VIRTUALISIERUNG IN NETZWERKEN

Mario Taschwer

Inf2school-Workshop, Klagenfurt

1. März 2013

AGENDA

- Einführung
- Virtualisierungskonzepte
- Demo: Desktop-Virtualisierung mit Virtualbox
- Netzwerkkonfiguration für Virtualbox-VMs
- Demo: Host-only Netzwerk in Virtualbox
- Vor- und Nachteile der Virtualisierung
- Demo: Server-Virtualisierung mit Proxmox VE
- Diskussion

VORBEMERKUNG

- Workshop-Titel ist ungünstig, besser: "Virtualisierung von Betriebssystemen"
  - Netzwerkkonfiguration wird besonders berücksichtigt
  - Praktische Übungen müssen entfallen, dafür Demos
- Nicht behandelt werden:
  - Speichervirtualisierung: z.B. SAN (storage area network)
  - Virtuelle lokale Netze (VLAN)
- Einführung und Virtualisierungskonzepte:
  - Siehe Foliensatz von Baun, Mauch (2011)

DEMO: VIRTUALBOX

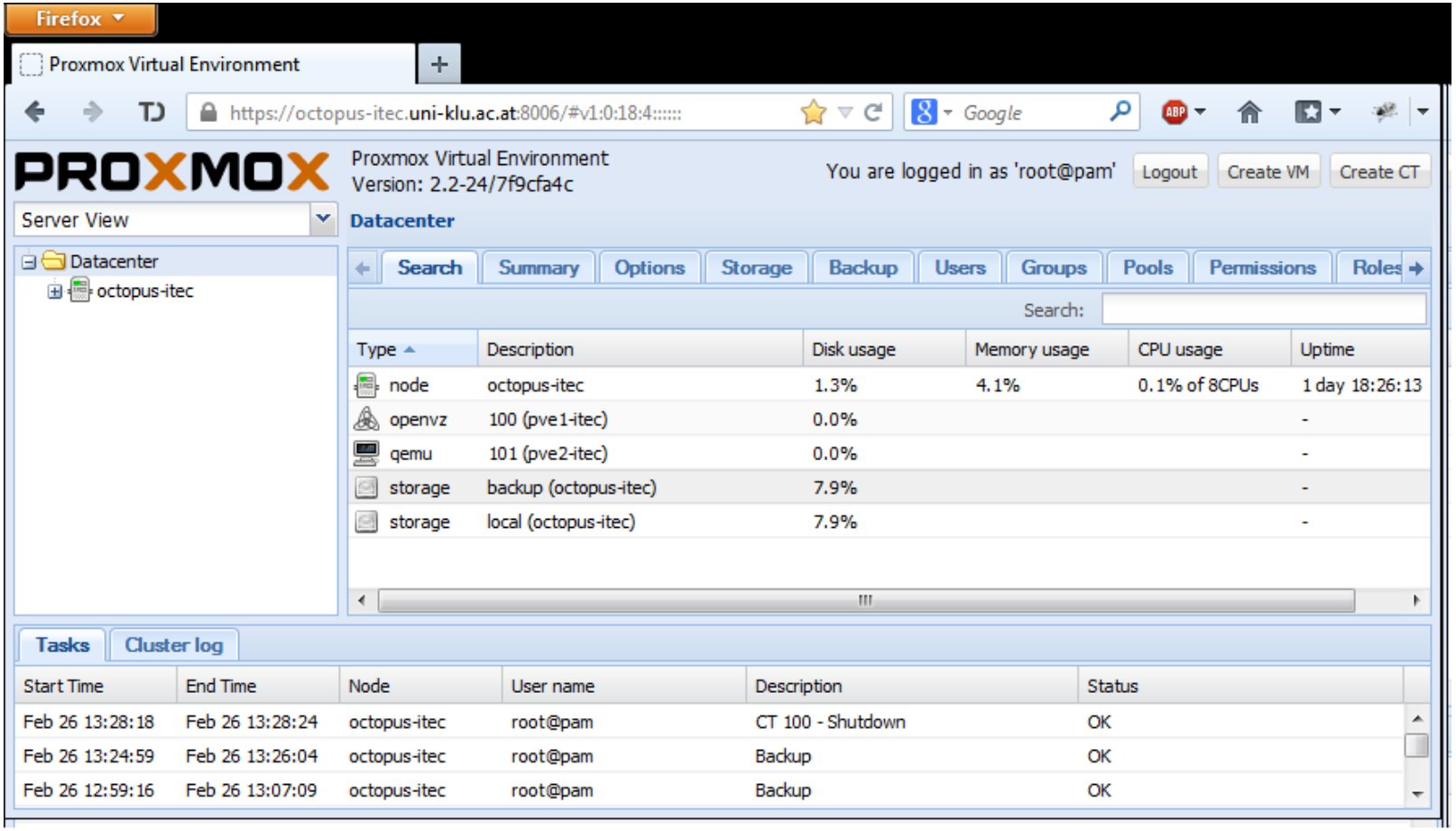
- Virtualbox-Images für freie Betriebssysteme:
  - http://virtualboxes.org/images/
- Linux-Gastsystem auf Win7-Wirtsystem:
  - Prozessor, Massenspeicher, Netzwerk (NAT)
  - Demo Internetzugriff
  - Installation der Virtualbox-Gasterweiterungen
    - https://forums.virtualbox.org/viewtopic.php?t=15679
  - Gemeinsame Ordner für Gast- und Wirtsystem
  - Klonen eines Gastsystems

NETZWERKE IN VIRTUALBOX

- NAT- und host-only Netzwerke in Virtualbox:
  - Siehe Foliensatz von Hahn (2012).

- Demo: lokale Netzwerkdienste in 2 Linux-VMs
  - Zuweisung der IP-Adressen im host-only Netzwerk
  - Ping-Tests, Zugriff auf ssh- und http-Dienste
  - Internetzugriff und Routing-Tabellen auf VMs

DEMO: PROXMOX VE (1)



DEMO: PROXMOX VE (2)

- Betriebssystem-Virtualisierung (Container, CT):
  - Template herunterladen (Storage ‘local’, Content)
    - Siehe auch: http://pve.proxmox.com/wiki/Category:Virtual_Appliances
  - Neuen Container erzeugen (Create CT)
  - Netzwerkschnittstelle venet:
    - in-host routing, keine MAC-Adresse
  - Internetzugriff und http-Dienst testen (Java-Console)
  - Backup im ‘suspend’- und ‘snapshot’-Modus

DEMO: PROXMOX VE (3)

- Vollständige Virtualisierung mit KVM:
  - Anlegen einer VM für Windows 7
  - Prozessor, Festplatte, DVD (Installationsmedium)
  - Netzwerk: bridge (vmbr0), statische IP-Adresse
  - Internetzugriff und Netzwerkdienste (RDP)
  - Live Snapshots
- Weiterführende Links:
  - http://unix-heaven.org/proxmox-ve-kvm-template
  - http://pve.proxmox.com/wiki/Resizing_disks

QUELLENANGABEN

- Virtualisierungskonzepte:
  - Baun, Mauch: VO Cluster-, Grid-, Cloud-Computing, SS 2011, http://www.informatik.hs-mannheim.de/~baun/CGC11/Skript/folien_cgc_vorlesung_13_SS2011.pdf
  - Hahn: Networking between host and guest VMs, 2012, http://de.slideshare.net/powerhan96/networking-between-host-and-guest-v-ms-in-virtual-box
- Virtualbox: https://www.virtualbox.org/
- Proxmox VE: http://pve.proxmox.com/
  - Artikel im Linux-Magazin 10/2012: http://www.linux-magazin.de/Ausgaben/2012/10/Proxmox-VE

13.Vorlesung Cluster-, Grid- und Cloud-Computing Hochschule Mannheim

Christian Baun, Viktor Mauch

Karlsruher Institut für Technologie
Steinbuch Centre for Computing
[baun|mauch]@kit.edu

24.6.2011

Virtualisierung – Grundlagen

- Durch **Virtualisierung** werden die Ressourcen eines Rechnersystems aufgeteilt und von mehreren unabhängigen Betriebssystem-Instanzen genutzt
- Virtualisierung ist stellvertretend für mehrere grundsätzlich verschiedene Konzepte und Technologien
- Jede **virtuelle Maschine** (VM) verhält sich wie ein vollwertiger Computer mit eigenen Komponenten, der in einer abgeschotteten Umgebung auf einer realen Maschine läuft
- In einer VM kann ein Betriebssystem mit Anwendungen genau wie auf einem realen Computer installiert werden.
  - Die Anwendungen merken nicht, dass sie sich in einer VM befinden
- Anforderungen der Betriebssystem-Instanzen werden von der Virtualisierungssoftware abgefangen und auf die real vorhandene oder emulierte Hardware umgesetzt
  - Die VM selbst bekommt davon auch nichts mit

Ursprung der Virtualisierung

- Virtualisierung ist kein neues Konzept
  - Einführung bereits vor ca. 40 Jahren bei Großrechnern
- IBM stellte in den 1960er-Jahren die Virtual Machine Facility/370, kurz VM/370 vor
  - Auf dieser Plattform wurde Mehrbenutzerbetrieb realisiert, indem mehrere Einzelbenutzerbetriebinstanzen in virtuellen Maschinen ausführt wurden
  - Jede VM stellte eine vollständige Nachbildung der darunter liegenden, physischen Hardware dar

