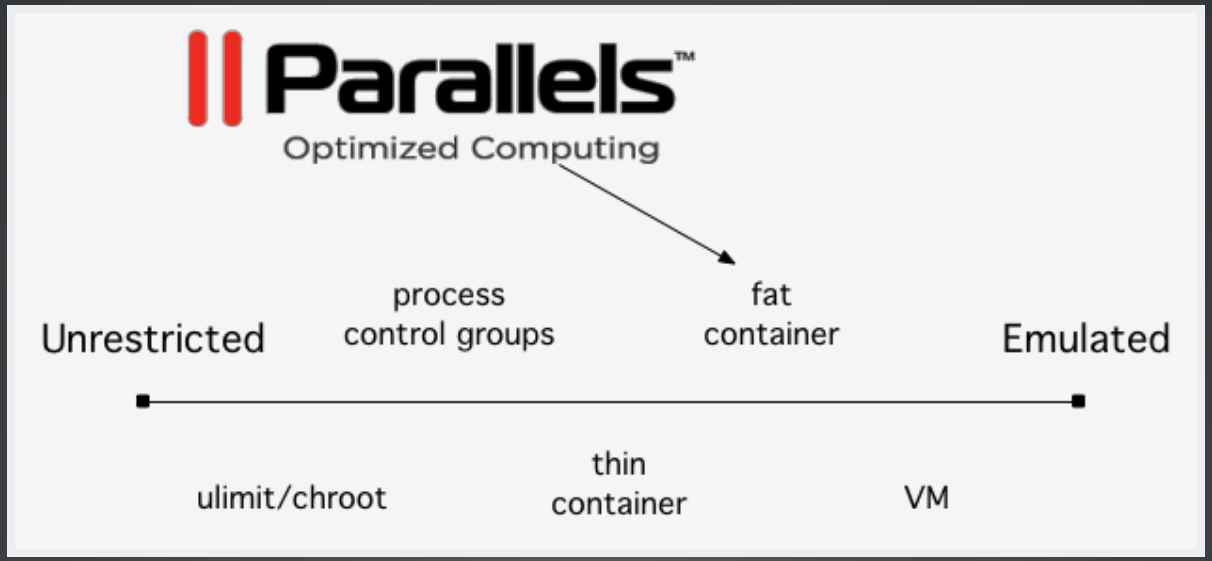


IMPACTS


CPU/Memory Overhead
Latency
Image Sizes
Boot Times
Unrestricted
process
control groups
fat
container
Emulated
ulimit/chroot
thin
container
VM
Density


Unrestricted
process
control groups
fat
container
Emulated
ulimit/chroot
thin
container
VM
“The Cloud”

VIRTUOZZO/OPENVZ ('THICK CONTAINERS')



DENSITY/EFFICIENCY: WE SHOULD CARE.

1. Profit
   - Extra servers, networking gear
   - Power Costs
2. Performance
   - See (1)
3. Social and Environmental Responsibility
   - Power usage = C02
   - Manufature = Precious Metals, Toxic Waste
   - Also, can be terrible conditions.

ON POWER

Datacenter Power Still Largely Driven by Coal

by Kevin Fogarty | August 19, 2013

0

9 Repost This Article

Despite green IT initiatives, only a fraction of datacenter power comes from renewable resources.

Studies showing the rapid consolidation of business and consumer IT into energy-efficient power centers vastly underestimate the amount of electricity used to run everything from smartphones to mainframes—and the vast majority of that power that comes from generators driven by coal.

Slashdot Poll

I wish my car could...

- Fly
- Navigate underwater
- Go subterranean
- Fly through space
- Shrink and drive through tiny things
- Expand and crush houses
- Not make that annoying noise

Vote

Read the **387** comments

Voted on **17339** times.

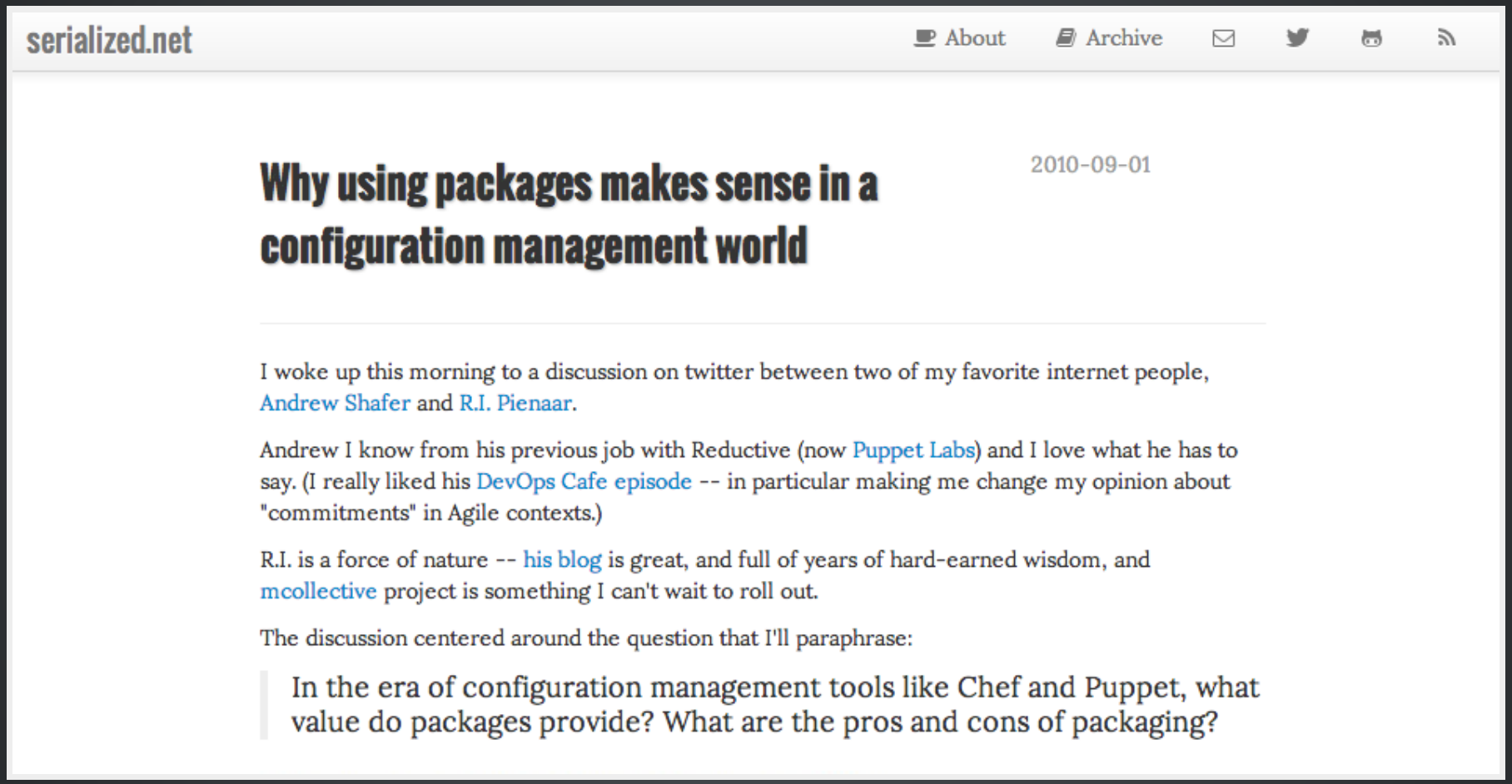
Latest Data Center Jobs

TL;DR TODAY'S CLOUD MIGHT NEED IMPROVEMENT

APPS & SERVER LAYER

How do we ship our applications?

