Devops’n the Operating System

John Willis
Director of Ecosystem Development
Docker, Inc.
@botchagalupe

- a.k.a. John Willis
- 35 Years in IT Operations
- Exxon, Canonical, Chef, Enstratius, Socketplane
- Devopsdays Core Organizer
- 35 Official Devopsdays
- Devopscafe on iTunes
- Organizer of Devops Enterprise Summit
Devops

Devops is a movement motivated to turn human capital into high performance organizational capital.
First Generation Configuration Management

Tivoli - Configuration Manager
BMC - Bladelogic
HP - Opsware
We Launch Startups.

Why are people paying 3 to 5 million for configuration management software? 3

Posted by Adam Jacob on 8/31/2007

In John Willis' response to Puppet, ILike and Infrastructure 2.0, he poses the question:

Maybe I should ask this one more time... and why are people paying in excess of 3 to 5 million a year for this. I think the answer revolves around two things.

People don't realize that manual systems administration is a prob

By "people" I mean everyone involved... many excellent systems administrators just don't see the need for week building my current systems by hand.

The result of this is that automation comes in too late in the game, after you've already got a couple of support structures built around doing it by hand. If you're a growing startup, this winds up impacting your like mine to help, or at least talking to Luke's own Reductive Labs, to help get you running quickly.

If you are the Fortune 1000, it's another thing altogether. You start to look for a way out, and someone has to be nimble enough to realize that Open Source provides a better path here.

You have to see it to believe it.

If you ask anyone working in technology whether they want the ability to rebuild the entire infrastructure. What follows, though, is "but our infrastructure can't work that way because of X" or "your systems must of functionality. They've never even seen it. So they believe you when they say it, but they don't have any. When you realize you must have it, and you already believe that it's impressively difficult to do, having someone tell you it's hard to be

Puppet, ilike and Infrastructure 2.0

Posted by Adam Jacob on 8/31/2007

John Willis, one of the founders of Gulf Breeze Software (an IBM Tivoli consulting house), met up with Luke Kanies and interviewed him about Puppet.

In addition to lots of insightful commentary on how Puppet is constructed, and a nice compare/contrast with how Tivoli is built (and you would be hard pressed to talk to someone who knows more about how Tivoli is built than John Willis, I expect,) there is also a section about ilike and HJK.

Uncomfortable with his recent celebrity at conferences, Luke told me that he has difficulty measuring his successes because he has his head so deep in the development and services of Puppet. One of his better success stories is with ilike.com, which allows users to download and share music. When ilike created one of the first Facebook applications, it grew from about 15 million users to over 6 million in a week. Luke, being the entrepreneur that he is, asked how ilike planned to manage that growth. He discovered that a services company in Seattle was managing ilike.com's infrastructure buildout using Puppet. In fact, one of the owners of that company told Luke that he makes a healthy living installing Puppet. Luke admitted that he felt pretty good to know that other people can make a living from his product.
Operations is a competitive advantage... (Secret Sauce for Startups!)

Second Generation Configuration Management

Cfengine
Puppet
Chef
History of Virtualization

- IBM 360/370 (1960/1970)
- CHROOT - Version 7 Unix 1979 (Bell Labs) and BSD in 1982 (Berkley)
- VMware (1998)
- FreeBSD Jails 2000
- XEN 2003
- Solaris Zones 2004
- OpenVZ 2005

**Amazon Web Services 2006**
- BTRFS (Oracle) 2007
- Namespaces 2007
- Cgroups (Google) 2007
- KVM 2007
- AIX LPARS (IBM) 2007
- Drawbridge (2008)
- Hyper-V (2008)
- Linux Containers - LXC (Parelles, IBM, Google) 2008
- Docker (Dotcloud Inc) 2013
- Rocket (Coreos) 2014
VM Image Sprawl in Real Life

Posted on Jan 1, 2009 by Randy Bias

A while back, Geva Perry and I were chatting about the issue of virtual machine image sprawl (Google Search), which is really little more than an extension of not-so-new traditional physical server sprawl problem. It's hard to get really hard data on how bad the vm sprawl problem is since most images exist behind firewalls or other walled gardens. However, there is one good place to get solid data and that's Amazon's own public image repository.

