Containerize your world

Dariusz Śmigiel

PyCon PL 2015

- Full virtualization
- Para-Virtualization
- Operating system-level virtualization
- Hardware-Assisted Virtualization

1964 - IBM developed Virtual Machine Monitor

1964 - IBM developed Virtual Machine Monitor

- partitions allowed mainframes to multitask
- run multiple applications and processes at the same time

1964 - IBM developed Virtual Machine Monitor

- partitions allowed mainframes to multitask
- run multiple applications and processes at the same time

Why?

- Very expensive hardware
- underutilization was a problem – provide way to fully leverage investment

1980s - 1990s

- hardware is cheaper

1980s - 1990s

- hardware is cheaper
- x86 architecture doesn't support virtualization

1980s - 1990s

- hardware is cheaper
- x86 architecture doesn't support virtualization
- relic of previous era

1980s - 1990s

- hardware is cheaper
- x86 architecture doesn't support virtualization
- relic of previous era
- client-server applications

1980s - 1990s

- hardware is cheaper
- x86 architecture doesn't support virtualization
- relic of previous era
- client-server applications
- One application per server

- Low Utilization 5%-15%
- Extra costs of power and cooling requirements
- IT management costs
- Insufficient failover and disaster protection

Host server

Underlying hardware, provides computing resources:

- processor
- memory
- disk
- network
- ...

Guest Virtual Machine

Separate and independent instance of an operating system and application software

Virtual Machine Monitor (hypervisor) - virtualizing fill set of hardware resources, including:

- processor(s)
- memory
- storage
- peripheral devices

1974 - "Formal Requirements for Virtualizable Third Generation Architectures"

1974 - "Formal Requirements for Virtualizable Third Generation Architectures"
Set of conditions sufficient for a computer architecture to support system virtualization efficiently.

- Equivalence/Fidelity - software should exhibit behavior essentially identical to equivalent hardware (barring timing effects);

1974 - "Formal Requirements for Virtualizable Third Generation Architectures"
Set of conditions sufficient for a computer architecture to support system virtualization efficiently.

- Equivalence/Fidelity - software should exhibit behavior essentially identical to equivalent hardware (barring timing effects);
- Efficiency/Performance - vast majority of machine instructions must be executed by hardware, without VMM intervention

1974 - "Formal Requirements for Virtualizable Third Generation Architectures"
Set of conditions sufficient for a computer architecture to support system virtualization efficiently.

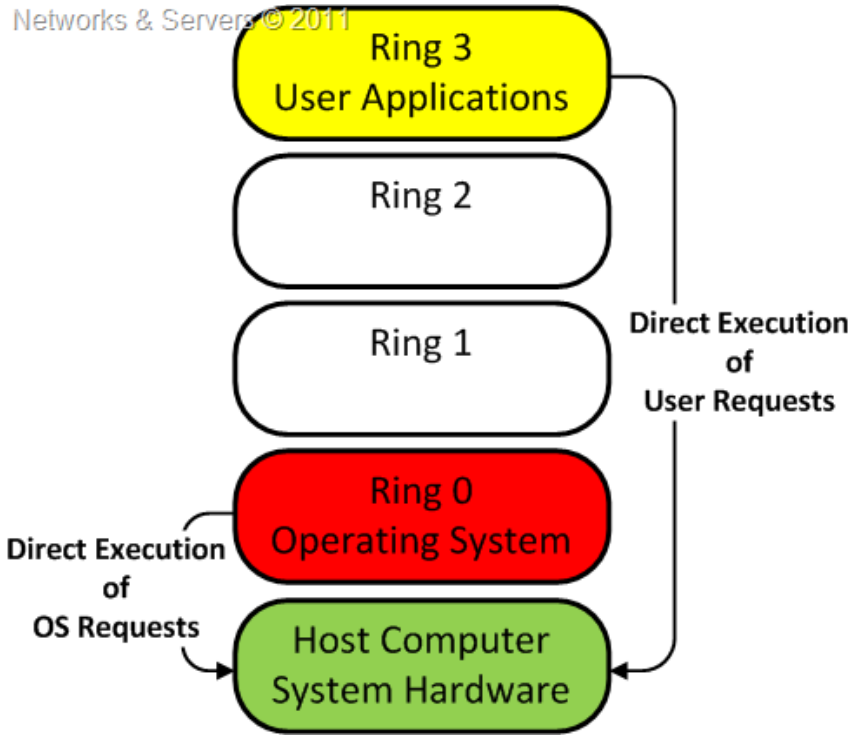
- Equivalence/Fidelity - software should exhibit behavior essentially identical to equivalent hardware (barring timing effects);
- Efficiency/Performance - vast majority of machine instructions must be executed by hardware, without VMM intervention
- Resource Control/Safety - VMM must be in complete control of virtualized resources.