I ♥ PACKAGES



serialized.net

About Archive

Why using packages makes sense in a configuration management world

2010-09-01

I woke up this morning to a discussion on twitter between two of my favorite internet people, Andrew Shafer and R.I. Pienaar.

Andrew I know from his previous job with Reductive (now Puppet Labs) and I love what he has to say. (I really liked his DevOps Cafe episode -- in particular making me change my opinion about "commitments" in Agile contexts.)

R.I. is a force of nature -- his blog is great, and full of years of hard-earned wisdom, and mcollective project is something I can't wait to roll out.

The discussion centered around the question that I'll paraphrase:

> In the era of configuration management tools like Chef and Puppet, what value do packages provide? What are the pros and cons of packaging?

A HYPOTHETICAL PYTHON APP

1. We need python 3.2. Damn, distro doesn't provide it.
2. Need a library that clashes with the system's version. Damn.
3. We package the whole mess, using `virtualenv` and some glue.
4. Oh, and we need redis 2.6 for Lua support. Distro has 2.4. Triple damn.
5. Oh, and the QA team has standardized on Centos.
6. Oh, BTW, the new search service is a node.js app, that's cool, right?

... 3 MONTHS LATER

7. There's a new openssl package out, very urgent security issue!

- Huh.
- Can I push it?
- Which apps use it?
- Which apps will break?

Oh, and we're also stuck if one of those apps gets exploited, or tries to use all the resources on the box...

NETFLIX MODEL

- Bake an entire server image
- Test as a unit
- Use for deploy and rollback

Progress! But, 'the computer' is a very chunky unit of abstraction.

THE TRICKY BITS

- Lots of sysadmin surface area
- Additional Daemons to run & manage
- Large images to juggle. 1 byte change? 0.5 GB image.
- Boot Times. 2 seconds - 15 mins (OpenStack nodepool)
- Density isn't great.

WHAT'S THE BETTER (WELL, MORE EFFICIENT) COMMUTER CAR?

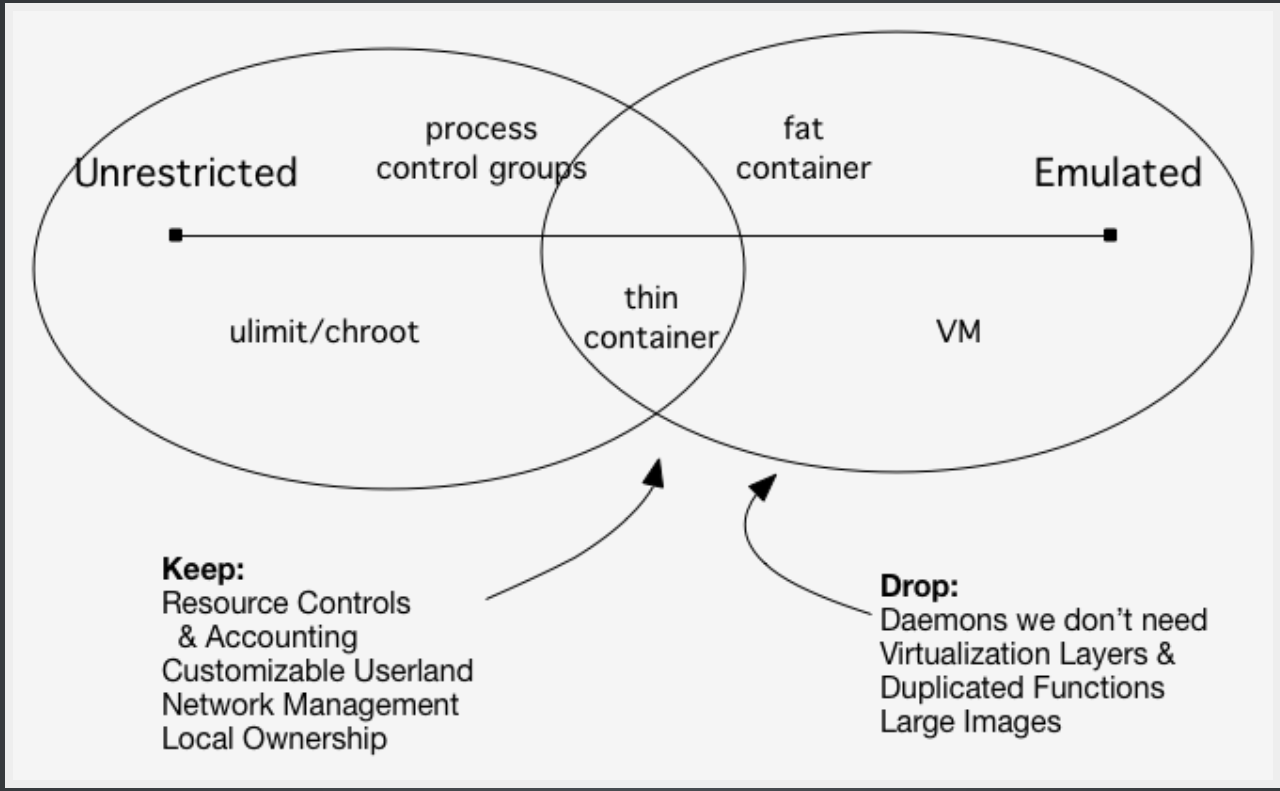
WHAT ABOUT PAAS?

Cool! "Process Virtualization Not Server Virtualization."

Sort of.