Quellen

- Creasy RJ. **The origin of the VM/370 time-sharing system**. IBM Journal of Research and Development 25 (1981), No. 5, 483–490
- Amit Singh. **An Introduction to Virtualization**. 2004 `http://www.kernelthread.com/publications/virtualization/`

Virtualisierungskonzepte

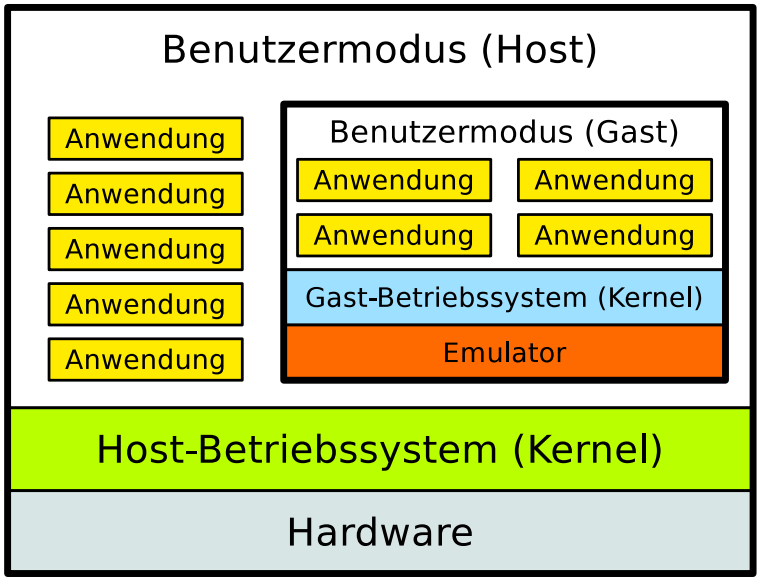
- Es existieren unterschiedliche Virtualisierungskonzepte:
  - Partitionierung
  - Hardware-Emulation
  - Anwendungsvirtualisierung
  - Virtueller Maschinen-Monitor
  - Paravirtualisierung
  - Hardware-Virtualisierung
  - Betriebssystem-Virtualisierung bzw. Container bzw. Jails
  - Speichervirtualisierung
  - Netzwerkvirtualisierung
  - ...

Partitionierung

- Bei Partitionierung können auf den Gesamtressourcen eines Computersystems Teilsysteme definiert werden
  - Jedes Teilsystem kann eine lauffähige Betriebssysteminstanz enthalten
  - Jedes Teilsystem ist wie ein eigenständiges Computersystem verwendbar
- Die Ressourcen (Prozessor, Hauptspeicher, Datenspeicher...) werden über die **Firmware** des Rechners verwaltet und den VMs zugeteilt
- Partitionierung kommt z.B. bei IBM Großrechnern (zSerie) oder Midrange-Systemen (pSerie) mit Power5/6 Prozessoren zum Einsatz
  - Ressourcenzuteilung ist im laufenden Betrieb ohne Neustart möglich
  - Auf einem aktuellen Großrechner können mehrere hundert bis tausend Linux-Instanzen gleichzeitig laufen
- Aktuelle CPUs unterstützen lediglich die Partitionierung der CPU selbst und nicht des Gesamtsystems (Intel Vanderpool, AMD Pacifica)
  - Partitionierung spielt im Desktop-Umfeld keine Rolle

Hardware-Emulation

- Emulation bildet die **komplette Hardware** eines Rechnersystems nach, um ein **unverändertes Betriebssystem**, das für eine **andere Hardwarearchitektur** (CPU) ausgelegt ist, zu betreiben
  - Ausnahme Wine: Wine emuliert keine Hardware, sondern nur die Schnittstellen eines Windows-Betriebssystems



- Nachteile der Emulation:
  - Entwicklung ist sehr aufwendig
  - Ausführungsgeschwindigkeit ist gegenüber Virtualisierung geringer
- Wichtige Unterscheidung: **Emulation** $\neq$ **Virtualisierung**
- Einige Emulatoren: Bochs, QEMU, PearPC, Wabi, DOSBox, Microsoft Virtual PC (ist in der Version für MacOS X/PowerPC ein x86-Emulator)

Auswahl an Emulatoren

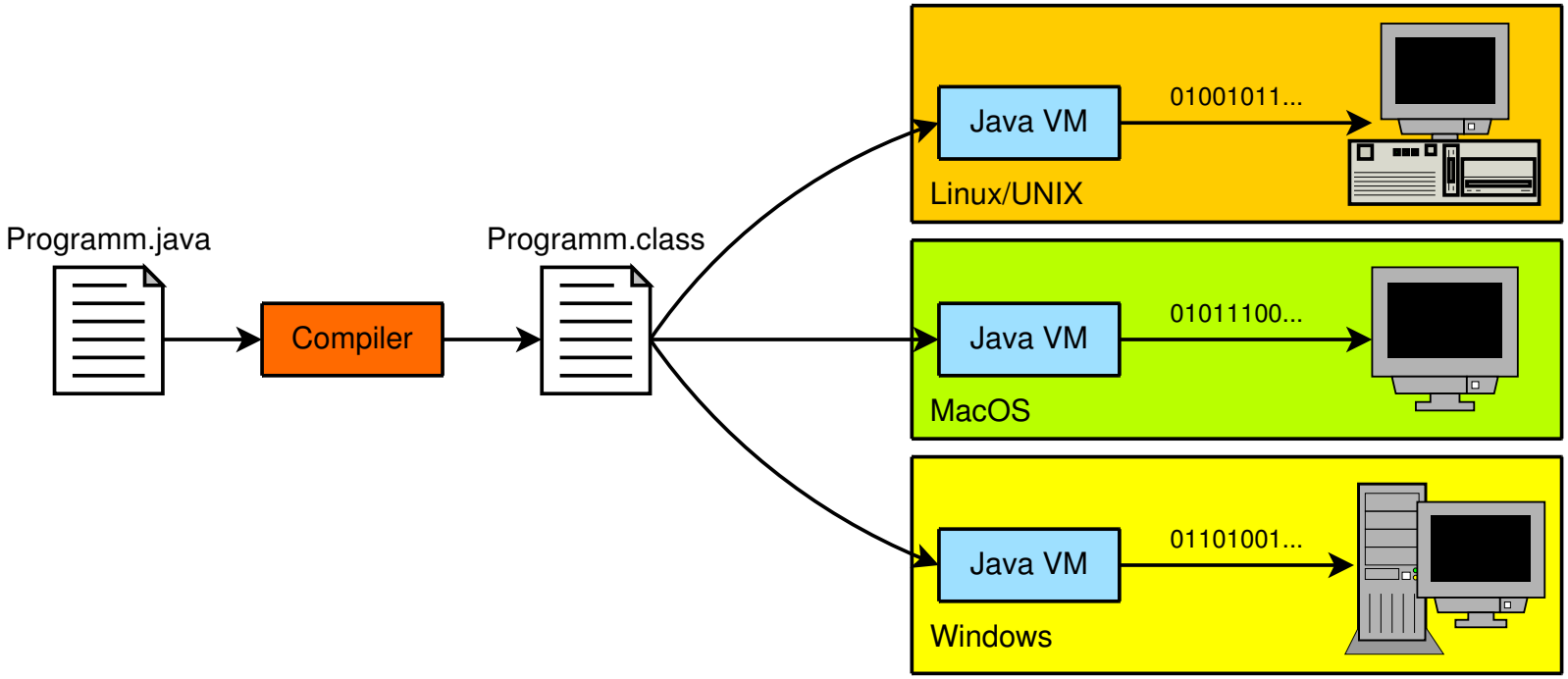
Name	Lizenz	Host	Emulierte Architektur	Gast-System
Bochs v2.3.6	LGPL	Linux, Solaris, MacOS, Windows, IRIX, BeOS	x86, AMD64	Linux, DOS, BSD, Windows, BeOS
QEMU v0.9.0	GPL	Linux, BSD, Solaris, BeOS, MacOS-X	x86, AMD64, PowerPC, ARM, MIPS, Sparc	Linux, MacOS-X, Windows, BSD
DOSBox v0.72	GPL	Linux, Windows, OS/2, BSD, BeOS, MacOS-X	x86	DOS
DOSEMU v1.4.0	GPL	Linux	x86	DOS, Windows bis 3.11
PearPC v0.4.0	GPL	Linux, MacOS-X Windows	PowerPC	Linux, MacOS-X, BSD
Baseilisk II v0.9-1	GPL	Linux, diverse UNIX, Windows NT4, BeOS, Mac OS, Amiga OS	680x0	MacOS $\leq$ 8.1
Wabi v2.2	proprietär	Linux, Solaris	x86	Windows 3.x
MS Virtual PC v7	proprietär	MacOS-X	x86	Windows, (Linux)
M.A.M.E. v0.137	MAME-Lizenz	Linux, Windows, DOS, BeOS, BSD, OS/2	diverse Arcade	diverse Arcade
SheepShaver	GPL	Linux, MacOS-X, BSD Windows, BeOS	PowerPC, 680x0	MacOS 7.5.2 bis MacOS 9.0.4
Hercules 3.07	QPL	Linux, MacOS-X, BSD Solaris, Windows	IBM-Großrechner	IBM System/360, 370, 390

- Die Tabelle erhebt keinen Anspruch auf Vollständigkeit!

