.Deb

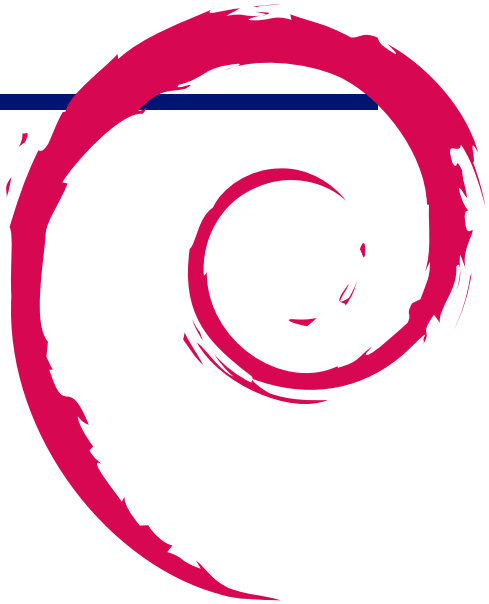
銀河系唯一のDebian専門誌

2022 年 10 月 15 日

KVM特集

デビアン勉強会

目次



1 最近の Debian 関連のミーティング報告

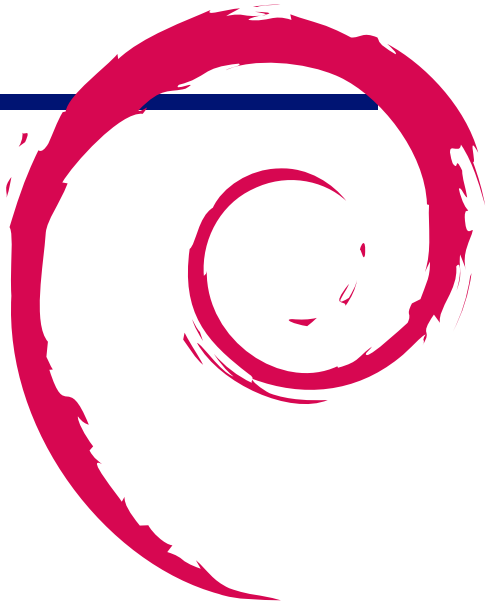
杉本 典充

1.1 2022 年 9 月度 東京エリア・関西合同 Debian 勉強会

2022 年 9 月 17 日 (土) に東京エリア Debian 勉強会と関西 Debian 勉強会の合同でオンラインによる Debian 勉強会を開催しました。参加者は 9 名でした。

セミナー発表は、Yosuke OTSUKI さんの「ffmpeg + GIMP と Blender で動画編集 (CLI で呼吸している人向け)」を行いました。

勉強会の終了後、参加者同士で Debian や OSS に関する話の情報交換を行いました。



2 事前課題

杉本 典充

今回の事前課題は以下です。

1. 使っているデスクトップ環境を教えてください
2. 使っているインプットメソッド (IM) と、かな漢字変換ソフトを教えてください

2.1 dictoss

1. GNOME on Wayland, GNOME on Xorg, MATE, Xfce
2. ibus-mozc, uim-anthy

2.2 Hiroyuki Yamamoto (yama1066)

1. KDE Plasma on Xorg
2. uim-mozc

2.3 NOKUBI Takatsugu (knok)

1. KDE Plasma on Xorg, Xfce
2. uim-mozc, uim-anthy

2.4 uwabami

1. その他
2. その他

2.5 yy_y_ja_jp

1. KDE Plasma on Xorg
2. uim-mozc

2.6 yosukesan (yosuke_san)

