Guidewire

Introduction to Docker and Containers

@docker
@charme_g

docker

Guidewire

Outline

- **Whom is this for?**
- What's the problem?
- What's a Container?
- Docker 101
- Docker images
- Docker future

docker

Guidewire

Ops

- any distro[1]
- any cloud[2]
- any machine (physical, virtual...)
- recent kernels[3]

[1] as long as it's Ubuntu or Debian ☺ others coming soon
[2] as long as they don't ship with their custom crappy kernel
[3] at least 3.8; support for RHEL 2.6.32 on the way

docker

Guidewire

Devs

- all languages
- all databases
- all O/S
- targetting Linux systems

Docker will eventually be able to target FreeBSD, Solaris, and maybe OS X.

docker

Guidewire

CFO, CIO, CTO, ...

- LESS overhead!
- MORE consolidation!
- MORE agility!
- LESS costs!

docker

Guidewire

Outline

- Whom is this for?
- **What's the problem?**
- What's a Container?
- Docker 101
- Docker images
- Docker future

docker

Guidewire

The Matrix From Hell

	my laptop	your laptop	QA	staging	prod on cloud VM	prod on bare metal
django web frontend	?	?	?	?	?	?
node.js async API	?	?	?	?	?	?
background workers	?	?	?	?	?	?
SQL database	?	?	?	?	?	?
distributed DB, big data	?	?	?	?	?	?
message queue	?	?	?	?	?	?

docker

Guidewire

Another Matrix from Hell

	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?
	?	?	?	?	?	?	?







Guidewire

Solution:
the *intermodal shipping container*

docker

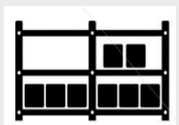
Guidewire

Solved!










Guidewire

Solution to the deployment problem: the *Linux* container

docker

Guidewire

Linux containers...

Units of software delivery (**ship it!**)

- run everywhere
  - regardless of kernel version
  - regardless of host distro
  - (but container and host architecture must match*)
- run anything
  - if it can run on the host, it can run in the container
  - i.e., if it can run on a Linux kernel, it can run

*Unless you emulate CPU with qemu and binfmt

docker

Guidewire

Outline

- Whom is this for?
- What's the problem?
- **What's a Container?**
- Docker 101
- Docker images
- Docker future

docker

Guidewire

High level approach: it's a lightweight VM

- own process space
- own network interface
- can run stuff as root
- can have its own /sbin/init (different from the host)

« Machine Container »

docker

Guidewire

Low level approach: it's chroot on steroids

- can also *not* have its own /sbin/init
- container = isolated process(es)
- share kernel with host
- no device emulation (neither HVM nor PV)

« Application Container »

docker

Guidewire

Separation of concerns: Dave the Developer

- inside my container:
  - my code
  - my libraries
  - my package manager
  - my app
  - my data

docker

Guidewire

Separation of concerns: Oscar the Ops guy

- outside the container:
  - logging
  - remote access
  - network configuration
  - monitoring

docker

Guidewire

How does it work?
Isolation with namespaces

- pid
- mnt
- net
- uts
- ipc
- user

docker

Guidewire

How does it work?
Isolation with cgroups

- memory
- cpu
- blkio
- devices

docker

Guidewire

If you're serious about security, you also need…

- capabilities
  - okay: cap_ipc_lock, cap_lease, cap_mknod, cap_net_admin, cap_net_bind_service, cap_net_raw
  - troublesome: cap_sys_admin (mount!)
- think twice before granting root
- grsec is nice
- seccomp (very specific use cases); seccomp-bpf
- beware of full-scale kernel exploits!

docker

Guidewire

How does it work? Copy-on-write storage

- unioning filesystems
  (AUFS, overlayfs)
- snapshotting filesystems
  (BTRFS, ZFS)
- copy-on-write block devices
  (thin snapshots with LVM or device-mapper)

This is now being integrated with low-level LXC tools as well!

docker

Guidewire

Efficiency

docker

Guidewire

Compute efficiency: *almost* no overhead

- processes are isolated,
  but run straight on the host
- CPU performance
  = native performance
- memory performance
  = a few % shaved off for (optional) accounting
- network performance
  = small overhead; can be reduced to zero

docker

Guidewire

Storage efficiency: many options!