- Local dev, QA, staging can be tough to clone their versions of.
- Hard upper bound on troubleshooting
- Generally fixed userland
- Lots of capability limits (daemons, storage, networking, etc)

DREAM WITH ME

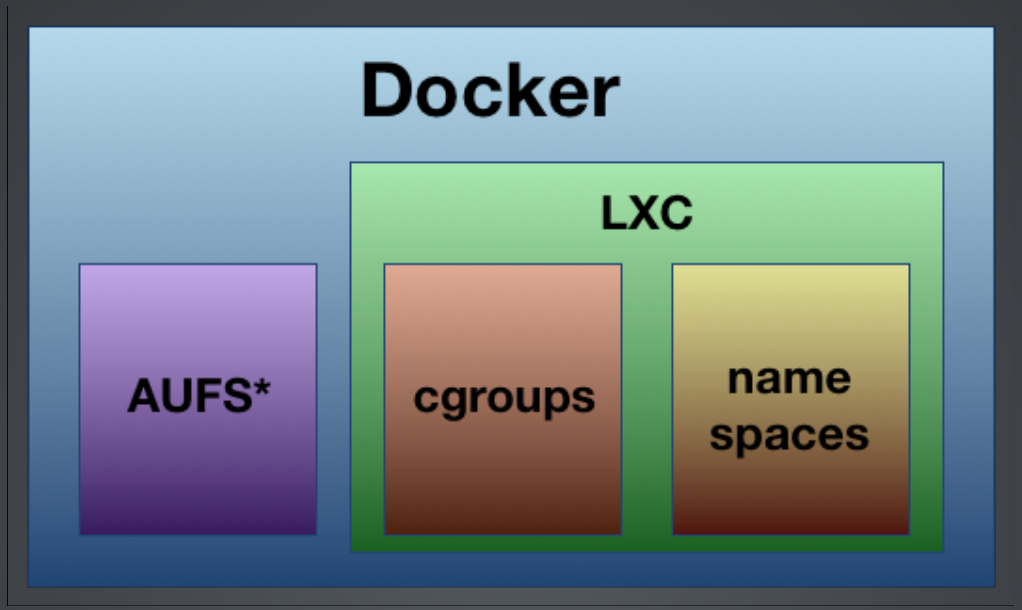



docker

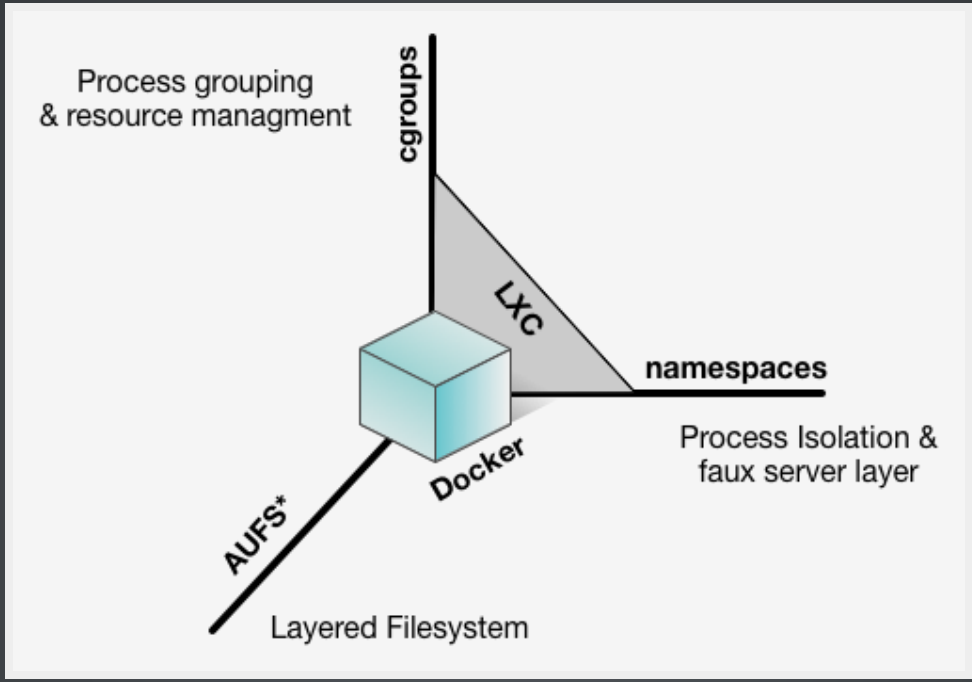
WHAT IS DOCKER?

“Docker is an open-source project to easily create lightweight, portable, self-sufficient containers from any application.”

WHAT IS DOCKER?



WHAT IS DOCKER?



redhat

THANKS REDHAT!

PORTABILITY & DENSITY

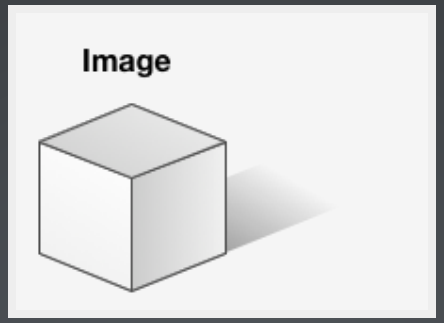
- Extremely Lightweight
- Build containers manually or with many flavors of repeatable process
- Run those on local, dev, QA, stage, prod, cloud, PaaS...

WHO IS DOCKER?

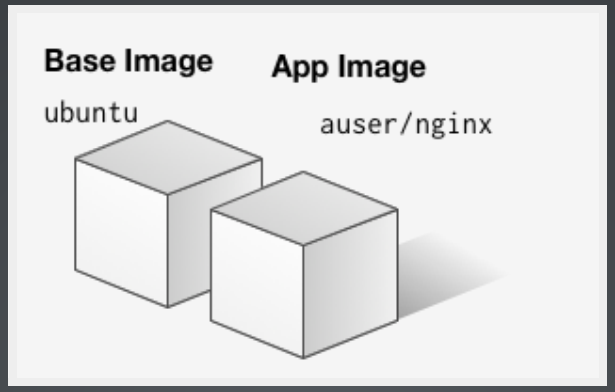
Tons of people!

- Open Sourced by dotCloud, part of a family of other great tools. (hipache)
- < 1 year old, but hundreds of committers, and it's blogged about constantly
- Quick Survey?