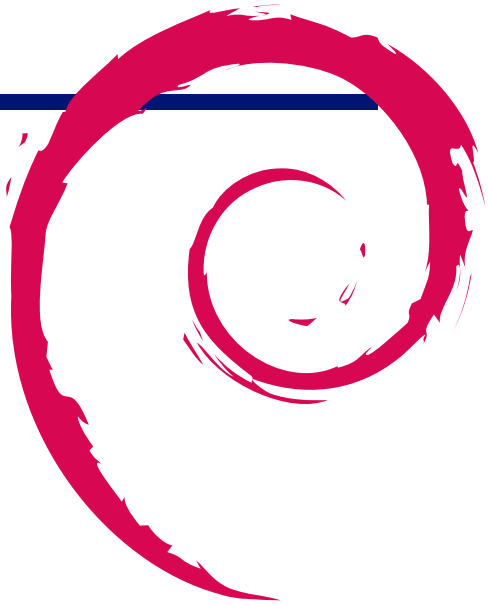
1. その他
2. その他

2.7 Kazuhiro NISHIYAMA (znz)

1. デスクトップ環境は使っていない
2. Linux 上で日本語変換処理は使っていない

2.8 kenhys

1. その他
2. fcitx-mozc



3 Debian で KVM を使う (CLI で呼吸している人向け)

杉本 典充

3.1 はじめに

自宅で使っている PC の入れ替えに伴い、Debian で KVM サーバを作りました。最近の Debian 勉強会では KVM の話題はないため、まとめてみました。今回は KVM をコマンドラインで操作する CLI ツールに焦点を当ててみます。

なお、動作環境については OS は Debian GNU/Linux 11 bullseye amd64、CPU は Intel Core i5-6500T とします。

3.2 KVM とは

KVM (for Kernel-based Virtual Machine) とは、「仮想化拡張機能 (Intel VT または AMD-V) を含む x86 ハードウェア上の Linux 用の完全仮想化ソリューション」です[*1]。公式の Web サイトには x86 とだけ書いてありますが、Debian Wiki の KVM の説明ページ (`https://wiki.debian.org/KVM`) には、ARM 系 CPU も KVM が使えると記載があります。

コンピュータの仮想化技術は 表 1 のように分類され、KVM は「完全仮想化 (ハイパーバイザ型)」に該当します[*2]。

3.3 KVM のインストール

3.3.1 KVM の動作条件確認

KVM を利用するには仮想化機能に対応した CPU が必要です。Intel 製 CPU であれば Intel-VT、AMD 製 CPU であれば AMD-V、ARM64 系 CPU であれば Arm Virtualization Extensions[*3] と呼ばれている機能を搭載しており、かつマザーボードの設定で有効にしている必要があります。

Intel 製 CPU または AMD 製 CPU であれば、

```
$ grep -E 'vmx|svm' /proc/cpuinfo
```

というコマンドで仮想化機能が有効になっているか調べることができます (Intel 製 CPU の場合は vmx、AMD 製 CPU の場合は svm の文字列が出現します)。

[*1] `https://www.linux-kvm.org/page/Main_Page`

[*2] 引用元：東京エリア Debian 勉強会 2013 年 4 月 杉本典充「debootstrap を有効活用してみよう」8.1 仮想化技術について を元に一部加筆

[*3] `https://developer.arm.com/documentation/ddi0406/c/System-Level-Architecture/The-System-Level-Programmers--Model/The-Virtualization-Extensions`

表 1　仮想化技術の分類

仮想化技術	実装例	仮想化環境の特徴
完全仮想化 (エミュレーション型)	QEMU VirtualBox	既存の OS を無修正のままゲスト環境として動作させる。ホスト OS で動作する仮想化アプリケーションがエミューションする。
完全仮想化 (ハイパーバイザ型)	KVM	既存の OS を無修正のままゲスト環境として動作させる。CPU 等の仮想化機能を使うことでホスト環境におけるオーバーヘッドを極力減らしている。
準仮想化	Xen	ホスト環境とやりとりする API を利用できるように修正が入った OS をゲスト OS として動作させる方式（＝既存の OS そのままでは動かない）
コンテナ型仮想化	OpenVZ LXC FreeBSD jail docker(筆者加筆)	ホスト環境とゲスト環境は同一カーネルで動作しつつ、ホスト環境から分離したゲスト環境を提供する。

```
$ grep "model name" /proc/cpuinfo | head -n 1
model name      : Intel(R) Core(TM) i5-6500T CPU @ 2.50GHz

$ grep -E 'vmx|svm' /proc/cpuinfo | head -n 1
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi
mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good
nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg
fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm
abm 3dnowprefetch cpuid_fault epb invpcid_single pti tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase
tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec
xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp
```