Anwendungsvirtualisierung

- Anwendungen werden lokal, unter Verwendung lokaler Ressourcen in einer virtuellen Umgebung ausgeführt, die alle Komponenten bereitstellt, die die Anwendung benötigt
  - VM befindet sich zwischen der auszuführenden Anwendung und dem Betriebssystem
- Populärstes Beispiel: Die Java Virtual Machine (JVM)
  - Die JVM ist der Teil der Java-Laufzeitumgebung (JRE), der für die Ausführung des Java-Bytecodes verantwortlich ist
  - Die JVM ist für Java-Programme die Schnittstelle zum Rechnersystem und dessen Betriebssystem
- Vorteil: Plattformunabhängigkeit
- Nachteil: Geringere Ausführungsgeschwindigkeit gegenüber nativer Programmausführung

Prinzip der Java Virtual Machine (JVM)



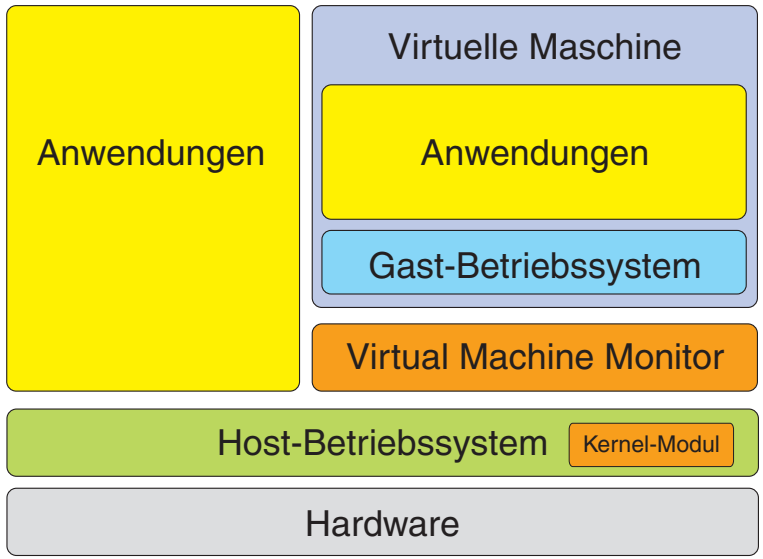
- Der Java-Compiler `javac` übersetzt den Quellcode in architekturunabhängige `.class`-Dateien, die Bytecode enthalten, der in der Java Virtual Machine lauffähig ist
- Das `java`-Programm startet eine Java-Applikation in einer Instanz der Java Virtual Machine

VMware ThinApp

- Weiteres Beispiel für Anwendungsvirtualisierung: VMware ThinApp
  - `http://www.vmware.com/products/thinapp/`
  - Bis 2008 unter dem Namen Thinstall bekannt
- Eine Windows-Anwendung wird in eine einzelne `.exe`-Datei gepackt
- Die Anwendung wird dadurch portabel und kann ohne lokale Installation verwendet werden
  - Die Anwendung kann u.a. auf einem USB-Stick ausgeführt werden
- Es erfolgen keine Einträge in der Windows Registry. Es werden auch keine Umgebungsvariablen und DLL-Dateien auf dem System erstellt
- Benutzereinstellungen und erstellte Dokumente werden in einer eigenen Sandbox gespeichert
- Nachteil: Funktioniert ausschließlich mit Microsoft Windows
  - Unter Linux ist die Nutzung mit Wine möglich

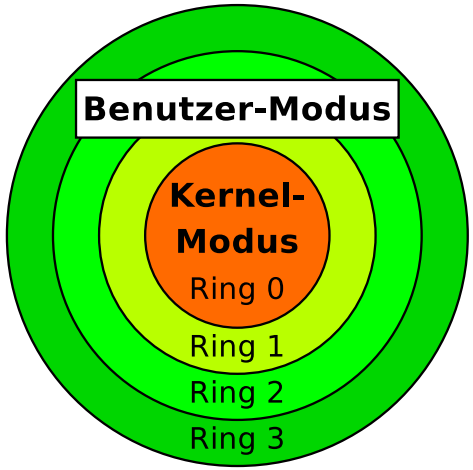
Vollständige Virtualisierung (1)

- Vollständige Virtualisierungslösungen bieten einer VM eine vollständige, virtuelle PC-Umgebung inklusive eigenem BIOS
  - Jedem Gastbetriebssystem steht ein eigener virtueller Rechner mit virtuellen Ressourcen wie CPU, Hauptspeicher, Laufwerken, Netzwerkkarten, usw. zur Verfügung
- Es kommt ein **Virtueller Maschinen-Monitor** (VMM) zum Einsatz
  - Den VMM bezeichnet man auch als **Typ-2-Hypervisor**
- Der VMM läuft *hosted* als Anwendung unter dem Host-Betriebssystem
- Der VMM verteilt die Hardwareressourcen des Rechners an die VMs
- Teilweise emuliert der VMM Hardware, die nicht für den gleichzeitigen Zugriff mehrerer Betriebssysteme ausgelegt ist
  - Ein Beispiel sind Netzwerkkarten
  - Die Emulation populärer Hardware vermeidet Treiberprobleme



Virtualisierungsgrundlagen der x86-Architektur (1)

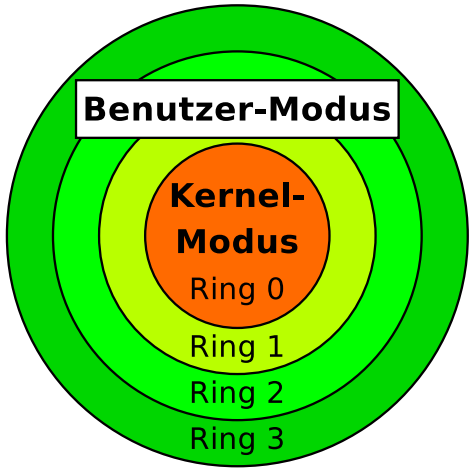
- Die Virtualisierung in der x86-Architektur basiert auf dem Schutzkonzept der Ringe
- x86-kompatible CPUs enthalten vier Privilegienstufen zum Speicherschutz, um die Stabilität und Sicherheit zu erhöhen
- Ein Prozess kann immer nur in einem einzelnen Ring ausgeführt werden und ist nicht in der Lage, sich selbstständig aus diesem zu befreien



- Die verbreiteten Betriebssysteme nutzen nur Ring 0 und 3
- Eine Ausnahme ist OS/2
  - OS/2 nutzt Ring 2 für Anwendungen, die auf Hardware und Eingabe-/Ausgabeschnittstellen zugreifen dürfen (z.B. Grafiktreiber)

Virtualisierungsgrundlagen der x86-Architektur (2)

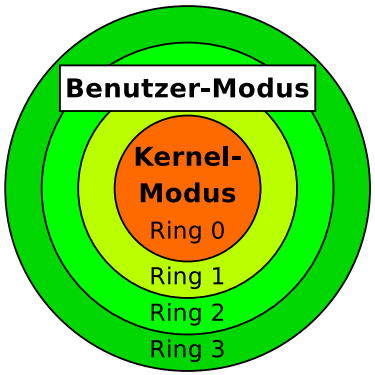
- Nur Prozesse in Ring 0 haben vollen Zugriff auf die Hardware und dürfen den vollständigen Befehlssatz der CPU nutzen
  - Ring 0 ist der Kernel-Bereich (Kernel-Space)
    - Hier läuft nur der Betriebssystemkern und zum Start des Betriebssystems nötige Hardwaretreiber
  - Ring 3 ist der Benutzerbereich (User-Space)
    - Hier laufen die Anwendungen

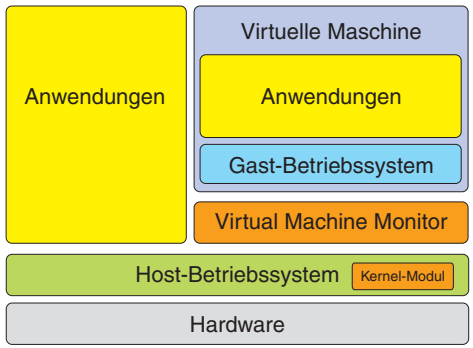


- Ruft ein Prozess in einem weniger privilegierten Ring eine privilegierte Operation auf, erzeugt die CPU eine Ausnahme (Exception)
  - Die Exception wird im benachbarten privilegierteren Ring abgefangen und dort behandelt
  - Ausnahmen, die nicht abgefangen werden können, verursachen eine allgemeine Schutzverletzung (General Protection Fault)
    - Der aufrufende Prozess stürzt ab
    - Handelt es sich bei dem Prozess um den Kernel, stürzt das System ab

Vollständige Virtualisierung (2)