Real Data So, around the time we were chatting I started collecting some information on the number of public Amazon Machine Image (AMIs). This isn't a perfect sampling, but should be pretty good for most purposes.
Chapter 5. Infrastructure As Code

Adam Jacob

YOU'RE SITTING AT HOME, WATCHING A MOVIE AND EATING POPCORN, with your feet up and the family gathered around you. The phone rings—it’s your on-call system administrator. “The datacenter has been hit by a tornado—it ripped right through our cage. What do we do?”

Once you get over the obvious answer,1 you start running down the list:

1. Pause the movie.
2. Sign up for an account with a cloud computing provider, to replace the raw computing, network, and storage resources you have lost.
3. Start uploading/downloading the off-site backups of your customer and application data to the new infrastructure.
4. Provision enough servers to bring the company back online, assigning an appropriate role to each new server resource (“web server,” “database server,” “monitoring server,” etc.).
5. Change your DNS to point to your new infrastructure, with a “we got hit by a tornado” page.
6. Restore the customer and application data.
7. Remove the “we got hit by a tornado” page.
8. Finish the movie.
Building with Legos

In the six years that I have been involved in building and releasing software here at Netflix, the process has evolved and improved significantly. When I started, we would build a WAR, get it setup and tested on a production host, and then run a script that would stop tomcat on the host being pushed to, rsync the directory structure and then start tomcat again. Each host would be manually pushed to using this process, and even with very few hosts this took quite some time and a lot of human interaction (potential for mistakes).

Our next iteration was an improvement in automation, but not really in architecture. We created a web based
Virtualization

- **Type 1 Virtualization**
  - VMware ESX, XEN, Hyper-V
  - (indirectly Amazon, Rackspace, etc..)

- **Type 2 Virtualization**
  - KVM, Virtualbox, QEMU, VMware Workstation
  - (indirectly Vagrant)

- **OS Level Virtualization**
  - OpenVZ, LXC, Docker
Hypervisors vs. Linux Containers

Containers share the OS kernel of the host and thus are lightweight. However, each container must have the same OS kernel.

Containers are isolated, but share OS and, where appropriate, libs / bins.

Type 1 Hypervisor

Type 2 Hypervisor

Linux Containers

http://www.slideshare.net/BodenRussell/realizing-linux-containerslxc
Why OS Level Virtualization

• Provision in milliseconds
• Near bare metal runtime performance
• VM-like agility - it’s still “virtualization”
• Lightweight - Just enough Operating System (JeOS)
• Supported with modern Linux kernel
• Growing in popularity
Introducing Containers

Containerization uses the kernel on the host operating system to run multiple root file systems

- Each root file system is called a **container**
- Each container also has its own
  - Processes
  - Memory
  - Devices
  - Network stack
Docker?

- Isolation
- Lightweight
- Simplicity
- Workflow
- Community
Density & Footprint – Docker

- In this test, we created 150 Docker containers with CentOS, started apache & then removed them
- Average footprint was ~10MB per container
- Average start time was 240ms

- Serially booting 150 containers which run apache
  - Takes on average 36 seconds
  - Consumes about 2% of the CPU
  - Negligible HDD space
  - Spawns around 225 processes for create
  - Around 1.5 GB of memory ~ 10 MB per container
  - Expect faster results once docker addresses performance topics in the next few months

- Serially destroying 150 containers running apache
  - On average takes 9 seconds
  - We would expect destroy to be faster – likely a docker bug and will triage with the docker community

http://www.slideshare.net/BodenRussell/realizing-linux-containerslxc
Docker and the Linux Kernel

- **Docker Engine** is the program that enables containers to be distributed and run
- Docker Engine uses Linux Kernel namespaces and control groups
- Namespaces give us the isolated workspace
Docker Client and Daemon

- Client / Server architecture
- Client takes user inputs and sends them to the daemon
- Daemon runs and distributes containers
- Client and daemon can run on the same host or on different hosts
- CLI client and GUI (Kitematic)
Understanding image layers

- An image is a collection of files and some metadata
- Images are comprised of multiple layers
- A layer is also just another image
- Each image contains software you want to run
- Every image contains a base layer
- Docker uses a copy on write system
- Layers are read only
- COW/Union Filesystems (AUFS/BTRFS)
Dockerfile Examples