	Union Filesystems	Snapshotting Filesystems	Copy-on-write block devices
Provisioning	Superfast Supercheap	Fast Cheap	Fast Cheap
Changing small files	Superfast Supercheap	Fast Cheap	Fast Costly
Changing large files	Slow (first time) Inefficient (copy-up!)	Fast Cheap	Fast Cheap
Diffing	Superfast	Superfast (ZFS) Kinda meh (BTRFS)	Slow
Memory usage	Efficient	Efficient	Inefficient (at high densities)
Drawbacks	Random quirks AUFS not mainline !AUFS more quirks	ZFS not mainline BTRFS not as nice	Higher disk usage Great performance (except diffing)
Bottom line	**Ideal for PAAS and high density things**	**This might be the Future**	**Dodge Ram 3500**

docker

Guidewire

Outline

- Whom is this for?
- What's the problem?
- What's a Container?
- **Docker 101**
- Docker images
- Docker future

docker

Guidewire

Guidewire

Docker-what?

- Open Source engine to **commoditize** LXC
- using copy-on-write for quick provisioning

STOP!

~~HAMMER~~ DEMO TIME.

docker


root@dockerhost:~#

Guidewire

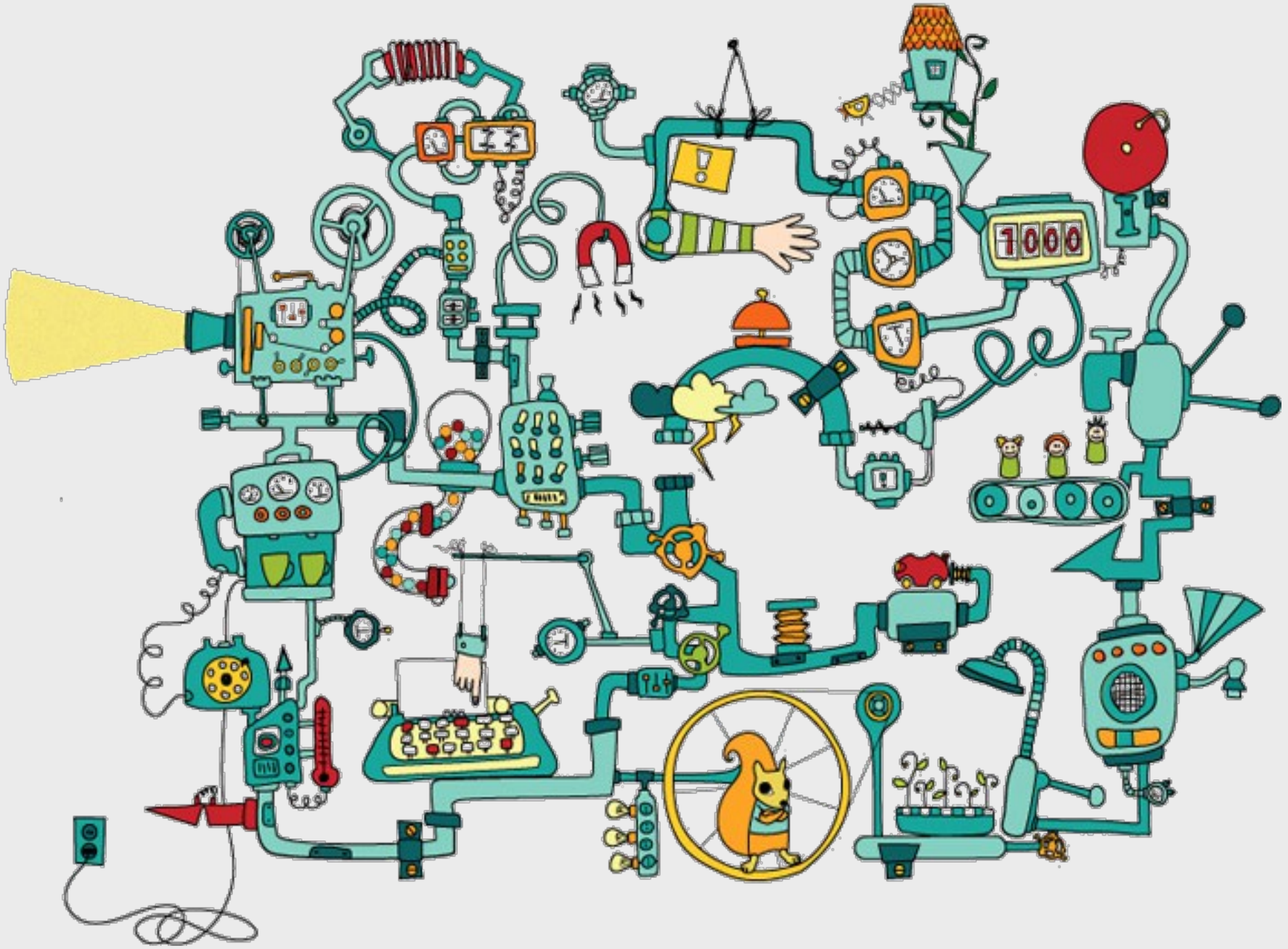
Yes, but...