- Vollständige Virtualisierung nutzt die Tatsache, dass x86-Systeme nur zwei von vier möglichen Privilegienstufen verwenden
  - Der VMM befindet sich in Ring 0 auf der Ebene des Betriebssystemkerns des Host-Betriebssystems und hat vollen Zugriff auf die Hardware
  - Die VMs befinden sich in einem der weniger privilegierten Ringe 1 oder 2
- Der VMM stellt für jede denkbare Ausnahme eine Behandlung zur Verfügung, die die privilegierten Operationen der Gastbetriebssysteme abfängt, interpretiert und ausführt
- Der VMM stellt sicher, dass die VMs nur über den Umweg des VMM Zugriff auf die Hardware erhalten
  - Kontrollierter Zugriff auf die gemeinsam genutzten Systemressourcen ist gewährleistet





Vollständige Virtualisierung (3)

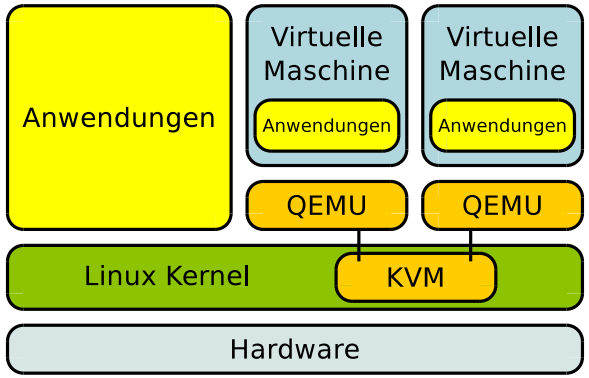
- Vorteile der Vollständigen Virtualisierung:
  - Kaum Änderungen an Host- und Gast-Betriebssystemen erforderlich
  - Zugriff auf die wichtigsten Ressourcen wird nur durchgereicht
    $\Longrightarrow$ Fast native Verarbeitungsgeschwindigkeit der Gast-Betriebssysteme
  - Jedes Gast-Betriebssystem hat seinen eigenen Kernel
    $\Longrightarrow$ Hohe Flexibilität
- Nachteile der Vollständigen Virtualisierung:
  - Wechsel zwischen den Ringen erfordern einen Kontextwechsel
    $\Longrightarrow$ Jeder Kontextwechsel verbraucht Rechenzeit
  - Fordert eine Anwendung im Gast-Betriebssystem die Ausführung eines privilegierten Befehls an, liefert der VMM eine Ersatzfunktion und diese weist die Ausführung des Befehls über die Kernel-API des Host-Betriebssystems an
    $\Longrightarrow$ Geschwindigkeitseinbußen

Beispiele für Vollständige Virtualisierung

- Beispiele für Virtualisierungslösungen, die auf dem Konzept des VMM basieren, sind:
  - VMware Server, VMware Workstation und VMware Fusion
  - Microsoft Virtual PC (in der Version für x86)
  - Parallels Desktop und Parallels Workstation
  - VirtualBox
  - Kernel-based Virtual Machine (KVM)
  - Mac-on-Linux (MoL)

Kernel-based Virtual Machine (KVM)

- KVM ist als Modul direkt im Linux-Kernel integriert
  - KVM-Basismodul: `kvm.ko`
  - Hardwarespezifische Module: `kvm-intel.ko` und `kvm-amd.ko`



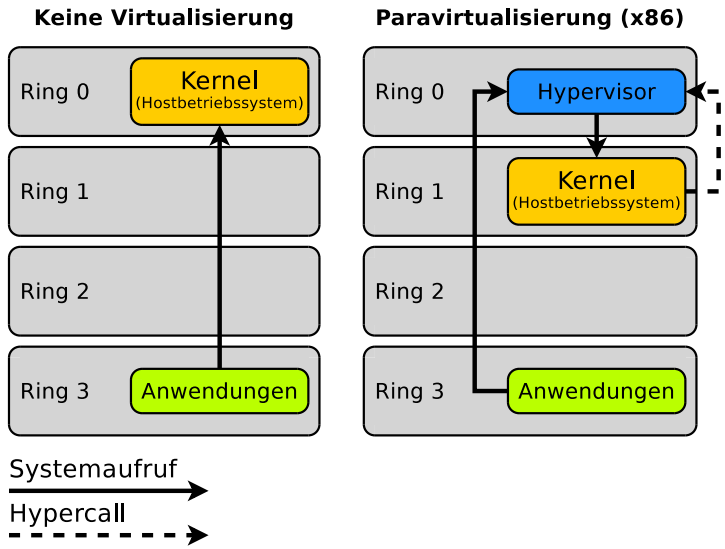
- Nach dem Laden der Module arbeitet der Kernel selbst als Hypervisor
- KVM kann nur mit CPUs mit Hardwarevirtualisierung arbeiten
  - Dadurch braucht KVM weniger Quellcode als z.B. Xen
- Neben den Kernelmodulen enthält KVM den Emulator QEMU
  - KVM stellt keine virtuelle Hardware zur Verfügung. Das macht QEMU
    - CPU-Virtualisierung stellt der Prozessor bereit (Intel VT oder AMD-V)
    - Der Speicher wird durch KVM virtualisiert
    - E/A wird durch einen QEMU-Prozess pro Gastsystem virtualisiert

Paravirtualisierung (1)

- Bei Paravirtualisierung wird keine Hardware virtualisiert oder emuliert
  - Gast-Betriebssystemen steht keine emulierte Hardwareebene, sondern eine API zur Verfügung
- Virtuell gestartete Betriebssysteme verwenden eine abstrakte Verwaltungsschicht, den **Hypervisor**, um auf die physischen Ressourcen wie Speicher, Ein-/Ausgabegeräte und Netzwerkinterfaces zuzugreifen
  - Der Hypervisor ist quasi ein auf ein Minimum reduziertes **Metabetriebssystem**, das die Hardwareressourcen unter den Gastsystemen verteilt, so wie ein Betriebssystem dieses unter den laufenden Prozessen tut
    - Der Hypervisor läuft *bare metal*
    - Der Hypervisor wird auch als **Typ-1-Hypervisor** bezeichnet
  - Ein Metabetriebssystem ermöglicht den unabhängigen Betrieb unterschiedlicher Anwendungen und Betriebssysteme auf einer CPU
- Das Host-Betriebssystem läuft nicht im privilegierten Ring 0, sondern im weniger privilegierten Ring 1
  - Ein Host-Betriebssystem ist wegen der Gerätetreiber nötig

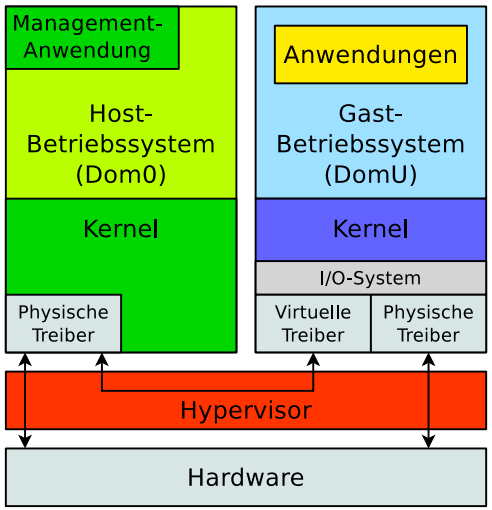
Paravirtualisierung (2)

- Das Betriebssystem läuft nicht mehr in Ring 0, sondern in Ring 1
  - Darum kann der Kernel keine privilegierten Anweisungen ausführen
  - Lösung: Der Hypervisor stellt **Hypercalls** zur Verfügung
- Hypercalls sind vergleichbar mit Systemaufrufen (System Calls)
  - Die Interrupt-Nummern sind verschieden
  - Fordert eine Anwendung die Ausführung eines Systemaufrufs an, wird eine Ersatzfunktion im Hypervisor aufgerufen
  - Der Hypervisor weist die Ausführung des Systemaufrufs über die Kernel-API des Betriebssystems an



- Erweiterung des Kernels um die Hypercall-Funktionalität macht eine Modifikation der Betriebssysteme notwendig
- Abfangen und Prüfen aller Systemaufrufe durch den Hypervisor führt nur zu geringen Geschwindigkeitseinbußen
- Beispiele: Xen, Citrix Xenserver, Virtual Iron, VMware ESX Server

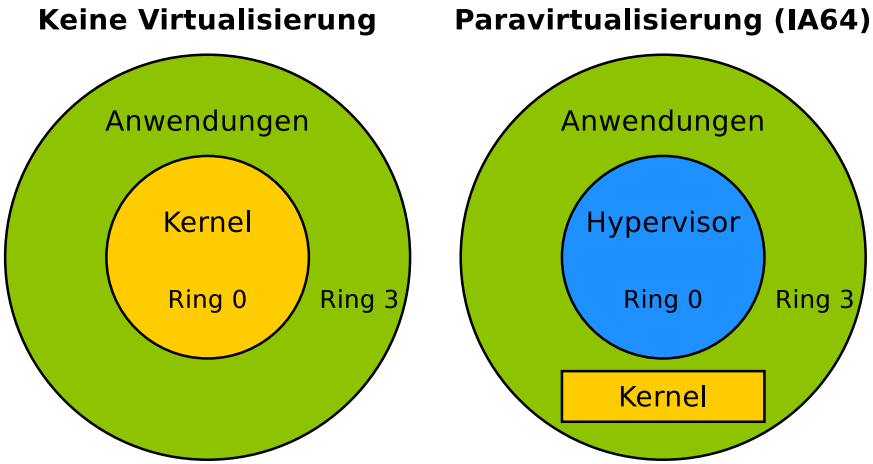
Paravirtualisierung (3)



