

Teknologi Virtualisasi: Virtualisasi Hardware dengan KVM & QEMU

STT Terpadu Nurul Fikri

Henry Saptano, M.Kom

November , 2020

Agenda

- Pengantar KVM
- Mekanisme KVM
- QEMU
- KVM & QEMU
- Komponen KVM
- Praktek

Pengantar KVM

- **Kernel-based Virtual Machine (KVM)** adalah solusi virtualisasi yang open source di sistem operasi Linux , untuk platform arsitektur CPU x86. Pada dasarnya ini adalah sebuah **modul kernel yang terintegrasi** kedalam kernel Linux.
- KVM menggunakan virtualisasi yang dibantu perangkat keras (hardware assisted) melalui ekstensi perangkat keras. Intel dan AMD menyediakan ekstensi perangkat keras dengan teknologi Intel VT-x dan AMD-V dan KVM memanfaatkan teknologi ini.

Pengantar KVM

- KVM terdiri dari dua komponen utama:
 - **Satu set modul Kernel (kvm.ko, kvm-intel.ko, dan kvm-amd.ko)** yang menyediakan infrastruktur virtualisasi inti dan driver khusus prosesor
 - **Program userspace (qemu-kvm)** yang menyediakan **emulasi** untuk perangkat virtual dan **mekanisme kontrol untuk mengelola VM Guest** (mesin virtual).
- Istilah KVM lebih tepat mengacu pada **fungsionalitas virtualisasi** tingkat Kernel, tetapi dalam praktiknya lebih umum digunakan untuk mereferensikan komponen ruang pengguna.

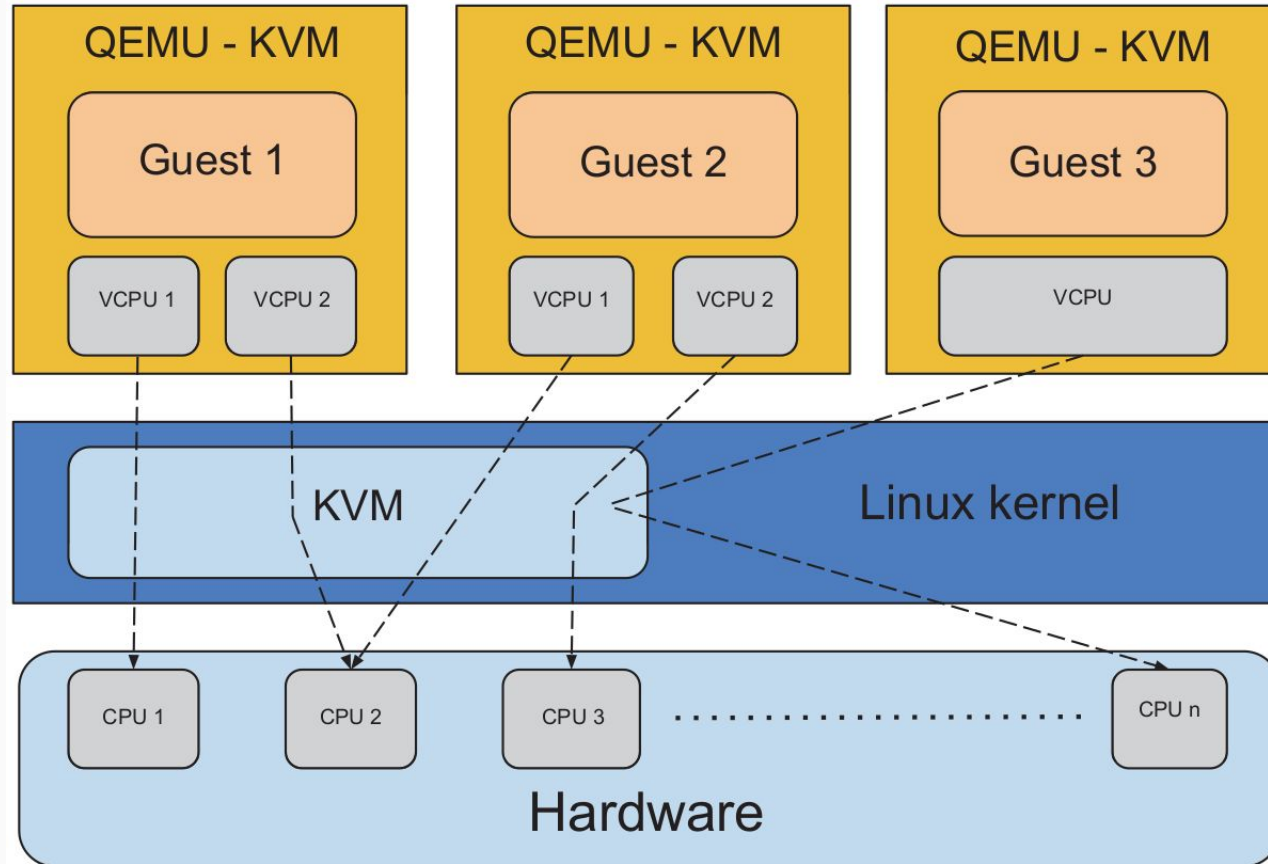
Mekanisme KVM

- KVM adalah modul kernel yang diimplementasikan sedemikian rupa sehingga kernel Linux memiliki peran **hypervisor** dan KVM hanya menangani emulasi tamu (vm).
- Modul KVM menangani semua komunikasi antara tamu (vm) dan perangkat keras, lihat Gambar-1. Modul kernel KVM diekspos ke tamu (vm) melalui antar muka **/dev/kvm**

