

# Next-Gen Cloud Computing and DevOps with Docker Containers

Hamilton Turner  
hamiltont@gmail.com

March 12, 2014

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Who am I?

## Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

## Who am I?

- Ph.D. Candidate at Virginia Tech
  - Defending in Fall 2014
- Member of Magnum Research Group (<http://www.magnum.io/>)
  - We focus on mobile and cloud applications

## How do I use Docker?

- Stress test 100+ web apps, without installing natively
- Manage computer cluster for research group
- Personal playground for learning new technologies

# DevOps Challenges: The Matrix From Hell

django web frontend	?	?	?	?	?	?
node.js async API	?	?	?	?	?	?
background worker tasks	?	?	?	?	?	?
SQL Database	?	?	?	?	?	?
distributed DB, big data tools	?	?	?	?	?	?
message queue	?	?	?	?	?	?
	my laptop	your laptop	QA system	staging	production on Cloud VM	production on bare metal

- Each environment looks different
- Different developer needs
- Different operations environments

## The 'Big Question'

- How do we make all these environments identical?!

Credit: Jérôme Petazzoni at <http://www.slideshare.net/jpetazzo/introduction-docker-linux-containers-lxc>

Who Am I?

The DevOps Challenge

Beyond VMs

The How of Docker

Docker 101

Docker Examples

Docker Limits

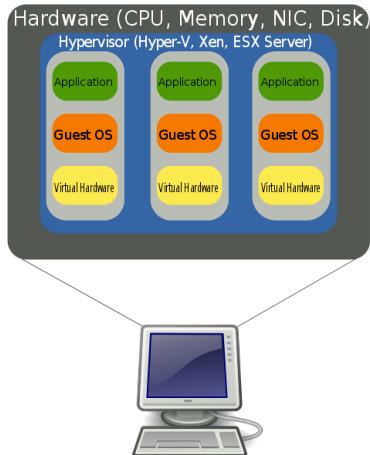
Hadoop Demo

Conclusions

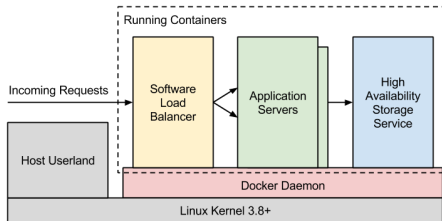
# Review of Virtual Machines

## Virtual Machines are one current solution

- An 'entire computer' is executed as a program
- No developer's local system has to be modified
- The entire VM disk image can be shared
- No more differences between dev and production environments!
- This is great! ....Right?



# Comparison of Docker Containers And VMs



Credit: quay.io - A Secure Hosting Solution For Private Docker Repositories

	Docker Container	Virtual Machine
Avg Host Resources Consumed	Low	High
Clean Startup Time	seconds	minutes/hours
Environment (FS) Sharing	Small (Union filesystem)	Large (Entire Snapshot)
Environment Reproducibility	High	Moderate (AWS image)
Software Modifications Needed?	Perhaps (one process)	Unlikely
Attack Surface	Untested	Small
System Monitoring	Use Linux Tools	Custom systems

# The How of Docker

Docker shares the kernel with the host, uses Linux namespaces+cgroups+union filesystems to isolate

- process trees (PIDs)
- mounts (mnt)
- network (net)
- inter-process communication (ipc)
- user accounts (user)
- hostnames (utc)
- memory
- CPU
- Disk access (blkio)
- Device access (devices)

Summary: Docker combines and standardizes a number of existing Linux components (kernel 3.8+)

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# The How of Docker, Union Filesystem Version

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

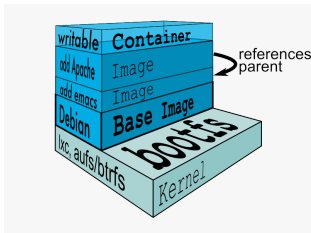
Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions



- Each *layer* of the FS is mounted on **top** of prior layers
- The first layer is the *base image*
- Current base images include debian, ubuntu, busybox, fedora, cent os, etc
- Each read-only layer is called an *image* (A layer is just a collection of files and folders!)
- The top layer is the only modifiable layer - it's termed the *container*