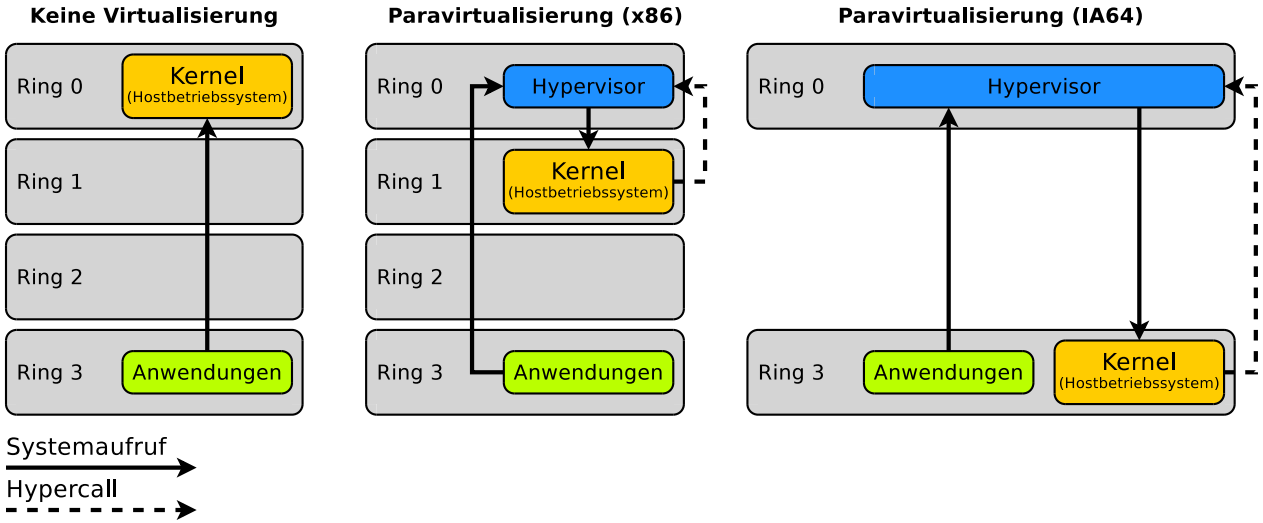
- Eine VM bezeichnet man (bei Xen) als **Domain**
- VMs nennt man **unprivilegierte Domain** (DomU)
- Der Hypervisor arbeitet unterhalb der VMs und die Entwickler können nicht alle Treiber selbst schreiben und pflegen
  - Darum startet der Hypervisor eine (Linux-)Instanz mit ihren Treibern und leiht sich diese Treiber
  - Diese spezielle Instanz heißt Domain0 (Dom0)
- Nachteile:
  - Kernel der Gast-Betriebssysteme müssen speziell für den Betrieb im paravirtualisierten Kontext angepasst sein
  - Rechteinhaber proprietärer Betriebssysteme lehnen eine Anpassung aus strategischen Gründen häufig ab
    $\Longrightarrow$ Funktioniert häufig nur mit OpenSource-Betriebssystemen
- Vorteil:
  - Geschwindigkeitseinbußen, die beim VMM entstehen, werden vermieden

Problem: x86-64-Architektur

- Die x86-64-Architektur (z.B. IA64) verzichtet auf die Ringe 1 und 2



- Der Hypervisor befindet sich wie bei der x86-32-Architektur in Ring 0
- Der Kernel mit seinem Betriebssystemkern wird bei der der x86-64-Architektur in Ring 3 zu den Anwendungen verschoben



- Der Betrieb der Hardwaretreiber und Anwendungen in einem Ring ist tendenziell unsicher

Zusammenfassung: Voll- vs. Paravirtualisierung

- Paravirtualisierung erfordert angepasste Gastsysteme
  - Hypervisor läuft *bare metal* anstatt eines klassischen Betriebssystems
  - Typ-1-Hypervisor
  - Hypervisor läuft in Ring 0 und hat vollen Zugriff auf die Hardware
  - Beispiele: VMware ESX(i), Xen, Microsoft Hyper-V
- Bei Vollvirtualisierung kann man unveränderte Systeme betreiben
  - VMM läuft *hosted* als Anwendung unter einem klassischen Betriebssystem wie Linux oder Windows
  - VMM = Typ-2-Hypervisor
  - VMM läuft in Ring 0 auf der Ebene der Anwendungen
  - Beispiele: VMware Workstation, KVM, Oracle VirtualBox, Parallels

Hardware-Virtualisierung (1)

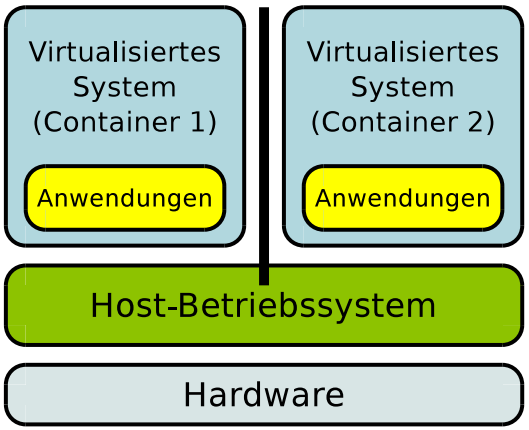
- Aktuelle Prozessoren von Intel und AMD enthalten Erweiterungen zur Hardware-Virtualisierung
  - Dank Hardware-Virtualisierung können unmodifizierte Betriebssysteme (z.B. Windows) mit Xen oder VMware ESX laufen
  - Die Lösungen von Intel und AMD sind ähnlich aber inkompatibel
- AMD erweitert seit Juni 2006 seine AMD64 CPUs um den sogenannten Secure-Virtual-Machine-Befehlssatz (**SVM**)
  - Die Lösung heißt **AMD-V** und war vorher als **Pacifica** bekannt
- Die Lösung von Intel heißt **VT-x** für IA32-CPUs und **VT-i** für Itanium
  - Intels Lösung lief vormals unter dem Stichwort **Vanderpool**
- Xen unterstützt ab Version 3 Hardware-Virtualisierung
- Auch Windows Server 2008 (Hyper-V) nutzt Hardwarevirtualisierung

Hardware-Virtualisierung (2)

- Kern der Neuerung ist eine Überarbeitung der Privilegienstruktur
  - Die neuen Befehle bei AMD und Intel bieten VMs eine Erweiterung zu den bereits beschriebenen Privilegienstufen Ring 0 und Ring 3
- Die Ringstruktur wurde durch eine Erweiterung von Ring 0 um eine Ebene, die neue Hypervisor-Schicht, ergänzt
  - Die Ebene wird als Root-Betriebsmodus oder Ring -1 bezeichnet
  - Der Hypervisor bzw. VMM läuft im Root-Betriebsmodus (Ring -1) und besitzt jederzeit die volle Kontrolle über die CPU und die Ressourcen, da damit ein höheres Privileg als Ring 0 implementiert ist
- VMs steht der gewohnte Zugriff auf Ring 0 zur Verfügung
  - Diese VMs heißen HVM (Hardware Virtual Machine)
- Vorteile:
  - Gastbetriebssysteme müssen nicht angepasst werden
    - Auch proprietäre Betriebssysteme (z.B. Windows) laufen als Gastsysteme
  - Der Kernel läuft nicht wie bei der Paravirtualisierung mit den Anwendungen auf einer Privilegienstufe

Betriebssystem-Virtualisierung / Container / Jails (1)

- Bei Betriebssystem-Virtualisierung laufen unter ein und demselben Kernel mehrere voneinander abgeschottete identische Systemumgebungen
- Es wird kein zusätzliches Betriebssystem, sondern eine isolierte Laufzeitumgebung virtuell in einem geschlossenen Container erzeugt
- Alle laufenden Anwendungen verwenden denselben Kernel
  - Betriebssystem-Virtualisierung heißt **Container** in SUN/Oracle Solaris
  - Betriebssystem-Virtualisierung heißt **Jails** in BSD



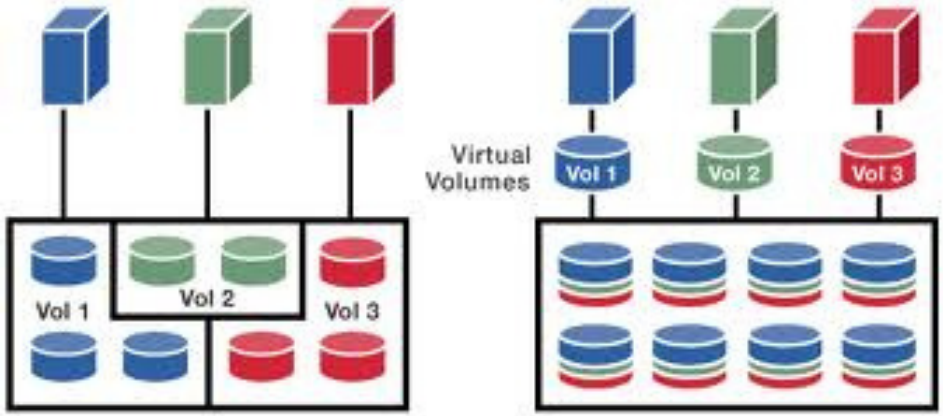
- Anwendungen sehen nur die Anwendungen, mit denen sie sich in einer virtuellen Umgebung befinden
- Ein Vorteil ist der geringe Overhead, da der Kernel in gewohnter Weise die Hardware verwaltet