3.3.2　KVM のカーネルモジュールの読み込み

KVM を利用するには linux カーネルのカーネルモジュールを読み込む必要があります。Intel 製 CPU の場合は "kvm_intel.ko"、AMD 製 CPU の場合は "kvm_amd.ko" という名前になっています。

modprobe コマンドを実行してカーネルモジュールを読み込みます。以下は Intel 製 CPU のコマンド例となります。

```
# modprobe kvm_intel
```

lsmod コマンドで "kvm_intel" または "kvm_amd" モジュールが表示されていれば必要なカーネルモジュールを読み込んでおり、KVM の動作条件を満たしています。

```
# lsmod | grep kvm
kvm_intel             331776  0
kvm                   937984  1 kvm_intel
irqbypass              16384  1 kvm
```

3.3.3　QEMU のインストール

KVM 上で仮想マシンを実行する QEMU をインストールするには以下コマンドを実行します。今回は CLI 操作を目的とするため、GUI 関連のパッケージをインストールしないよう "–no-install-recommends" オプションを指定しています[*4]。

```
# apt install --no-install-recommends qemu-system libvirt-clients libvirt-daemon-system
```

[*4] https://wiki.debian.org/KVM#Installation

KVM 上に仮想マシンをコマンドラインでインストールする virt-install コマンドを含む virtinst パッケージをインストールします。libosinfo-bin パッケージには virt-install コマンドの ”–os-variant” オプションに指定できる値を調べることができる osinfo-query コマンドが入っています[*5]。

```
# apt install virtinst libosinfo-bin
```

上記コマンドをインストールすると libvirt グループが作成されます。以降の手順で仮想マシンを操作するコマンドを紹介しますが、そのとき libvirt グループ に所属しているユーザが仮想マシンを操作できます。自分のユーザを libvirt グループに追加しておきます。

```
# adduser <youruser> libvirt
```

3.3.4 KVM 上で動作する仮想マシンのネットワークの構成

KVM 上の仮想マシンが接続するネットワーク構成は、NAT と Bridge の 2 種類があります。

NAT 構成

NAT 構成は、ホストマシンの中に仮想マシン同士が通信する専用の NAT ネットワークを作る構成をいい、KVM 上の仮想マシン用ネットワークのデフォルト設定になっています。NAT 構成の場合、仮想マシン用ネットワークが KVM のホストマシンの中で完結するため、開発用 PC やネットワークが頻繁に切り替わるノート PC では NAT 構成の利用が向いています。

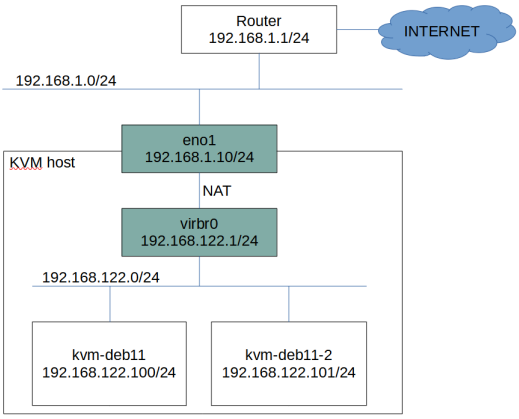


図 1　KVM ホストマシンと仮想マシンのネットワーク構成 (NAT 構成)

