LinuC レベル 1 Version10.0
技術解説無料セミナー
2020/06/21 開催

主題	2.05 仮想化サーバー

本日の講師

エスディーテック株式会社
末永 貴一

■自己紹介

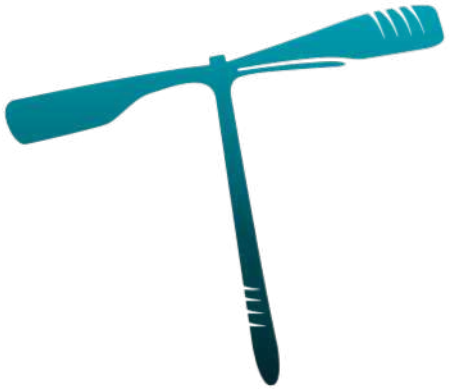
#LinuC学習中

sdtech
Sustainable Design & Technology

エスディーテック株式会社
- 組込み機器向けUI/UXソフトウェアの研究開発
  http://www.sdtech.co.jp

Linux関連文章の執筆
- ・LPIC Level2 1回で合格必達テキスト+問題集
- ・LPI-Japan コラム【Linux道場 入門編】
- ・@IT 「Linuxをいまから学ぶコツ教えます」
- ・@IT 「Linuxに触れよう」
- ・日経Linux「Xと次世代「Wayland」を知る」

など



LinuCレベル１／レベル２ Version10.0とは

#LinuC学習中

■LinuCとは

クラウド時代の即戦力エンジニアであることを証明するLinux技術者認定資格

✓現場で「今」求められている新しい技術要素に対応

- オンプレミス／仮想化を問わず様々な環境下でのサーバー構築
- 他社とのコラボレーションの前提となるオープンソースへの理解
- システムの多様化に対応できるアーキテクチャへの知見

✓全面的に見直した、今、身につけておくべき技術範囲を網羅

今となっては使わない技術やコマンドの削除、アップデート、新領域の取り込み

✓Linuxの範疇だけにとどまらない領域までカバー

セキュリティや監視など、ITエンジニアであれば必須の領域もカバー

■Version10.0と従来の出題範囲の比較

#LinuC学習中

	テーマ	Version 10.0	従来
LinuC-1	仮想技術	・仮想マシン／コンテナの概念 ・クラウドセキュリティの基礎	← （Version10.0で新設）
	オープンソースの文化	・オープンソースの定義や特徴 ・コミュニティやエコシステムへの貢献	← （Version10.0で新設）
	その他	→ （Version10.0で削除）	アクセシビリティ、ディスククォータ、プリンタの管理、SQLデータ管理、他
LinuC-2	仮想化技術	・仮想マシンの実行と管理（KVM） ・コンテナの仕組みとDockerの導入	← （Version10.0で新設）
	システムアーキテクチャ	・クラウドサービス上のシステム構成 ・高可用システム、スケーラビリティ、他	← （Version10.0で新設）
	その他	・統合監視ツール（zabbix） ・自動化ツール（Ansible）	← （Version10.0で出題範囲に含む）
		→ （Version10.0で削除）	RAID、記憶装置へのアクセス方、FTPサーバーの保護、他

#LinuC学習中

今回のテーマ

主題2.05：仮想化サーバー

- ✓ 仮想マシン
- ✓ ハイパーバイザー
- ✓ KVM
- ✓ QEMU
- ✓ 仮想マシン環境構築
- ✓ 準仮想化

仮想マシンの基礎

■仮想マシンとは・・・

#LinuC学習中

✓コンピュータ（機械）の動作をエミュレーションするもの
- 機械（ハードウェア）自体の動作をソフトウェア的に再現する。
- 動作しているOS（システム）上に別のコンピュータを実現させる。

✓同一コンピュータ上に複数のコンピュータ・OSを動作可能とする
- 一般的にはOS上に異なるOSを複数動させることができる環境
- 同一コンピュータ上に同時に複数のOS（ゲストOS）を動作させることができる。

以上のような仮想マシンを実現させるために必要な演算管理、メモリ管理、入出力管理等を行う制御プログラムを**ハイパーバイザー**（仮想化モニター、仮想化OS）と呼ぶ。

参考：
シミュレータは外から見た振る舞いを再現するもの。エミュレータは中身の動作まで再現するもの。

仮想マシンの基礎

■ハイパーバイザーの種類

#LinuC学習中

✓ Type1 ハイパーバイザー

- ハードウェア上で直接動作する。
- ゲストOSはこのハイパーバイザー上で動作する。
- ベアメタルハイパーバイザー、ネイティブハイパーバイザーとも呼ばれる。
- Xen、KVM等

