Virtualisatie

en KVM

Oscar Buse

14 februari 2017

Linux User Group Nijmegen

Overzicht onderwerpen

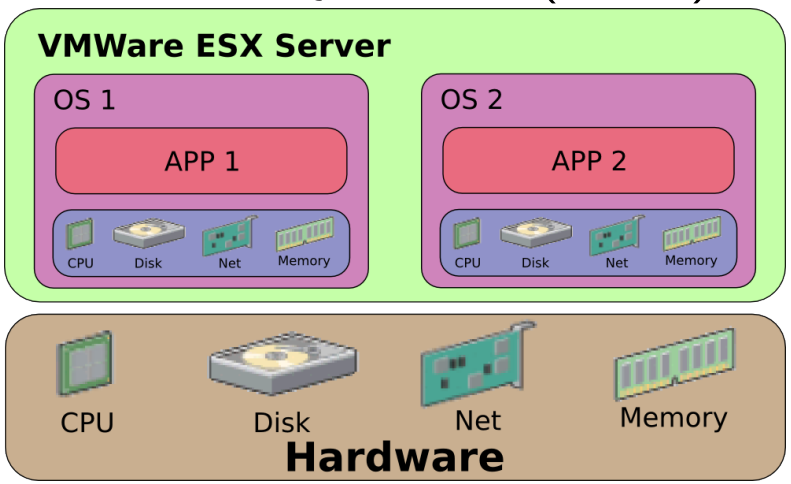
- Terminologie.
- Historie.
- Definitie van virtualisatie?
- KVM, QEMU en libvirt.
  - KVM - command line en grafisch.
  - Virtuele netwerken.
  - Storage pools en volumes.
  - Maken van een VM, installatie OS.
- Concluderend.

Terminologie

virtualisatie wikipedia: "virtualization refers to the act of creating a virtual (rather than actual) version of something, including virtual computer hardware platforms, storage devices, and computer network resources."

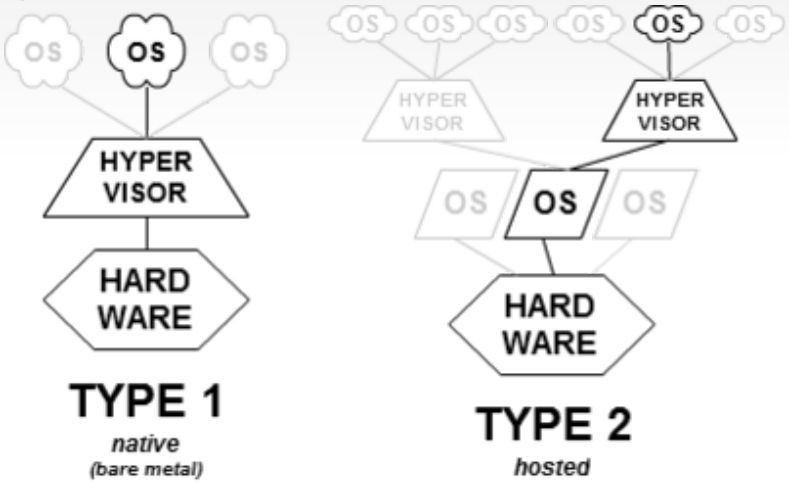
host het "host" systeem: het systeem wat guest systemen ondersteunt.

guest de virtuele systemen (VM's).



hardware emulatie hardware wat door het host systeem (de it hypervisor) als virtuele hardware aan de guests wordt gepresenteerd.

hypervisor software welke runt op het host systeem en de onderliggende hardware voor guest systemen virtualiseert (beschikbaar maakt middels hardware emulatie).



container "lichtgewicht VM.." (later meer).

Historie

- Vóór virtualisatie: veel "losse" servers:
  - Verspilling van CPU, RAM, disk, netwerk en (niet te vergeten) power resources (bij InterNLnet ging het stroomverbruik van het datacenter met meer dan 20% naar beneden (90% > 6x%)).
  - Hoge kosten.
  - Lange "Time to market".
  - Meer storingsgevoelig (..).
  - ...
- Belangrijk (in het begin): Xen en VMware:
  - Xen: *para virtualisatie:* aangepaste kernel voor **guest** systeem nodig.
  - VMware: hypervisor ESX. '*binary translation*': guest calls door de hypervisor vertaald (geen aangepast guest systeem nodig).

- HVM (Hardware Virtual Machine). Bv. Intel VT-x en AMD-V (vanaf ca. 2006).
  - Extra instructies op de CPU speciaal voor virtualisatie.
  - Maakt volledige virtualisatie makkelijker (juist vanwege de complexe x86 architectuur was VT-x/AMD-V erg welkom).
  - Tevens een enorme snelheidswinst.
  - Xen, VMWare, KVM, ...
  - Alleen voor cpu/memory. Access van netwerk en disk nog steeds via emulatie (door de hypervisor). Wel vaak via speciale drivers voor zowel guest als hypervisor (PV's: Para virtualized Drivers).

Dus wat is virtualisatie?

Er is niet 1 definitie: erg afhankelijk van implementatie:

- ~~para?~~
- ~~binary translation?~~
- volledig?
- containers?