- « I don't need Docker;
  I can do all that stuff with LXC tools, rsync,
  some scripts! »
- correct on all accounts;
  but it's also true for apt, dpkg, rpm, yum, etc.
- the whole point is to **commoditize**,
  i.e. make it ridiculously easy to use

docker

Guidewire

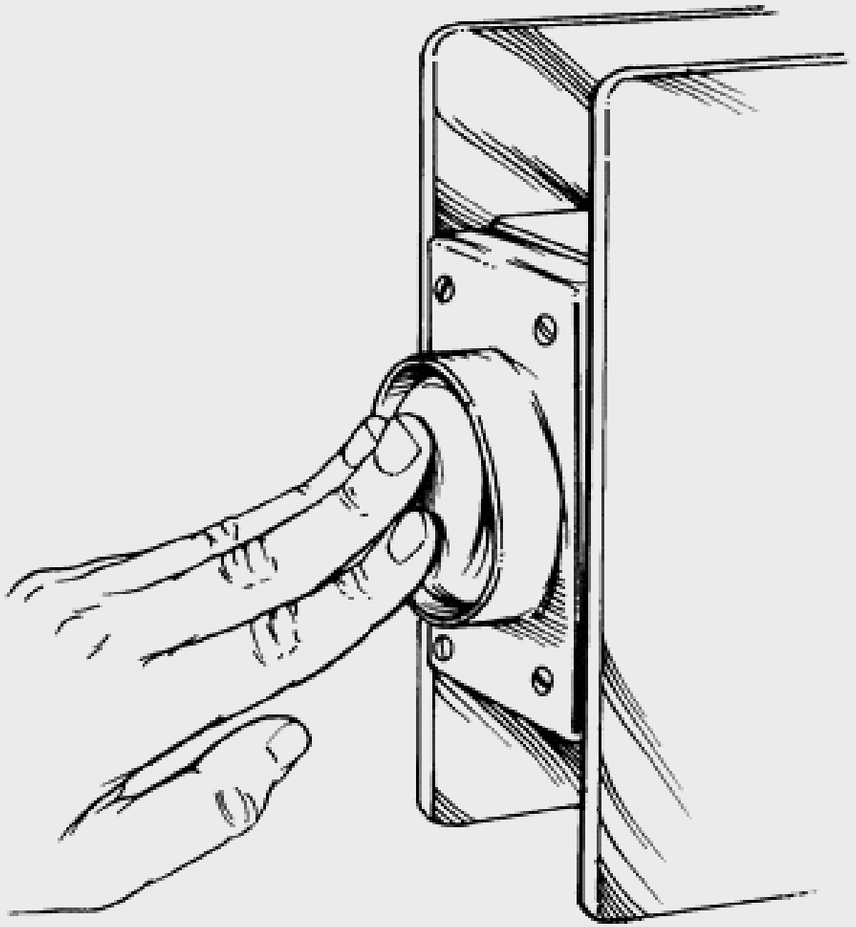
Containers before Docker



docker

Guidewire

Containers after Docker



docker

Guidewire

What this really means…

- instead of writing « very small shell scripts » to manage containers, write them to do the rest:
  - continuous deployment/integration/testing
  - orchestration
- = use Docker as a building block
- re-use other people images (yay ecosystem!)

