Red Hat Enterprise Linux 6

仮想化管理ガイド

仮想環境の管理

Last Updated: 2022-10-25

Red Hat Enterprise Linux 6 仮想化管理ガイド

仮想環境の管理

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

仮想化管理ガイドでは、ホストの物理マシン、ネットワーク、ストレージ、デバイス、およびゲスト仮想マシンの管理およびトラブルシューティングについて説明します。 注記: 本書は開発中であり、大幅に変更され、プレビューとしてのみ提供されます。含まれる情報および命令は完了とみなされるべきではないため、注意して使用する必要があります。 専門知識を深めるためにも、Red Hat Virtualization (RH318) トレーニングコースも読みください。

目次

第1章 サーバーのベストプラクティス

以下のタスクおよびヒントは、 で Red Hat Enterprise Linux ホストのパフォーマンスを高めるのに役に立ちます。その他のヒントは、『Red Hat Enterprise Linux 仮想化チューニングおよび最適化ガイド』に記載されています。

- SELinux を Enforcing モードで実行します。**setenforce** コマンドを使用して、SELinux が Enforcing モードで実行するように設定します。

  ```
  # setenforce 1
  ```

- **AutoFS**、**NFS**、**FTP**、**HTTP**、**NIS**、**telnetd**、**sendmail** などの不要なサービスを削除または無効にします。

- サーバー上にはプラットフォームの管理に必要な最低限のユーザーアカウントのみを追加します。不要なユーザーアカウントは削除してください。

- ホストでは不必要なアプリケーションは実行しないようにしてください。ホストでアプリケーションを実行すると仮想マシンのパフォーマンスに影響を与えるため、その影響がサーバーの安定性に及ぶ可能性があります。サーバーがクラッシュする可能性のあるアプリケーションでも、サーバー上のすべての仮想マシンがダウンします。

- 仮想マシンのインストールおよびイメージには集中管理できる場所を使用します。仮想マシンのイメージは **/var/lib/libvirt/images/** に格納します。仮想マシンのイメージをこれ以外のディレクトリーに格納する場合は、そのディレクトリーを SELinux ポリシーに追加し、インストールを開始する前にラベルの再設定を必ず行ってください。集中管理ができる共有可能なネットワークストレージの使用を強くお勧めします。

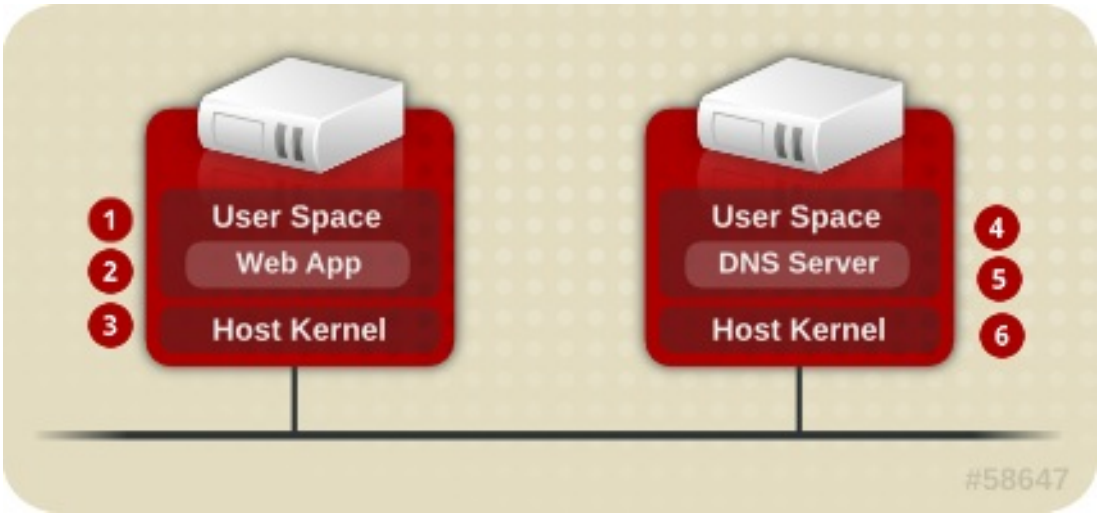
第2章 SVIRT

sVirt は、SELinux と仮想化を統合する Red Hat Enterprise Linux に含まれるテクノロジーです。sVirt は、ゲスト仮想マシンを使用する際のセキュリティーを向上させるために強制アクセス制御 (MAC) を適用します。この統合テクノロジーにより、セキュリティーが向上し、ハイパーバイザーのバグに対してシステムがハードコーディングされます。ホストの物理マシンまたは別のゲスト仮想マシンの攻撃を防ぐことが特に便利です。

本章では、sVirt が Red Hat Enterprise Linux 6 の仮想化テクノロジーと統合する方法を説明します。

非仮想化環境

非仮想化環境では、ホスト物理マシンは互いに物理的に分離されており、各ホスト物理マシンには、Web サーバーや DNS サーバーなどのサービスで設定される自己完結型の環境があります。これらのサービスは、独自のユーザー空間、ホストの物理マシンのカーネルおよび物理ハードウェアに直接通信し、サービスをネットワークに直接提供します。以下のイメージは、仮想化されていない環境を表しています。



??????
ユーザー空間: 全ユーザーモードのアプリケーションと一部のドライバーが実行されるメモリー領域。

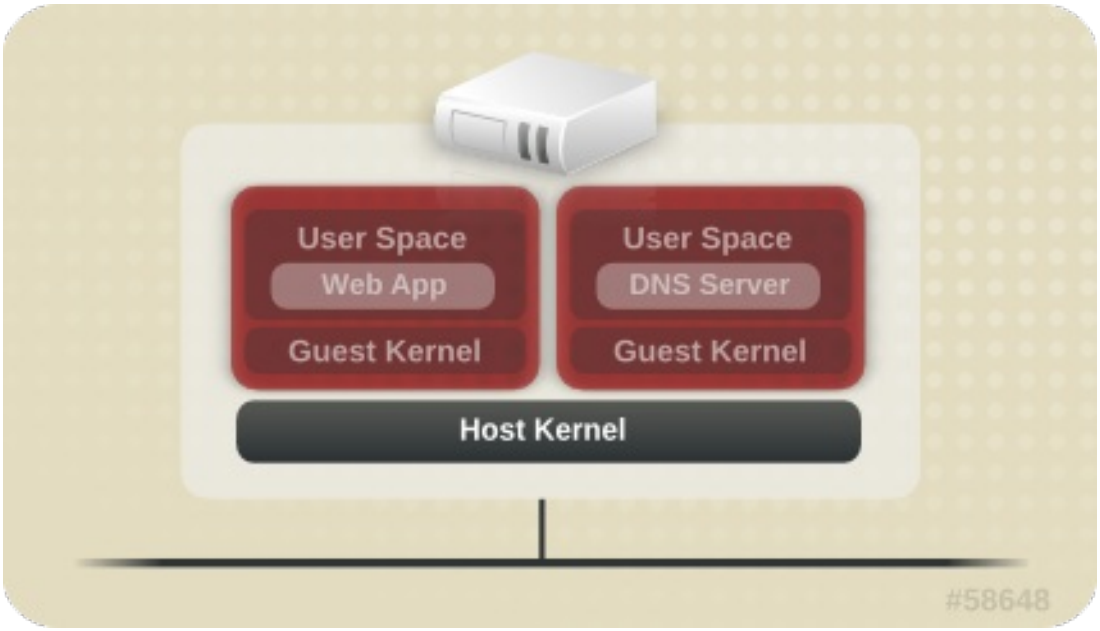
???
Web アプリ (Web アプリケーションサーバー): ブラウザーからアクセスできる Web コンテンツを配信します。