# Docker 101: Run Interactive Container

```
$ sudo docker run -i -t ubuntu /bin/bash
```

- `sudo` : Docker has to be run as root!
- `run` : we are running a container
- `-i -t` : we want a **terminal** (stdin and stdout), and we want to be connected to those so we can **interact** with the container
- `ubuntu` : The base image for this container
- `/bin/bash` : Let's run bash

```
$ sudo docker run -i -t ubuntu /bin/bash
root@03711559d57d:/# cat /etc/*release*
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04 LTS"
root@03711559d57d:/# exit
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions



# Docker 101: Run Non-Interactive Container

Flags **-i** and **-t** are good for interacting with a container, but for scripting or long-running tasks, you'll want to use detached (**-d**) mode

```
$ sudo docker run -d ubuntu /bin/bash -c "echo hi"  
94490365f464bab1f009ec0971e1691213b4562dbaeb04b2e33ad  
$
```

Odd things:

- There was no 'hi'
- You were given this long string
- You are back at your original shell, even though you ran bash

In detached mode, docker immediately returns a container ID. This ID can be used to fetch container stdout, check container status, stop the container, etc

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Docker 101: Run Non-Interactive Container, Part Two

Ok, let's see what's happening using our container ID

```
$ sudo docker run -d ubuntu /bin/bash -c "echo hi"  
d2026870efedf09e29dbea146d399e60493e9dd0ebbf6124347d6  
$ sudo docker logs d2026870efedf09e29dbea146d399e6049  
hi
```

Container ID's can be referenced by unique prefix too

```
$ sudo docker logs d202  
hi
```

**docker ps** shows you what containers are running

```
$ sudo docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS  
d2026870ef ubuntu:12.04 /bin/bash -c while t 1 minute ago Up 1 min
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# More on Container IDs

Typically, you will want to store the ID

```
$ MY_ECHO=$(sudo docker run -d ubuntu /bin/bash  
-c "echo hi")  
$ sudo docker logs $MY_ECHO  
hi
```

- Detached Mode (e.g. **docker run -d**)
  - Docker run response is the container ID
  - To capture the output, we use `$(...)`
  - This output is stored into variable `MY_ECHO`, and later retrieved with `$MY_ECHO`
- Interactive Mode (e.g. **docker run -i -t**)
  - Run container, modify, then exit. Container is now stopped
  - Use **docker ps -a** to show *all* containers, incl. stopped ones
  - Or use **docker ps -l -q** to show the *last* container ID

```
$ sudo docker ps -a  
CONTAINER ID  IMAGE          COMMAND                  CREATED        STATUS  
d2026870ef    ubuntu:12.04   /bin/bash -c while t    1 minute ago  Exit 0  
$ sudo docker ps -q -l  
d2026870ef
```

Note: Docker now supports container **names**

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

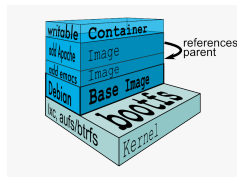
Docker Limits

Hadoop Demo

Conclusions

# Storing A Container For Reuse (a.k.a. Building an Image)

- Recall: the **container** filesystem is the final union with a stack of images
- docker commit** converts this container filesystem into an **image**
- The **image** can then be used to run other containers



```
$ APP=$(sudo docker run -d ubuntu /bin/bash -c  
  ''echo hi > config.out'')  
$ sudo docker commit $APP hamiltont/myapp  
$ sudo docker run -i -t hamiltont/myapp /bin/bash  
root@3a1f0c28b822:/# cat config.out  
hi
```

If you could share this image...then others could build new containers based on this image!