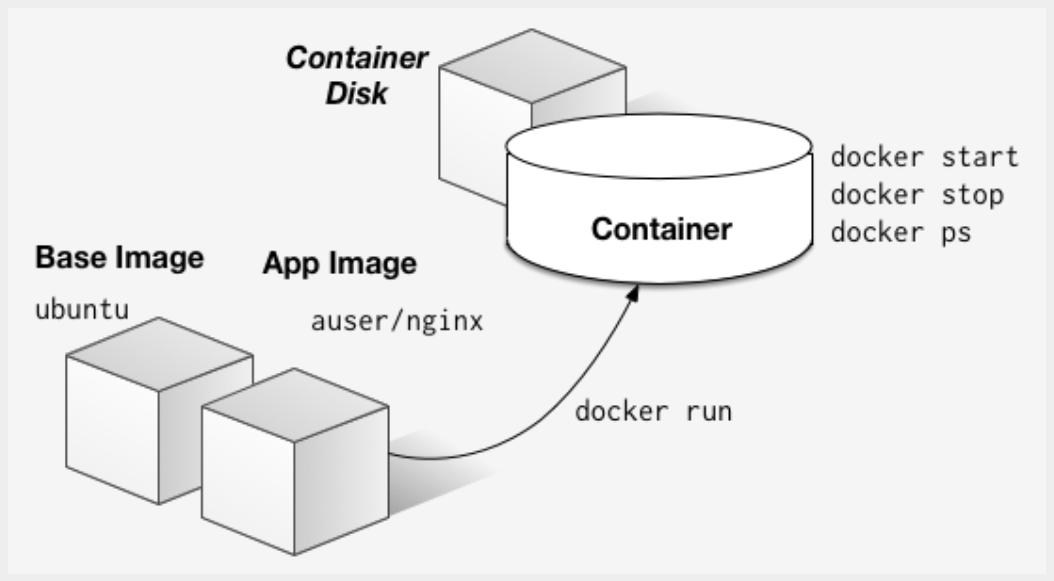
DOCKER LIFECYCLE



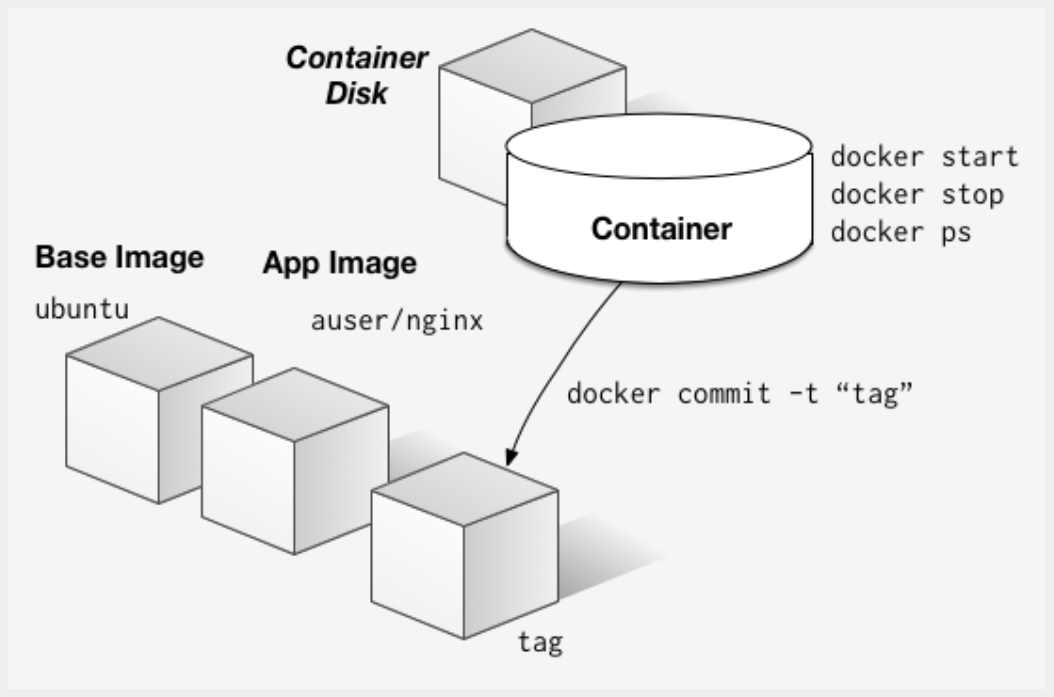
DOCKER LIFECYCLE



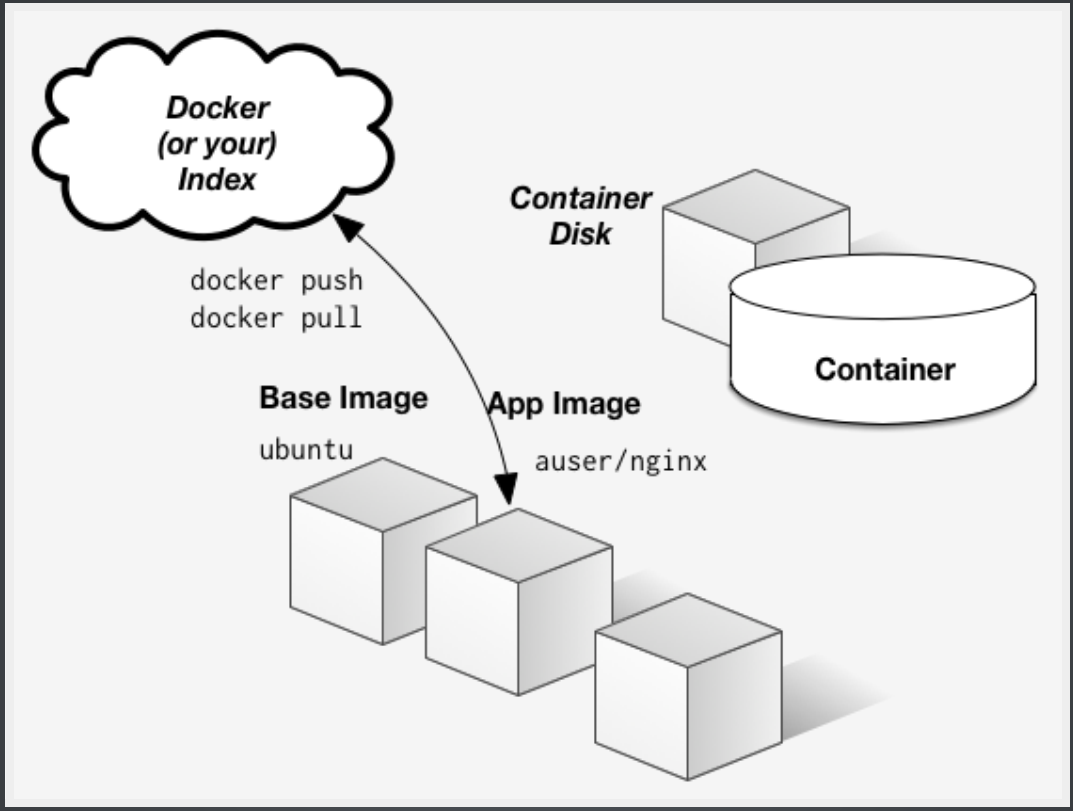
DOCKER LIFECYCLE



DOCKER LIFECYCLE



DOCKER LIFECYCLE



DOCKER CLI BASICS

$ sudo docker

- **ps**: List *containers*
- **images**: List *images*
- **run**: Create container from image
- **start/stop**: Stop a container
- **build**: Create an image from a Dockerfile
- **inspect**: Inspect an image
- **logs**: Review STD(OUT|ERR) from a container

HOW DOES IT HELP?

- **Density!**
  - Great (native) out of the box
  - Add 'idle mode' for extra BLAMMO*
  - Can also limit RAM/CPU/blk io/net
- Private and Public Repos
- Image Stacking = Minimal Upgrade Payloads
- **Consistency** @ every stage
- Minimum Viable Moving Parts

* By BLAMMO I mean that docker containers start fast enough that for some workloads you can afford to create or start them on demand, which makes your effective density far, far higher.