Mekansime KVM

- Sebuah tamu (vm) biasanya diinisialisasi dengan alat / program ruang pengguna misalnya **QEMU** yang memiliki **dukungan KVM**. KVM bertanggung jawab atas pengelolaan tamu (vm) untuk menjadi tuan rumah (host). Ini memerlukan pengelolaan register prosesor, register MMU dan register terkait perangkat keras untuk perangkat keras PCI yang diemulasi.
- Prosesor tamu (vm) diemulasi dengan menjalankan utas (thread) di ruang pengguna dan dijadwalkan dengan proses Linux lainnya oleh hypervisor, yaitu kernel Linux.

Gambar-1. Gambaran umum dari struktur lapisan KVM



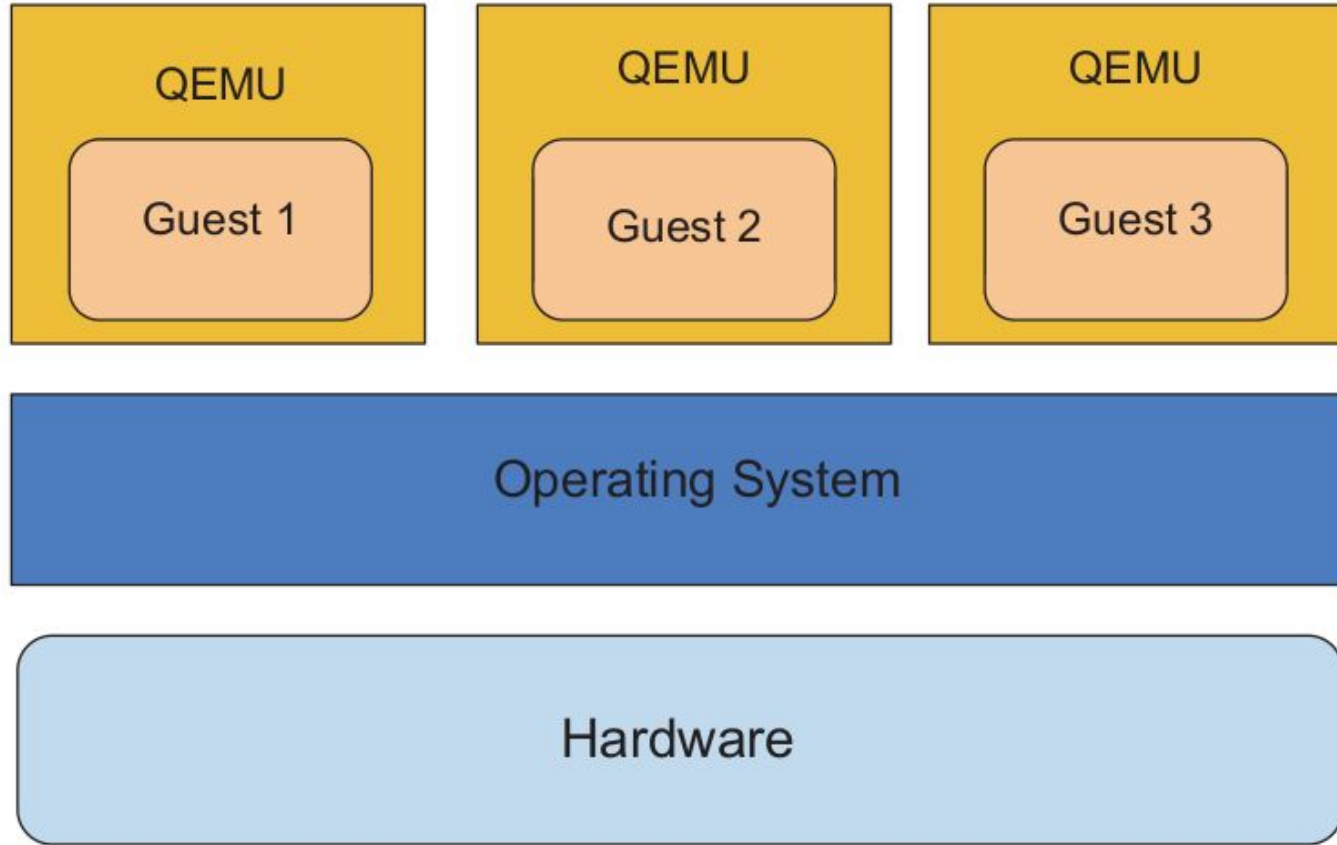
Mekansime KVM

- Memori fisik tamu (vm) dipetakan ke virtual host dan ditangani oleh alat ruang pengguna dengan menyimpan tabel halaman bayangan. Biasanya terjemahan memori diemulasi dalam perangkat lunak tetapi dengan penambahan teknologi baru Intel dan AMD untuk dukungan virtualisasi, penanganan pemetaan memori telah dipindahkan ke perangkat keras. Ini memungkinkan tamu (vm) untuk mengelola tabel halamannya (page tables) sendiri. Akses I / O dan penyimpanan juga ditangani oleh alat ruang pengguna