??????
ホストカーネル: ホストの物理マシンの特権付きカーネル、カーネル拡張、およびほとんどのデバイスドライバーを実行するために厳密に予約されます。

???
DNS サーバー: DNS レコードを格納し、ユーザーが IP アドレスの代わりに論理名を使用して Web ページにアクセスできるようにします。

仮想化環境

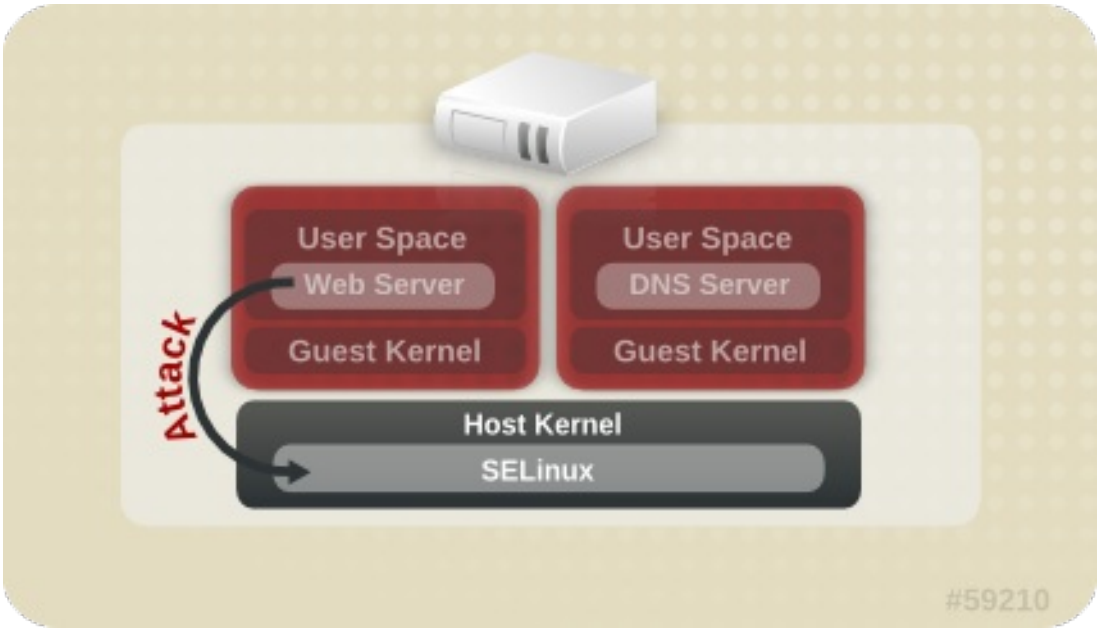
仮想化環境では、複数の仮想オペレーティングシステムを、ホスト物理マシン上にある単一のカーネルで実行できます。以下のイメージは、仮想化環境を表しています。



2.1. セキュリティーおよび仮想化

サービスが仮想化されていない場合は、マシンは物理的に分離されています。通常、エクスプロイトは影響を受けるマシン内に封じ込められ、ネットワーク攻撃は明らかな例外となります。仮想化環境でサービスをグループ化すると、システムに追加の脆弱性が発生します。ハイパーバイザーにゲスト仮想マシンが悪用できるセキュリティー上の欠陥がある場合、このゲスト仮想マシンは、ホスト物理マシンだけでなく、そのホスト物理マシンで実行されている他のゲスト仮想マシンも攻撃できる可能性があります。これらの攻撃はゲスト仮想マシンを超えて拡大する可能性があり、他のゲスト仮想マシンも攻撃にさらされる可能性があります。

sVirt は、ゲスト仮想マシンを分離し、悪用された場合にさらに攻撃を開始する能力を制限するための取り組みです。これは次のイメージに示されています。このイメージでは、攻撃がゲスト仮想マシンから抜け出して他のゲスト仮想マシンに侵入することはできません。



SELinux では、MAC (Mandatory Access Control) の実装で、仮想インスタンスにプラグ可能なセキュリティーフレームワークを導入しました。sVirt フレームワークを使用すると、ゲスト仮想マシンとそのリソースに一意のラベルを付けることができます。ラベルを付けると、異なるゲスト仮想マシン間のアクセスを拒否できるルールを適用できます。

2.2. SVIRT のラベル付け

SELinux の保護下にある他のサービスと同様に、sVirt はプロセスベースのメカニズムと制限を使用して、ゲスト仮想マシンに追加のセキュリティーレイヤーを提供します。通常の使用では、sVirt がバックグラウンドで動作していることに気付くことさえありません。本セクションでは、sVirt のラベリング機能を説明します。

次の出力に示すように、sVirt を使用すると、仮想化された各ゲスト仮想マシンプロセスにラベルが付けられ、動的に生成されたレベルで実行されます。各プロセスは、レベルが異なる他の仮想マシンから分離されています。

```
# ps -eZ | grep qemu

system_u:system_r:svirt_t:s0:c87,c520 27950 ?  00:00:17 qemu-kvm
```

以下の出力に示すように、実際のディスクイメージには、プロセスに一致するように自動的にラベルが付けられます。

```
# ls -lZ /var/lib/libvirt/images/*

 system_u:object_r:svirt_image_t:s0:c87,c520   image1
```

次の表に、sVirt を使用するときに割り当てることができるさまざまなコンテキストラベルの概要を示します。

表2.1 sVirt コンテキストラベル

SELinux コンテキスト	タイプ/説明
system_u:system_r:svirt_t:MCS1	ゲスト仮想マシンのプロセス。MCS1 はランダムな MCS フィールドです。約 500,000 個のラベルがサポートされます。
system_u:object_r:svirt_image_t:MCS1	ゲスト仮想マシンのイメージ。これらのイメージの読み取り/書き込みができるのは、同じ MCS フィールドを持つ **svirt_t** プロセスのみです。
system_u:object_r:svirt_image_t:s0	ゲスト仮想マシンの共有の読み取り/書き込みコンテンツ。すべての **svirt_t** プロセスは svirt_image_t:s0 ファイルに書き込むことができます。

sVirt を使用する場合は、静的ラベリングを実行することもできます。静的ラベルを使用すると、管理者はゲスト仮想マシンの MCS/MLS フィールドなど、特定のラベルを選択できます。静的にラベルを付けた仮想化ゲスト仮想マシンを実行する管理者は、イメージファイルに正しいラベルを設定する必要があります。ゲスト仮想マシンは常にそのラベルで起動し、sVirt システムは静的にラベル付けされた仮想マシンの内容のラベルを変更しません。これにより、MLS 環境で sVirt コンポーネントを実行できます。また、要件に応じて、システムで異なる機密レベルを持つゲスト仮想マシンを複数実行することもできます。