docker

Guidewire

Docker-what?
The Big Picture

- Open Source engine to commoditize LXC
- using copy-on-write for quick provisioning
- allowing to **create and share** *images*
- **standard format** for containers
  (stack of layers; 1 layer = tarball+metadata)
- standard, *reproducible* way to *easily* build
  trusted images (Dockerfile, Stackbrew...)

docker

Guidewire

Docker-what?
Under The Hood

- rewrite of dotCloud internal container engine
  - original version: Python, tied to dotCloud's internal stuff
  - released version: Go, legacy-free
- the Docker daemon runs in the background
  - manages containers, images, and builds
  - HTTP API (over UNIX or TCP socket)
  - embedded CLI talking to the API
- Open Source (GitHub public repository + issue tracking)

docker

Guidewire

Outline

- Whom is this for?
- What's the problem?
- What's a Container?
- Docker 101
- **Docker images**
- Docker future

docker

Guidewire

Authoring images with run/commit

1) docker run ubuntu bash
2) apt-get install this and that
3) docker commit <containerid> <imagename>
4) docker run <imagename> bash
5) git clone git://.../mycode
6) pip install -r requirements.txt
7) docker commit <containerid> <imagename>
8) repeat steps 4-7 as necessary
9) docker tag <imagename> <user/image>

docker

Guidewire

Authoring images with a Dockerfile

```
FROM ubuntu

RUN apt-get -y update
RUN apt-get install -y g++
RUN apt-get install -y erlang-dev erlang-manpages
erlang-base-hipe ...
RUN apt-get install -y libmozjs185-dev libicu-dev
libtool ...
RUN apt-get install -y make wget

RUN wget http://.../apache-couchdb-1.3.1.tar.gz | tar
-C /tmp -zxf-
RUN cd /tmp/apache-couchdb-* && ./configure && make
install

RUN printf "[httpd]\nport = 8101\nbind_address =
0.0.0.0" >
    /usr/local/etc/couchdb/local.d/docker.ini
```

docker

Guidewire

Authoring Images with Chef/Puppet/Ansible/Salt/...

Plan A: « my other VM is a container »

- write a Dockerfile to install $YOUR_CM
- start tons of containers
- run $YOUR_CM in them

Good if you want a mix of containers/VM/metal

But slower to deploy, and uses more resources

docker

Guidewire

Authoring Images with Chef/Puppet/Ansible/Salt/...

Plan B: « the revolution will be containerized »

- write a Dockerfile to install $YOUR_CM
- … and *run* $YOUR_CM as part of build process
- deploy fully baked images

Faster to deploy

Easier to rollback

docker

Guidewire

Outline

- Whom is this for?
- What's the problem?
- What's a Container?
- Docker 101
- Docker images
- **Docker future**

docker

Guidewire

Docker: the community

- Docker: >200 contributors
- <7% of them work for ~~dotCloud~~ Docker inc.
- latest milestone (0.6): 40 contributors
- ~50% of all commits by external contributors
- GitHub repository: >800 forks

docker

Guidewire

Docker: the ecosystem

- Cocaine (PAAS; has Docker plugin)
- CoreOS (full distro based on Docker)
- Deis (PAAS; available)
- Dokku (mini-Heroku in 100 lines of bash)
- Flynn (PAAS; in development)
- Maestro (orchestration from a simple YAML file)
- OpenStack integration (in Havana, Nova has a Docker driver)

docker

Guidewire

Docker long-term roadmap

Docker 1.0:

- dynamic discovery
- remove AUFS, THINP, LXC, etc.
  - execution? chroot!
  - storage? cp!
  - we can run everywhere \o/
- re-add everything as plugins

docker

Guidewire

Thank you! Questions?

http://docker.io/

http://docker.com/

https://github.com/dotcloud/docker

@docker

@charme_g

docker