# Sharing An Image For Reuse

- Images can be shared using a registry
- **docker push** and **docker pull**
- There is a public registry available, or your company can host it's own private registry Check out quay.io
- If **docker run** does not find the image locally, it will automatically search known registries

```
$ sudo docker push hamiltont/myapp
$ sudo docker pull hamiltont/myotherapp
```

- The **images** subcommand can be used to list local images

```
$ sudo docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE
hamiltont/myapp	latest	d100b411c51e	2 minutes ago	204.4 MB
hamiltont/myotherapp	latest	7cb2d9010d39	11 days ago	410.6 MB
ubuntu	12.04	9cd978db300e	5 weeks ago	204.4 MB
ubuntu	latest	9cd978db300e	5 weeks ago	204.4 MB

# Benefits of Using Union Filesystems For Images

- Hypothetical:
  - I run a ubuntu container, make changes, and exit
  - I commit my changes to an image and run **docker push**
  - My colleague wants to **docker pull** my image
  - What do they need to download?
- Answer:
  - Just your changes!
  - They have probably already downloaded the ubuntu base image
- No inherent need for multi-GB images
- Download only files, not arbitrary filesystem junk
- While YMMV,  
80% of images are  $\leq 50MB$ , 95% are  $\leq 500MB$

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Linking Host/Container Filesystems

A nice separation of concerns would place application inside the container, logs outside the container.

The **-v** flag can mount a host volume to the container

```
$ sudo mkdir /app_logs
$ sudo docker run -i -t -v /app_logs:/logs ubuntu
/bin/bash
root@842fa9699353:/# cd /logs/
root@842fa9699353:/logs# echo "My application log"
> log.out
root@842fa9699353:/logs# exit
$ cat /app_logs/log.out
My application log
```

**-v** can also be used to access configuration on the host

```
$ sudo mkdir /app_conf
$ sudo docker run -i -t -v /app_conf:/etc/app ubuntu
/bin/bash
root@842fa9699353:/# my_app --conf /etc/app
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Exposing Container Network Ports

Docker container ports are not published unless requested.  
The **-p** flag can be used to publish a port

```
$ SERVER=$(docker run -d -p 8000 ubuntu /bin/bash  
-c 'while true; do sleep 5; done')  
$ sudo docker port $SERVER 8000  
0.0.0.0:49158
```

Breakdown:

- Run a bash process inside the container, looping indefinitely
- **-p 8000** caused Docker to find an unused host port and link it with the container-internal port 8000
- We used the **port** subcommand to find this public port
- There is nothing listening on port 8000 in the container, so this is kind of boring

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions



# Exposing Container Network Ports, Part Two

So let's run an actual webserver!

Method 1: Build my own webserver image

Method 2: Reuse someone else's pre-built image

```
$ WEB_SERVER=$(sudo docker run -t -d  
-p 8000 hamiltont/python-simplehttpserver)  
$ sudo docker logs $WEB_SERVER  
Serving HTTP on 0.0.0.0 port 8000 ...  
$ sudo docker port $WEB_SERVER 8000  
0.0.0.0:49186
```

Note:

- I chose to reuse hamiltont/python-simplehttpserver
- Navigating to `http://localhost:49186` will now connect me to the webserver
- The container knew what command to run! More on this next...

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Building Images with Dockerfiles

- We know how to run a container, modify it, and commit it as an image
- A **Dockerfile** lists the steps needed to build an images
- Similar to a Makefile
- **docker build** is used to run a Dockerfile
- Can define default command for **docker run**, ports to expose, etc

```
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

EXPOSE 8101
CMD ["/usr/local/bin/couchdb"]
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Locating Community Images



## docker index

- There are hundreds of community-contributed and/or official images online at <http://index.docker.io>
- This is the official registry, you can also host your own
- You can also use the **docker search** subcommand to interact with [index.docker.io](http://index.docker.io)