LESS MOVING PARTS FTW

“We've switched to Docker from Vagrant/Virtualbox to isolate test runs on our CI server. We have seen major speed improvements in some cases, but mostly we are seeing much more stable runs with fewer tests that just randomly fail on CI, or cases where the VM would stop responding completely or fail to launch.”

-- Dan Ivovich


LET'S GET OUR HANDS
DIRTY

ANY CONTAINERS RUNNING?

```
vagrant@precise64:~/docker$ sudo docker ps
ID                  IMAGE               COMMAND             CREATED
```

OK, START ONE BASED ON THE UBUNTU IMAGE.

```
vagrant@precise64:~/docker$ sudo docker run -i -t ubuntu:12.10 bash
root@7f407e484d62:/# apt-get update
# lots of apt-get stuff
root@7f407e484d62:/# apt-get install redis-server
# lots of apt-get stuff
root@7f407e484d62:/# exit
vagrant@precise64:~/docker$ sudo docker ps -a
ID                  IMAGE               COMMAND             CREATED
7f407e484d62        ubuntu:12.10        bash                56 seconds ago
```

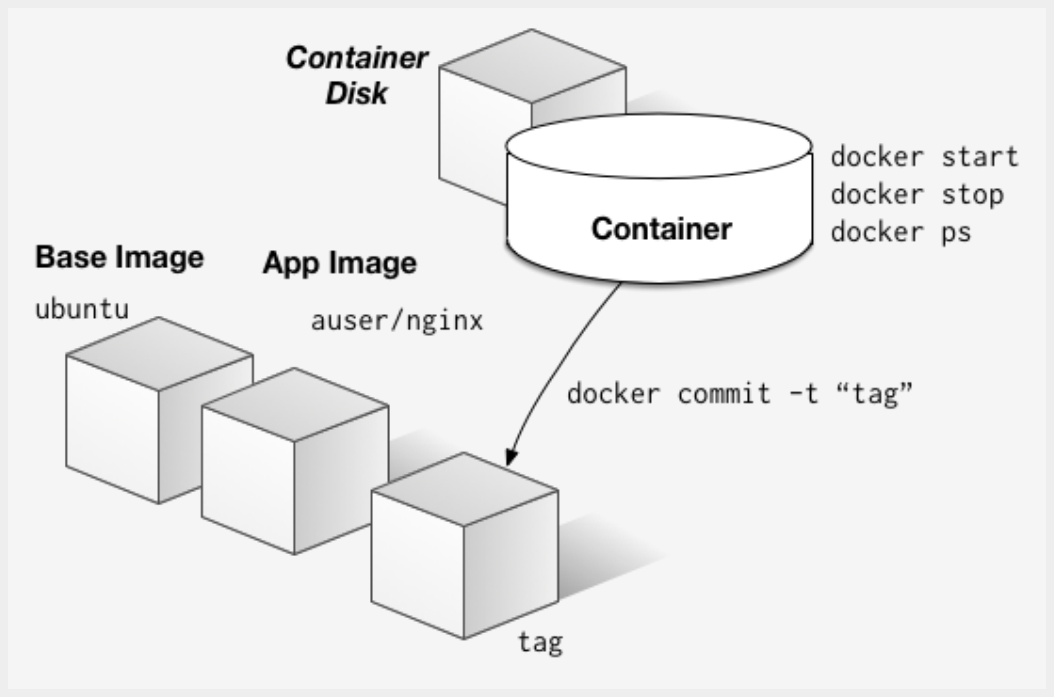
TAG IT!

```
vagrant@precise64:~/docker$ sudo docker commit -m "installed redis server" 7f
517800321f48
vagrant@precise64:~/docker$ sudo docker images
REPOSITORY          TAG                 ID                  CREATED
ubuntu              12.04               8dbd9e392a96        6 months ago
ubuntu              12.10               b750fe79269d        7 months ago
ubuntu              latest              8dbd9e392a96        6 months ago
ubuntu              precise             8dbd9e392a96        6 months ago
ubuntu              quantal             b750fe79269d        7 months ago
jbarratt/redis01    latest              517800321f48        56 seconds ago
```

BUT, WE MADE AN IMAGE, NOT CONTAINER.

```
vagrant@precise64:~/docker$ sudo docker ps
ID                  IMAGE               COMMAND             CREATED
```

DOCKER LIFECYCLE



TRY IT AGAIN, WITH MORE DAEMON!

```
vagrant@precise64:/var$ sudo docker run -p 6379 -d -i \
    -t jbarratt/redis01 /usr/bin/redis-server
8222ea69d3a3

vagrant@precise64:/var$ sudo docker ps
ID                  IMAGE                       COMMAND                 PORTS
8222ea69d3a3        jbarratt/redis01:latest     /usr/bin/redis-server   49153->

vagrant@precise64:/var$ telnet 172.17.42.1 49153
Connected to 172.17.42.1.
Escape character is '^]'.
    > rpush users josh
    > rpush users sam
    > rpush users susan
    > llen users
:3
    > lrange users 0 5
josh sam susan
```

WANT STDOUT?

```
vagrant@precise64:/var$ sudo docker attach 8222ea69d3a3
[1] 23 Oct 00:06:36 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[1] 23 Oct 00:06:36 - 0 clients connected (0 slaves), 791000 bytes in use
...
```

WHAT'S GOING ON?

```
vagrant@precise64:/var$ ps xwf
495  ?        Ss     0:00 /bin/sh -e -c /usr/bin/docker -d /bin/sh
496  ?        Sl     0:24  \_ /usr/bin/docker -d
3115 pts/1    Ss+    0:00      \_ lxc-start -n 8222ea(...) -f /var/lib/docker
3120 pts/1    Sl+    0:48          \_ /usr/bin/redis-server
```

CONTAINER-EYE VIEW... 1 != 3120

```
vagrant@precise64:/var$ sudo lxc-attach -n 8222ea(...)
fc51fae86a46603bc3762 /bin/bash
root@8222ea69d3a3:/var# ps auxwwf
USER       PID TTY      STAT START   TIME COMMAND
root         1 ?        Sl+  Oct22   0:52 /usr/bin/redis-server
```

NAMESPACES ARE MAGIC

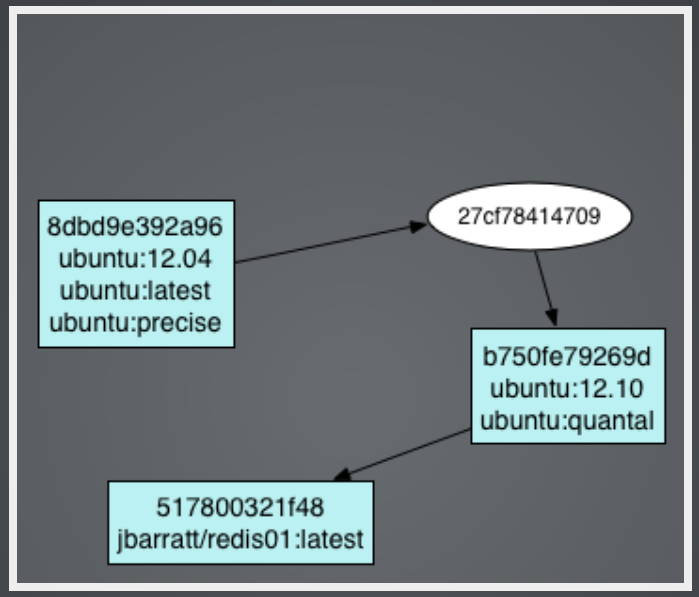
- pid
- net
- ipc
- mnt
- uts (hostname)
- user (unpriv. control)