x86 architecture offers protection levels (rings).

x86 architecture offers protection levels (rings).
Ring 0 has the highest privilege – operating system kernel (system space, kernel mode or supervisor mode)

x86 architecture offers protection levels (rings).
Ring 0 has the highest privilege – operating system kernel (system space, kernel mode or supervisor mode)
Ring 1-3 run appications – typically Ring 3.

x86 architecture offers protection levels (rings).
Ring 0 has the highest privilege – operating system kernel (system space, kernel mode or supervisor mode)
Ring 1-3 run appications – typically Ring 3.



OS expects to be running on bare-metal hardware (Ring 0).
Some instructions can't be virtualized: have different semantics when not executed in Ring 0.
Problem with trapping and translating instruction at runtime.

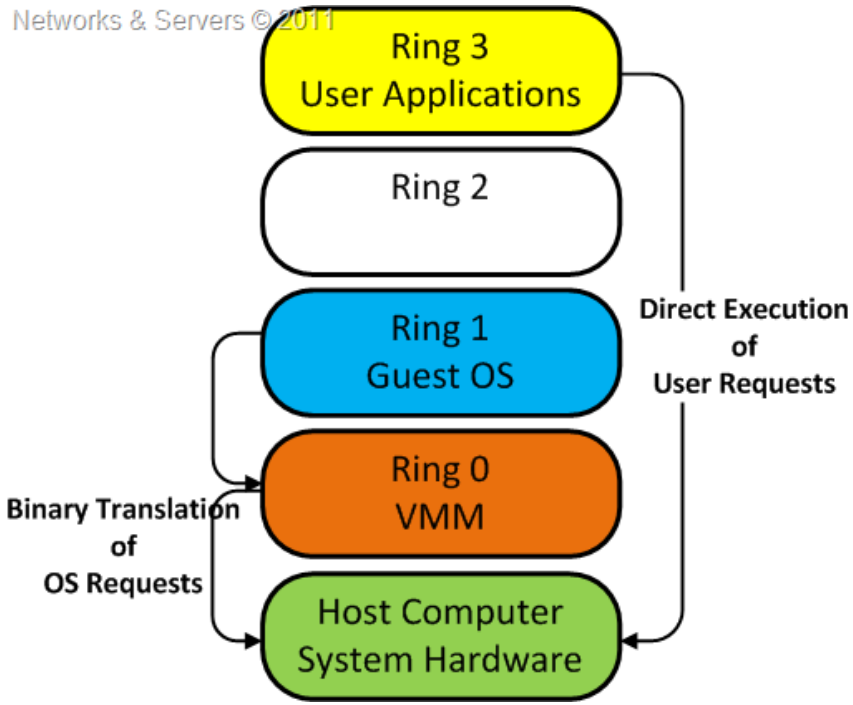
In 1998, VMware virtualized x86 platform.

In 1998, VMware virtualized x86 platform.

Developed binary translation techniques: allow running VMM in Ring 0 (isolation and performance).

In 1998, VMware virtualized x86 platform.
Developed binary translation techniques: allow running VMM in Ring 0 (isolation and performance).
Operating System is moved to user-level ring (with greater privilege than Ring 3, but less than Ring 0).

In 1998, VMware virtualized x86 platform.
Developed binary translation techniques: allow running VMM in Ring 0 (isolation and performance).
Operating System is moved to user-level ring (with greater privilege than Ring 3, but less than Ring 0).



Guest OS makes sytem calls to emulated hardware.

Guest OS makes sytem calls to emulated hardware.
Calls are intercepted by virtualization hypervisor which maps them onto real, underlying hardware.