第3章 仮想マシンのクローン作成

ゲストコピーの作成に使用されるゲスト仮想マシンインスタンスには、以下の 2 つのタイプがあります。

- **クローン** は、1 台の仮想マシンのインスタンスです。クローンを使用すると、同じ仮想マシンのネットワークを設定したり、別の宛先に配布したりできます。
- **テンプレート** は、クローン作成のソースとして使用するように設計された仮想マシンのインスタンスです。テンプレートから複数のクローンを作成し、各クローンにマイナーな変更を加えることができます。これは、この変更がシステムに与える影響を確認する際に役立ちます。

クローンとテンプレートはどちらも仮想マシンインスタンスです。これらの違いは、使用方法にあります。

作成されたクローンが正しく機能するには、通常、クローンを作成する前に、クローンを作成する仮想マシンに固有の情報と設定を削除する必要があります。削除する情報は、クローンの使用方法によって異なります。

削除する情報および設定は、次のいずれかのレベルになります。

- **プラットフォームレベル**の情報および設定には、仮想化ソリューションが仮想マシンに割り当てたものが含まれます。例には、ネットワークインターフェイスカード (NIC) の数と、その MAC アドレスが含まれます。
- **ゲストオペレーティングシステムレベル**情報および設定には、仮想マシン内で設定されたものが含まれます。例には SSH 鍵が含まれます。
- **アプリケーションレベル**情報および設定には、仮想マシンにインストールされているアプリケーションで設定したものが含まれます。例には、アクティベーションコードおよび登録情報が含まれます。

注記

情報およびアプローチは各アプリケーションに固有のものであるため、本章には、アプリケーションレベルの削除に関する情報は記載されていません。

その結果、仮想マシン内の情報および設定の一部を削除する必要がありますが、その他の情報および設定は、仮想化環境 (Virtual Machine Manager や VMware など) を使用して仮想マシンから削除する必要があります。

3.1. クローンを作成する仮想マシンの準備

仮想マシンのクローンを作成する前に、ディスクイメージで virt-sysprep ユーティリティーを実行するか、次の手順に従って仮想マシンを準備する必要があります。

手順3.1 クローンを作成する仮想マシンの準備

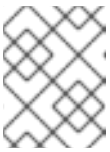
1. **仮想マシンのセットアップ**
   a. クローンまたはテンプレートに使用する仮想マシンを構築します。
      - クローンに必要なソフトウェアをインストールします。
      - オペレーティングシステムに一意でない設定を設定します。

- 固有でないアプリケーション設定を設定します。

2. **ネットワーク設定を削除します。**

a. 以下のコマンドを使用して、永続的な udev ルールを削除します。

```
# rm -f /etc/udev/rules.d/70-persistent-net.rules
```

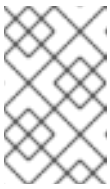


注記

udev ルールを削除しない場合は、最初の NIC の名前が eth0 ではなく eth1 になります。

b. **/etc/sysconfig/network-scripts/ifcfg-eth[x]** で以下の編集を行い、ifcfg スクリプトから一意のネットワークの詳細を削除します。

i. HWADDR 行および Static 行を削除します。



注記

HWADDR が新しいゲストの MAC アドレスと一致しない場合、ifcfg は無視されます。したがって、ファイルから HWADDR を削除することが重要です。

```
DEVICE=eth[x]
BOOTPROTO=none
ONBOOT=yes
#NETWORK=10.0.1.0       <- REMOVE
#NETMASK=255.255.255.0  <- REMOVE
#IPADDR=10.0.1.20       <- REMOVE
#HWADDR=xx:xx:xx:xx:xx  <- REMOVE
#USERCTL=no             <- REMOVE
# Remove any other *unique* or non-desired settings, such as UUID.
```

ii. HWADDR または一意の情報が含まれていない DHCP 設定が残っていることを確認します。

```
DEVICE=eth[x]
BOOTPROTO=dhcp
ONBOOT=yes
```

iii. ファイルに以下の行が含まれていることを確認します。

```
DEVICE=eth[x]
ONBOOT=yes
```

c. 以下のファイルが存在する場合は、そのファイルに同じ内容が含まれていることを確認してください。

- **/etc/sysconfig/networking/devices/ifcfg-eth[x]**
- **/etc/sysconfig/networking/profiles/default/ifcfg-eth[x]**

注記

NetworkManager または特殊な設定を仮想マシンで使用した場合は、追加の固有情報が ifcfg スクリプトから削除されていることを確認してください。

3. **登録の詳細を削除します。**

   a. 以下のいずれかを使用して、登録の詳細を削除します。

   - Red Hat Network (RHN) 登録済みゲスト仮想マシンの場合は、以下のコマンドを実行します。

```
# rm /etc/sysconfig/rhn/systemid
```

   - Red Hat Subscription Manager (RHSM) の登録済みゲスト仮想マシンの場合は、次のコマンドを使用します。

     - 元の仮想マシンを使用しない場合は、以下のコマンドを実行します。

```
# subscription-manager unsubscribe --all
# subscription-manager unregister
# subscription-manager clean
```

     - 元の仮想マシンを使用する場合は、次のコマンドのみを実行します。

```
# subscription-manager clean
```

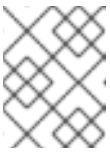
注記

元の RHSM プロファイルはポータルに残ります。

4. **その他の固有の詳細の削除**

   a. 次のコマンドを使用して、sshd の公開鍵と秘密鍵のペアを削除します。

```
# rm -rf /etc/ssh/ssh_host_*
```



注記

ssh キーを削除すると、このホストを信頼しない ssh クライアントの問題が回避されます。

   b. 複数のマシンで実行している場合に、競合する可能性があるその他のアプリケーション固有の識別子や設定を削除します。

5. **次回のシステムの起動時に設定ウィザードを実行するように仮想マシンを設定します。**

   a. 以下のいずれかの方法で、仮想マシンを次回起動したときに、関連する設定ウィザードが実行されるように設定します。

   - Red Hat Enterprise Linux 6 以前の場合は、以下のコマンドを使用して、root ファイルシステムに .unconfigured という名前の空のファイルを作成します。

     ■

```
# touch /.unconfigured
```

- Red Hat Enterprise Linux 7 の場合は、次のコマンドを実行して、最初の起動ウィザードおよび初期設定ウィザードを有効にします。

```
# sed -ie 's/RUN_FIRSTBOOT=NO/RUN_FIRSTBOOT=YES/'
/etc/sysconfig/firstboot
# systemctl enable firstboot-graphical
# systemctl enable initial-setup-graphical
```

注記

次回の起動時に実行するウィザードは、仮想マシンから削除された設定によって異なります。また、クローンの初回ブートでは、ホスト名を変更することが推奨されます。

3.2. 仮想マシンのクローン作成

クローンの作成を続行する前に、仮想マシンをシャットダウンします。**virt-clone** または **virt-manager** を使用して、仮想マシンのクローンを作成できます。

3.2.1. virt-clone を使用したゲストのクローン作成

virt-clone を使用すると、コマンドラインから仮想マシンのクローンを作成できます。

virt-clone を正常に完了するには、root 権限が必要であることに注意してください。

virt-clone コマンドには、コマンドラインで渡すことができるオプションが多数用意されています。これには、一般的なオプション、ストレージ設定オプション、ネットワーク設定オプション、およびその他のオプションが含まれます。**--orginal** のみが必要です。オプションの一覧を表示するには、次のコマンドを実行します。

```
# virt-clone --help
```