- Nachteil: Alle virtuellen Umgebungen nutzen den gleichen Kernel
  - Es werden nur unabhängige Instanzen eines Betriebssystemes gestartet
  - Verschiedene Betriebssysteme können nicht gleichzeitig verwendet werden

Betriebssystem-Virtualisierung / Container / Jails (2)

- Diese Art der Virtualisierung nutzt man, um Anwendungen in isolierten Umgebungen mit hoher Sicherheit zu betreiben
- Besonders Internet-Service-Provider, die (virtuelle) Root-Server oder Webdienste auf Mehrkernprozessorarchitekturen anbieten, nutzen diese Form der Virtualisierung
  - Wenig Performance-Verlust, hoher Grad an Sicherheit
- Beispiele sind:
  - SUN/Oracle Solaris
  - OpenVZ für Linux
  - Linux-VServer
  - FreeBSD Jails
  - Virtuozzo (kommerzielle Variante von OpenVZ)
  - FreeVPS

Speichervirtualisierung



Bildquelle: `wiwiki.wi-inf.uni-essen.de`

- Der Speicher wird in Form virtueller Laufwerke den Benutzern zur Verfügung gestellt
- Der logische Speicher wird vom physischen Speicher getrennt

- Vorteile:
  - Nutzer sind nicht an die physischen Grenzen der Speichermedien gebunden
  - Physischen Speicher umstrukturieren/erweitern stört die Nutzer nicht
  - Redundantes Vorhalten erfolgt transparent im Hintergrund
  - Besserer Auslastungsgrad, da der verfügbare physische Speicher effektiver auf die vorhandenen Benutzer aufgeteilt werden kann
- Nachteil: Professionelle Lösungen zur Speichervirtualisierung sind teuer
- Bekannte Anbieter: EMC, HP, IBM, LSI und SUN/Oracle

Netzwerkvirtualisierung

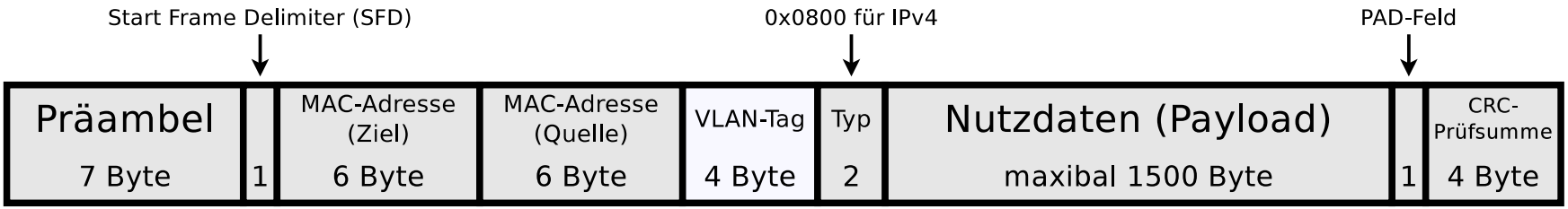
- Basiert auf virtuellen lokalen Netzen (**V**irtual **L**ocal **A**rea **N**etworks)
- Verteilt aufgestellte Geräte können durch VLANs in einem einzigen logischen Netzwerk zusammengefasst werden
  - Nützlich bei der Konzeption der IT-Infrastruktur verteilter Standorte
  - Es ist mit VLANs möglich, ein physisches Netzwerk in logische Teilnetze, sogenannte Overlay-Netzwerke, zu trennen
    - VLAN-fähige Switches leiten die Datenpakete eines VLAN nicht in ein anderes VLAN weiter
    - Ein VLAN bildet ein nach außen isoliertes Netzwerk über bestehende Netze
  - Zusammengehörende Systeme und Dienste können mit VLANs in einem eigenen Netz konsolidiert werden, um somit die übrigen Netze nicht zu beeinflussen
  - Ein VLAN bildet ein Netzwerk über fremde oder nicht vertrauenswürdige Netze und kann so helfen, verteilte Standorte in eine virtuelle Infrastruktur zu integrieren
- Nachteil: Steigender Aufwand für die Netzwerkadministration

Statische und dynamische VLANs

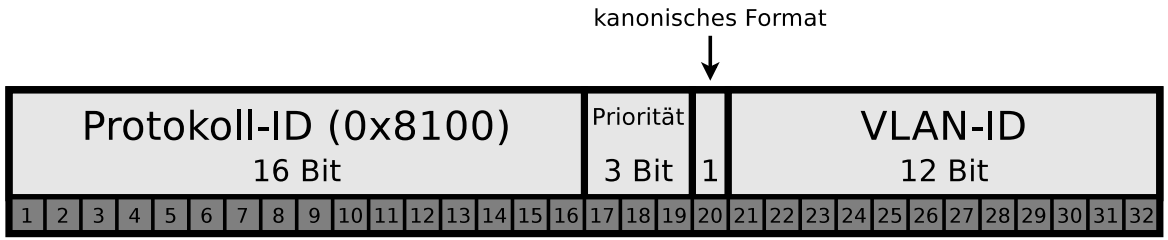
- Es existieren unterschiedliche Typen von VLANs

1. Der älteste Standard sind die **statischen VLANs**
   - Die Anschlüsse (Ports) eines Switches werden in mehrere logische Switches unterteilt
   - Jeder Anschluss ist fest einem einzigen VLAN zugeordnet oder verbindet unterschiedliche VLANs
   - Statische VLANs sind schlecht automatisierbar
2. Aktuell ist das **paketbasierte, dynamische VLAN** nach IEEE 802.1Q
   - Die Netzwerkpakete enthalten eine spezielle VLAN-Markierung (*Tag*)
   - Vor der Standardisierung 1998 durch das IEEE Konsortium existierten diverse proprietäre paketbasierte VLAN Lösungen
     - Cisco Inter-Switch Link (ISL)
     - Virtual LAN Trunk (VLT) von 3Com
   - Diese Lösungen wurden im herstellerübergreifenden Standard IEEE 802.1Q zusammengefasst
   - Dynamische VLANs können mit Hilfe von Skripten rein softwaremäßig erzeugt, verändert und wieder entfernt werden

Ethernet mit VLAN-Markierung nach IEEE 802.1Q



- Der IEEE 802.1Q-Standard definiert die VLAN-Markierung (*Tag*)
- Die eingefügte VLAN-Markierung umfasst 32 Bit
  - Die Protokoll-ID (16 Bit) hat immer den Wert `0x8100`
  - 3 Bit repräsentieren die Priorität
    - Eine von 8 möglichen Prioritäten kann angegeben werden
    - Damit ist eine Priorisierung bestimmter Daten (z.B. VoIP) möglich
  - Das kanonische Format (1 Bit) bestimmt das höchstwertige Bit der MAC-Adressen (0 = Ethernet, 1 = Token Ring)
  - 12 Bit enthalten die ID des VLAN, zu dem das Netzwerkpaket gehört



Gründe für Virtualisierung (1)

- Bessere Ausnutzung der Hardware
  - Serverkonsolidierung: Zusammenlegen vieler (virtueller) Server auf weniger physischen Servern
    - Kostensenkung bei Hardware, Verbrauchskosten (Strom, Kühlung), Stellplätze, Administration, usw.
  - Gartner geht davon aus, dass durch (Server-)Virtualisierung die Investitionen in neue Hard- und Software um bis zu 70% sinken können
  - Im Rechenzentrum sind Kosteneinsparungen von bis zu 50% erreichbar
- Vereinfachte Administration
  - Anzahl physischer Server wird reduziert
  - Ausgereifte Managementwerkzeuge existieren
  - VMs können im laufenden Betrieb verschoben werden (Live Migration)
    - Wartung und Technologiewechsel (der Virtualisierungsplattform) ohne Betriebsunterbrechung möglich (gilt nicht für die VMs!)
- Vereinfachte Bereitstellung
  - Neue Infrastrukturen und Server können innerhalb von Minuten manuell oder automatisch erzeugt werden

Gründe für Virtualisierung (2)

- Maximale Flexibilität
  - VMs können leicht vervielfältigt und gesichert werden
  - Snapshots vom aktuellen Zustand einer VM können erzeugt und wieder hergestellt werden
- Höhere Sicherheit
  - VMs sind gegenüber anderen VMs und dem Host-System isoliert
  - Unternehmenskritische Anwendungen können in einer VM gekapselt und so in einer sicheren Umgebung laufen
  - Ausfall einer VM tangiert die übrigen VMs und den Host nicht
- Optimierung von Software-Tests und Software-Entwicklung
  - Gleichzeitiger Betrieb mehrerer Betriebssysteme
  - Testumgebungen können schnell aufgesetzt werden
- Unterstützung alter Anwendungen
  - Legacy-Betriebssysteme oder Legacy-Anwendungen, für die keine Hardware mehr zu bekommen ist, können reanimiert werden