Guest OS makes sytem calls to emulated hardware.
Calls are intercepted by virtualization hypervisor which maps them onto real, underlying hardware.
Hypervisor provides complete independence and autonomy.

Guest OS makes sytem calls to emulated hardware.
Calls are intercepted by virtualization hypervisor which maps them onto real, underlying hardware.
Hypervisor provides complete independence and autonomy.
Hypervisor monitors and controls physical resources.

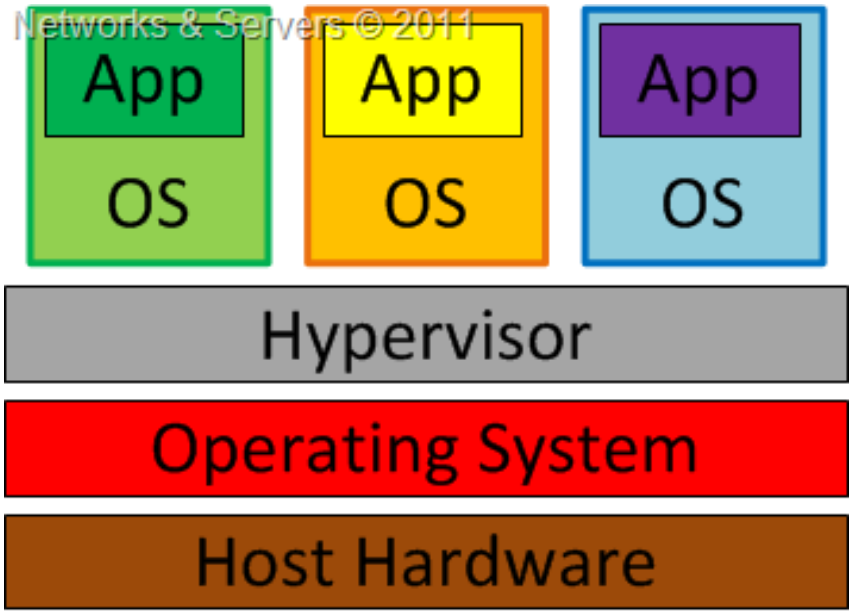
- Type 1 Hypervisor – bare-metal virtualization
- Type 2 Hypervisor – guest OS virtualization
- Embedded Hypervisor – Kernel Linux Virtualization

Guest OS virtualization (Hosted Virtualization).

Guest OS virtualization (Hosted Virtualization).
OS provides abstraction layer, that allows other OSes to reside within - creating VMs.

Guest OS virtualization (Hosted Virtualization).
OS provides abstraction layer, that allows other OSes to reside within - creating VMs.
Examples: Virtualbox, VMware Server, Microsoft Virtual PC

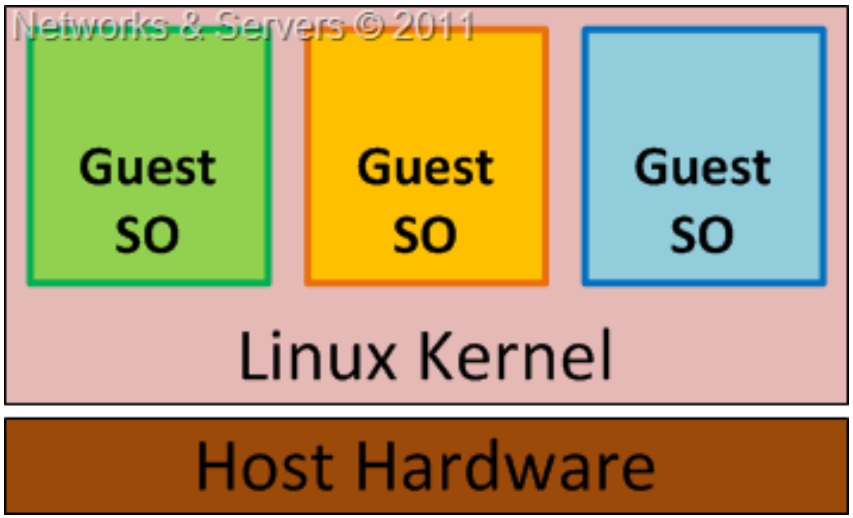
Guest OS virtualization (Hosted Virtualization).
OS provides abstraction layer, that allows other OSes to reside within - creating VMs.
Examples: Virtualbox, VMware Server, Microsoft Virtual PC



Kernel Level Virtualization - kernel contains extensions designed to manage and control multiple virtual machines.