```
$ sudo docker search wordpress
```

NAME	DESCRIPTION	STARS	TRUSTED
ctlc/wordpress		4	[OK]
jbfink/docker-wordpress	Same as jbfink/wordpress, just a trusted b	2	[OK]
skxsx/wordpress	Wordpress & SSH in a container.	2	
eugeneware/docker		1	[OK]
tutum/wordpress	Wordpress Docker image — listens in port 80	1	[OK]
jbfink/wordpress	Wordpress 3.8	1	

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Docker 101 Review: Topics Covered

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

- Running Docker Containers In Interactive or Detached Mode
- Container IDs (and names)
- Viewing output of detached container (**docker logs**)
- Committing containers to create images
- Sharing images via push/pull
- Mounting Filesystems From Host inside container
- Exposing container network ports
- Basic concept of Dockerfiles

# Docker 101: Things We Didn't Cover

- Detach from running container, then re-attaching
- Diff'ing filesystems
- Storing container/image as tar file
- Using **docker kill** to stop a running container
- Deleting unused containers/images
- Examining processes inside container from the host
- Linking containers together (via filesystem or network)
- Trusted builds
- Limiting container memory and CPU consumption

Any questions at this point?

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

**Docker 101**

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# A (small) selection of Dockerized Projects

Data Storage	Redis, MySQL, MongoDB, memcached
Server Stacks	Nginx, Apache+PHP, SSH
Development Environments	Python, Go, Ruby, Java, Chef/Puppet, node.js, X11+SSH Desktop
Blogging / CMS	Wordpress, Ghost
Single-use tool usage	Redis CLI, Latex, subuser
Continuous Integration	Jenkins, Drone
Proxy Software	Hipache, Nginx/Apache/node.js
PaaS	Cocaine (from Yandex), Deis, Flynn, Bowery, Dokku

Who Am I?

The DevOps Challenge

Beyond VMs

The How of Docker

Docker 101

Docker Examples

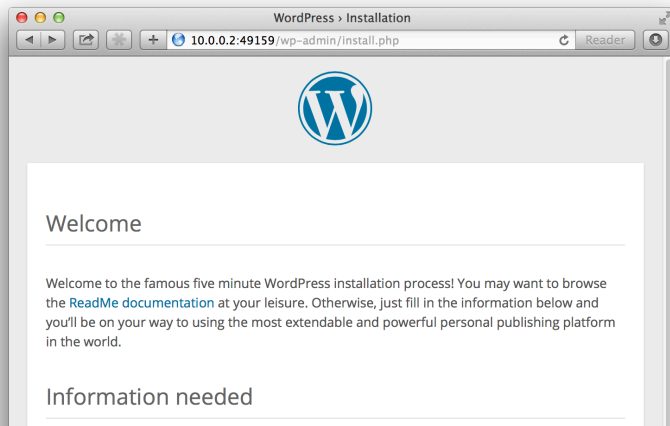
Docker Limits

Hadoop Demo

Conclusions

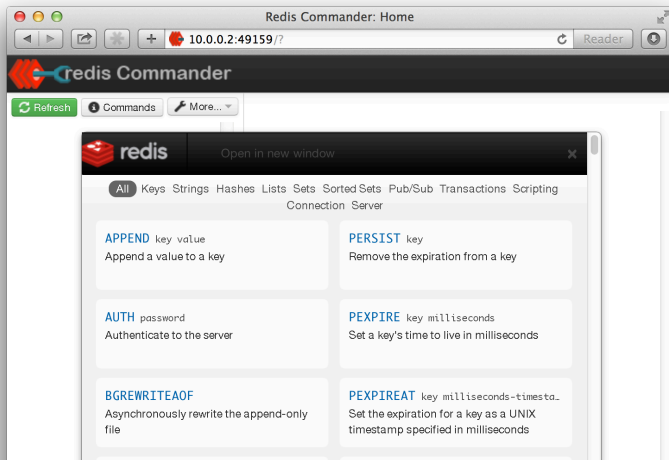
# Example: Wordpress

```
# WP=$(docker run -d -p 80 tutum/wordpress)
# docker port $WP 80
0.0.0.0:49159
```



# Example: Redis Commander Web GUI

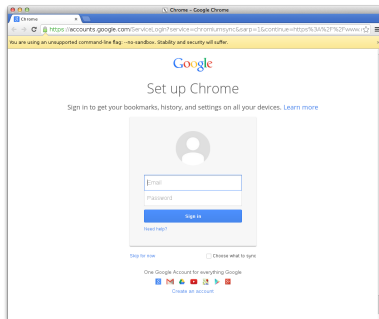
```
# RD=$(docker run -d -p 8081 elsdoerfer/redis-commander)
# docker port $RD 8081
0.0.0.0:49159
```





# Example: Sandboxed Dev Environment

```
# docker run -t -i -p 22 magglass1/docker-browser-over-ssh
IP address: 172.17.0.4
Password: N24DjBM86gPubuEE
Firefox: ssh -X webuser@172.17.0.4 firefox
Google Chrome: ssh -X webuser@172.17.0.4 google-chrome --no-sandbox
```



# Example: SSH Server

```
# SSH=$(docker run -d -p 22 dhrp/sshd)
# docker port $SSH 22
0.0.0.0:49160
```

Ok, SSH is running. Now to connect!