Nachteile und Grenzen der Virtualisierung

- Leistungsverlust
  - Aktuelle Virtualisierungstechnologien sind so ausgereift, dass sich der Leistungsverlust mit 5-10% nicht sonderlich auswirkt
  - Da aktuelle Mehrkernprozessorsysteme (Intel VT/VT-x und AMD-V) mit Virtualisierung besonders effektiv genutzt werden können, spielt der Leistungsverlust eine zunehmend untergeordnete Rolle
- Nicht jede Hardware kann angesprochen oder emuliert werden
  - Kopierschutzstecker (Hardwaredongles) sind ein Problem
  - Beschleunigte Grafik kann nicht immer realisiert werden
- Beim Ausfall eines Hosts würden mehrere virtuelle Server ausfallen
  - Ausfallkonzepte und redundante Installationen sind notwendig
- Virtualisierung ist komplex
  - Zusätzliches Know-how ist notwendig

Virtualisierung im Cloud Computing

- Anwendungsvirtualisierung (JVM) in Plattformen wie der GAE
- Partitionierung spielt wegen der hohen Anschaffungskosten nur in Bereichen wie Cloud Gaming eine Rolle
- Vollständige Virtualisierung spielt im Dienstangebot von Cloud-Anbietern mit Ausnahme von KVM zurzeit kaum eine Rolle
- KVM kann von einigen Private Cloud-Lösungen verwendet werden
- Xen-basierte Lösungen (Paravirtualisierung) sind die Basis der AWS
- Viele Private Cloud-Lösungen verwenden Xen
- Betriebssystem-Virtualisierung kann in Cloud-Umgebungen helfen, die Hardware effizienter auszunutzen
- In Cloud-Rechenzentren ermöglicht Speichervirtualisierung den Speicher zu konsolidieren und besser auszunutzen
- Einige öffentlich zugängliche und private Cloud-Dienste nutzen VLANs, um die Instanzen und deren Netzwerkkommunikation vom Produktionsnetzwerk der physischen Infrastruktur zu trennen

Fazit zur Virtualisierung

- Virtualisierung bietet ein großes Einsparpotential, eröffnet aber auch neue Angriffspunkte, insbesondere auf der Ebene des Hypervisors
  - VMware vertreibt die schlanke Virtualisierungs-Lösung ESXi, bei der ein nur 32 MB großer Virtualisierungs-Kernel die Virtualisierungsfunktionen mit minimalem Betriebssystem direkt auf der Hardware realisiert
- Virtualisierung spielt in den nächsten Jahren auch wegen der besseren Energieeffizienz und unkomplizierten Nutzung von Mehrkernprozessoren eine wachsende Rolle
- Hardwarevirtualisierung ist durch die neuen Prozessorgenerationen fast überall verfügbar
- Hardware-Emulation $\neq$ Virtualisierung
  - Ziel der Emulation ist die Nachbildung einer anderen Hardwarearchitektur

Networking Between Host and Guest VMs

(Host-only Networking with VirtualBox)

- Ideal network configurations for VirtualBox
- Network Address Translate
- Host-only Networking

Alex Hahn

2012.02

Scenario

Objectives

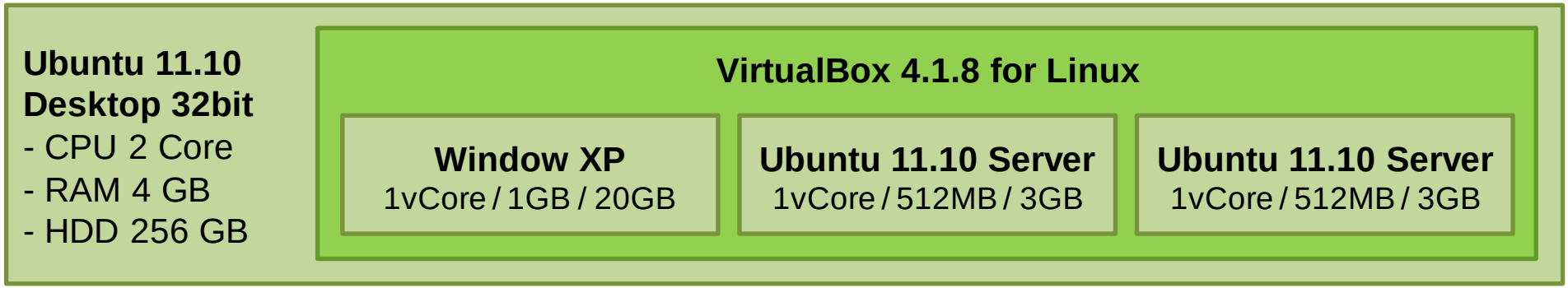
To develop and test in distributed environment you might use virtualizing tool like VirtualBox to imitate the environment of where number of servers and complex networks resides. But still, you have to move around through VM windows. In order to access all virtual servers using SSH just like we do in the real world, several things should be done first.

- Setup Host-only Networking
- Configure guest OS

Requirements

To achieve the goal, a virtualization tool (VirtualBox will be used here) and a little bit of networking knowledge is necessary which will be handled below.

Of course you need your own laptop and Linux server OS for guest VMs.

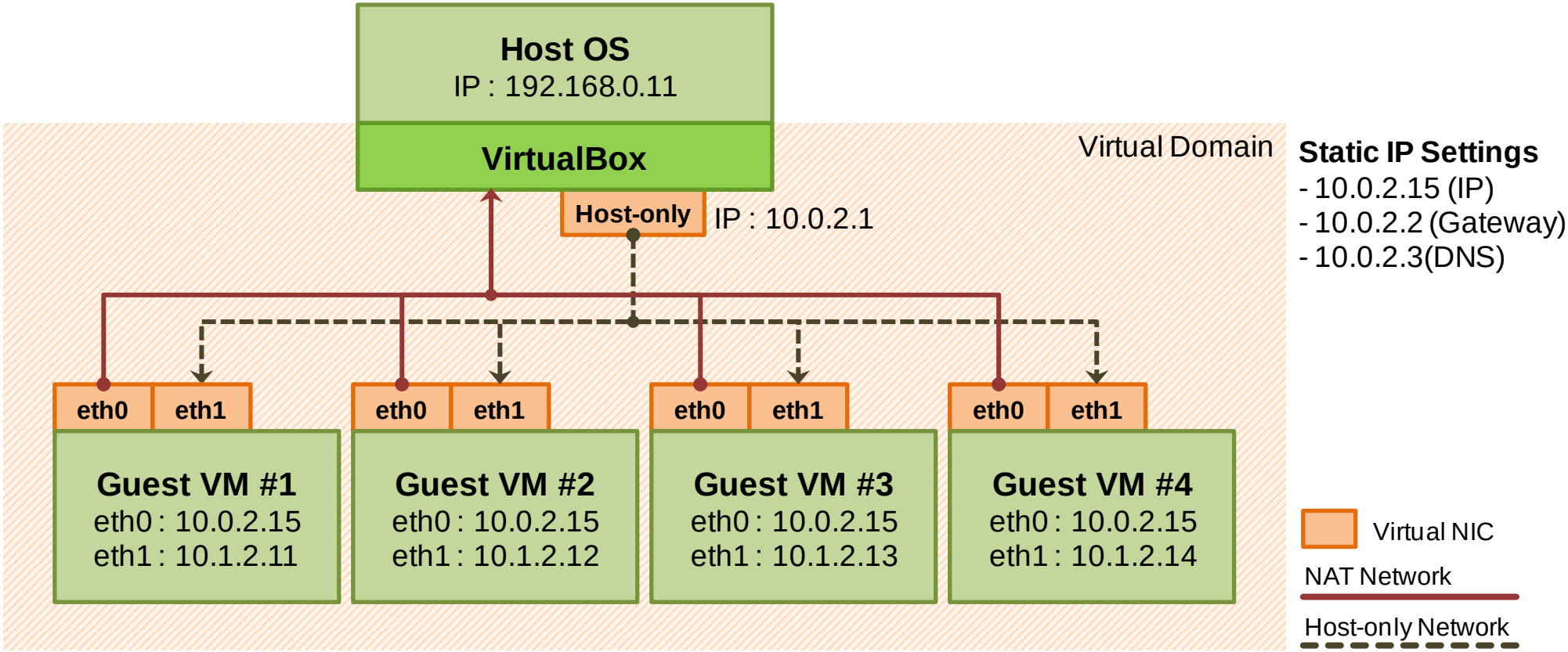


[An example of virtual environment]