また、**virt-clone** の man ページでは、各コマンドオプション、重要な変数、および例が記載されています。

以下の例は、デフォルト接続で、demo と呼ばれるゲスト仮想マシンのクローンを作成する方法を示しています。これにより、新しい名前とディスククローンパスが自動的に生成されます。

例3.1 virt-clone を使用したゲストのクローン作成

```
# virt-clone --original demo --auto-clone
```

以下の例は、demo と呼ばれる QEMU ゲスト仮想マシンのクローンを、複数のディスクで作成する方法を示しています。

例3.2 virt-clone を使用したゲストのクローン作成

```
# virt-clone --connect qemu:///system --original demo --name newdemo --file
/var/lib/xen/images/newdemo.img --file /var/lib/xen/images/newdata.img
```

3.2.2. virt-manager を使用したゲストのクローン作成

この手順では、**virt-manager** ユーティリティーを使用して、ゲスト仮想マシンのクローンを作成する方法を説明します。

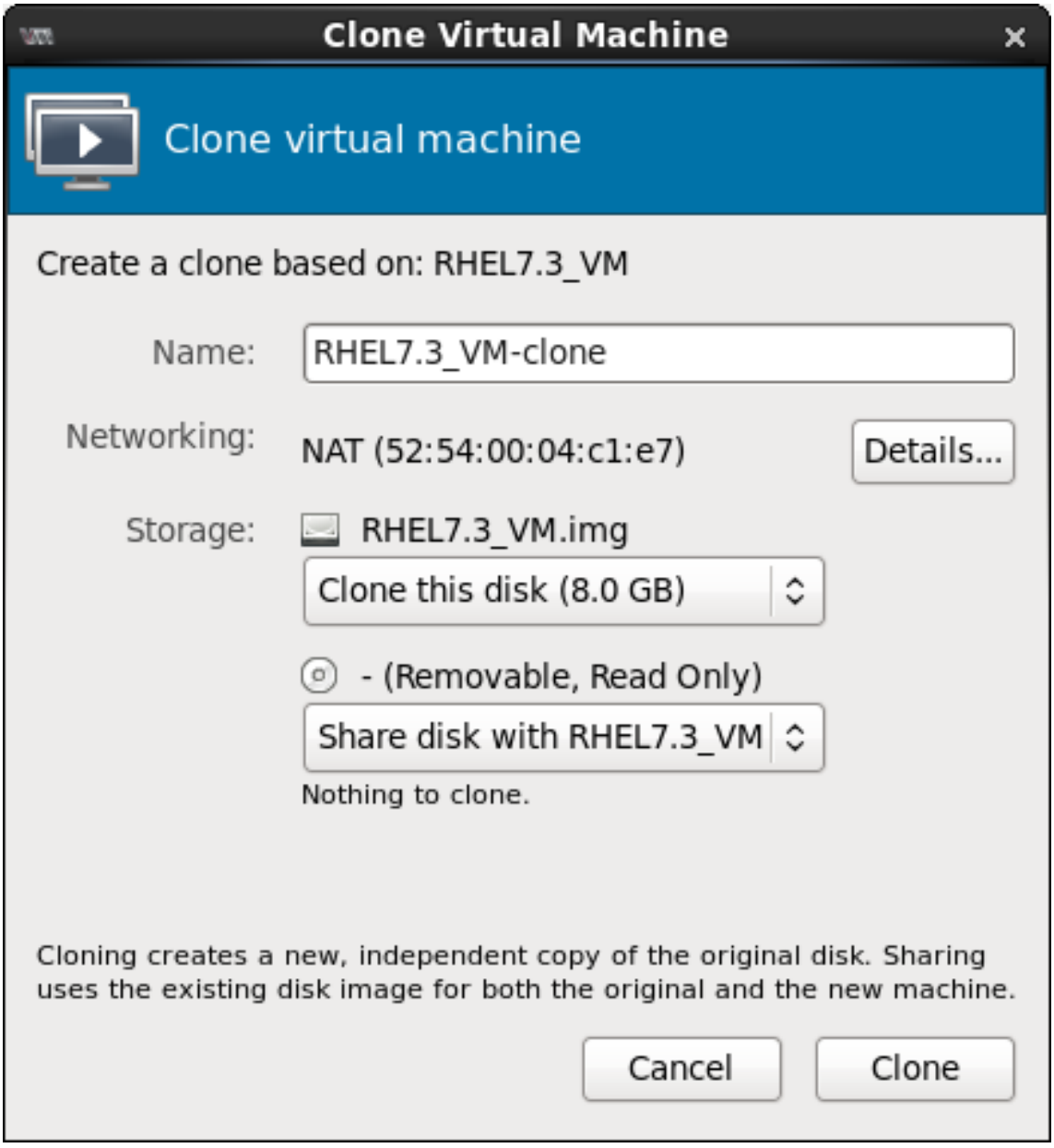
手順3.2 virt-manager を使用した仮想マシンのクローンの作成

1. **virt-manager を開く**
   virt-manager を起動します。**Applications** メニューおよび **System Tools** サブメニューから **Virtual Machine Manager** アプリケーションを起動します。または、root で **virt-manager** を実行します。