以下コマンドを実行すると、仮想マシン向けのネットワークとして virbr0 インタフェースが起動します。デフォルトでは、192.168.122.0/24 のネットワークが作成され、仮想マシンから見たデフォルトゲートウェイは 192.168.122.1/24 となります。

```
$ virsh --connect=qemu:///system net-start default
ネットワーク default が起動されました

$ ip addr show dev virbr0
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:db:b9:5b brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
```

この状態のままでは、PC の起動時に virbr0 インタフェースは自動起動しません。もし、PC の起動時に自動で virbr0 インタフェースを起動するようにしたい場合は以下コマンドを実行してください。

[*5] $ osinfo-query os を実行すると一覧を表示します。

```
$ virsh --connect=qemu:///system net-autostart default
```

Bridge 構成

Bridge 構成は、ホストマシンが接続するネットワークセグメントに仮想マシンも接続する構成のことをいいます。Bridge 構成の場合は仮想マシンの IP アドレスにホストマシンと同じセグメントの IP アドレスを割り当てるため、KVM のホストマシンとは別のホストからでも仮想マシンへ直接通信できるメリットがあります。

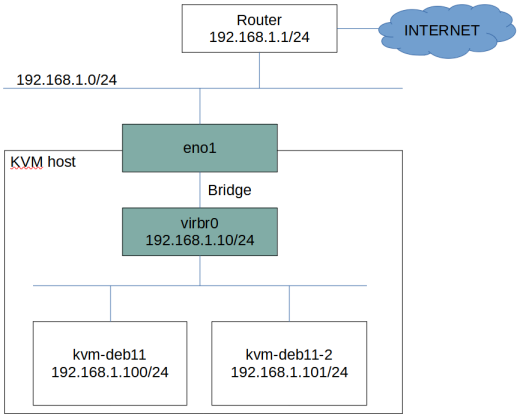


図 2　KVM ホストマシンと仮想マシンのネットワーク構成 (Bridge 構成)

Bridge 構成にするには以下コマンドで追加のパッケージをインストールします。

```
# apt install bridge-utils
```

次に有線 LAN の設定を Bridge 構成に変更します[*6]。
変更前の有線 LAN で固定の IP アドレスに設定している設定ファイルは以下です。

```
$ cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eno1
iface eno1 inet static
  address 192.168.1.10
  netmask 255.255.255.0
  gateway 192.168.1.1
  dns-nameservers 192.168.1.1
  bridge_ports eno1
  bridge_stp off

# This is an autoconfigured IPv6 interface
iface eno1 inet6 auto
```

以下のようにネットワークの設定を変更して Bridge のインタフェースに IP アドレスを設定し、有線 LAN のインタフェースへ Bridge するよう紐つけます。

[*6] ネットワークの設定変更をするため、ssh 経由で操作している場合に設定ミスがあるとネットワークを再起動した後につながらなくなる危険性があります。できるだけホストマシンのコンソールへログインできる状況で作業する方がよいです。

```
$ sudo vi /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
allow-hotplug eno1
iface eno1 inet manual

auto br0
iface br0 inet static
  address 192.168.1.10
  netmask 255.255.255.0
  gateway 192.168.1.1
  dns-nameservers 192.168.1.1
  bridge_ports eno1
  bridge_stp off

# This is an autoconfigured IPv6 interface
iface eno1 inet6 auto
```