FROM scratch
ADD ubuntu-trusty-core-cloudimg-amd64-root.tar.gz /

# a few minor docker-specific tweaks
# see https://github.com/docker/docker/blob/master/contrib/mkimage/debootstrap
RUN echo '#!/bin/sh' > /usr/sbin/policy-rc.d \
    && echo 'exit 101' > /usr/sbin/policy-rc.d \
    && chmod +x /usr/sbin/policy-rc.d \
    \ 
    && dpkg-divert --local --rename --add /sbin/initctl \
    && cp -a /usr/sbin/policy-rc.d /sbin/initctl \
    && sed -i 's/^exit.*$/exit 0/' /sbin/initctl \
    \ 
    && echo 'force-unsafe-io' > /etc/dpkg/dpkg.cfg.d/docker-apt-speedup \
    \ 
    && echo 'DPkg::Post-Invoke { "rm -f /var/cache/apt/archives/*/deb /var/cache/apt/archives/partial/*/deb /var/cache/apt/' \
    && echo 'APT::Update::Post-Invoke { "rm -f /var/cache/apt/archives/*/deb /var/cache/apt/archives/partial/*/deb /var/cache/apt/' \
    && echo 'Dir::Cache::pkgcache ""; Dir::Cache::srcpkgcache "";' >> /etc/apt/apt.conf.d/docker-clean \
    \ 
    && echo 'Acquire::Languages "none";' > /etc/apt/apt.conf.d/docker-no-languages \
    \ 
    && echo 'Acquire::GzipIndexes "true"; Acquire::CompressionTypes::Order:: "gz";' > /etc/apt/apt.conf.d/docker-gzip-index

# enable the universe
RUN sed -i 's/^#s*\(deb.*universe\)$\//\1/g' /etc/apt/sources.list

# overwrite this with 'CMD [' in a dependent Dockerfile
CMD ["/bin/bash"]
Dockerfile Examples

```bash
Dockerfile
Change variants to be libc variants instead of build-style variants

Dockerfile.builder
Enforce UTC via /etc/localtime from the builder container

busybox.tar.xz
Add tarballs (2016-01-14)
```
Socketplane Example
Docker and Windows

- Azure
  - Azure Container Service
  - Swarm Integration

- Windows Server 2016
  - Windows Server Containers
  - Hyper-V Containers
Immutable Infrastructure

Saturday, August 13, 2011

Building with Legos

In the six years that I have been involved in building and releasing software, it has evolved and improved significantly. When I started, we would build a WAR, upload it to a production host, and then run a script that would stop the web container on the host and start it again. Each host would be manually pushed with very few hosts this took quite some time and a lot of human interaction.

Our next iteration was an improvement in automation, but not really in architecture. We would handle the process of stopping and starting things as well as extracting the new code. This meant that people could push to a number of servers at once and each server would be back up before a new deployment had failed.

Automated configuration tools (such as CFEngine, Puppet, or Chef) allow you to specify how servers should be configured, and bring new and existing machines into compliance. This helps to avoid the problem of fragile SnowflakeServers. Such tools can create PhoenixServers that can be torn down and rebuilt at will. An Immutable Server is the logical conclusion of this approach, a server that once deployed, is never modified, merely replaced with a new updated instance.
“The least-cost way to ensure that the behavior of any two hosts will remain completely identical is always to implement the same changes in the same order on both hosts.”
Management Methods

- Divergence
- Convergence
- Congruence
Assessing Docker and Containers for Five Software Delivery Use Cases

27 April 2015  G00275476

Analyst(s): Richard Watson

Summary
Docker offers application-focused, container-based virtualization to DevOps-minded developers and administrators. This document assesses Docker for use cases spanning development and test, continuous integration (CI), production deployment, and building private PaaS.

What is Docker?  Use Cases  Try It!  Install & Docs  Blog

May 26, 2015

DOCKER AND THE THREE WAYS OF DEVOPS
written by John Willis, Evangelist at Docker

Have you read Gene Kim’s The Phoenix Project? Some of the principles behind the Phoenix Project and an upcoming book I am co-authoring with Gene (The DevOps Cookbook) have been referred to as the “Three Ways of DevOps”. These are particular patterns of applying DevOps principles in a way that yields high performance outcomes.

We assert that the Three Ways describe the values and philosophies that frame the processes, procedures, practices of DevOps, as well as the prescriptive steps.

