Dockah, Dockah, Dockah!

Containerization as a shift of paradigm for the GNU/Linux OS

Daniel Riek
Sr. Director Systems Design & Engineering
In the beginning there was Stow...

… and /usr/local, and tar, and binaries mounted from the network, ...

- Works well for a Unix host.
- Too fragile due to dependencies on each host's environment.
- Not efficient for managing artifacts.
- Doesn't scale in distributed environments (aka PCs).
Then, There Be RPM...

... and dpkg, and...

- Frozen binary distribution, reproducible builds.
- Metadata, signatures.
- Transport format for a curated content stream from a trusted source.
- Management of installed artifacts, updates.
- Implicit move to a single instance, single version model.
- Implements a late-binding model for deploying software in Ops based on an ABI contract.
- Layers move across dev/test/ops independently.
Traditional application deployment

- Single userspace runtime shared between applications.
- Environment and life cycle defined by host OS.
- OS generations define software lifecycle.
- Trend to isolate apps on hardware level.
- Managed by IT, very limited delegation.
- Stable, long maintenance, few updates, hardware-centric.
Traditional application deployment

- Very limited flexibility, resources underutilized
- Component-level life cycle management fragile in large environments.
- Stacks too complex to map into a common single instance, single version namespace.
- Side-effects of shared dependencies can not be managed at scale, induce unwanted change.
- Full distribution model with monolithic generations of content create artificial tie of userspace stacks to low level components and hardware.
Userspace Stacks Too Complex

Module Counts

http://www.modulecounts.com/

Include
- Bower (JS)
- Clojars (Clojure)
- CPAN
- CPAN (search)
- CRAN (R)
- Crates.io (Rust)
- GoDoc (Go)
- Hackage (Haskell)
- Hex.pm (Elixir/Erlang)
- LuaRocks (Lua)
- Maven Central (Java)
- MELPA (Emacs)
- npm (node.js)
- nuget (.NET)
- Packagist (PHP)
- Pear (PHP)
- PyPI
- Rubygems.org
Application Deployment via Virt & IaaS

- Application isolation per VM.
- Guest environment and lifecycle defined by application.
- Application and runtime abstracted from hardware.
- Higher flexibility at cost of increased redundancy and overhead.
- Complex multi-level management of host and VM layers
- Delegation along the Host / VM boundary.
Application Deployment via Virt & IaaS

- Too much overhead per application.
- Management too complex.
- Delegation model inefficient.
- Life cycle management in practice still too fragile.
- Power shifting from Ops to application owner.
  - Everything business is software now, so the business spends the money, not IT.
App Delivery Using Docker Containers

- Separation of system runtime and application-centric userspace runtimes
- Application consist of orchestrated services.
- Services packaged with individual runtime stack in containers.
- Maximal flexibility, minimal overhead.
- Delegation along container boundaries.
App Delivery Using Docker Containers

- Multi-instance, multi-version, multi-tenant.
  - Multi-tenant in the enterprise required to support multiple lines of business.
- Everything is a cluster, everything is scale-out, everything is high-available.
- Integration with IaaS.
Docker as Aggregate Packaging

- Frozen binary distribution, reproducible builds.
- Metadata, signatures (soon).
- Transport format for a curated content stream from a trusted source.
- Management of installed artefacts, updates.
- Move to a multi-instance, multi-version, multi-tenant model.
- Implements an early-binding model for deploying applications packaged by a developer.
- Whole stack artefacts move across dev/test/ops unmodified.
A Shift In the Personas

- The traditional Linux Distribution provides a curated content stream to ops.
- Ops aggregate the content from the distro, 3rd party ISVs and developers based on the ABI contract.
A Shift In the Personas

- The container-driven model makes the developer the primary consumer above the core system runtime.
The Atomic Architecture and Red Hat's Container Strategy
Open Standards

Red Hat works with the open source community to drive standards for containerization.

<table>
<thead>
<tr>
<th>ISOLATION WITH LINUX CONTAINERS</th>
<th>CONTAINER FORMAT WITH DOCKER</th>
<th>ORCHESTRATION WITH KUBERNETES</th>
<th>REGISTRY / CONTAINER DISCOVERY</th>
<th>PAAS &amp; DEVOPS WITH OPENShift</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Isolating applications on host operating system</td>
<td>• Interface for communications, configuration, data persistence, provisioning</td>
<td>• Orchestrate containers at scale</td>
<td>• Easily find and consume trusted container images</td>
<td>• End-to-end application lifecycle workflow on a PaaS leveraging the Red Hat ecosystem.</td>
</tr>
<tr>
<td>• Security</td>
<td>• Content agnostic</td>
<td>• Define app topologies</td>
<td>• Federate consumption libraries</td>
<td>• Portability across host systems</td>
</tr>
</tbody>
</table>
Project Atomic

- The Atomic Architecture describes the end-to-end design of an containerized OS.
- Project Atomic is an umbrella project to drive the Red Hat vision of the next generation Operating System: Transition from a monolithic UNIX OS model to a modularized, multi-Instance, multi-version, multi-tennant, application/service-centric environment.
- Not RH-specific, but driven by Red Hat into RH family of Linux OSs.
Kubernetes: Multi-Container App Definition

- Docker packages individual services into containers.
  - Multi-service containers possible but not ideal.