QEMU

- QEMU adalah singkatan dari **Quick Emulator** dan merupakan **emulator prosesor open source** yang dapat menjalankan sistem operasi yang tidak dimodifikasi dalam mesin virtual. Lihat Gambar-2. untuk ilustrasi QEMU.
- Bersama dengan KVM , QEMU dapat berperan sebagai Virtualizer. Menjadi solusi open source untuk menerapkan virtualisasi penuh dengan bantuan teknologi virtualisasi pada perangkat keras arsitektur prosesor Intel (VTx) dan AMD (AMD-v)
- QEMU dapat diinstal pada beberapa sistem operasi seperti Linux, Mac OS X dan Windows dan beberapa lainnya. Arsitektur prosesor host dan target bisa berbeda atau sama.

Gambar-2. Ringkasan lapisan arsitektur QEMU.



QEMU

- QEMU terdiri dari beberapa subsistem, emulator CPU, perangkat yang diemulasi, perangkat generik, deskripsi mesin untuk pembuatan instance perangkat yang diemulasi, debugger, dan antarmuka pengguna.
- QEMU dapat meniru beberapa arsitektur perangkat keras seperti ARM, x86, PowerPC.
- QEMU juga dapat meniru perangkat keras seperti kartu jaringan, tampilan VGA, dan periferal lainnya

QEMU

- Perangkat keras seperti jaringan, penyimpanan, antarmuka IO, PCI yang diekspos ke tamu (vm) diemulasi dan ditangani oleh QEMU kecuali saat KVM dipanggil dalam hubungannya dengan QEMU. Kemudian QEMU dimulai dengan parameter **--enable-KVM** dan **emulasi CPU di QEMU dimatikan**.
- Memori fisik tamu (vm) ditiru dengan memetakannya ke dalam ruang alamat prosesnya sendiri. QEMU mengalokasikan memori untuk tamu di awal. Di host, QEMU dijalankan di ruang pengguna dan dijadwalkan oleh sistem operasi.
- Instruksi CPU tamu (vm) dijalankan pada CPU host melalui hypervisor tanpa terjemahan instruksi dalam kasus di mana arsitektur CPU tamu dan host identik.

QEMU - Portable dynamic translations

- **Proses penerjemahan** dimulai dengan **membagi semua instruksi CPU tamu (vm)** menjadi unit yang lebih kecil dan lebih sederhana yang disebut operasi mikro.
- Operasi mikro ini kemudian secara manual dikodekan ke dalam fungsi C dan dikompilasi oleh GCC ke file objek host. Set operasi mikro jauh lebih kecil dari set instruksi CPU tamu.

QEMU - Portable dynamic translations


- QEMU kemudian menggunakan alat **dyngen** pada saat menjalankan untuk membongkar instruksi tamu ke operasi mikro. Alat dyngen memetakan instruksi mikro ini ke instruksi yang telah dikompilasi sebelumnya di file objek dan menggabungkannya untuk dieksekusi pada host. Terjemahan biner dapat mencapai kecepatan eksekusi yang hampir sama dengan mesin host

KVM and QEMU - Type 1 or Type 2 hypervisor

Halaman web KVM dan QEMU dengan jelas menunjukkan bahwa **KVM membutuhkan QEMU** untuk menyediakan **fungsi-fungsi hypervisor penuh**. Dengan sendirinya, KVM lebih merupakan penyedia infrastruktur virtualisasi.

KVM and QEMU - Type 1 or Type 2 hypervisor

[linux-kvm.org/page/Main_Page](#)
Gmail YouTube Maps Quickstart... Tutorial —... Get mongo... Building R... aleksxp's P... PN Realtime V... M Real time... Welcome t... roug



Create account Log in

Search

Home Status and Features ▾ Develop ▾ Conferences ▾ About ▾

Discussion View source History

Kernel Virtual Machine

KVM (for Kernel-based Virtual Machine) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, `kvm.ko`, that provides the core virtualization infrastructure and a processor specific module, `kvm-intel.ko` or `kvm-amd.ko`.

Using KVM, one can run multiple virtual machines running unmodified Linux or Windows images. Each virtual machine has private virtualized hardware: a network card, disk, graphics adapter, etc.

KVM is open source software. The kernel component of KVM is included in mainline Linux, as of 2.6.20. The userspace component of KVM is included in mainline QEMU, as of 1.3.

Blogs from people active in KVM-related virtualization development are syndicated at <http://planet.virt-tools.org/>

KVM and QEMU - Type 1 or Type 2 hypervisor

QEMU sendiri adalah **hypervisor Tipe-2**. Ini mencegat instruksi yang dimaksudkan untuk Virtual CPU dan menggunakan sistem operasi host untuk menjalankan instruksi tersebut pada CPU fisik. **Saat QEMU menggunakan KVM** untuk akselerasi perangkat keras, kombinasinya menjadi **hypervisor Tipe-1**. Perbedaan ini cukup jelas terlihat dari uraian di situs QEMU.