✓ Type2 ハイパーバイザー

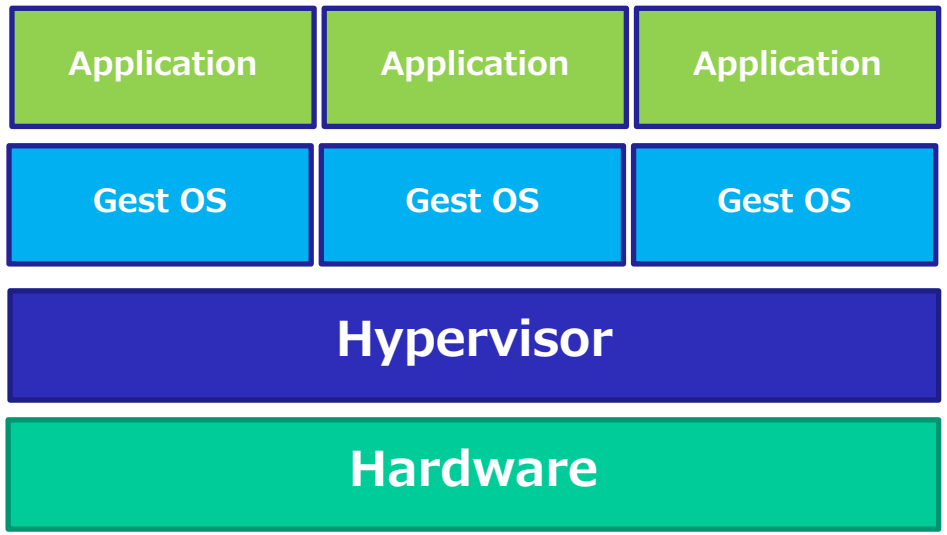
- ハードウェア上で動作するOS上でアプリケーションとして動作する。
- ハイパーバイザーアプリケーション上でゲストOSは動作する。
- ホストハイパーバイザーとも呼ばれるが、厳密にはハイパーバイザーではない。
- VirtualBox、QEMU等

仮想マシンの基礎

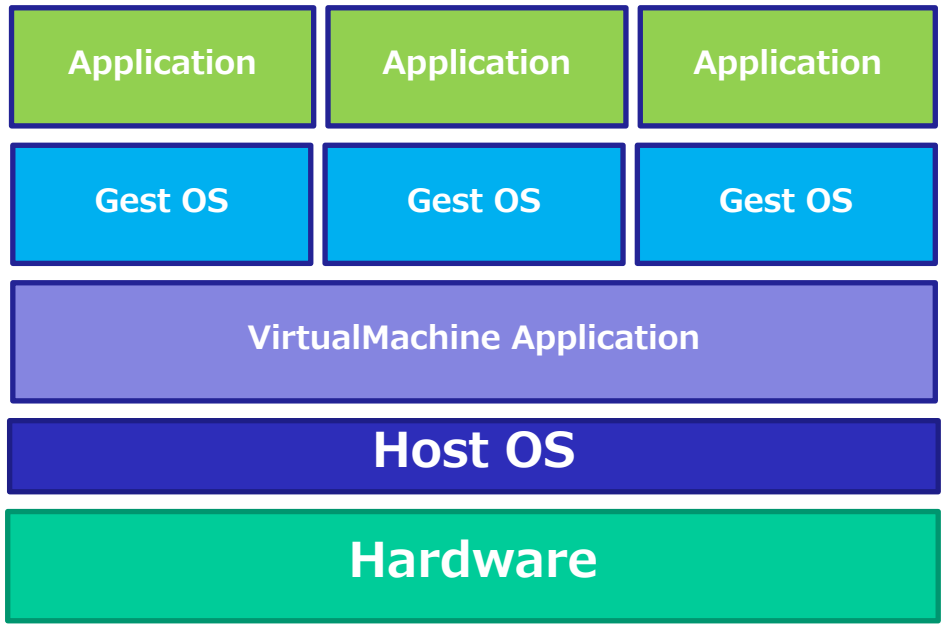
■ハイパーバイザーの種類

#LinuC学習中

Type1



Type2



KVMとは

■KVM（Kernel-based Virtual Machine）とは

#LinuC学習中

✓ LinuxをType1ハイパーバイザーとして機能させる技術

- Linuxカーネルに組み込まれた仮想化技術（kvm.ko）。
- Linux Kernelの機能を使用する。
- CPUの仮想化支援機能（Intel VTやAMD-V）を利用
- QEMU（PCエミュレーター）と合わせて使用する。