設定変更を反映するため、ネットワークを再起動します。

```
$ sudo systemctl restart networking
```

ホストマシンから wget や apt を実行して、ネットワークに接続できることを確認してみてください。

3.4 KVM 上の仮想マシンを操作するコマンド

3.4.1 qemu-img コマンド

qemu-img コマンドは、KVM 上で動作する仮想マシンの仮想ディスクを作成、変更するコマンドです。

仮想ディスクを作成するには ”create” サブコマンドで作成できます。以下の例は、KVM で利用できる qcow2 形式 の 10 GB の仮想ディスクを作成します。qcow2 形式の仮想ディスクは実際にデータを書き込んだ分だけファイルサイズが増えていくため (スパースファイル、またはスパースディスクといいます)、作成直後のファイルサイズはとても小さいです。

```
# qemu-img create -f qcow2 /var/lib/libvirt/images/kvm-deb11.img 10g
Formatting '/var/lib/libvirt/images/kvm-deb11.img', fmt=qcow2 cluster_size=65536 extended_l2=off
compression_type=zlib size=10737418240 lazy_refcounts=off refcount_bits=16

# ls -lh /var/lib/libvirt/images/kvm-deb11.img
-rw-r--r-- 1 root root 193K 10 月 12 23:11 /var/lib/libvirt/images/kvm-deb11.img

# file /var/lib/libvirt/images/kvm-deb11.img
kvm-deb11.img: QEMU QCOW2 Image (v3), 10737418240 bytes
```

qemu-img コマンドのオプションには他の仮想化ソフトが採用している仮想ディスクの形式と相互変換する ”convert” サブコマンドもあります。以下の例は、VirtualBox 向けの VDI 形式から KVM 向けの qcow2 形式へ変換するコマンドです。

```
$ qemu-img convert -f vdi disk-virtualbox.vdi -O qcow2 disk-kvm.img
```

3.4.2 virt-install コマンド

仮想マシンの OS インストールをシリアル接続したコンソールで操作して行う場合

virt-install コマンドを使うと、KVM 上に仮想マシンをインストールすることができます。コマンドライン環境で仮想マシンをインストールする場合は、起動した仮想マシンのコンソールへシリアル接続するようパラメータを指定します (–graphics none、–console オプション、–extra-args オプション)。以下のように virt-install コマンドを実行すると、起動した仮想マシンのコンソールで Debian インストーラの処理が始まり、キーボード入力で仮想マシンへ Debian のインストールを行うことができます。

```
$ sudo virt-install \
--name kvm-deb11 \
--disk path=/var/lib/libvirt/images/kvm-deb11.img,format=qcow2,bus=virtio \
--vcpus 1 \
--ram 1024 \
--os-type linux \
--os-variant generic \
--network bridge=br0,model=virtio \
--location 'http://deb.debian.org/debian/dists/bullseye/main/installer-amd64/' \
--graphics none \
--console pty,target_type=serial \
--extra-args 'console=ttyS0,115200n8 serial'
```

仮想マシンの OS インストールを preseed 機能を使って自動インストールする場合

Debian インストーラには設定ファイルにしたがって OS を自動インストールする preseed[*7] という機能があります。virt-install コマンドのパラメータに preseed の設定ファイルを指定することで仮想マシンへ Debian を自動インストールでき、検証環境の作成や大量に仮想マシンをインストールする場合に便利です。

Debian 11 bullseye 向けの preseed 設定ファイルのサンプルは以下 URL にあります。

- `https://www.debian.org/releases/bullseye/example-preseed.txt`

preseed の設定ファイルを作成し[*8]、/var/lib/libvirt/images/preseed.cfg としてホストマシンに置きます。そして virt-install コマンドのパラメータに "–initrd-inject" を指定すると 仮想マシンの / 直下に "–initrd-inject" で指定したファイルがコピーされます。すると、仮想マシン上で起動した Debian インストーラの処理が /preseed.cfg を検出し、シリアル接続した仮想マシンのコンソール上で Debian のインストールが自動で進んでいきます[*9]。

```
$ sudo qemu-img create -f qcow2 /var/lib/libvirt/images/kvm-deb11-2.img 10g

$ sudo virt-install \
--name kvm-deb11-2 \
--disk path=/var/lib/libvirt/images/kvm-deb11-2.img,format=qcow2,bus=virtio \
--vcpus 1 \
--ram 1024 \
--os-type linux \
--os-variant generic \
--network bridge=br0,model=virtio \
--location 'http://deb.debian.org/debian/dists/bullseye/main/installer-amd64/' \
--initrd-inject=/var/lib/libvirt/images/preseed.cfg \
--graphics none \
--console pty,target_type=serial \
--extra-args 'console=ttyS0,115200n8 serial'
```