KVM and QEMU - the x86 dependency

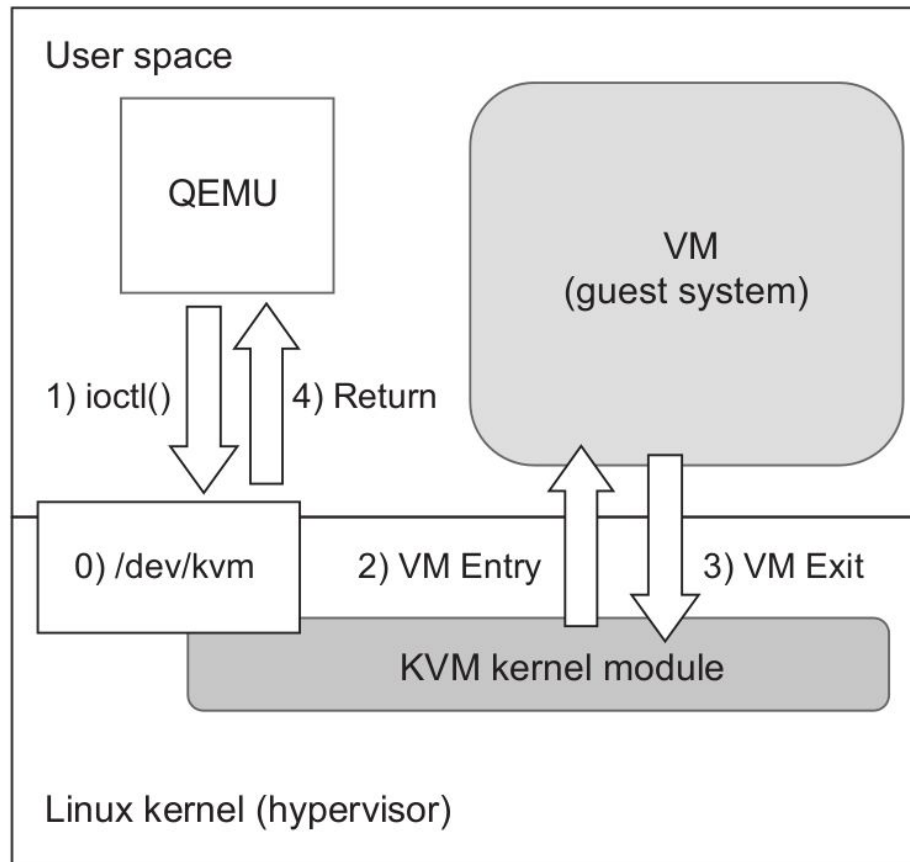
Karena KVM benar-benar merupakan sebuah driver untuk kemampuan CPU fisik, KVM sangat erat kaitannya dengan arsitektur CPU (arsitektur x86). Artinya, manfaat akselerasi perangkat keras hanya akan tersedia jika CPU Mesin Virtual juga menggunakan arsitektur yang sama (x86). Jika VM perlu menjalankan Power PC CPU tetapi server hypervisor memiliki CPU Intel, maka KVM tidak akan berfungsi. Anda harus menggunakan QEMU sebagai Jenis Virtualisasi dan berjalan dengan kinerja yang overhead.

QEMU/KVM execution flow

Alur eksekusi QEMU / KVM ditampilkan di Gambar-3.

- Pertama, **file bernama /dev/kvm** dibuat oleh modul kernel KVM (langkah 0 pada gambar). File ini memungkinkan QEMU untuk menyampaikan berbagai permintaan ke modul kernel KVM untuk menjalankan fungsi hypervisor.
- Ketika QEMU mulai menjalankan sistem tamu, itu berulang kali membuat panggilan sistem `ioctl ()` dengan menentukan file khusus ini (atau deskriptor file yang diturunkan darinya)

QEMU/KVM execution flow



QEMU/KVM execution flow

- Ketika tiba waktunya untuk mulai menjalankan sistem tamu, QEMU kembali memanggil **ioctl ()** untuk menginstruksikan modul kernel KVM untuk memulai sistem tamu (langkah 1).
- Modul kernel, pada gilirannya, melakukan Entri VM (langkah 2) dan mulai menjalankan sistem tamu. Nanti, ketika sistem tamu akan mengeksekusi instruksi sensitif, VM Keluar dilakukan (langkah 3), dan KVM mengidentifikasi alasan keluar.

QEMU/KVM execution flow

- Jika intervensi QEMU diperlukan untuk menjalankan tugas I / O atau tugas lain, kontrol ditransfer ke proses QEMU (langkah 4), dan QEMU menjalankan tugas tersebut. Saat eksekusi selesai, QEMU kembali membuat panggilan sistem `ioctl ()` dan meminta KVM untuk melanjutkan pemrosesan tamu (yaitu, aliran eksekusi kembali ke langkah 1). Aliran QEMU / KVM ini pada dasarnya diulangi selama emulasi VM.