UNION FILESYSTEMS ARE MAGIC

```
root@precise64:/var/lib/docker/containers# ls
15f3c09e26eef579dfa53035fbdde536f07fb124c473b146df73fd4c7f2be33c
7a3dad48db07177ffdd6ae77104585a3a0f43d88400f4c3affc258685c835dd0
7f407e484d62e50ce05c14c83d47b136d590f37054084a44f3370a721bb0116a
8222ea69d3a344edd2a734423c53ef17dc4de194bb4fc51fae86a46603bc3762
root@precise64:/var/lib/docker/containers# ls 8222(...)
bb4fc51fae86a46603bc3762/rw/
dump.rdb

root@precise64:/var/lib/docker/containers# docker diff 8222
A /dump.rdb

vagrant@precise64:/var$ sudo lxc-attach -n 8222(...) /bin/bash
root@8222ea69d3a3:/var# mkdir /foo && touch /foo/bar
root@8222ea69d3a3:/var# exit

vagrant@precise64:~# docker diff 8222
A /dump.rdb
A /foo
A /foo/bar
```

DOCKER TRACKS THE LINKAGES...

```
vagrant@precise64~$ sudo docker images -viz | dot -Tpng -o docker.png
```



CGROUPS ARE MAGIC

```
root@precise64:~# cat /sys/fs/cgroup/cpuacct/lxc/8222(...)/cpuacct.usage
63976740039
root@precise64:~# cat /sys/fs/cgroup/cpuacct/lxc/8222(...)/cpuacct.usage
63986569761
root@precise64:~# cat /sys/fs/cgroup/memory/lxc/8222(...)/memory.usage_in_byt
2433024
root@precise64:~# cat /sys/fs/cgroup/memory/lxc/8222(...)/memory.limit_in_byt
9223372036854775807
```

BUT IS IT FAST?

```
root@precise64:~# docker stop 8222
root@precise64:~# time docker start 8222
    real    0m0.150s

root@precise64:~# time service redis-server start
Starting redis-server: redis-server.
    real    0m0.165s

root@precise64:~# time docker run -p 6379 -d -i -t jbarratt/redis01 /usr/bin/
    real    0m0.147s
```

THAT SUCKS. YOU BUILT THAT BY HAND.

- Yep. The Devops Gods are angry. REPEATABLE BUILDS! they say.
- The cool part? You can automate it however you want.

ENTER DOCKERFILE

```
# redis image
# VERSION 0.1

FROM ubuntu:12.10
MAINTAINER Joshua Barratt jbarratt@serialized.net

RUN apt-get -qq update
RUN apt-get install -y redis-server

EXPOSE 6379
CMD ["/usr/bin/redis-server"]
```

```
vagrant@precise64:/vagrant/redis$ sudo docker build -t jbarratt/redis-dockerf
Uploading context 10240 bytes
Step 1 : FROM ubuntu:12.10
 ---> b750fe79269d
Step 2 : MAINTAINER Joshua Barratt jbarratt@serialized.net
 ---> Using cache
 ---> 0501ea3da390
Step 3 : RUN apt-get -qq update
 ---> Using cache
 ---> 134cfdc556cb
Step 4 : RUN apt-get install -y redis-server
 ---> Using cache
 ---> 01ba2afd847f
Step 5 : EXPOSE 6379
 ---> Using cache
 ---> 9ac5730000e7
Step 6 : CMD ["/usr/bin/redis-server"]
 ---> Running in 713d6ab6d7d8
 ---> 60d75fbc2eac
Successfully built 60d75fbc2eac

vagrant@precise64:/vagrant/redis$ sudo docker run jbarratt/redis-dockerfile

vagrant@precise64:/vagrant/redis$ sudo docker ps
ID                  IMAGE                                 COMMAND
bbd790b7dfcd        jbarratt/redis-dockerfile:latest   /usr/bin/redis-server
```

LAYERING DOCKERFILES

```
FROM jbarratt/redis-dockerfile
MAINTAINER Joshua Barratt jbarratt@serialized.net

ADD fixtures.sh /tmp/fixtures.sh
RUN /tmp/fixtures.sh
```

```
#!/bin/bash

/usr/bin/redis-server &
for ((i=1; i <=100; i++))
do
    redis-cli rpush numbers $i
done
redis-cli shutdown
```

LAYERING DOCKERFILES

```
vagrant@precise64:/vagrant/redis-fixture$ sudo docker build -t jbarratt/redis
Uploading context 10240 bytes
Step 1 : FROM jbarratt/redis-dockerfile
 ---> 60d75fbc2eac
Step 2 : MAINTAINER Joshua Barratt jbarratt@serialized.net
 ---> Running in 0d8bf423b2bc
 ---> a5c3e1b51ceb
Step 3 : ADD fixtures.sh /tmp/fixtures.sh
 ---> c4170f7490e1
Step 4 : RUN /tmp/fixtures.sh
 ---> Running in 0ca9a42ca15d
[10] 23 Oct 05:21:32 - Client closed connection
[10] 23 Oct 05:21:32 - Accepted 127.0.0.1:54281
[10] 23 Oct 05:21:32 # User requested shutdown...
[10] 23 Oct 05:21:32 * Saving the final RDB snapshot before exiting.
[10] 23 Oct 05:21:32 * DB saved on disk
[10] 23 Oct 05:21:32 # Redis is now ready to exit, bye bye...
 ---> 835789acdc85
Successfully built 835789acdc85
```

```
vagrant@precise64:/vagrant/redis-fixture$ sudo docker run jbarratt/redis-fixt
[8] 23 Oct 05:28:30 * The server is now ready to accept connections on port 6
[8] 23 Oct 05:28:31 - DB 0: 1 keys (0 volatile) in 4 slots HT.

vagrant@precise64:/vagrant/redis-fixture$ sudo docker ps
ID                  IMAGE                                STATUS              P
8eced7aaefbf        jbarratt/redis-fixture:latest        Up 28 seconds       4
bbd790b7dfcd        jbarratt/redis-dockerfile:latest     Up 25 minutes       4

vagrant@precise64:/vagrant/redis-fixture$ redis-cli -p 49173

redis 127.0.0.1:49173> lrange numbers 0 5
1) "2" 2) "3" 3) "4" 4) "5" 5) "6" 6) "7"
```