```
$ ssh -p 49161 root@10.0.0.2
root@10.0.0.2's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.11.0-18-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@7d45b427eca1:~#
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

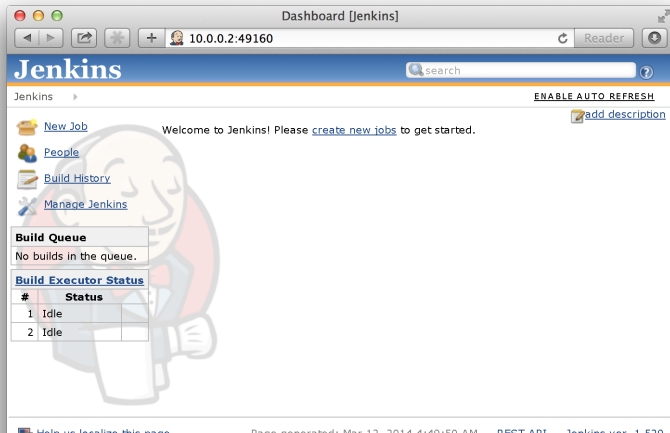
Hadoop Demo

Conclusions

# Example: Continuous Integration

```
# JEN=$(docker run -d -p 8080 --cpu-shares=20 lzhang/jenkins)
# docker port $JEN 8080
0.0.0.0:49160
```

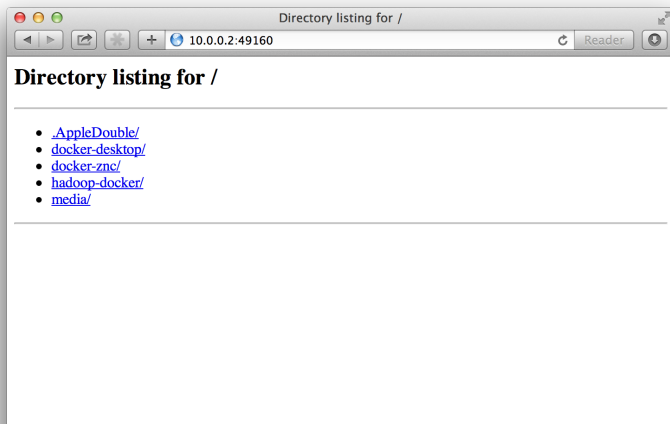
Note: I've limited the CPU shares allowed



# Example: Quick Web Server

```
# WEB=$(docker run -d -v /Programming:/www -p 8000 hamiltont/python-simplehttpd)
# docker port $WEB 8000
0.0.0.0:49160
```

Note: I've mounted my /Programming folder to /www



Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Example: Using Redis CLI Without Installing It

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

```
# docker run -i -t crosbymichael/redis-cli -h  
redis.myhost.com 'KEYS *'
```

- 1) "four"
- 2) "one"
- 3) "two"

- Redis is not installed on the host
- I can use complex (and stateful!) commands without modifying the host machine
- I could even pull my dotfiles onto a production machine
- No noticable run delay

# Example: Preexisting Automation Tools

- Docker doesn't exclude existing tools like Chef, Puppet, etc
- Current base systems are limited to (mostly) Debian
- Most current automation systems can be used seamlessly
- But...you no longer have to worry as much about OS updates as the Docker image is static ;-)