Kernel Level Virtualization - kernel contains extensions designed to manage and control multiple virtual machines.
Examples: KVM

Kernel Level Virtualization - kernel contains extensions designed to manage and control multiple virtual machines.
Examples: KVM



- truly isolated guest OS

- truly isolated guest OS
- guest OS is not aware of being virtualized

- truly isolated guest OS
- guest OS is not aware of being virtualized
- VMM provides standarized hardware environment

- truly isolated guest OS
- guest OS is not aware of being virtualized
- VMM provides standarized hardware environment
- full virtualization offers the best isolation and security

- problem with performance;
- hypervisor must contain interfaces to resources of machine - device drivers.

Kernel of OS allows for multiple, isolated user-space instances.

Kernel of OS allows for multiple, isolated user-space instances.
Runs on top of existing host.

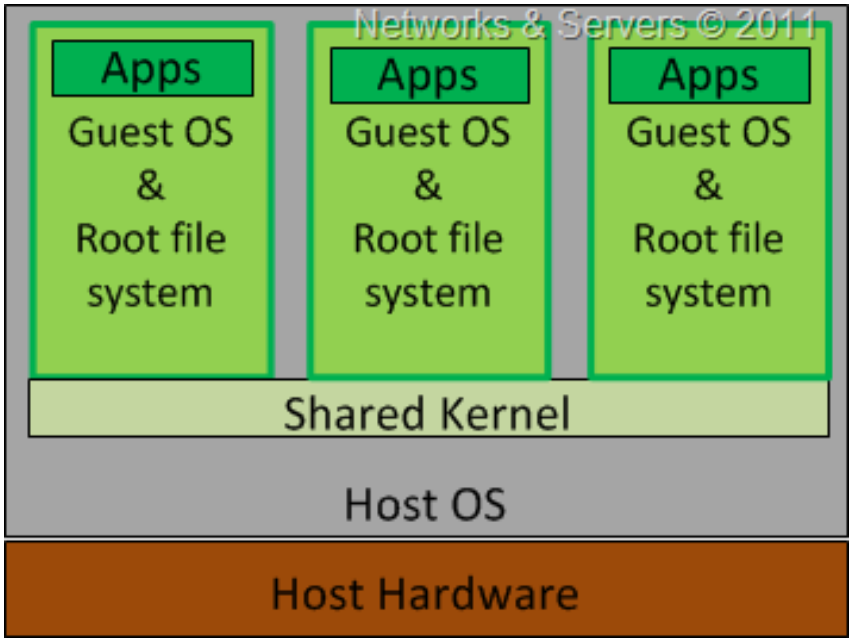
Kernel of OS allows for multiple, isolated user-space instances.
Runs on top of existing host.
Provides set of libraries that application interact with.

Kernel of OS allows for multiple, isolated user-space instances.
Runs on top of existing host.
Provides set of libraries that application interact with.
Provides "illusion" of running on dedicated machine.

Containers, Virtual Private Servers, Virtual Environments

Containers, Virtual Private Servers, Virtual Environments
Examples: Docker, rkt, OpenVZ, lxc

Containers, Virtual Private Servers, Virtual Environments
Examples: Docker, rkt, OpenVZ, lxc



- virtualization usually imposes little to no overhead;

- virtualization usually imposes little to no overhead;
- good for webhosting companies with multiple virtual web servers;

- virtualization usually imposes little to no overhead;
- good for webhosting companies with multiple virtual web servers;
- patches and modifications are applied once, for every container

- every guest OS must be identical or similar to host, in terms of version number and patch level

- 14 September 2006: Rohit Seth (Google) provides first version patchset of containers to Linux kernel;

- 14 September 2006: Rohit Seth (Google) provides first version patchset of containers to Linux kernel;
- 29 May 2007: Paul Menage, based on Rohit's work, sends updated patchset.