QEMU/KVM execution flow

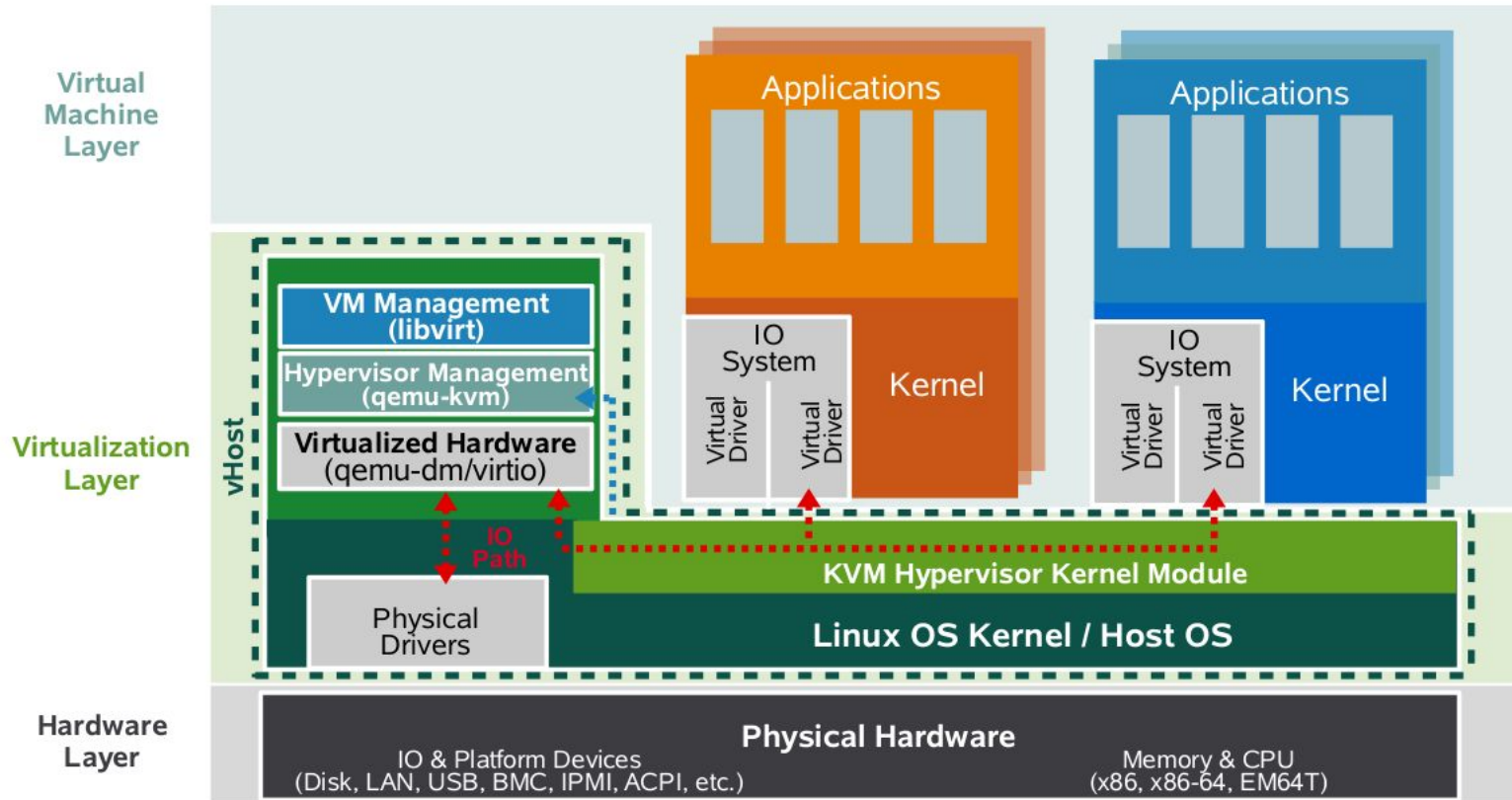
QEMU / KVM dengan demikian memiliki struktur yang relatif sederhana.

- Implementasi modul kernel KVM mengubah kernel Linux menjadi hypervisor.
- Ada satu proses QEMU untuk setiap sistem tamu. Saat beberapa sistem tamu berjalan, jumlah proses QEMU yang sama sedang berjalan.

QEMU/KVM execution flow

- QEMU adalah program multi-utas (multi-thread), dan satu CPU virtual (VCPU) dari sistem tamu sesuai dengan satu utas QEMU. Langkah 1–4 pada Gambar-3 dilakukan dalam satuan utas.
- Utas (thread) QEMU diperlakukan seperti proses pengguna biasa dari sudut pandang kernel Linux. Penjadwalan untuk utas yang sesuai dengan CPU virtual sistem tamu, misalnya, diatur oleh Penjadwal kernel Linux dengan cara yang sama seperti utas proses lainnya.

KVM Virtualization Architecture



KVM Virtualization Architecture

- Dengan KVM (atau Mesin Virtual Kernel), **modul kernel dimuat ke dalam kernel Linux** yang mengubahnya menjadi hypervisor.
- KVM pada dasarnya akan menjadi Hypervisor Tipe I karena berjalan langsung di atas perangkat keras.
- Dengan KVM, kernel Linux menjadi hypervisor "gemuk" karena tidak hanya menengahi akses ke perangkat keras yang mendasarinya tetapi juga memuat driver fisik dan berbagi akses ke perangkat keras yang mendasarinya dengan driver virtual.
- Emulasi perangkat dan manajemen VM ditangani oleh versi QEmu yang dimodifikasi yang berjalan di ruang pengguna.