## CentOS+Chef:

```
# docker run -i -t raybeam/chef /bin/bash
bash-4.1# knife
ERROR: You need to pass a sub-command
(e.g., knife SUB-COMMAND)
... <snip>
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Docker Is Not a Panacea: Considerations

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

- No Shared Libraries
  - The price of process isolation
- ADD-only filesystem
  - Scenario: Run a huge build and then `rm /tmp/*`
  - Result: All of `/tmp` will be downloaded, and then masked with an empty directory
- Root access required for simple operations!
  - Progress is being made on this front
- Orchestration of Containers is limited...more on this next

# Docker Service Orchestration and Service Discovery

If my application has 10 containers, how do I organize them all?  
No clear winner here, but many solutions in progress:

- Raw Docker
  - **docker run -v** can be used to call out interesting volumes within containers
  - **volumes from** can be used to share these volumes with other containers
  - Containers can be **linked** to share their network
- Fig: Uses simple config to start/link containers
- Serf: General service discovery solution
- Shipyard: Web-based system for managing docker-driven applications
- CoreOS: OS designed for running cloud apps such as Docker
- SkyDock: DNS-based Docker service discovery

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions



# Rough Test of Docker Overhead

- Launch and sleep tiny containers (busybox, 125Kb)
- Docker overhead hopefully trumps this
- Using 4GB, 4-core system

```
# while true
> do
>   docker run -d busybox /bin/ash -c
"while true; do echo alive; sleep 60000; done"
>   docker ps | wc -l
> done
```

## Fast Fail Results:

- 250 containers launched before "Too many open files"
- <2GB memory used, load of 3 (all sleeping)

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Recap: Docker and Virtual Machines

Q: Are Docker Containers just 'Better Virtual Machines'?

- Docker monitors one process, VMs have system daemons running (cron, syslog, upstart, etc)
  - You could run/monitor a process manager (e.g. supervisord)
- Let's consider separation of concerns
  - Given: Containers are light, VMs are heavy
  - It's unlikely you allocate two VMs and communicate - you shove your entire ecosystem into a single VM
  - This jumbles concerns together and reduces maintainability, predictability, and security
- Containers emulate processes, VM's emulate machines
- Often 'Dev' means you work inside the container, and 'Ops' means you work outside the container

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

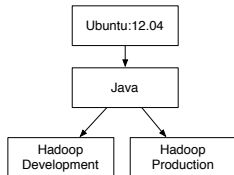
Hadoop Demo

Conclusions

# Docker and Hadoop

## What do I need?

- Development Environment
  - With full source code / docs
  - Good base image for configuring IDE
- Production Environment
  - Minimal Hadoop footprint
  - All dependencies
  - Native 64-bit libraries
- Commonalities?
  - Yes, Java
  - Ok, so three images: Java, Dev, Production



Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Developing Oracle Java7 Image

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

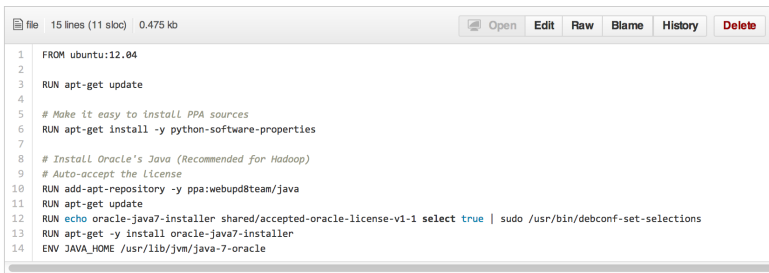
Docker  
Examples

Docker Limits

**Hadoop Demo**

Conclusions

- Need to base our image on a stable parent
- Need to carefully record our steps
- Want to share this with all developers
- Use Dockerfile!



The screenshot shows a code editor window with a light gray header bar. The header bar contains a file icon, the text "file", "15 lines (11 sloc)", "0.475 kb", and a row of buttons: "Open", "Edit", "Raw", "Blame", "History", and "Delete". The main area of the editor displays a Dockerfile with 15 lines of code, each preceded by a line number from 1 to 14. The code is as follows:

```
1 FROM ubuntu:12.04
2
3 RUN apt-get update
4
5 # Make it easy to install PPA sources
6 RUN apt-get install -y python-software-properties
7
8 # Install Oracle's Java (Recommended for Hadoop)
9 # Auto-accept the license
10 RUN add-apt-repository -y ppa:webupd8team/java
11 RUN apt-get update
12 RUN echo oracle-java7-installer shared/accepted-oracle-license-v1-1 select true | sudo /usr/bin/debconf-set-selections
13 RUN apt-get -y install oracle-java7-installer
14 ENV JAVA_HOME /usr/lib/jvm/java-7-oracle
```

# Developing Hadoop-Dev Image

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

**Hadoop Demo**

Conclusions

- Build on top of Java7 image
- Install build tools
- Install hadoop build dependencies
- Run build process
- Of Note: This image is huge!  $> 1GB$  due to all the build dependencies downloaded by maven

# Developing Hadoop Docker