   Virtual Machine Manager にあるゲスト仮想マシンの一覧から、クローンを作成するゲスト仮想マシンを選択します。

   クローンを作成するゲスト仮想マシンを右クリックし、**Clone** を選択します。Clone Virtual Machine ウィンドウが開きます。

   図3.1 Clone Virtual Machine ウィンドウ



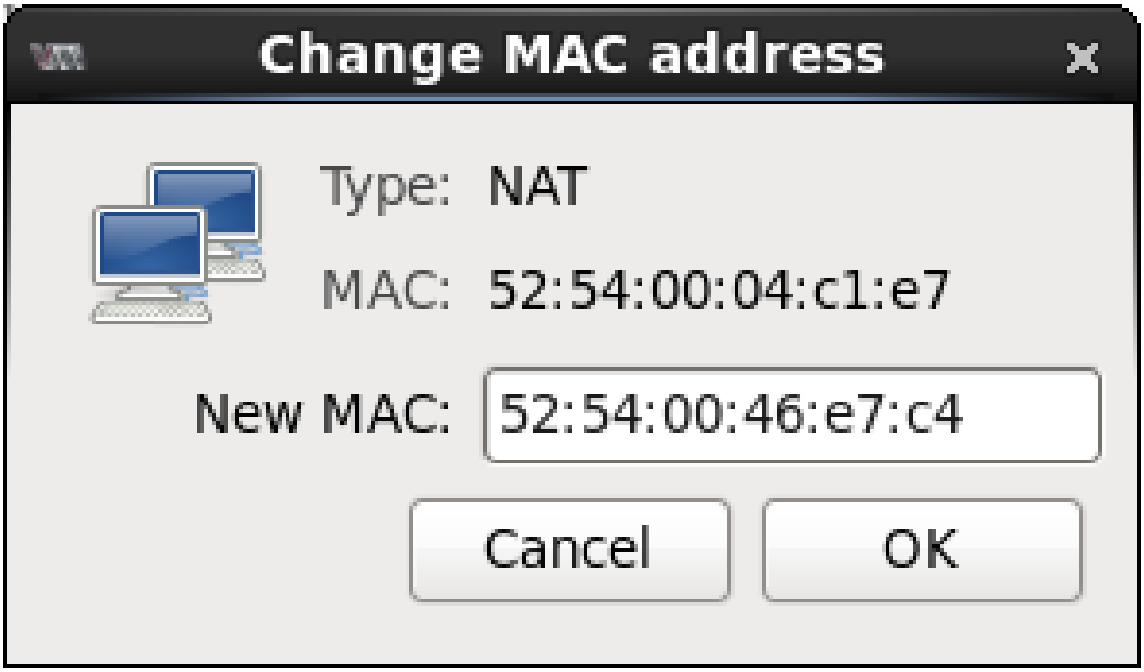
2. **クローンの設定**

   - クローンの名前を変更する場合は、クローンの新しい名前を入力します。
   - ネットワーク設定を変更する場合は、**Details** を選択します。

     クローンの新しい MAC アドレスを入力します。

     OK をクリックします。

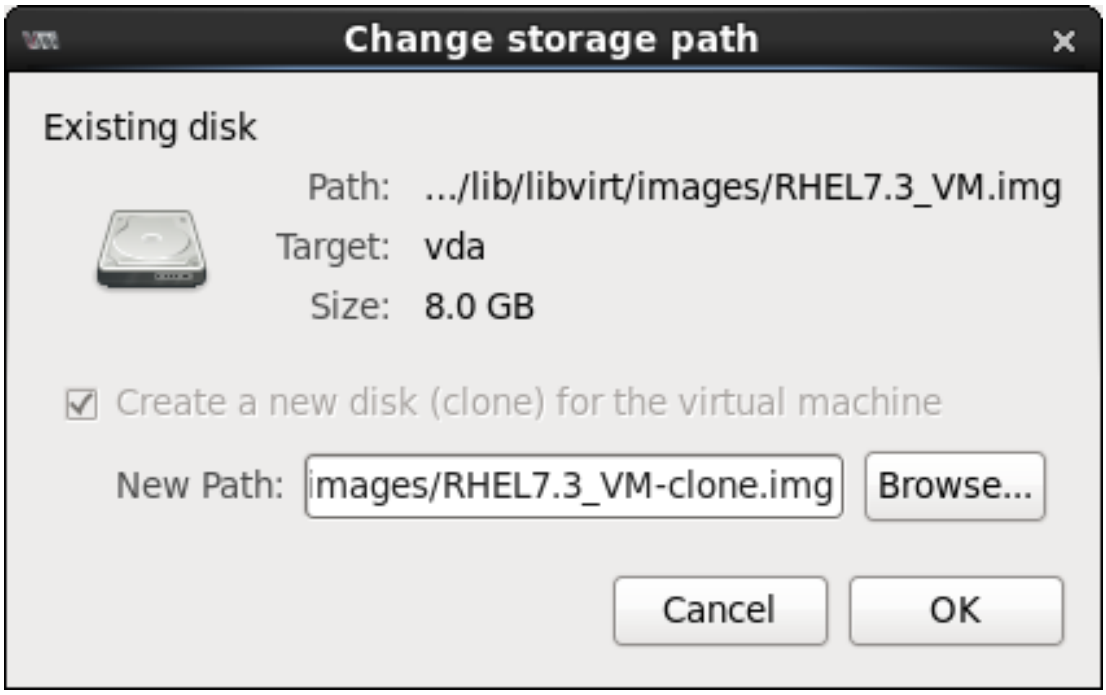
     図3.2 Change MAC Address ウィンドウ



   - クローンを作成したゲスト仮想マシンのディスクごとに、次のいずれかのオプションを選択します。

     - **Clone this disk** - ディスクは、クローンとして作成されたゲスト仮想マシンのクローンとして作成されます。
     - **Share disk with *guest virtual machine name*** - ディスクは、クローンを作成されるゲスト仮想マシンとそのクローンで共有されます。
     - **Details** - ストレージパスの変更 画面を開きます。これにより、ディスクへの新しいパスの選択が可能になります。

図3.3 Change **storage path** ウィンドウ



3. **ゲスト仮想マシンのクローンを作成する**
   Clone をクリックします。

第4章 KVM のライブマイグレーション

本章では、あるホストの物理マシンで実行中のゲスト仮想マシンを別のホスト物理マシンに移行する方法について説明します。いずれのインスタンスでも、ホストの物理マシンが KVM ハイパーバイザーを実行しています。

移行では、ゲスト仮想マシンをあるホストの物理マシンから別のホストの物理マシンに移動するプロセスを説明します。これは、ゲスト仮想マシンがハードウェア上で直接ではなく仮想化環境で実行しているため可能です。移行は、以下の場合に役立ちます。

- 負荷分散: ゲスト仮想マシンは、ホスト物理マシンが過負荷になった場合、または別のホスト物理マシンが十分に活用されていない場合に、使用率の低いホスト物理マシンに移動できます。
- ハードウェア独立性: ホストの物理マシンでハードウェアデバイスをアップグレード、追加、または削除する必要がある場合、ゲスト仮想マシンを他のホストの物理マシンに安全に移動できます。つまり、ゲスト仮想マシンでは、ハードウェアの改善のためのダウンタイムが発生しません。
- 省エネ: ゲスト仮想マシンを他のホスト物理マシンに再配布できるため、電源をオフにして、低使用期間でエネルギーを節約し、コストを削減できます。
- 地理的な移行: 待ち時間を短縮するため、または深刻な状況では、ゲスト仮想マシンを別の場所に移動できます。

移行は、ゲスト仮想マシンのメモリーと仮想デバイスの状態を移行先ホストの物理マシンに送信することで機能します。共有されたネットワークストレージを使用して、ゲスト仮想マシンのイメージを移行することが推奨されます。仮想マシンを移行するときは、共有ストレージに libvirt が管理するストレージプールを使用することもお勧めします。

移行はライブで実行でき、実行できません。

ライブマイグレーションでは、ゲスト仮想マシンはソースホスト物理マシン上で実行を継続し、そのメモリーページは宛先ホスト物理マシンに順番に転送されます。移行中、KVM は、すでに転送されたページの変更についてソースを監視し、すべての初期ページが転送されたときにこれらの変更の転送を開始します。KVM は、移行中の転送速度も推定するため、転送するデータの残りの量が一定の設定可能な期間 (デフォルトでは 10 ミリ秒) に達すると、KVM が元のゲスト仮想マシンを一時停止し、残りのデータを転送して、移行先ホストの物理マシンで同じゲスト仮想マシンを再開します。

ライブで実行されない移行は、ゲスト仮想マシンを一時停止してから、ゲスト仮想マシンのメモリーのイメージを移行先のホスト物理マシンに移動します。次に、ゲスト仮想マシンが宛先ホスト物理マシンで再開され、ソースホスト物理マシンで使用されていたゲスト仮想マシンのメモリーが解放されます。このような移行を完了するのにかかる時間は、ネットワークの帯域幅と遅延によって異なります。ネットワークが頻繁に使用されているか、帯域幅が狭い場合、移行にははるかに長い時間がかかります。

元のゲスト仮想マシンがページを KVM が宛先ホストの物理マシンに転送するよりも速く変更する場合は、ライブ移行が完了しないため、オフライン移行を使用する必要があります。

