

(Kernel) Isolation – PV, HVM, OS-V technologies in Linux

Introduction and description of the isolation differences between HM, PV and OS-level virt. technologies.

Zdeněk Kubala
Senior QA Engineer
zkubala@suse.com
@n1djz88

Paravirtualization



Paravirtualization (a.k.a. PV - Xen)

- Not kernel module, uses a hypervisor(domain 0)
- Guest OS has to be **aware** of the fact it is being paravirtualized(Kernel 3.0+).
- Hypervisor provides **ABI** to communicate and Guest OS calls it



Paravirtualization (a.k.a. PV)

- “No” Performance losses (direct access to resources)
- Faster boot - Can boot kernel directly (no bootloader)
- Guests uses **own** kernel
- Isolation on the underlying OS – processes could be secured by Apparmor/Selinux



The background features abstract geometric shapes in teal and green. A large teal shape is on the left, and a green shape is on the right, separated by a white diagonal line. The text is centered in the teal area.

Hardware-assisted virtualization

Hardware-assisted virtualization (a.k.a. HVM)

For example: KVM or Xen

- **Using „hypervisor“ - guests are completely isolated**

binary translation to trap and virtualize non-virtualized instructions => emulation

- Has own bootloder
- Has own kernel
- **Not** modified OS.



Hardware-assisted virtualization (a.k.a. HVM)

- All resources are handled in-directly through emulation.
- Nowadays **PVHVM** can be used if OS supports it (Kernel 2.6.32+)
- Needs CPU flags (Intel *vmx* | AMD *svm*)



Operating-system- level virtualization

The background features abstract geometric shapes in two shades of green. A large teal shape occupies the left and top portions, while a lighter green shape is on the right. A white diagonal line separates the two green areas.

Operating-system-level virt. (a.k.a. containers)

When we talk about containers we can think about a book in a shelf. There are multiple chapters in the book. Every chapter has a different "story" but they belong to the same piece of book.



Operating-system-level virt. (a.k.a. containers)

- Sometimes called as "**jail on steroids**".
- Containers provide an additional layer of the security by isolating resources on a OS level
- Can be used together with apparmor/SELinux to enhance security



Operating-system-level virt. (a.k.a. containers)

- Solves issues with shared libraries(multiple versions) and helps with keeping OS clean
- Easily destroyed
- Sharing the kernel with the host



Differences between virtual machines & containers

The background features a large teal shape on the left and a large green shape on the right, separated by a white diagonal line. The shapes are angular and geometric, resembling stylized arrows or chevrons pointing towards the right.

Differences between virt. machines & containers

- VMs are "heavier" to setup/start - in general
- OS boot takes up to minutes (PV/HVM difference)
- HW isolation on a hypervisor level(HVM/PV/PVHVM)
- Qemu process represents virtual machine, storage backend involved




Differences between virt. machines & containers

- Lightweight (MiB - "hundreds of MiB")
- Can be application oriented
- Isolation on an OS level - process tree



Containers technologies

The background features abstract geometric shapes. A large teal shape occupies the left and top-left areas, while a bright green shape is on the right. A white diagonal line separates the teal and green areas, creating a sense of depth and movement.

Containers technologies

- chroot *1982
- OpenVZ *2005
- lxc(lxd) *2008
- docker *2013
- systemd-nspawn *2013



Containers technologies

chroot *1982

- partial file system isolation
- nested virtualization



Containers technologies

OpenVZ *2005

- file system isolation
- disk quotas (ZFS)
- IO limiting
- memory limits
- cpu quotas
- network isolation
- partial nested virtualization
- live migration
- root isolation



Containers technologies

lxc(lxd) *2008

- file system isolation
- partial disk quotas (lvm/btrfs)
- partial IO limiting (btrfs)
- memory limits
- cpu quotas
- network isolation
- partial nested virtualization
- root isolation



Containers technologies

docker *2013

- file system isolation
- IO limiting (since 1.10)
- memory limits
- cpu quotas
- network isolation
- partial nested virtualization
- root isolation (since 1.10)



Containers technologies

systemd-nspawn *2013

- file system isolation
- disk quotas
- partial IO limiting (systemd+Cgroups)
- memory limits (systemd+Cgroups)
- cpu quotas (systemd+Cgroups)
- network isolation
- nested virtualization
- root isolation





What are they using
to isolate resources?