KVM Components

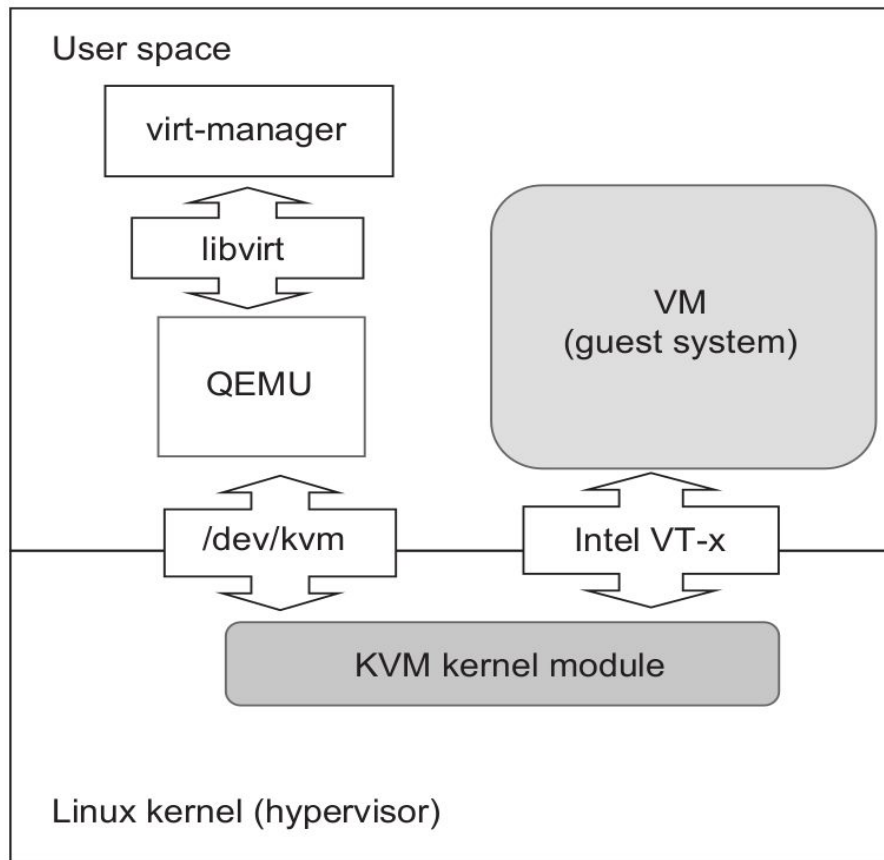
Struktur keseluruhan KVM, dari GUI hingga kernel Linux, mencakup lima komponen utama

1. **Virt-manager & Virsh** - Antarmuka pengguna GUI / CUI yang digunakan untuk mengelola VM; itu memanggil fungsi VM menggunakan libvirt
2. **Libvirt** - Pustaka alat dan antarmuka umum untuk perangkat lunak virtualisasi server yang mendukung Xen, VMware ESX / GSX, dan, tentu saja, QEMU / KVM

KVM Components

3. **QEMU** - Emulator yang berinteraksi dengan modul kernel KVM dan menjalankan berbagai jenis pemrosesan sistem tamu seperti I / O; satu proses QEMU sesuai dengan satu sistem tamu.
4. **KVM Kernel module** - Dalam arti sempit, KVM adalah modul kernel Linux; itu menangani VM Exit dari sistem tamu dan menjalankan instruksi VM Entry
5. **Linux Kernel** - Karena QEMU berjalan sebagai proses biasa, penjadwalan sistem tamu terkait ditangani oleh kernel Linux itu sendiri.