- Applications typically consist of multiple services.

- Kubernetes provides the generic definition and orchestration layer for containerized applications.
Containerization vs. Virtualization

- Generally complementary concepts
- Virtualization: vertical abstraction
- Containerization: horizontal segmentation
- Containers used to replace virtualization where container paradigms more applicable:
  - Application isolation
  - Lightweight delegation
  - “Application Virtualization”
  - Density
- Containers on top of Virt/Cloud common.
Red Hat Registry

- A Red Hat image registry to deliver Red Hat content.
- Everything can be used with RHEL or Atomic.
A Container Development Kit (CDK) to build RHEL-based containers.
Based on vagrant with official Red Hat-provided vagrant images.
Supported on Linux, Windows, Mac.
Building a Docker Image – EAP Example

- Build tool chain encapsulated in Vagrant environment – will set up RHEL environment.
- Developer pulls EAP image from Red Hat
- Complete layer stack is automatically retrieved.
- Layers are statically linked.
- Developer builds new layered image
  - Build controlled by Dockerfile
  - Content pulled in from RHEL, JBoss, 3rd Party, custom content repositories.
  - Dockerfile and source based (STI – Source To Image) builds provided by OpenShift
Container content built directly from native components, externally produced binary artifacts, source.
RED HAT ATOMIC ENTERPRISE PLATFORM
AND OPENShift 3

CONTAINER
CONTAINER
CONTAINER

DEVOPS TOOL AND EXPERIENCE

LANGUAGE RUNTIMES, MIDDLEWARE, DATABASES, AND OTHER SERVICES

CONTAINER ORCHESTRATION AND MANAGEMENT

CONTAINER API

RHEL
RHEL ATOMIC HOST

PHYSICAL INFRASTRUCTURE
CONTAINER-BASED APPLICATION DELIVERY SOLUTIONS

Core container runtime.
System services provided as containers on Atomic Host.
Hybrid container / locally installed app model supported in RHEL Server.
CONTAINER-BASED APPLICATION DELIVERY SOLUTIONS

Run and orchestrate multi-container based applications at scale
CONTAINER-BASED APPLICATION DELIVERY SOLUTIONS

- Application lifecycle management
- Continuous integration
- Developer experience
- Source-to-image

Develop, build, and manage container-based applications
Run and orchestrate multi-container based applications at scale

- Managed cluster of container-optimized hosts
- Orchestration and service aggregation

Run and orchestrate multi-container based applications at scale
CONTAINER-BASED APPLICATION DELIVERY SOLUTIONS

RED HAT CLOUD SUITE
- Unified management from bare metal to containers
- Scalable infrastructure
- Hybrid deployment management

OPENSHIFT ENTERPRISE by Red Hat
- Application lifecycle management
- Continuous integration
- Developer experience
- Source-to-image

RED HAT ATOMIC ENTERPRISE PLATFORM
- Managed cluster of container-optimized hosts
- Orchestration and service aggregation

Seamlessly manage from infrastructure to applications
Build scalable infrastructure based on OpenStack
Develop, build, and manage container-based applications
Run and orchestrate multi-container based applications at scale
Develop, build, and manage container-based applications
Run and orchestrate multi-container based applications at scale

RED HAT ENTERPRISE LINUX, INCLUDING ATOMIC HOST
CERTIFIED HARDWARE, VIRT & CLOUD ECOSYSTEM
OpenStack Integration

Red Hat Atomic Enterprise Platform

Service
VM
KVM

OpenShift
Kubernetes
Container
Container
VM

OpenStack shared services
compute, networking, storage

standard hardware
Atomic App - Full Application Portability

- Docker is great for individual component-containers. Most applications are more than one container.
- Kubernetes adds multi-container application orchestration. Kubernetes configs have to be copied manually.
- Atomic App defines packaging and transport multi-container applications.
- New upstream project to define spec and reference implementation.
- Included in Red Hat CDK work.

https://github.com/projectatomic/atomicapp
Conclusion

- Containers are the future of application packaging.
- DevOps is an opportunity, change induced by containerization goes far beyond it, changing the fundamentals of software distribution.
- We have to rethink existing concepts of the GNU/Linux distribution.
- Red Hat is embracing the shift of paradigm towards containerization.
- A lot of input available from RH Summit Presentations:
  - https://www.redhat.com/summit/2015/presentations/