3.4.3 virsh コマンド

virsh とは

virsh コマンドは Debian では libvirt-clients パッケージで提供されており、仮想マシンを制御する様々なハイパーバイザーに対応するコマンドラインのフロントエンドツールです。Linux では KVM や Xen のハイパーバイザーの操作に virsh を利用することができます。

virsh は、"virsh { サブコマンド }" のような形のコマンドを実行することで仮想マシンを制御できます。

virsh help

"virsh help" コマンドを実行すると、サブコマンドの一覧が表示されます。

[*7] 付録 B preseed を利用したインストールの自動化 `https://www.debian.org/releases/bullseye/amd64/apb.ja.html`

[*8] 詳しくは、第 144 回東京エリア Debian 勉強会「preseed で Debian を自動インストール」を参照。

[*9] 使った preseed の設定ファイルは `https://github.com/dictoss/utils/blob/master/debian-preseed/preseed-debian11.cfg`

```
$ virsh help
グループ別コマンド:

 Domain Management (ヘルプのキーワード 'domain'):
    attach-device                  XML ファイルによるデバイスの接続
    attach-disk                    ディスクデバイスの接続
    attach-interface               ネットワークインターフェースの接続
    autostart                      ドメインの自動起動
    blkdeviotune                   ブロックデバイスの I/O チューニングパラメーターの設定・取得
    blkiotune                      ブロック I/O パラメーターの取得・設定
    blockcommit                    ブロックのコミット操作の開始
  (以下、省略)
```

また、”virsh help { サブコマンド }” を実行すると、サブコマンドの使い方とオプションを調べることができます。

```
$ virsh help start
  名前
    start - 停止状態の（定義済み）ドメインの起動

  形式
    start <domain> [--console] [--paused] [--autodestroy] [--bypass-cache] [--force-boot] [--pass-fds <string>]
  (以下、省略)
```

virsh –connect

KVM のハイパーバイザーに接続するには ”virsh –connect” を実行します。接続先を示す ”qemu:///system” は URI 形式になっており、接続に利用するプロトコルを切り替えたり、リモート先のホストを指定することができます。

以下の例は、KVM のハイパーバイザーが動作しているホストマシン上からローカル接続する場合のコマンドです。

```
$ virsh --connect qemu:///system
virsh にようこそ、仮想化対話式ターミナルです。

入力: 'help' コマンドのヘルプ
      'quit' 終了

virsh #
```

”virsh –connect” で KVM のハイパーバイザーへ接続すると以降は対話的な操作に移ります。接続中はサブコマンドのみで操作が可能になります。

list サブコマンド

list サブコマンドは、仮想マシンの一覧を出力することができます。オプションの指定値で一覧に表示する仮想マシンが変わります。

単に ”list” と実行した場合は起動中の仮想マシンのみ表示し、”list –inactive” と実行した場合は停止中の仮想マシンのみを表示し、”list –all” と実行した場合はすべての仮想マシンを表示します。

```
virsh # list
 Id   名前         状態
--------------------------
 1    kvm-deb11    実行中
```

```
virsh # list --inactive
 Id   名前               状態
---------------------------------------
 -    kvm-deb11-2        シャットオフ
```

```
virsh # list --all
 Id   名前               状態
---------------------------------------
 1    kvm-deb11          実行中
 -    kvm-deb11-2        シャットオフ
```

virsh edit

edit サブコマンドは、仮想マシンの設定ファイル (XML 形式) を開き、編集することができます。edit サブコマンドを実行すると環境変数 EDITOR で指定しているエディタで XML ファイルを開きます[*10]。

```
virsh # edit kvm-deb11


<domain type='kvm'>
  <name>kvm-deb11</name>
  <uuid>6dec70d3-09e5-40c6-b269-eaf20a2651d9</uuid>
  <memory unit='KiB'>1048576</memory>
  <currentMemory unit='KiB'>1048576</currentMemory>
  <vcpu placement='static'>1</vcpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-5.2'>hvm</type>
    <boot dev='hd'/>
  </os>
  （以降、省略）
```

start サブコマンド

start サブコマンドは、仮想マシンを起動することができます。start サブコマンドを実行すると仮想マシンはバックグラウンドで起動します。

```
virsh # start kvm-deb11
Domain 'kvm-deb11' started
```