```
FROM hamiltont/oracle-java7

RUN apt-get update

# Needed to download Hadoop release from Apache website
RUN apt-get install -y wget

# Download and build Hadoop's protobuf dependency
RUN apt-get install -y zlib1g-dev autoconf automake build-essential maven cmake libssl-dev
RUN wget http://protobuf.googlecode.com/files/protobuf-2.5.0.tar.gz
RUN tar -xvzf protobuf-2.5.0.tar.gz -C /tmp && \
    cd /tmp/protobuf-2.5.0 && \
    ./configure && \
    make && make install && \
    ldconfig

# Download build Hadoop
RUN wget http://apache.mirrors.tds.net/hadoop/common/hadoop-2.2.0/hadoop-2.2.0-src.tar.gz
RUN tar -xvzf hadoop-2.2.0-src.tar.gz -C /tmp && \
    cd /tmp/hadoop-2.2.0-src

# Minor patch needed for compilation
ADD hadoop-auth-pom.xml.patch /tmp/hadoop-auth-pom.xml.patch
RUN patch /tmp/hadoop-2.2.0-src/hadoop-common-project/hadoop-auth/pom.xml /tmp/hadoop-auth-pom.xml.patch

# Build Hadoop (also builds native 64-bit library)
RUN cd /tmp/hadoop-2.2.0-src && \
    mvn package -Pdist,native -DskipTests -Dtar
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Developing Hadoop Image

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

- Use **docker cp** to extract the tar from hadoop-dev
- Add that to hadoop, extract tar file
- Update environment PATH
- Install SSH server
- Pull in configuration files
- Automatically start SSH, hadoop processes



# Developing Hadoop Docker

```
FROM hamiltont/oracle-java7

RUN apt-get update

# If you built the hadoop-dev container then it generates this .tar.gz
# docker cp can be used to pull it to your host filesystem
ADD hadoop-2.2.0.tar.gz /

RUN mkdir /hadoop
ADD hadoop-2.2.0.tar.gz /tmp/
RUN tar -xzf /hadoop/hadoop-2.2.0.tar.gz -C /hadoop --strip-components 1

# Download/setup Hadoop's SSH dependency
RUN apt-get install -y openssh-server
RUN mkdir /var/run/ssh
RUN ssh-keygen -t rsa -P "" -f /root/.ssh/id_rsa
RUN cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys

# Manually accept host verification key
RUN /usr/sbin/sshd && \
  ssh -o StrictHostKeyChecking=no localhost exit &&\
  ssh -o StrictHostKeyChecking=no 0.0.0.0 exit &&\
  ssh -o StrictHostKeyChecking=no 127.0.0.1 exit

# Ensure hadoop workers that SSH into root@localhost can still find Java
RUN echo "export JAVA_HOME=$JAVA_HOME" > /tmp/tmp_profile && \
  cat /root/.bashrc >> /tmp/tmp_profile && \
  mv /tmp/tmp_profile /root/.bashrc

# Setup up supervisor
RUN apt-get install -y supervisor &&\
  mkdir -p /var/log/supervisor
ADD supervisord.conf /etc/supervisor/conf.d/supervisord.conf
```

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Conclusions: Who and What is Docker

Officially, Docker is...

- Open Source, 200+ contributors
- Corporate parent named Docker as well ;-)
- Very active IRC + mailing lists
- A project to combine and standardize existing features of Linux

Unofficially, Docker is...

- The forefront in Linux Containers
- A huge step beyond current VM's w.r.t. machine utilization and DevOps workflow
- A pragmatic improvement that is here to stay

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions

# Thank You For Your Time

## Questions?

Please feel free to reuse/modify presentation if you wish, just remember to leave my name in there somewhere. It's online at <https://github.com/hamiltont/intro-to-docker>

Who Am I?

The DevOps  
Challenge

Beyond VMs

The How of  
Docker

Docker 101

Docker  
Examples

Docker Limits

Hadoop Demo

Conclusions