CLI API IS GREAT (AS IS THE *REST*)

See Dokku for craziness...

```
#!/bin/bash

if [ $# -eq 0 ]
then; echo "Usage: ./rebuilt.sh (tag)"; exit; fi

if [[ $EUID -ne 0 ]]
then ; echo "Docker needs sudo or root." ; exit ; fi

OLDCONTAINER=$(docker ps | grep $1 | cut -d ' ' -f 1)

if [ -z $OLDCONTAINER ]
then
    echo "No container found tagged $1"
    exit
fi

docker build -t $1 .
docker stop $OLDCONTAINER
docker run -d $1
```

THE DOCKER INDEX IS COOL

```
vagrant@precise64:~/docker-registry$ sudo docker pull crosbymichael/ipython
Pulling repository crosbymichael/ipython
5c69fc56f7ef: Download complete
b83fe8498b94: Download complete
e7162f69e18e: Download complete
....
vagrant@precise64:~/docker-registry$ sudo docker run -d crosbymichael/ipython
5bcdaa4f7a2f
vagrant@precise64:~/docker-registry$ sudo docker ps
ID                  IMAGE                              COMMAND               PO
5bcdaa4f7a2f        crosbymichael/ipython:latest       /bin/sh -c ipython n  49
```

IPy AllThingsOpenDemo

localhost:8888/d766f82b-7c9d-4f22-9a3b-371ff844680a

IP[y]: Notebook AllThingsOpenDemo (autosaved)

File Edit View Insert Cell Kernel Help

Code Cell Toolbar: None

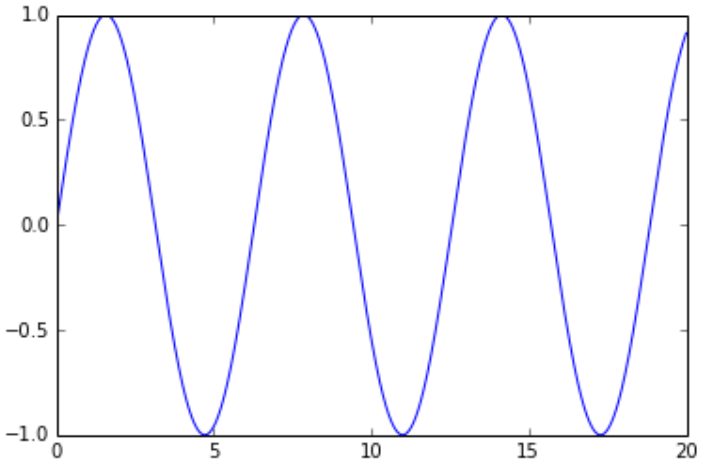
In [1]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import pylab
import numpy as np
```

In [3]:
```
x = np.linspace(0, 20, 1000)  # 100 evenly-spaced values from 0 to 50
y = np.sin(x)
pylab.plot(x, y)
```

Out[3]:
```
[<matplotlib.lines.Line2D at 0x28db510>]
```




ECOSYSTEM:
HUGE AND GROWING.

NOT JUST DOCKER

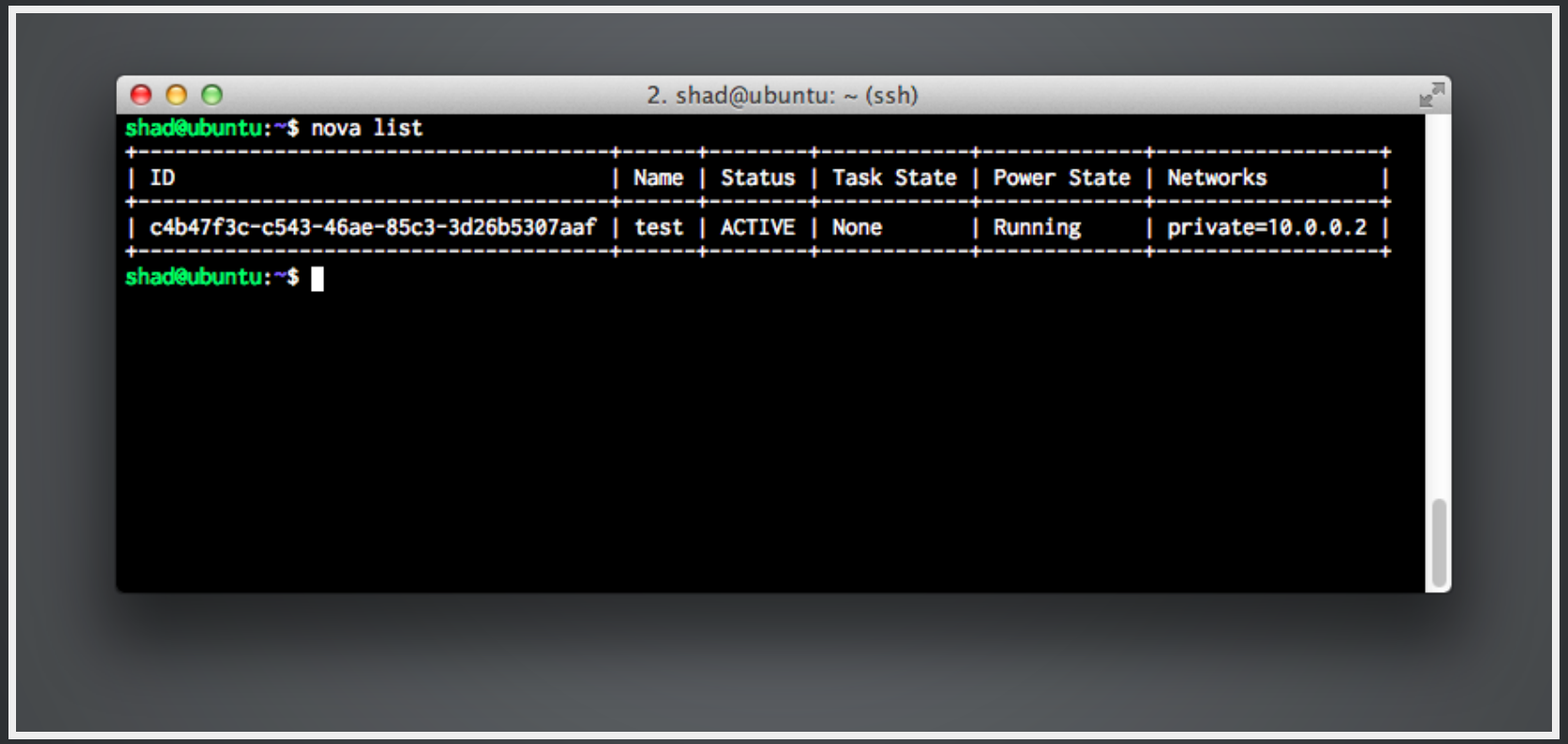
- LXC vanilla // OpenStack
- CloudLinux
- lmctfy

BUT DOCKER IS *CRAZY.*

- **PaaS**: Cocaine (!!!!), dotCloud, Deis, OpenShift, Flynn, Dokku..
- **IaaS**: OpenStack (Standard in Havana)
- **VPS**: Many/Most providers possible if not easy (Linode, DO)
- **Distros**: CoreOS
- **Config Management**: Puppet, Chef, Salt, Ansible
- **Orchestration**: Heat, Maestro, Mesos, Toscanni
- **Dashboards**: Docker-UI, Shipyard, Horizon

OPENSTACK: BOOTING A CONTAINER

OPENSTACK: CHECKING RUNNING CONTAINERS



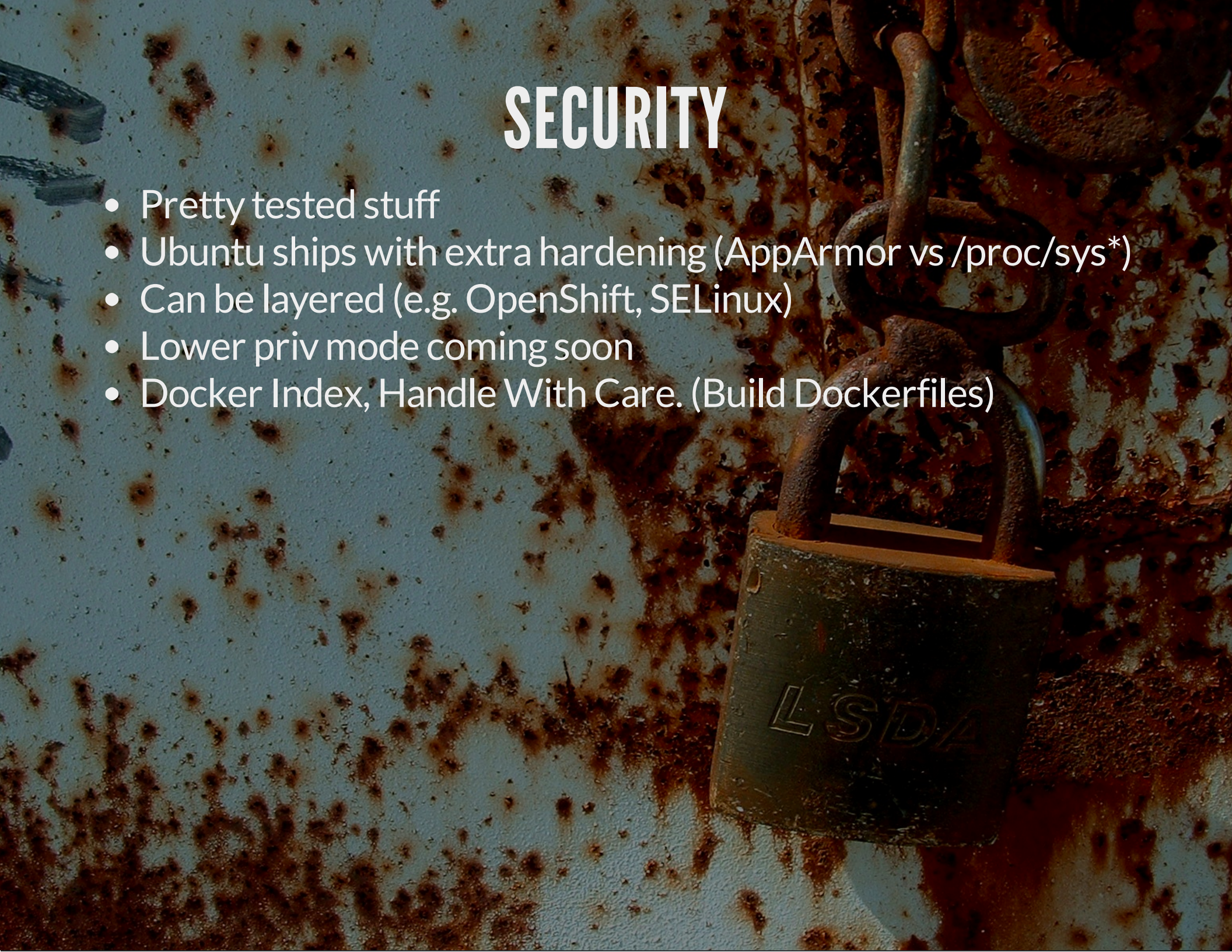
OPENSTACK HEAT ORCHESTRATION