4.1. ライブマイグレーションの要件

ゲスト仮想マシンを移行するには、以下が必要です。

移行の要件

- 次のいずれかのプロトコルを使用して、共有ストレージにインストールされたゲスト仮想マシン。

- ファイバーチャネルベースの LUN
- iSCSI
- FCoE
- NFS
- GFS2
- SCSI RDMA プロトコル (SCSI RCP) - Infiniband アダプターおよび 10GbE iWARP アダプターで使用されるブロックエクスポートプロトコル

- 移行プラットフォームおよびバージョンは、テーブル 表4.1「ライブマイグレーションの互換性」 に対して確認する必要があります。また、Red Hat Enterprise Linux 6 は、共有ストレージ上の raw イメージと qcow2 イメージを使用したゲスト仮想マシンのライブマイグレーションをサポートしていることにも注意してください。

- 両方のシステムで、適切な TCP/IP ポートが開いている必要があります。ファイアウォールが使用されている場合は、詳細なポート情報の https://access.redhat.com/site/documentation/ で見つかる 『Red Hat Enterprise Linux Virtualization セキュリティーガイド』 を参照してください。

- 共有ストレージメディアをエクスポートする別のシステム。ストレージは、移行に使用される 2 つのホストマシンのいずれかに配置しないでください。

- 共有ストレージは、移行元システムと移行先システムの同じ場所にマウントする必要があります。マウントするディレクトリー名は同じである必要があります。別のパスを使用してイメージを保持することもできますが、推奨されません。virt-manager を使用して移行を行う場合は、パス名が同じである必要があることに注意してください。ただし、virsh を使用して移行を実行する場合は、移行を実行するときに **--xml** オプションまたはプリフックを使用して、さまざまなネットワーク設定とマウントディレクトリーを使用できます。共有ストレージがなくても、オプション **--copy-storage-all** (非推奨) を使用して移行を成功させることができます。**prehooks** の詳細は、 libvirt.org を参照してください。XML オプションの詳細については、20章 *ドメイン XML の操作* を参照してください。

- パブリックブリッジ + タップネットワーク内の既存のゲスト仮想マシンで移行を試みる場合、移行元と移行先のホスト物理マシンは同じネットワークに配置する必要があります。この手順を行わないと、ゲストの仮想マシンネットワークが移行後に動作しません。

- Red Hat Enterprise Linux 5 および 6 では、KVM ゲスト仮想マシンのデフォルトのキャッシュモードが ***none*** に設定されているため、ディスクの状態に一貫性がありません。キャッシュオプションを ***none*** (たとえば、**virsh attach-disk cache none** を使用) に設定すると、***O_DIRECT*** フラグを使用してゲスト仮想マシンのファイルが開かれます (**オープン** システムコールを呼び出す場合)。つまり、ホストの物理マシンのキャッシュを回避し、ゲスト仮想マシンでキャッシュのみを提供します。キャッシュモードを ***none*** に設定すると、不整合の問題が回避され、使用すると、仮想マシンのライブマイグレーションが可能になります。キャッシュを ***none*** に設定する方法は、 「ゲストへのストレージデバイスの追加」 を参照してください。

libvirtd サービスが有効になっていて (**#chkconfig libvirtd on**)、実行されている (**#service libvirtd start**) ことを確認してください。また、効果的に移行する機能は、**/etc/libvirt/libvirtd.conf** 設定ファイルのパラメーターの設定に依存することに注意してください。

手順4.1 libvirtd.conf の設定

1. **libvirtd.conf** を開くには、root で次のコマンドを実行する必要があります。

```
# vim /etc/libvirt/libvirtd.conf
```

2. 必要に応じてパラメーターを変更し、ファイルを保存します。
3. **libvirtd** サービスを再起動します。

```
# service libvirtd restart
```

4.2. ライブマイグレーションと RED HAT ENTERPRISE LINUX バージョンの互換性

ライブマイグレーションは、表 表4.1「ライブマイグレーションの互換性」 のようにサポートされます。

表4.1 ライブマイグレーションの互換性

移行の方法	リリースタイプ	例	ライブ移行のサポート	注記
前方	メジャーリリース	5.x → 6.y	サポート対象外	
前方	マイナーリリース	5.x → 5.y (y>x, x>=4)	完全対応	問題がある場合は報告する必要があります
前方	マイナーリリース	6.x → 6.y (y>x, x>=0)	完全対応	問題がある場合は報告する必要があります
後方	メジャーリリース	6.x → 5.y	サポート対象外	
後方	マイナーリリース	5.x → 5.y (x>y,y>=4)	サポート対象	既知の問題は、移行に関する問題のトラブルシューティング を参照してください。
後方	マイナーリリース	6.x → 6.y (x>y, y>=0)	サポート対象	既知の問題は、移行に関する問題のトラブルシューティング を参照してください。

移行に関する問題のトラブルシューティング

- **SPICE の問題**: Red Hat Enterprise Linux 6.0→6.1 からの移行時に、SPICE に互換性のない変更があることが判明しました。このような場合、クライアントは切断してから再接続し、オーディオとビデオが一時的に失われる可能性があります。これは一時的なものであり、すべてのサービスが再開されます。
- **USB の問題**: Red Hat Enterprise Linux 6.2 は、移行サポートを含む USB 機能を追加しました

が、USB デバイスをリセットし、デバイス上で実行されているアプリケーションを中止させる特定の警告がないわけではありません。この問題は Red Hat Enterprise Linux 6.4 で修正され、今後のバージョンでは発生しません。6.4 よりも前のバージョンでこれが発生するのを防ぐには、USB デバイスが使用されている間は移行を行いません。

- **移行プロトコルの問題** - 後方移行が "unknown section error" で終了する場合は、一時的なエラーである可能性があるため、移行プロセスを繰り返すことで問題を修復できます。そうでない場合は、問題を報告してください。

ネットワークストレージの設定

共有ストレージを設定し、共有ストレージにゲスト仮想マシンをインストールします。

あるいは、「共有ストレージの例: 単純な移行のための NFS」 の NFS の例を使用します。

4.3. 共有ストレージの例: 単純な移行のための NFS

重要

この例では、NFS を使用して、ゲスト仮想マシンのイメージを別の KVM ホストの物理マシンと共有します。大規模なインストールでは実用的ではありませんが、移行技術のみを実証することが推奨されます。この例を、複数のゲスト仮想マシンの移行または実行に使用しないでください。また、***sync*** パラメーターが有効になっている必要もあります。これは、NFS ストレージを適切にエクスポートするために必要です。さらに、NFS をソースホスト物理マシンにマウントすることを強くお勧めします。ゲスト仮想マシンのイメージは、ソースホスト物理マシンにある NFS マウントディレクトリーに作成する必要があります。また、NFS ファイルロックは KVM でサポートされていないため、**使用しないで**ください。

大規模なデプロイメントでは、iSCSI ストレージの方が適しています。設定の詳細は、「iSCSI ベースのストレージプール」 を参照してください。

また、このセクションに記載されている手順は、『Red Hat ストレージ管理ガイド』 に記載されている詳細な手順に代わるものではないことに注意してください。NFS の設定、IP テーブルのオープン、およびファイアウォールの設定については、このガイドを参照してください。

1. **ディスクイメージ用のディレクトリーを作成します**
   この共有ディレクトリーには、ゲスト仮想マシンのディスクイメージが含まれます。これを行うには、**/var/lib/libvirt/images** とは異なる場所にディレクトリーを作成します。以下に例を示します。