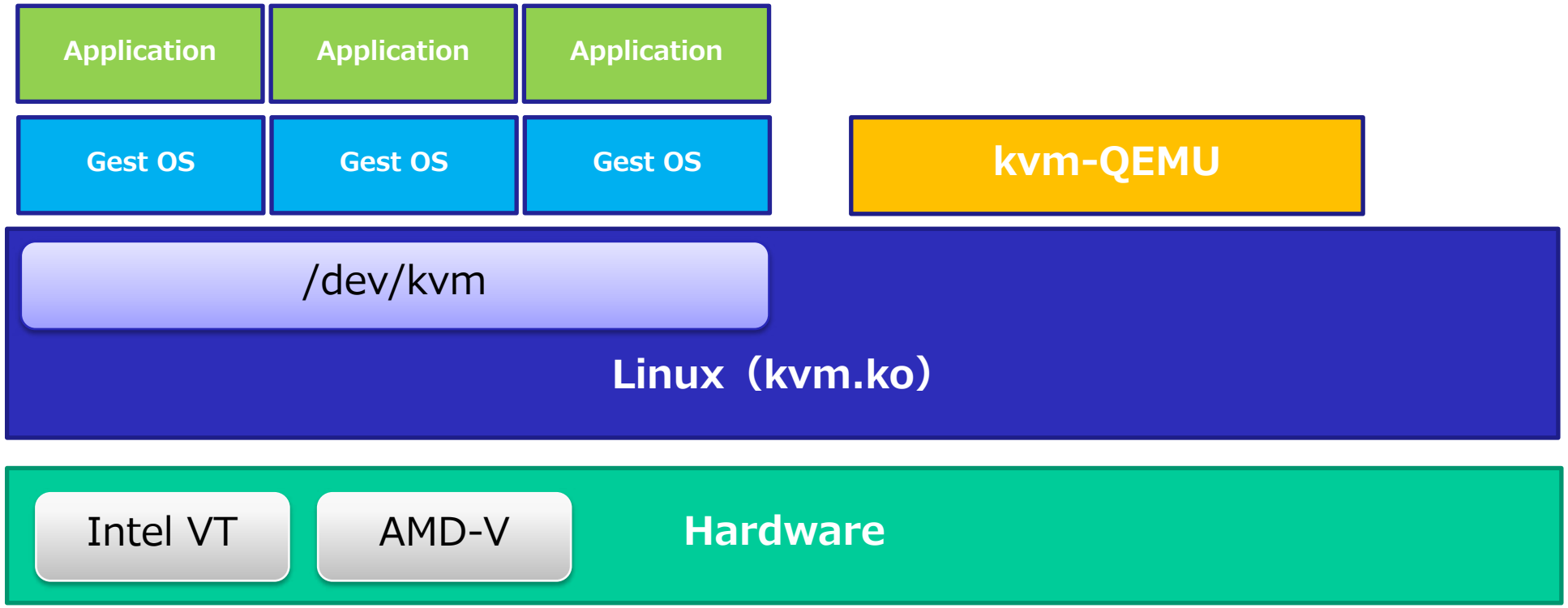
➢ 仮想化拡張機能の確認は/proc/cpuinfoやlscpuコマンドで確認

例：
$ grep -E 'svm|vmx' /proc/cpuinfo

KVMとは

■KVM（Kernel-based Virtual Machine）とは

#LinuC学習中



QEMUとは

■QEMU

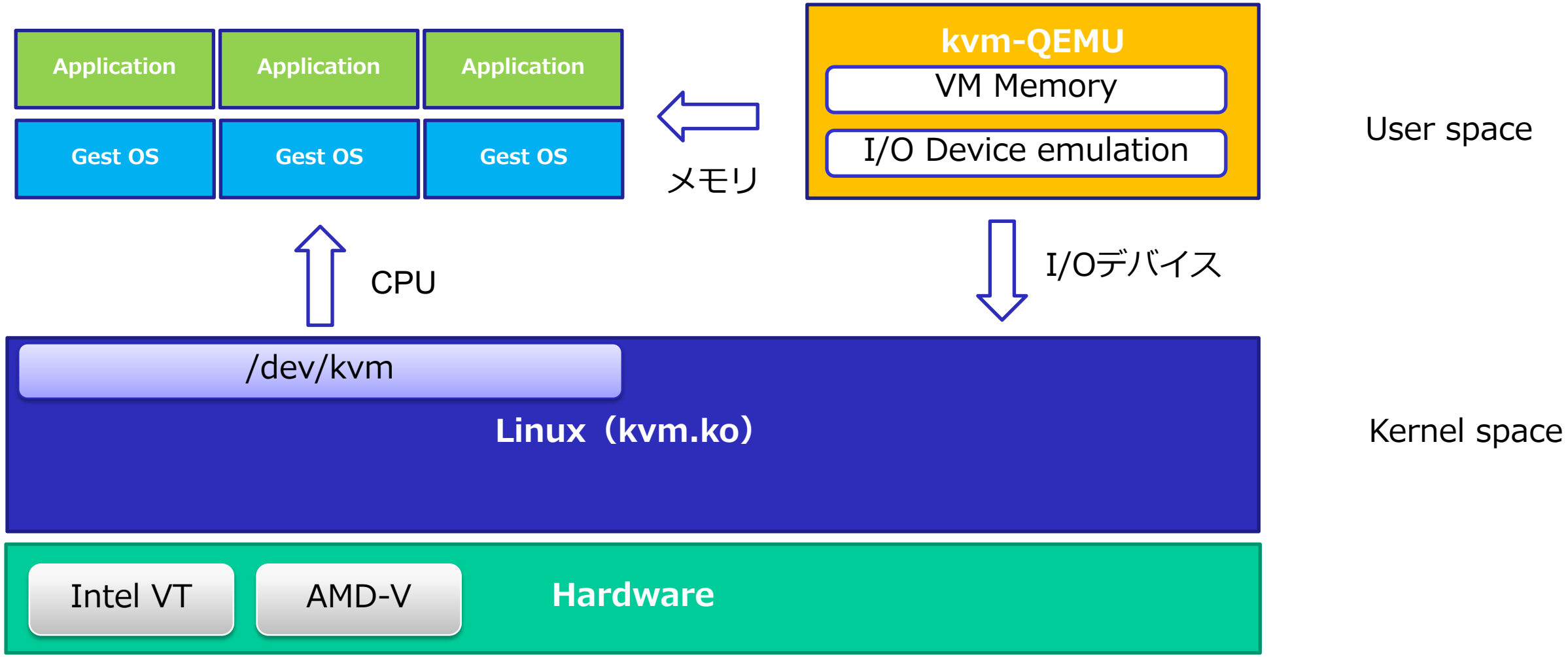
#LinuC学習中

- ✓ 機械全体をエミュレートするマシンエミュレーター
  - システムエミュレーションを行うことができる（Type2ハイパーバイザー）
  - 様々な周辺ハードウェアもエミュレーションする
  - KVMと組合わせることで完全仮想化環境を提供
    - KVMは演算管理系、QEMUはメモリ管理・入出力デバイスエミューレーションを行う
  - ユーザエミュレーション機能はWine等で利用されている

QEMUとは

■QEMU

#LinuC学習中



KVM環境構築

■KVMを動させるために必要なパッケージ

#LinuC学習中

✓ 必須パッケージ

- qemu-kvm：QEMU
- libvirt-clients：仮想マシンを操作するコマンド等
- libvirt-daemon-system or libvirt：libvirtd等が含まれるサービス

✓ 補助パッケージ

- virt-manager ：仮想マシン管理用ツール
- libosinfo-bin：OSの種別情報
- libguestfs-tools：仮想マシンのディスクイメージにアクセするツール群
- bridge-utils：ブリッジネットワークユーティリティ

➢ libvirtdの動作確認はsystemctlコマンドで確認

例：
$ systemctl status libvirtd.service