```
$ docker pull samalba/wordpress
$ docker pull samalba/mysql
$ heat stack-create wordpress -f templates/wordpress_mysql.yml
```

```
$ heat stack-list
+--------------------------------------+------------+-----------------+
| id                                   | stack_name | stack_status    |
+--------------------------------------+------------+-----------------+
| 239cc009-00fe-4c6f-ac67-f8873df38f08 | wordpress  | CREATE_COMPLETE |
+--------------------------------------+------------+-----------------+
```

```
docker ps
ID            IMAGE                       COMMAND                  PORTS
5513c532b4dc  samalba/wordpress:latest    /usr/sbin/apache2 -D     49174->80
b98d764d109d  samalba/mysql:latest        /start_mysqld.sh         49173->3306
```

Hybrid Workloads. Cool.



SECURITY

- Pretty tested stuff
- Ubuntu ships with extra hardening (AppArmor vs /proc/sys*)
- Can be layered (e.g. OpenShift, SELinux)
- Lower priv mode coming soon
- Docker Index, Handle With Care. (Build Dockerfiles)

NEW CHALLENGES: INCREASED SURFACE AREA & SPRAWL. (OPENSSL!)

- Can't just check w/ each box's package manager.
- More apps means more apps means more to learn. Making it easy to deploy node doesn't mean your ops team knows how, follows the right security lists, etc.

BUT

- Less cases where this happens, only the libs we care about. (not sshd too, for example.)
- Low surface area if an exploit does happen
- Lower friction to do upgrades (mysql vs apache2)
- Proper Docker layering actually means less sprawl than you think.

HOW DO I PLAY?

```
curl http://get.docker.io | sudo sh
```

Or

```
# Install vagrant, then...
$ git clone https://github.com/dotcloud/docker.git
$ vagrant up
$ vagrant ssh
$ sudo docker
```

SHOULD I DO MORE THAN PLAY?

- Not rated as "production ready" due to API volatility
- Very usable for nearline applications (e.g. testing)

BONUS: DOCKCEPTION!

```
FROM ubuntu
MAINTAINER Joshua Barratt jbarratt@serialized.net

RUN apt-get -qq update
RUN apt-get install -y python3

ADD talk /talk

EXPOSE 8080:8080
WORKDIR /talk
CMD ["python3", "-m", "http.server", "8080"]
```

LINKAGE

- @jbarratt
- github.com/jbarratt/dockertalk
- docker.io
- blog.docker.io/tag/docker-weekly