   ```
   # mkdir /var/lib/libvirt-img/images
   ```

2. **NFS 設定ファイルに新しいディレクトリーパスを追加します。**
   NFS 設定ファイルは、**/etc/exports** にあるテキストファイルです。ファイルを開き、手順 1 で作成した新しいファイルへのパスを追加して編集します。

   ```
   # echo "/var/lib/libvirt-img/images" >> /etc/exports/[NFS-Config-FILENAME.txt]
   ```

3. **NFS の起動**

   a. **iptable** (2049 など) の NFS のポートが開いていることを確認し、 **/etc/hosts.allow** ファイルに NFS を追加します。

b. NFS サービスを開始します。

```
# service nfs start
```

4. **ソースと宛先の両方に共有ストレージをマウントします**
次のコマンドを 2 回実行して、ソースシステムと宛先システムの両方に **/var/lib/libvirt/images** ディレクトリーをマウントします。ソースシステムで 1 回、宛先システムでもう一度。

```
# mount source_host:/var/lib/libvirt-img/images /var/lib/libvirt/images
```

警告

この手順で作成するディレクトリーが、「ライブマイグレーションの要件」 で概説されている要件に準拠していることを確認してください。さらに、ディレクトリーに正しい SELinux ラベルを付ける必要がある場合があります。詳細は、Red Hat Enterprise Linux ストレージ管理ガイド の NFS の章を参照してください。

4.4. VIRSH を使用した KVM のライブ移行

virsh コマンドを使用すると、ゲスト仮想マシンを別のホスト物理マシンに移行できます。**migrate** コマンドは、以下の形式のパラメーターを受け付けます。

```
# virsh migrate --live GuestName DestinationURL
```

ライブマイグレーションが望ましくない場合は、**-live** オプションが削除される可能性があることに注意してください。追加オプションは、「virsh migrate コマンドの追加オプション」 に一覧表示されています。

GuestName パラメーターは、移行するゲスト仮想マシンの名前を表します。

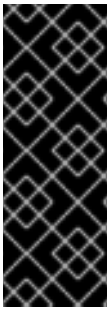
DestinationURL パラメーターは、移行先ホストの物理マシンの接続 URL です。移行先システムで同じバージョンの Red Hat Enterprise Linux を実行し、同じハイパーバイザーを使用し、**libvirt** を実行している必要があります。

注記

通常の移行およびピアツーピア移行の ***DestinationURL*** パラメーターには、以下のセマンティクスがあります。

- 通常の移行: ***DestinationURL*** は、ソースゲスト仮想マシンから見たターゲットホスト物理マシンの URL です。
- ピアツーピアの移行: ***DestinationURL*** は、移行元ホストの物理マシンから表示されるターゲットホストの物理マシンの URL です。

コマンドを入力すると、インストール先システムの root パスワードを求められます。



重要

移行を成功させるには、移行元サーバーの **/etc/hosts** ファイルに移行先ホストの物理マシンのエントリーが必要です。次の例に示すように、宛先ホストの物理マシンの IP アドレスとホスト名をこのファイルに入力し、宛先ホストの物理マシンの IP アドレスとホスト名を置き換えます。

```
10.0.0.20 host2.example.com
```

例:virsh を使用したライブマイグレーション

この例では、**host1.example.com** から **host2.example.com** に移行します。使用環境に合わせて、ホストの物理マシン名を変更します。この例では、**guest1-rhel6-64** という名前の仮想マシンを移行します。

この例では、共有ストレージを完全に設定し、すべての前提条件 (ここでは 移行の要件) を満たしていることを前提としています。

1. **ゲスト仮想マシンが実行していることを確認します。**
   移行元システムで、**host1.example.com** を実行し、**guest1-rhel6-64** が実行していることを確認します。

   ```
   [root@host1 ~]# virsh list
   Id Name                 State
   ----------------------------------
    10 guest1-rhel6-64     running
   ```

2. **ゲスト仮想マシンの移行**
   次のコマンドを実行して、ゲスト仮想マシンを移行先 **host2.example.com** にライブ移行します。リンク先の URL の末尾に **/system** を追加し、フルアクセスが必要であることを libvirt に指示します。

   ```
   # virsh migrate --live guest1-rhel6-64 qemu+ssh://host2.example.com/system
   ```

   コマンドを入力すると、インストール先システムの root パスワードを求められます。

3. **待機**
   負荷やゲスト仮想マシンのサイズによっては、移行に時間がかかる場合があります。**virsh** はエラーのみを報告します。ゲスト仮想マシンは、完全に移行するまで、移行元ホストの物理マシンで実行し続けます。

   注記

   移行中、完了率インジケーターの数は、プロセスが完了する前に複数回減少する可能性があります。これは、移行の開始後に変更されたソースメモリーページを再度コピーする必要があるため、全体的な進行状況の再計算が原因で発生します。したがって、この動作は予期されたものであり、移行に問題があることを示すものではありません。

4. **ゲスト仮想マシンが移行先ホストに到達していることを確認する**
   宛先システム **host2.example.com** から、**guest1-rhel6-64** が実行されていることを確認します。

   ```
   [root@host2 ~]# virsh list
   ```

```
 Id Name                 State
----------------------------------
 10 guest1-rhel6-64      running
```

これで、ライブマイグレーションが完了しました。

注記

libvirt は、TLS/SSL ソケット、UNIX ソケット、SSH、暗号化されていない TCP など、さまざまなネットワーク方式をサポートしています。他の方法の使用に関する詳細は、5 章 *ゲストのリモート管理* を参照してください。

注記

実行されていないゲスト仮想マシンは、**virshmigrate** コマンドを使用して移行できません。実行していないゲスト仮想マシンを移行するには、以下のスクリプトを使用する必要があります。

```
virsh dumpxml Guest1 > Guest1.xml
virsh -c qemu+ssh://<target-system-FQDN>  define Guest1.xml
virsh undefine Guest1
```

4.4.1. virsh を使用した移行に関する追加のヒント

複数の同時ライブマイグレーションを実行できます。各移行は、個別のコマンドシェルで実行されます。ただし、これは慎重に行ってください。各移行インスタンスでは、双方 (ソースおよびターゲット) から 1 つの MAX_CLIENT を使用するため、注意して計算を行う必要があります。デフォルト設定は 20 であるため、設定を変更せずに 10 インスタンスを実行できます。設定を変更する必要がある場合は、手順 手順4.1「libvirtd.conf の設定」 を参照してください。

1. 手順4.1「libvirtd.conf の設定」 の説明に従って、libvirtd.conf ファイルを開きます。
2. Processing controls のセクションを探します。

```
#################################################################
#
# Processing controls
#

# The maximum number of concurrent client connections to allow
# over all sockets combined.
#max_clients = 20


# The minimum limit sets the number of workers to start up
# initially. If the number of active clients exceeds this,
# then more threads are spawned, upto max_workers limit.
# Typically you'd want max_workers to equal maximum number
# of clients allowed
#min_workers = 5
#max_workers = 20


# The number of priority workers. If all workers from above
```