KVM Components



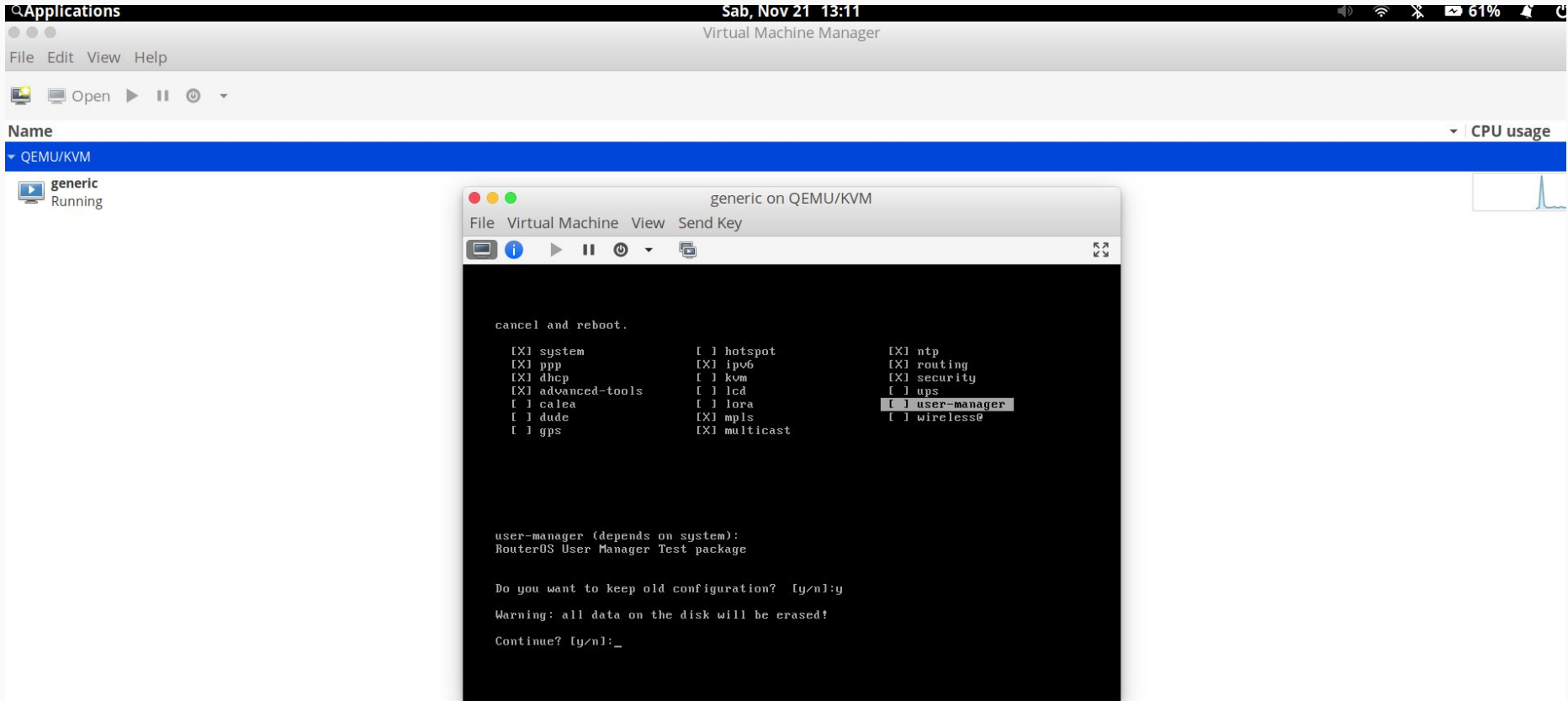
KVM Components

- Instalasi KVM components
- Virt-manager
- Virsh

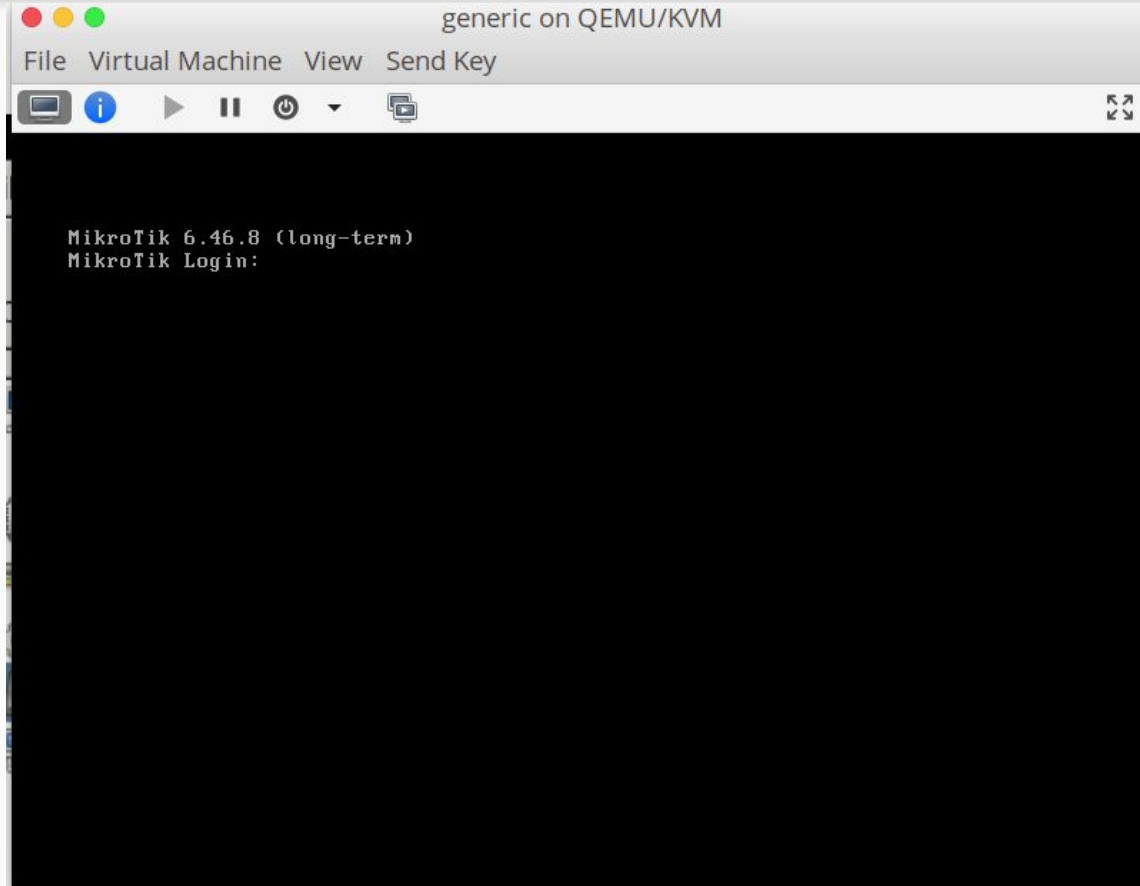
Instalasi kvm components

```
$ sudo apt-get install qemu-system-x86 qemu-kvm  
libvirt-bin libvirt-clients virt-manager
```