- 14 September 2006: Rohit Seth (Google) provides first version patchset of containers to Linux kernel;
- 29 May 2007: Paul Menage, based on Rohit's work, sends updated patchset.
- "process containers" changes name to cgroups (control groups) lands in Linux kernel

- 14 September 2006: Rohit Seth (Google) provides first version patchset of containers to Linux kernel;
- 29 May 2007: Paul Menage, based on Rohit's work, sends updated patchset.
- "process containers" changes name to cgroups (control groups) lands in Linux kernel
- Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network) of a collection of processes.

- provide unified interface

- provide unified interface
- resource limitation: can be set to not exceed configured memory limit;

- provide unified interface
- resource limitation: can be set to not exceed configured memory limit;
- prioritization: may get larger share of CPU utilization or I/O throughput

- provide unified interface
- resource limitation: can be set to not exceed configured memory limit;
- prioritization: may get larger share of CPU utilization or I/O throughput
- accounting: measures how much resources systems use (billing purposes)

- provide unified interface
- resource limitation: can be set to not exceed configured memory limit;
- prioritization: may get larger share of CPU utilization or I/O throughput
- accounting: measures how much resources systems use (billing purposes)
- control: freezing groups of processes

LXC combines cgroups and support for isolated namespaces to provide isolated environment for applications.

LXC combines cgroups and support for isolated namespaces to provide isolated environment for applications.
Originally, LXC containers were not very secure. Before kernel 3.8, root user of guest system, could run code on host system with root privileges.
LXC 1.0 (released on 20 February 2014), added "unprivileged containers" - cannot access hardware directly.

LXC combines cgroups and support for isolated namespaces to provide isolated environment for applications.
Originally, LXC containers were not very secure. Before kernel 3.8, root user of guest system, could run code on host system with root privileges.
LXC 1.0 (released on 20 February 2014), added "unprivileged containers" - cannot access hardware directly.
LXC 1.0 is intended to be supported for five years.

LXC combines cgroups and support for isolated namespaces to provide isolated environment for applications.
Originally, LXC containers were not very secure. Before kernel 3.8, root user of guest system, could run code on host system with root privileges.
LXC 1.0 (released on 20 February 2014), added "unprivileged containers" - cannot access hardware directly.
LXC 1.0 is intended to be supported for five years.
LXC runs on vanilla Linux kernel.

Docker was released as open source in March 13, 2013.

Docker was released as open source in March 13, 2013.
Used LXC as default execution environment.

Docker was released as open source in March 13, 2013.
Used LXC as default execution environment.
With Docker 0.9 release (March 10, 2014), Docker changes default LXC to libcontainer (written in Go, to access kernel's container APIs directly).

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

- composable: all tools downloadable, but independent and composable

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

- composable: all tools downloadable, but independent and composable
- security: isolation should be pluggable

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

- composable: all tools downloadable, but independent and composable
- security: isolation should be pluggable
- image distribution

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

- composable: all tools downloadable, but independent and composable
- security: isolation should be pluggable
- image distribution
- open: format and runtime should be well-specified and developed by a community.

At September 7, 2013 Docker removed manifest with informations about "standard container" assumptions.
CoreOS starts with rkt (rocket) to provide standarized format for containers:

- composable: all tools downloadable, but independent and composable
- security: isolation should be pluggable
- image distribution
- open: format and runtime should be well-specified and developed by a community.

rkt 0.1.0 released at December 1, 2014.

Open Container Initiative

On June 22, 2015 at Dockercon there was an information about "Open Container Project". Initiative to gather all companies around containers and protect from fragmentation.

–

Cloud Native Computing Foundation

On July 21, 2015 19 companies created open source foundation. Foundation aims to specify how clouds should be architected to serve modern applications.

- Networks & Servers: http://networksandservers.blogspot.com/2011/11/server-virtualization-explained.html
- Origins: http://lwn.net/Articles/236038/
- cgroups: https://en.wikipedia.org/wiki/Cgroups
- cgroups redesign: http://www.linux.com/news/featured-blogs/200-libby-clark/733595-all-about-the-linux-kernel-cgroups-redesign
- LXC: https://en.wikipedia.org/wiki/LXC
- Docker: https://en.wikipedia.org/wiki/Docker_(software)
- Docker 0.9: https://blog.docker.com/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/
- rkt: https://coreos.com/blog/rocket/
- Open Container Initiative: https://www.opencontainers.org/pressrelease/