"start –console" オプションを指定すると仮想マシンの起動と同時に仮想マシンのコンソールにシリアル接続します。

```
virsh # start --console kvm-deb11
Domain 'kvm-deb11' started
Connected to domain 'kvm-deb11-2'
Escape character is ^] (Ctrl + ])

（以降、コンソールに OS の起動ログが表示される）

kvm-deb11 login:
```

virsh console

console サブコマンドは、起動中の仮想マシンのコンソールへシリアル接続することができます。

```
virsh # console kvm-deb11
Connected to domain 'kvm-deb11'
Escape character is ^] (Ctrl + ])

kvm-deb11 login:
```

virsh destroy

destroy サブコマンドは、仮想マシンを強制的に停止することができます。なお、stop サブコマンドはありません。

```
virsh # destroy kvm-deb11
Domain 'kvm-deb11' destroyed
```

virsh undefine

undefine サブコマンドは、仮想マシンの登録を削除します。undefine サブコマンドにオプションを指定せずに実行した場合は仮想マシンに接続していた仮想ディスクは削除されずに残ります[*11]。

[*10] XML ファイルの定義は `https://libvirt.org/formatdomain.html` を参照。
[*11] 仮想ディスクも一緒に削除したい場合は、–storage {string}、–remove-all-storage などのオプションが利用できます。

```
virsh # undefine kvm-deb11
Domain 'kvm-deb11' has been undefined
```

virsh define

define サブコマンドは、仮想マシンの構成を定義した XML ファイルから仮想マシンを登録できます。

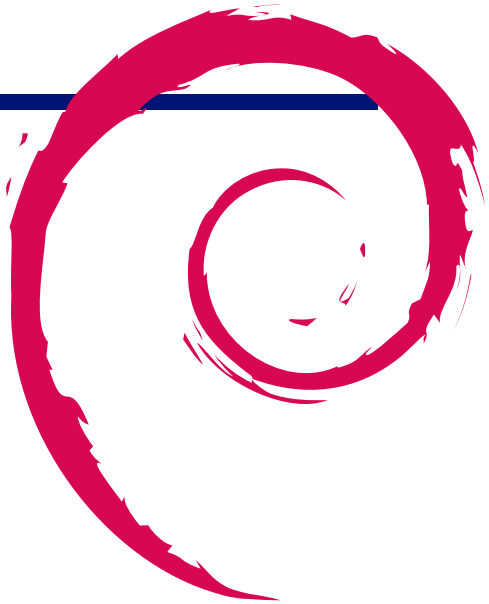
```
virsh # define /etc/libvirt/qemu/kvm-deb11-3.xml
```

3.5 おわりに

Debian 上で KVM を使えるように設定し、virt-install コマンドや virsh コマンドを説明しました。GUI から操作するより CLI から操作した方が便利な場合もありますので、有効に活用してみてください。

3.6 参考文献

- Home - linux-kvm.org `http://www.linux-kvm.org/page/Main_Page`
- KVM - Debian Wiki `https://wiki.debian.org/KVM`
- 第 144 回東京エリア Debian 勉強会「preseed で Debian を自動インストールをしてみよう」 `https://tokyodebian-team.pages.debian.net/pdf2016/debianmeetingresume201610-presentation-sugimoto.pdf`



4 メモ

Debian 勉強会資料

2022 年 10 月 15 日　　初版第 1 刷発行

東京エリア Debian 勉強会 （編集・印刷・発行）