Belangrijk dat je de mogelijkheden kent :)

KVM, QEMU en libvirt

- KVM:
  - Onderdeel van de kernel van het host systeem.
  - Geen complete hypervisor.
  - Maakt gebruik van hardware virtualisatie (VT-x/AMD-V). Verzorgt de mapping tussen fysieke en virtuele CPU's.
- QEMU (Quick EMUlator):
  - Compleet op zichzelf. Emuleert hardware (ook disken, netwerk, PCI, USB, ...)
  - Werkt samen met KVM maar kan ook geheel zelfstandig als hypervisor dienen.
- libvirt: software voor het managen van de hypervisor en guests.
  - daemon libvirtd
  - xml voor het definieren van VM's (en containers), netwerk, storage
  - cli -en grafische tools
  - voor KVM, QEMU, Xen, VMWare ESX, LXC, ...

Hypervisor is hier een combinatie van KVM en QEMU.

Installatie KVM/Qemu/libvirt

Bijvoorbeeld voor CentOS:

```
# yum -y install qemu-kvm libvirt virt-install bridge-utils
# systemctl enable libvirtd
# systemctl start libvirtd
```

Ook nodig voor het maken van een VM:

- Netwerk (automatisch wordt bij de install virbr0 (192.168.122.1/24) actief (voor het default virtueel netwerk)).
- Storage ("pools").

Keuze voor beheer uit cli (`virsh`) en/of grafische tool (`virt-manager`).

Virtueel netwerk

VM's kunnen we een eigen virtueel netwerk geven.
Dit kan met virt-manager (grafisch).
Maar kan ook met een template en de cli:

```
# cat /root/netwerk1.xml
<network>
  <name>netwerk1</name>
  <bridge name='virbr1' stp='on' delay ='0' />
  <forward mode='nat'>
    <nat>
      port start='1024' end='65535' />
    </nat>
  </forward>
  <ip address='192.168.1.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.1.2' end='192.168.1.10' />
    </dhcp>
  </ip>
</network>
```

Zie ook: `http://libvirt.org/formatnetwork.html`

```
# virsh
virsh# net-define /root/netwerk1.xml
virsh# net-autostart netwerk1
virsh# net-start netwerk1
virsh# net-info netwerk1
virsh# net-list
virsh# net-list --all
```

Storage pools en volumes

Met libvirt kun je storage pools maken.

- de default storage pool is in /var/lib/libvirt/images
- een storage pool bevat storage volumes (de images)

Maak een nieuwe pool:

```
# mkdir /var/lib/libvirt/pool1
# virsh
virsh# pool-define-as pool1 dir - - - - /var/lib/libvirt/pool1
(virsh# pool-define-as pool1 --type dir \
                             --target /var/lib/libvirt/pool1)
virsh# pool-autostart pool1
virsh# pool-start pool1

virsh# vol-create-as pool1 debian8.raw 10G
```

Maak een VM

```
# virt-install -r 1024 --vcpus=1 -n debian8 \
  -f /var/lib/libvirt/pool1/debian8.raw \
  --network bridge=virbr1 \
  --os-type=linux \
  --os-variant=generic \
  --initrd-inject=/root/preseed.cfg --extra-args="auto" --name d-i
  <redhat>
  --initrd-inject=/var/tmp/debian-minimal.ks \
  --extra-args="ks=file:/debian-minimal.ks \
  </redhat>
  --cdrom /home/user1/Downloads/debian-8.5.0-amd64-netinst.iso
```

Enkele commando's:

```
virsh# list --all
virsh# list --autostart
virsh# autostart debian8
virsh# start debian8
virsh# destroy debian8
virsh# undefine debian8
# virsh help | less
```

Verkrijg het IP-adres:

```
virsh# domifaddr debian8
```

Migreren van VM's

```
# virsh destroy "name_of_vm"
# scp /tmp/"vmname".xml "hostX:"
# scp /var/lib/libvirt/images/"vmname".img "hostX:"
```

Op de "hostX": verwijder uuid's in xml-file.

```
hostX:# virsh define "vmname".xml
hostX:# virsh start "vmname"
```

Hypervisor vs container

De functie van de hypervisor is wat bleekjes tegenwoordig.
Enkele nadelen:

- Netwerk/disk access via "paravirtualized drivers" (PV drivers). Verschillende PV-drivers nodig per hypervisor (VMware's ESX, Xen, KVM) én per guest OS (windows 7, windows 10, redhat, ubuntu etc..) > veel code!
- OS-patching in je VM's
- Relatief veel resources nodig per VM (vergeleken met containers).
- Management niet meer zo'n USP: er komen steeds meer "container management" tools zoals bv. "Kubernetes".
- Security is ook sterk verbeterd mbt containers. Zeker sinds de extra instructies op de CPU (VT-x/AMD-V).
- Een hypervisor is goed in het ondersteunen van een hybride omgeving (verschillende OSsen). Maar wat is het voordeel daarvan?

We kijken dan ook niet alleen naar de "gewone" (hypervisor) virtualisatie (met KVM) maar ook naar containers:

- Vandaag: KVM (+ QEMU en libvirt)
- Volgende keer: containers (docker)
- Daarna: Kubernetes (..)

Bier?