Introduction to Virtualization and Containers

Phil Hopkins

@twitterhandle
Virtualization – What is it?
What the heck is a hypervisor? Why are there so many of them? What is a container and all the related bits and pieces? Why would we want to use them? And what is Metal-as-a-Service, Infrastructure-as-a-service, Platform-as-a-Service, or Storage-a-a-S, or all the other *aaS? How about OPNFV and Cloud Foundry? What do I need to know and how does this all fit together? This presentation will answer these questions.
Virtualization is an abstraction of computer resources. Access to the resources are consistent before and after abstraction. Resource abstraction is not limited by implementation, including the underlying physical implementation.
Virtual Machine
Virtual Machine
Operating system
Bin/Libs
Ap1
Ap2
Bin/Libs
Ap3
Ap4
Bin/Libs
Operating system
Type 2 Hypervisor
A Virtual Machine is a software construct that mimics the characteristics of a physical server.
Types of Virtualization

- Server Virtualization
- Client / Desktop / Application Virtualization
- Network Virtualization
- Storage Virtualization
- Service / Application Infrastructure Virtualization
2 ways to achieve server virtualization

• Emulators
  - Linux - Bochs, QEMU
  - Very Slow

• Segmented Use of the Host Processor
  - Most virtualization hypervisors use this technique
  - Provided by a Hypervisor
  - Faster
Hypervisors

• Type-1, native or bare-metal hypervisors
  – XEN
• Type-2 or hosted hypervisors
  – VirtualBox
• KVM on Linux has been classified as both a type 1 and a Type 2
Virtual Machine
Virtual Machine
Type 2 Hypervisor
O/S

App 1
App 2
Bin/Libs
Operating Systems

App 3
App 4
Bin/Libs
Operating Systems
Paravirtualization

- Provides specialized APIs to virtual machines to optimize their performance
  - Similar yet not identical to the underlying hardware-software interface
- Support as part of the kernel
  - drivers
- Support has been offered as part of many of the general Linux distributions since 2008
Cloud Computing

- Provides software interface to the hypervisor
- Allows remote creation and management of virtual environments
- Usually includes sever types of virtualization
  - Storage
  - Network
  - Machine
- Examples:
  - AWS, OpenStack
chroot
1979/1982

jail
2000

Process
Containers
2006

NameSpaces
Present
<table>
<thead>
<tr>
<th>MNT NS</th>
<th>UTS NS</th>
<th>PID NS</th>
<th>IPC NS</th>
<th>NET NS</th>
<th>USER NS</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>globalhost</td>
<td>PID COMMAND</td>
<td>SHMID OWNER</td>
<td>lo: UNKNOWN...</td>
<td>root 0:0</td>
</tr>
<tr>
<td>/proc</td>
<td>rootns.com</td>
<td>1/sbin/init</td>
<td>32452 root</td>
<td>eth0: UP...</td>
<td>ntp 104:109</td>
</tr>
<tr>
<td>/mnt/fsrd</td>
<td></td>
<td>2 [kthrreadd]</td>
<td>43321 boden</td>
<td>eth1: UP...</td>
<td>mysql 105:110</td>
</tr>
<tr>
<td>/mnt/fsrw</td>
<td></td>
<td>3 [ksoftirqd]</td>
<td>SEMID OWNER</td>
<td>br0: UP...</td>
<td>boden 106:111</td>
</tr>
<tr>
<td>/mnt/cdrom</td>
<td></td>
<td>4 [cpuset]</td>
<td></td>
<td>app1 IP:5000</td>
<td></td>
</tr>
<tr>
<td>/run2</td>
<td></td>
<td>5 /sbin/udevd</td>
<td></td>
<td>app2 IP:6000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>globalhost</td>
<td>6 /bin/sh</td>
<td></td>
<td>app3 IP:7000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>rootns.com</td>
<td>7 /bin/bash</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**global (i.e. root) namespace**
Why? It's still virtualization!

- Each container has:
  - its own network interface (and IP address)
    - can be bridged, routed... just like $your_favorite_vm
  - its own filesystem
    - Debian host can run Fedora container (&vice-versa)
  - isolation (security)
    - container A & B can't harm (or even see) each other
  - isolation (resource usage)
    - soft & hard quotas for RAM, CPU, I/O...
Kubernetes
Kubernetes Features

- Containerized infrastructure
- Application-centric management
- Auto-scalable infrastructure
- Environment consistency across development testing and production
- Loosely coupled infrastructure, where each component can act as a separate unit
- Higher density of resource utilization
- Predictable infrastructure which is going to be created
XaaS - *Everything as a Service
Where did XaaS come from?

- Software as a Service (SaaS) — provider’s applications running on a cloud infrastructure.
- Platform as a Service (PaaS) — Deploys various acquired applications onto the cloud infrastructure.
- Infrastructure as a Service (IaaS) — processing, storage and other computing resources Available for operating systems and applications.
Also called Servicizing

- Product-service systems (PSS)
  - cohesive delivery of products and services
- Servicizing
  - transaction value from a combination of products and services
  - provide the function of the product
  - also the service component of a product
  - customers just want the function that the product provides
What do we see today?

- Software-as-a-Service (SaaS)
- Infrastructure-as-a-Service
- Platform-as-a-Service
- Storage-as-a-Service
- Desktop-as-a-Service
- Disaster recovery-as-a-Service
- Others
OPNFV
What is it?

- NFV
  - Network Functions Virtualization
- OP
  - Open Platform
- Collaborative open source platform for network functions virtualization
- Started by the Linux Foundation in 2014
NFV - What is it?

- Implement Network Functions in Software
  - load balancers
  - Firewalls
  - Customer Premises Equipment (CPE)
  - Evolved Packet Core (EPC)
  - IP Multi-media Subsystem (IMS)
  - Broadband Network Gateways (BNG)
  - And more
- Lowers TCO by virtualizing these functions
- Avoids restrictions the hardware implementations create
OPNFV Objectives

• Create an integrated and verified open source platform that can investigate and showcase foundational NFV functionality
• Provide proactive cooperation of end users to validate OPeNFW’s strides to address community needs
• Form an open environment for NFV products founded on open standards and open source software
• Contribute and engage in open source projects that will be influenced in the OPeNFW reference platform
Cloud Foundry
What is it?

- Platform as a service (PaaS)
- Governed by the Cloud Foundry Foundation
  - Sponsored by the Linux Foundation
- Originally developed by VmWare
- Open Source
  - Source code is under an Apache License 2.0
What is PaaS?

- Usually delivers a computing platform
  - including operating system
  - programming-language execution environment
  - database
  - web server
- Deploys cloud applications
- Applications are created using provider tools:
  - Programming languages
  - Libraries
  - Services
  - Other tools
- Users can not manage or control the underlying cloud infrastructure
- Users have control over the deployed applications
Cloud Foundry Projects

- Application Runtime Project Management Committee (PMC)
  - Directs strategy, development and quality control of the core components
- BOSH PMC
  - release engineering, deployment, lifecycle management, and monitoring of distributed systems
- Extensions PMC
  - extensions to the Cloud Foundry Runtime and BOSH platform
- Open Service Broker API PMC
  - a single, simple way to deliver services to applications running within cloud native platforms