仮想マシンの作成

■virt-installコマンド

#LinuC学習中

- ✓ 仮想マシンを作成するコマンド
  - libvirライブラリを使用したコマンドラインベースのツール
  - インストール設定をオプションで指定する。

オプション名	説明
--name	仮想マシンの名称
--hvm	完全仮想化の指定
--memory	仮想マシンに割り当てるメモリサイズ（単位MB）
--vcpu	仮想マシンに割り当てるCPUコア数
--os-variant	インストールするゲストOS
--disk	仮想マシンが使用するストレージ
--network	仮想マシンが使用するネットワークインターフェイス
--cdrom	仮想マシンが使用するCD-ROMデバイス（ISOファイルも指定可能）
--location	インストールするゲストOSの場所
--initrd-inject	キックスタートファイルのパス指定
--extra-args	カーネルパラメータ

KVM環境構築

■仮想マシンのインストール

#LinuC学習中

- ✓ virt-installを使用した仮想マシンの作成、ゲストOSのインストール
  - コマンド実行例

```
$ virt-install ¥
  --name centos7_test¥
  --ram 1024 ¥
  --vcpus 1 ¥
  --os-variant centos7.0 ¥
  --disk pool=default,size=5,format=qcow2 ¥
  --network network=default ¥
  --location /home/linuc/work/CentOS-7-x86_64-Minimal-2003.iso ¥
  --graphics none ¥
  --extra-args "console=ttyS0”
```