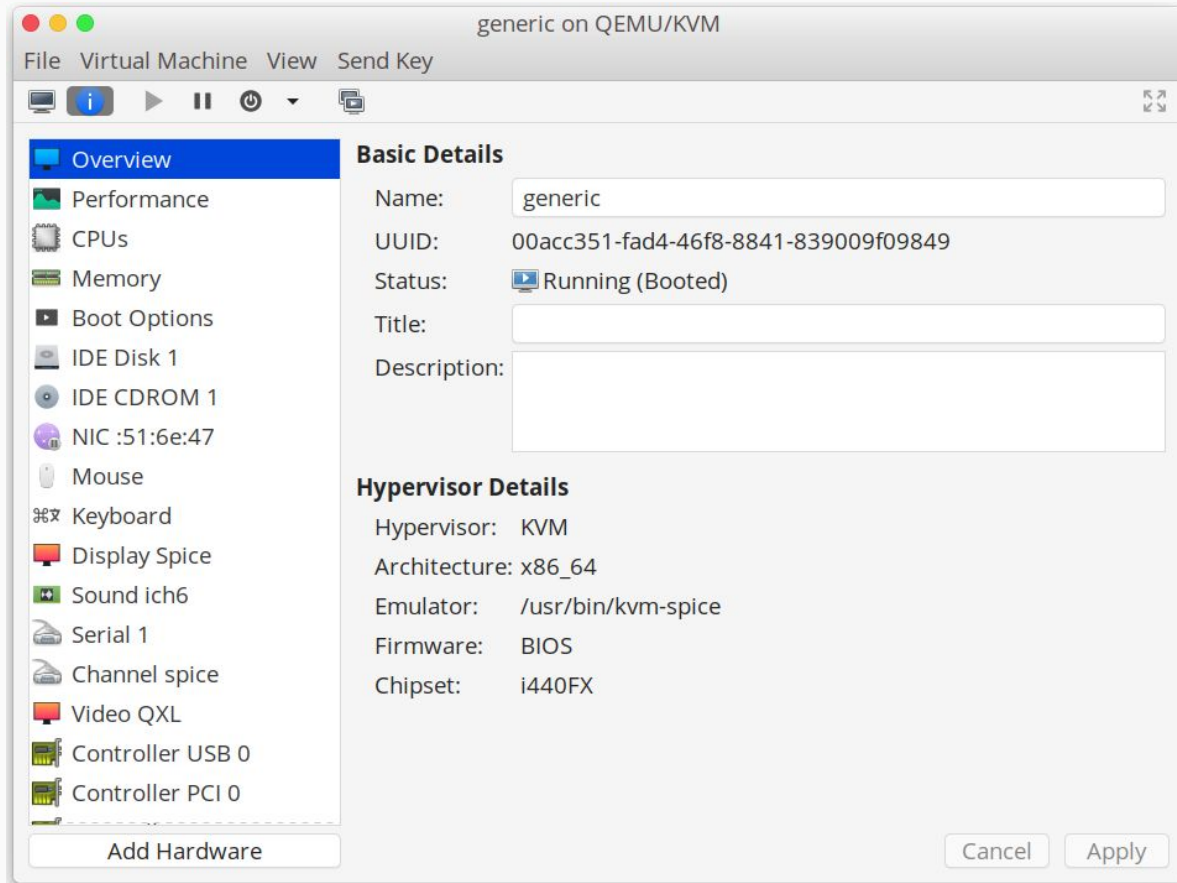
Virt-manager - main view



Virt-manager - console view



Virt-manager - VM details/settings



- `virsh list [--all]`
 - list running (or all) VMs
- `virsh start VM`
 - start the VM named *VM*
- `virsh shutdown VM`
 - shutdown VM (properly)
- `virsh destroy VM`
 - kill a VM (power off)
- `virsh console VM`
 - connect to the serial console of a VM
- `virsh define FILE`
 - create VM definition from this XML file
- `virsh undefine VM`
 - erase the machine definition (danger!)

- `virsh dumpxml VM`
 - show the XML
- `virsh edit VM`
 - open XML in editor

```
<domain type='kvm'>
  <name>noc.ws.nsrc.org</name>
  <uuid>4641a945-abab-1c0b-0fb0-2db681c28130</uuid>
  <memory>1048576</memory>
  <currentMemory>1048576</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-1.0'>hvm</type>
    <boot dev='hd' />
  </os>
  ...
```

Praktek/demo