Virtual Machine Network Structure

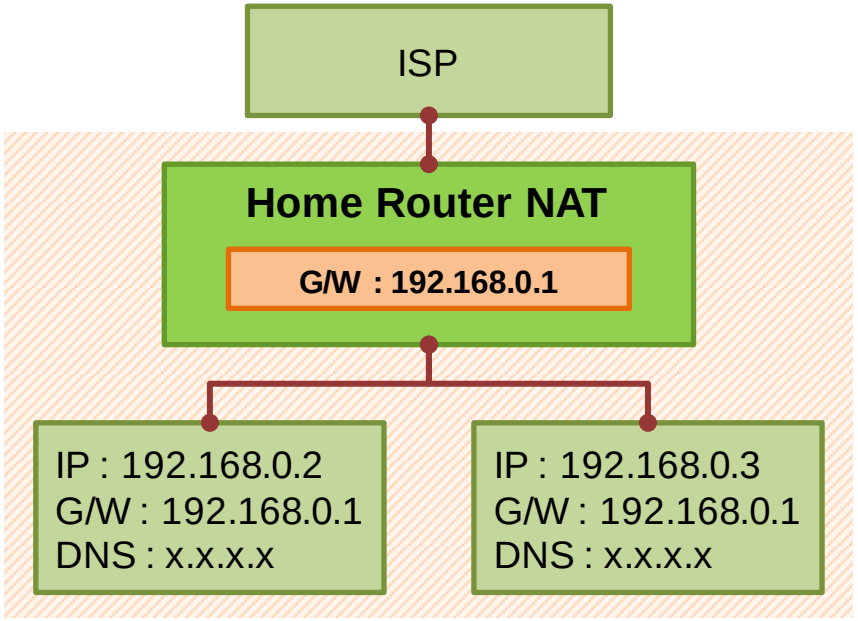
◈ Network configuration

- ✓ Separate network is necessary for host OS to access guest VMs
- ✓ To do that, each guest VMs should have two network adapters
- ✓ One is for guest VM to access outbound network, namely NAT adapter
- ✓ Another is for host OS to access each guest VM, namely Host-only adapter

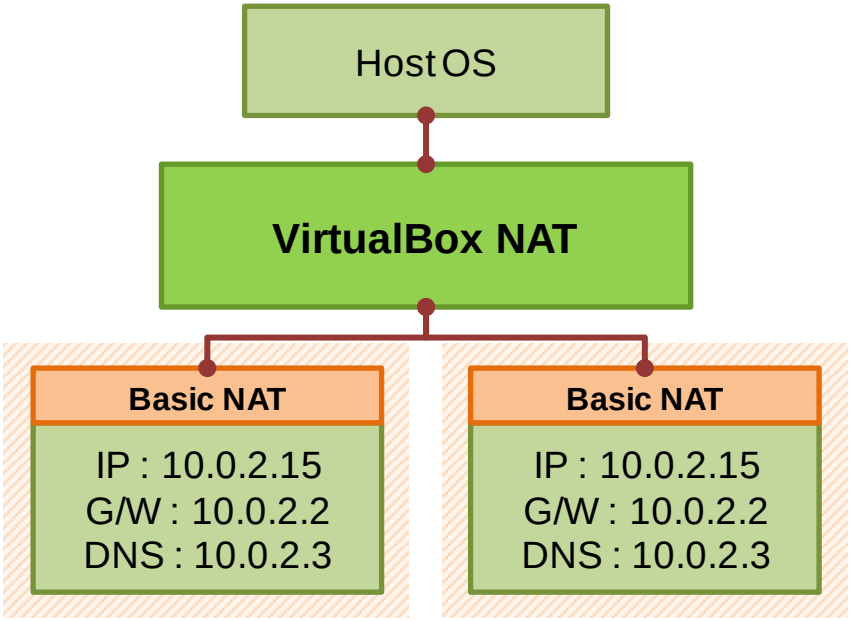


◈ NAT (Network Address Translation)

- ✓ NAT feature is included in most of our home routers.
- ✓ It creates a single separate sub network below the inbound IP (public IP) so that number of machines can get an internet access and can share with each other too.
- ✓ But in VirtualBox, NAT is attached to each VM which is so called 'Basic NAT' that does one-to-one translation, and in contrast home routers do one-to-many translation.
- ✓ That's why with default adapters for VMs, they can't see each other, even from the host.
- ✓ Actually, IP, Gateway, DNS is somewhat 'hard coded' in VirtualBox's NAT Adapter.



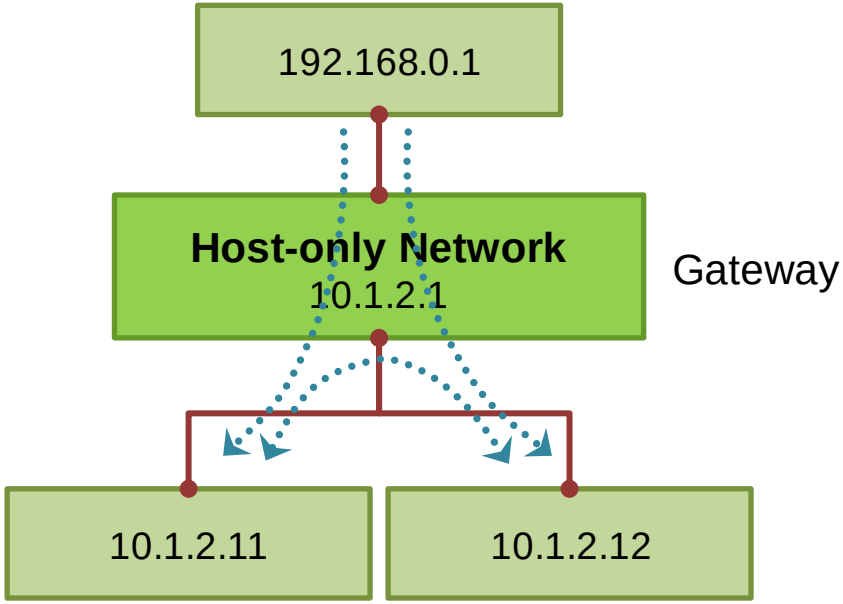
☐ All machines are allocated to one sub network.



☐ Each VM has its own sub network.

◈ Host-only Networking

- ✓ NAT is useful for guest VMs to access outbound network but as it is treated as a separate network, there is no way for host OS to access each guest OS.
- ✓ Bridged networking is used to handle this problem in real world, but it implies complexity.
- ✓ Host-only networking is kind of hybrid between internal and bridged networking.
- ✓ In VirtualBox (Host), Host-only network is a gateway for host (external network) to access guest VMs (internal network) to communicate with each other. So, it doesn't need to be created as many the number of guest VMs.



Considerations

- **Each NAT adapters and Host-only adapter should not reside in same sub-network**
  - ✓ eth0 -10.0.2.15 and eth1-10.0.2.12 wouldn't work
  - ✓ eth0 -10.0.2.15 and eth1-10.1.2.11 is good
  - ✓ Where eth0 is for NAT and eth1 is for Host-only
- **Each VM should have different host name.**
  - ✓ VM1 : ubuntu1, VM2 : ubuntu2
  - ✓ "$ sudo hostname ubuntu2" on new VM and reboot
- **Make sure to set different MAC address for new VM**
  - ✓ Rather installing a fresh new VM, use clone feature in VirtualBox
  - ✓ First, take a snapshot of well setup VM and make a clone
  - ✓ At clone, check 'Reinitialize the MAC address of all network cards' option
- **After booting second VM, reset 'udev' network rules to set new IPs properly**
  - ✓ Remove /etc/udev/rules.d/70-persistent-net.rules (recreated at new boot)
  - ✓ Check 'ifconfig' for eth0 and eth1
- **Use static IP rather than DHCP on Host-only Network and set host alias.**
  - ✓ IP of Host-only adapter will change when guest VM reboots, so turn off DHCP in VirtualBox
  - ✓ When guest IPs are fixed define an alias for each guest and configure hosts file in host OS

Instructions

Step by Step configuration

- ✓ Step 1 : Setup default networking for guest VM with NAT
- ✓ Step 2 : Create Host-only Networking in VirtualBox
- ✓ Step 3 : Setup secondary adapter for each guest VM
- ✓ Step 4 : Configure guest VM
- ✓ Step 5 : Easier access to guest VMs with ‘hosts’ configuration

Testing

Host Machine

- ✓ Ping to all guest Host-only IPs (10.1.2.11, 10.1.2.12 …)
- ✓ SSH to all guest VM

Guest Machine

- ✓ Ping to ‘www.google.com’ to make sure external network works well including DNS.
- ✓ Ping to gateway (10.1.2.1)
- ✓ Ping to other guest VM.
- ✓ Check SSH daemon running. (ps –ef | grep sshd)
- ✓ Check firewall open (default is all open in ubuntu server)

References

◈ Reference Sites

- ✓ http://christophermaier.name/blog/2010/09/01/host-only-networking-with-virtualbox
- ✓ http://allisterx.blogspot.com/2008/05/additions-and-ssh-access-to-virtualbox.html
- ✓ http://serverfault.com/questions/308229/virtual-box-host-only-adapter-configuration
- ✓ http://jackal777.wordpress.com/2012/02/13/internet-access-in-virtualbox-host-only-networking/
- ✓ http://www.ubuntugeek.com/how-to-set-up-host-interface-networking-for-virtualbox-on-ubuntu.html
- ✓ http://superuser.com/questions/144453/virtualbox-guest-os-accessing-local-server-on-host-os
- ✓ http://www.virtualbox.org/manual/ch09.html#changenat