  - Linuxであればキックスタートファイルを利用すると便利

```
  --initrd-inject /home/linuc/work/centos7.ks.cfg ¥
  --extra-args "inst.ks=file:/centos7.ks.cfg console=ttyS0"
```

KVM環境構築

■仮想マシンの実行

#LinuC学習中

- ✓ 仮想マシンの起動
  - virtshコマンドのstartコマンドを使用して仮想マシンを起動

    $ virsh start *VM*名
- ✓ 仮想マシンの終了
  - シャットダウン

    $ virsh shutdown *VM*名
  - 強制終了

    $ virsh destroy *VM*名
- ✓ 仮想マシンへの接続
  - グラフィックス機能OFFの場合

    $ virsh console *VM*名

  - グラフィックス機能OFFの場合

    $ virt-viewer *VM*名

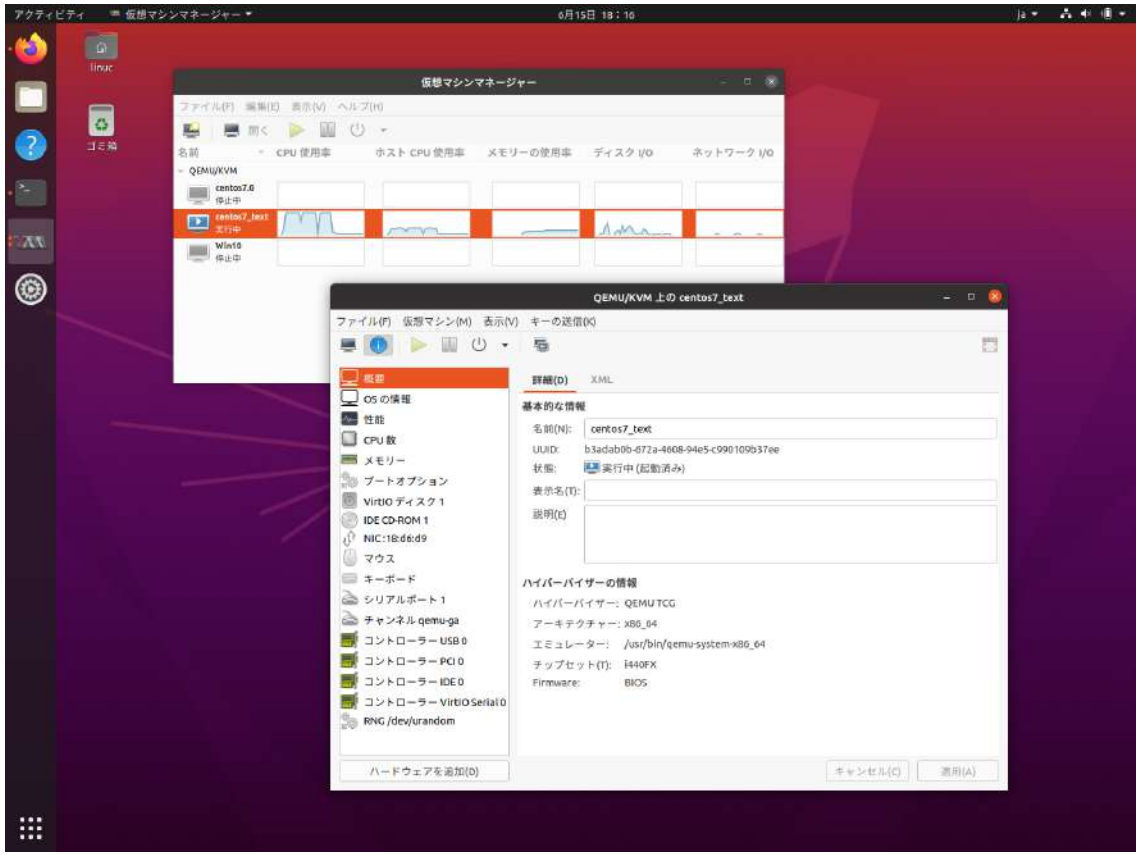
仮想マシンの管理

■virt-manager

✓GUIベースの仮想マシン管理ツール

- 仮想マシンの作成、監視、削除等が行えるツール
- GUIで仮想マシンの様々な設定が行える

#LinuC学習中



準仮想化

■準仮想化技術

#LinuC学習中

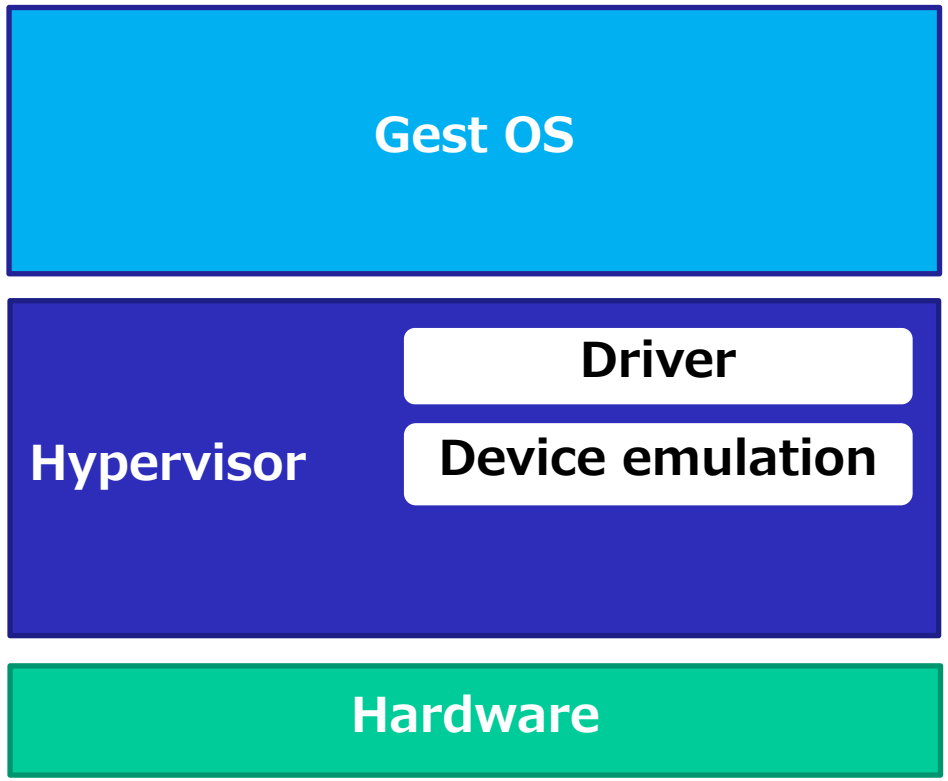
- ✓ 完全仮想化と準仮想化
  - 完全仮想化ではゲストOSは仮想化環境であることを意識しない
    - ➢ ゲストOSは修正なくハイパーバイザーの汎用的なエミュレーション環境を使用する。
  - 準仮想化ではゲストOSは仮想化環境であることを認識する
    - ➢ 効率化のためにゲストOS内に修正を行う。
- ✓ virtio
  - ゲストOSのIOを効率化するための準仮想化技術
  - OS専用のvirtioドライバを使用することでIO（ストレージ、ネットワーク等）を効率化