Gene Kim
Immutable Infrastructure

Sample command to start play container image

```
- run "
  --expose 80
  -p 9000:80
  giltarchitecture/apidoc-api-1-2-3
  -Dhttp.port=90
  -Dconfig.resource=xxx.conf
"
```
Unikernels are specialized virtual machine images complied from the modular stack of application code, system libraries and configuration.
Enter Unikernels

Docker Blog

Categories: General Engineering Community

January 21, 2016

Unikernel Systems Joins Docker

By Mano Marke - Posted in Docker, News - Tagged with docker, unikernel, unikernel system, unikernels

I’m happy to announce today that Unikernel Systems is part of Docker!

Unikernels compile your source code into a custom operating system that includes only the functionality required by the application logic. That makes them small, fast, and improves efficiency. Unikernel Systems was formed last year to build tools that allow developers to take advantage of a growing number of unikernel projects.
Unikernels
Unikernels

https://queue.acm.org/detail.cfm?id=2566628
Unikernels

<table>
<thead>
<tr>
<th>Unikernel</th>
<th>Language</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirage[13]</td>
<td>OCaml</td>
<td>Xen, kFreeBSD, POSIX, WWW/js</td>
</tr>
<tr>
<td>Drawbridge[16]</td>
<td>C</td>
<td>Windows &quot;picoprocess&quot;</td>
</tr>
<tr>
<td>HalVM[8]</td>
<td>Haskell</td>
<td>Xen</td>
</tr>
<tr>
<td>ErlangOnXen</td>
<td>Erlang</td>
<td>Xen</td>
</tr>
<tr>
<td>OSv[2]</td>
<td>C/Java</td>
<td>Xen, KVM</td>
</tr>
<tr>
<td>GUK</td>
<td>Java</td>
<td>Xen</td>
</tr>
<tr>
<td>NetBSD &quot;rump&quot;[9]</td>
<td>C</td>
<td>Xen, Linux kernel, POSIX</td>
</tr>
<tr>
<td>ClickOS [14]</td>
<td>C++</td>
<td>Xen</td>
</tr>
</tbody>
</table>

https://queue.acm.org/detail.cfm?id=2566628
Unikernels

RUMP KERNELS

*You can make an omelette without breaking the kitchen*

Rump kernels enable you to build the software stack you need without forcing you to reinvent the wheels. The key observation is that a software stack needs driver-like components which are conventionally tightly-knit into operating systems — even if you do not desire the limitations and infrastructure overhead of a given OS, you do need drivers.

We solve the problem by providing free, reusable, componentized, kernel quality drivers such as file systems, POSIX system calls, PCI device drivers and TCP/IP and SCSI protocol stacks. As a production-ready example, we offer the Rumpun unikernel, which clocks in at a few thousand lines of code plus rump kernel components, and supports POSIX' software directly on both raw hardware and cloud hypervisors such as KVM and Xen. Examples of Rump kernels integrated into 3rd party platforms also exist.

The article Rise and Fall of the Operating System provides an extended high-level motivation for rump kernels. The book Design and Implementation of the Anykernel and Rump Kernels gives a technical description of the fundamental operating principles and terminology. Further information is available on the wiki or interactively via the community. You can also hire consultants for commercial support.

http://rumpkernel.org/
Why Unikernels

• **Performance**
  • user-kernel context switches
  • instantiation times
  • Memory footprint

• **Security**
  • less attack surface
  • No known architecture patterns

• **Fine-grained optimisation**
  • as unikernels are constructed through a coherent compiler tool-chain, whole-system optimisation can be carried out across device drivers and application logic, potentially improving specialisation further
Enter Unikernels

Part of this is a numbers game – to run a reasonable system you might need to run 50 different services, and install 200 packages on every host. An attacker has to compromise just one of those to win - Gareth Rushgrove
Unikernel Examples

• DNS Server 446 KB
• Web Server 674 KB
• OVS Switch 393 KB
Unikernel Opportunities

• Composition and Orchestration
• Logging and Monitoring
• Networking
• Debugging
• Forces Immutability
Unikernels are unfit for production

Recently, I made the mistake of rhetorically asking if I needed to spell out why unikernels are unfit for production. The response was overwhelming: whether people feel that unikernels are wrong-headed and are looking for supporting detail or are unikernel proponents and want to know what the counter-arguments could possibly be, there is clearly a desire to hear the arguments against running unikernels in production.

production: **Unikernels are entirely undebuggable.** There are no processes, so of course there is no `ps`, no `htop`, no `strace` — but there is also no `netstat`, no `tcpdump`, no `ping`! And these are just the crude, decades-old tools. There is certainly nothing modern like `DTrace` or `MDB`. From a debugging