What are they using to isolate resources?

- PID namespace - Process identifiers and capabilities
- UTS namespace - Host and domain name
- MNT namespace - File system access and structure



What are they using to isolate resources?

- IPC namespace -Process communication over shared memory
- NET namespace -Network access and structure
- USR namespace -User names and identifiers



What are they using to isolate resources?

- Cgroups

Resource protection(cpu usage, memory usage, io)



When to choose containers and when vms

When to choose containers

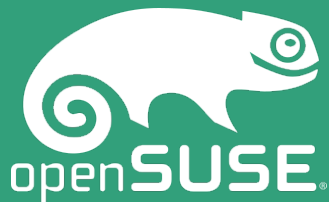
- testing a new application(from source - from the internet)
- fast deployments - "iso" template for the application(or for whole cycle)



When to choose vms

- wider isolation (running in the process, access to resources is filtered/emulated HVM or through api/drivers PV)
- "sandboxes" for customers





Questions?

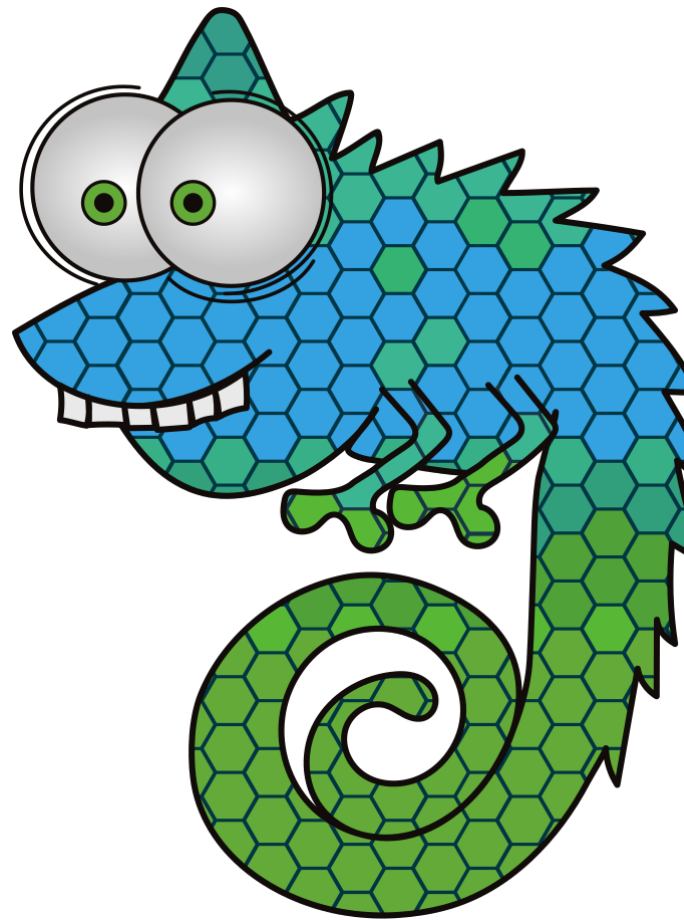
Sources:

https://en.wikipedia.org/wiki/Hardware-assisted_virtualization

https://en.wikipedia.org/wiki/Operating-system-level_virtualization

<http://www.linux-magazine.com/Issues/2016/184/systemd-nspawn>

Zdeněk Kubala
Senior QA Engineer
zkubala@suse.com
@n1djz88



Thank you

Zdeněk Kubala
Senior QA Engineer
zkubala@suse.com
@n1djz88

Join Us at www.opensuse.org



License

This slide deck is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license.

It can be shared and adapted for any purpose (even commercially) as long as Attribution is given and any derivative work is distributed under the same license.

Details can be found at <https://creativecommons.org/licenses/by-sa/4.0/>

General Disclaimer

This document is not to be construed as a promise by any participating organisation to develop, deliver, or market a product. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. openSUSE makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. The development, release, and timing of features or functionality described for openSUSE products remains at the sole discretion of openSUSE. Further, openSUSE reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All openSUSE marks referenced in this presentation are trademarks or registered trademarks of SUSE LLC, in the United States and other countries. All third-party trademarks are the property of their respective owners.

Credits

Template
Richard Brown
rbrown@opensuse.org

Design & Inspiration
openSUSE Design Team
<http://opensuse.github.io/branding-guidelines/>