準仮想化

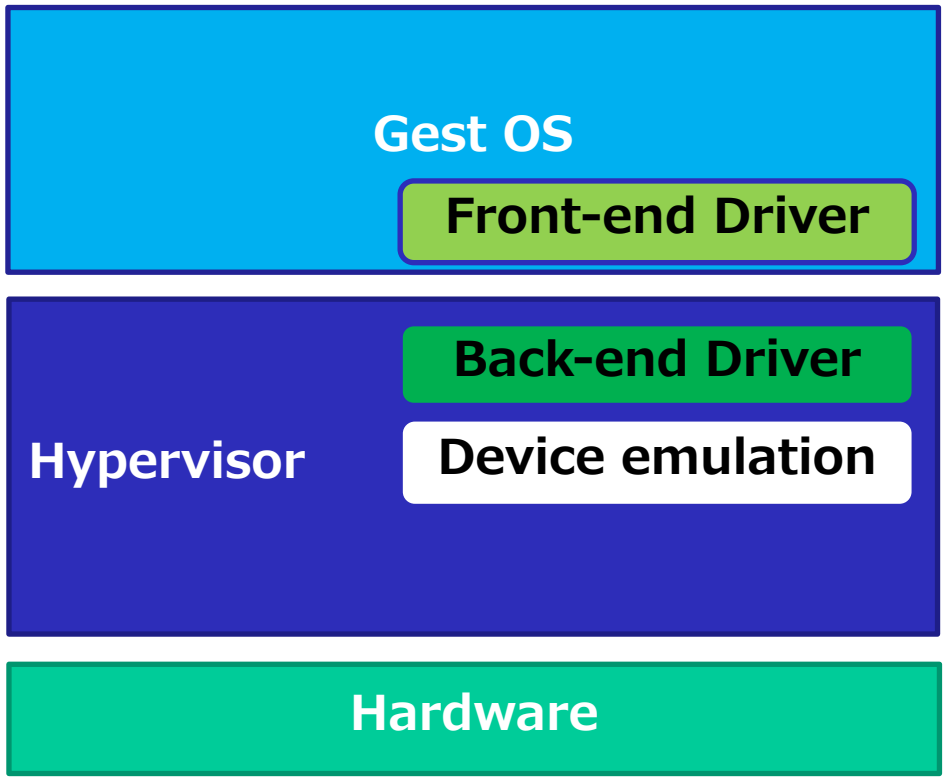
■準仮想化技術

#LinuC学習中

完全仮想化



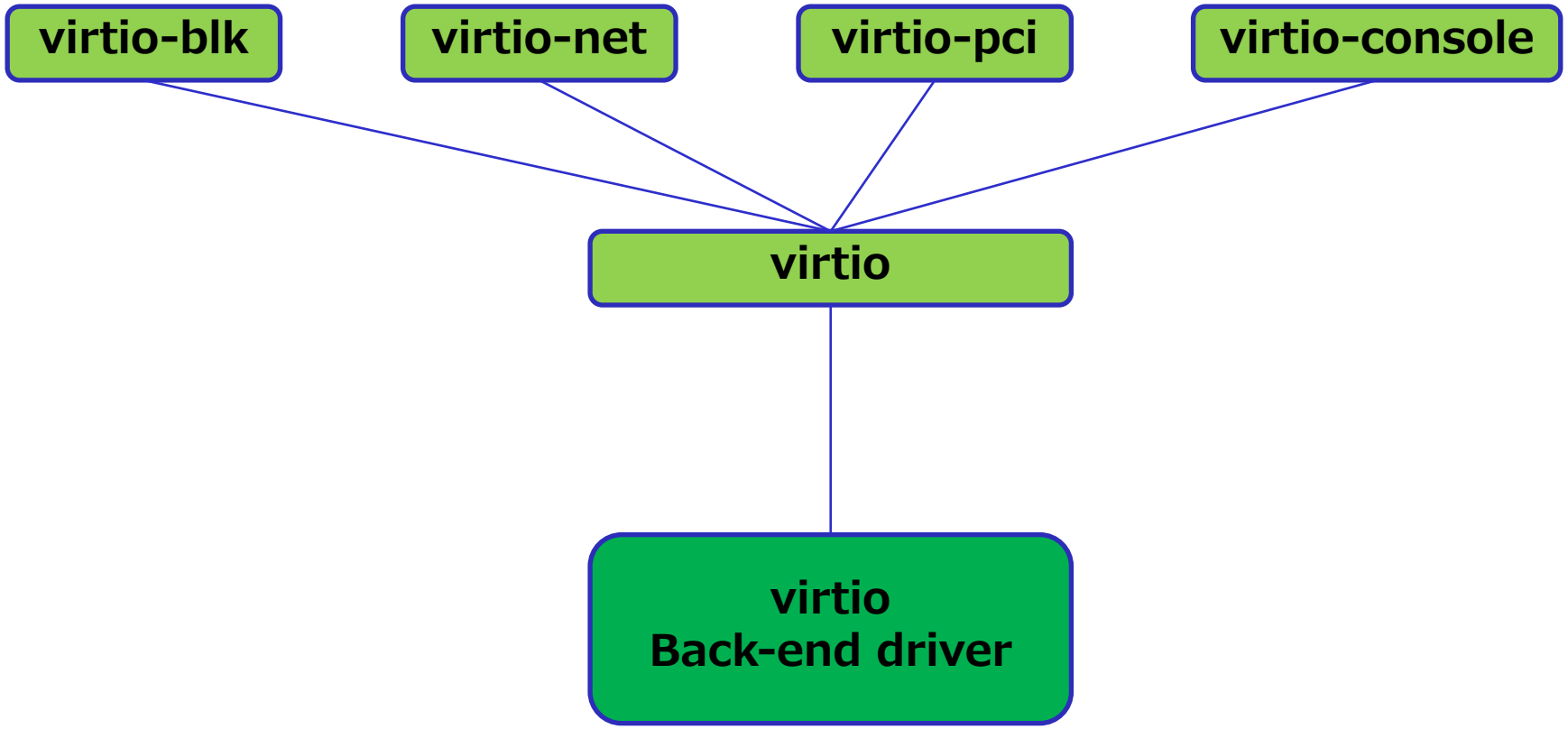
準仮想化



準仮想化

■virtio概要

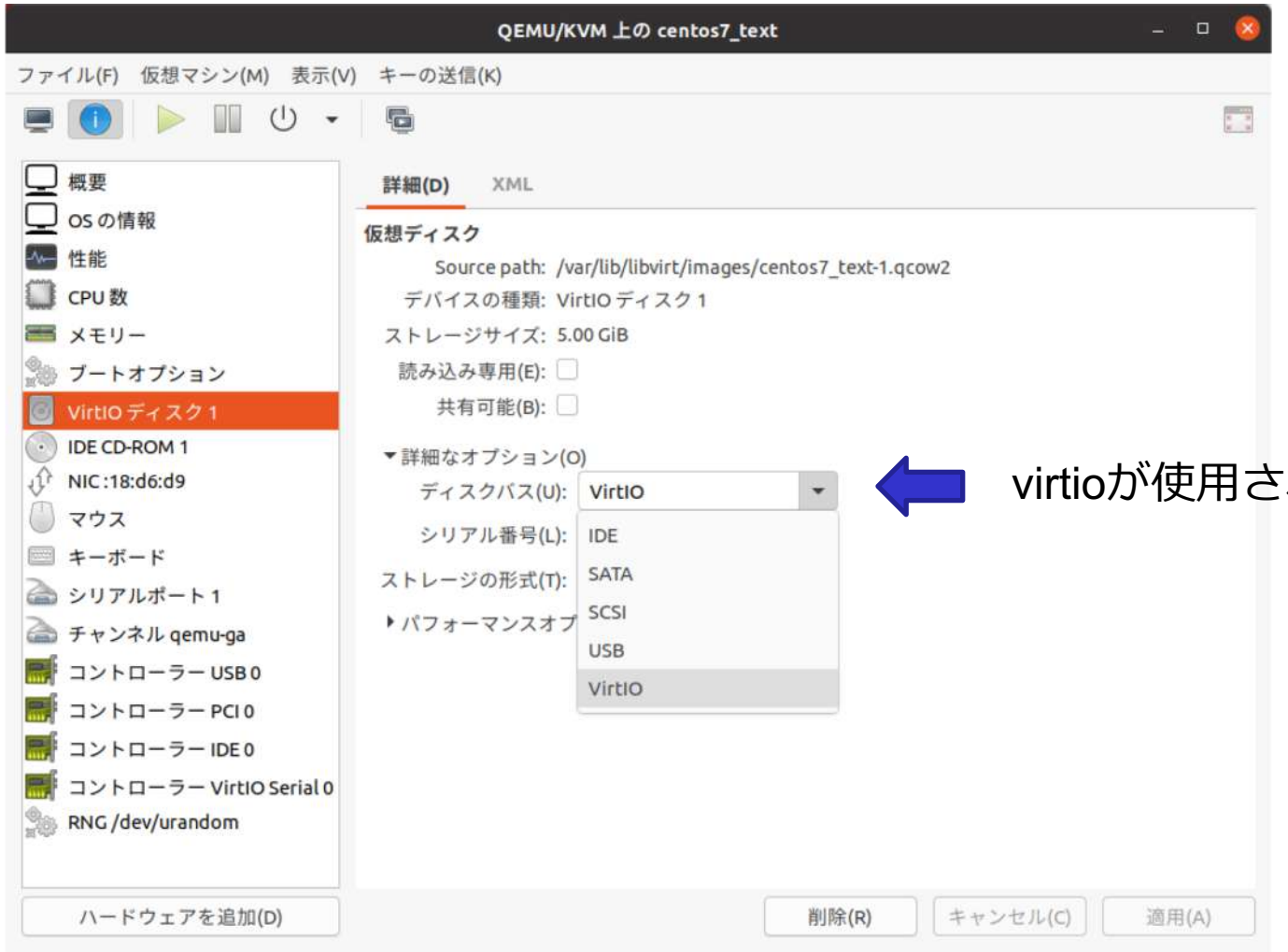
#LinuC学習中



準仮想化

■準仮想化技術

#LinuC学習中



virtioが使用されている

■まとめ

#LinuC学習中

- ✓ 仮想マシン
  - 完全仮想化ではゲストOSは仮想化環境であることを意識しない
- ✓ ハイパーバイザー
  - ゲストOSのIOを効率化するための準仮想化技術
- ✓ KVM
  - ゲストOSのIOを効率化するための準仮想化技術
- ✓ QEMU
  - ゲストOSのIOを効率化するための
- ✓ 仮想マシン環境構築
  - virt-install
  - virsh
  - virt-manager
- ✓ 準仮想化
  - virtio

#LinuC学習中

Q & A

#LinuC学習中

ありがとうございました

https://www.sdtech.co.jp