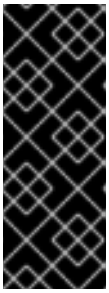
```
# pool will stuck, some calls marked as high priority
# (notably domainDestroy) can be executed in this pool.
#prio_workers = 5

# Total global limit on concurrent RPC calls. Should be
# at least as large as max_workers. Beyond this, RPC requests
# will be read into memory and queued. This directly impact
# memory usage, currently each request requires 256 KB of
# memory. So by default upto 5 MB of memory is used
#
# XXX this isn't actually enforced yet, only the per-client
# limit is used so far
#max_requests = 20

# Limit on concurrent requests from a single client
# connection. To avoid one client monopolizing the server
# this should be a small fraction of the global max_requests
# and max_workers parameter
#max_client_requests = 5

#################################################################
```

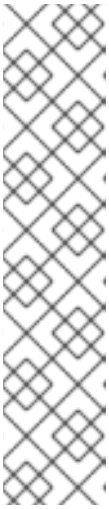
3. ***max_clients*** および ***max_workers*** パラメーターの設定を変更します。両方のパラメーターの番号が同じであることが推奨されます。***max_clients*** は、移行ごとに 2 つのクライアント (各側に 1 つ) を使用します。***max_workers*** は、実行フェーズ中に移行元で 1 つのワーカーと、移行先で 0 のワーカーを使用し、終了フェーズ中に移行先でワーカーを 1 つ使用します。



重要

max_clients パラメーターおよび ***max_workers*** パラメーターの設定は、libvirtd サービスへのゲスト仮想マシンのすべての接続の影響を受けます。つまり、同じゲスト仮想マシンを使用しているユーザーで、同時に移行を実行しているすべてのユーザーは、***max_clients*** および ***max_workers*** パラメーター設定で設定される制限にも古いことになります。このため、同時ライブマイグレーションを実行する前に、最大値を慎重に検討する必要があります。

4. ファイルを保存し、サービスを再起動します。



注記

起動したにもかかわらず認証されていない ssh セッションが多すぎるために、移行接続が切断する場合があります。デフォルトでは、**sshd** で許可されるセッションは 10 セッションのみで、常に "pre-authenticated state" となります。この設定は、sshd 設定ファイル (ここでは **/etc/ssh/sshd_config**) の **MaxStartups** パラメーターで制御されます。これには調整が必要な場合があります。この制限は DoS 攻撃 (および一般的なリソースの過剰使用) を防ぐために設定されているため、このパラメーターの調整は慎重に行ってください。この値を高く設定しすぎると、目的が無効になります。このパラメーターを変更するには、ファイル**/etc/ssh/sshd_config**を変更し、**MaxStartups** 行の先頭から**#**を削除して、**10** (デフォルト値) をより大きな数値に変更します。必ず保存して、**sshd** サービスを再起動してください。詳細は、**sshd_config** の man ページを参照してください。

4.4.2. virsh migrate コマンドの追加オプション

--live のほかに、virsh migrate では以下のオプションを利用できます。

- **--direct** - 直接移行に使用されます。
- **--p2p** - ピアツーピア移行に使用されます。
- **--tunnelled** - トンネル化された移行に使用されます。
- **--persistent** - ドメインを移行先ホストの物理マシンの永続状態に残します。
- **--undefinesource** - ソースホストの物理マシンのゲスト仮想マシンを削除します。
- **--suspend** - ドメインを移行先ホストの物理マシンの一時停止状態のままにします。
- **--change-protection** - 移行の進行中に、互換性のない設定変更がドメインに行われないように強制します。ハイパーバイザーによるサポートがある場合にこのオプションが暗黙的に有効になりますが、ハイパーバイザーに変更保護の対応がない場合に明示的に使用して移行を拒否できます。
- **--unsafe** - すべての安全手順を無視して、強制的に移行を実行します。
- **--verbose** - 移行の進行状況を表示します。
- **--abort-on-error** - 移行プロセス中にソフトエラー (I/O エラーなど) が発生すると移行を取り消します。
- **--migrateuri**: 通常省略される移行 URI です。
- **--domain** *[string]*- ドメイン名、ID、または uuid
- **--desturi***[string]*: クライアント (通常の移行) またはソース (p2p 移行) から見た宛先ホスト物理マシンの接続 URI
- **--migrateuri**: 移行 URI。通常は省略できます。
- **--timeout** *[seconds]*- ライブマイグレーションカウンターが N 秒を超えるとゲスト仮想マシンを強制的に中断します。ライブマイグレーションでのみ使用できます。タイムアウトが開始されると、一時停止されたゲスト仮想マシンで移行が続行されます。
- **--dname** *[string]* - 移行時にゲスト仮想マシンの名前を新規の名前に変更します (サポートされている場合)。
- **--xml** - 指定されたファイル名を使用して、宛先で使用する別の XML ファイルを提供できます。このファイル名は、基となるストレージへのアクセスで、ソースと宛先の名前の相違を考慮するなど、ホスト固有のドメイン XML の部分にさらに多くの変更を加えます。このオプションは通常、省略されます。

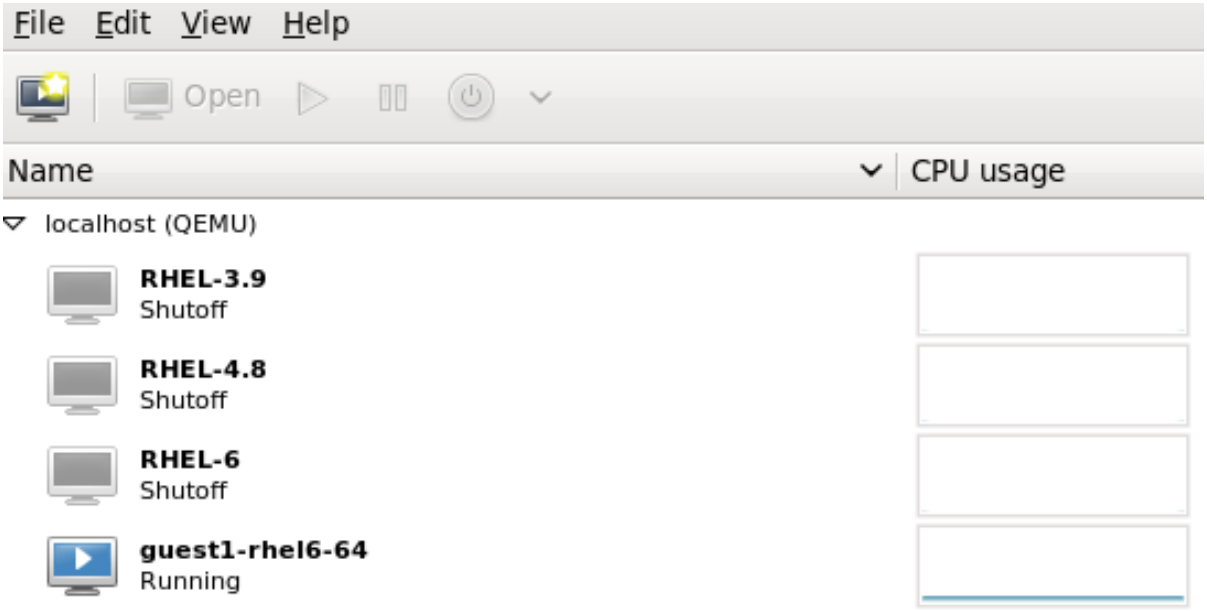
詳細は、man ページの virsh を参照してください。

4.5. VIRT-MANAGER を使用した移行

本セクションでは、**virt-manager** を使用した KVM ゲスト仮想マシンの、ホスト物理マシンから別のホスト物理マシンへの移行を説明します。

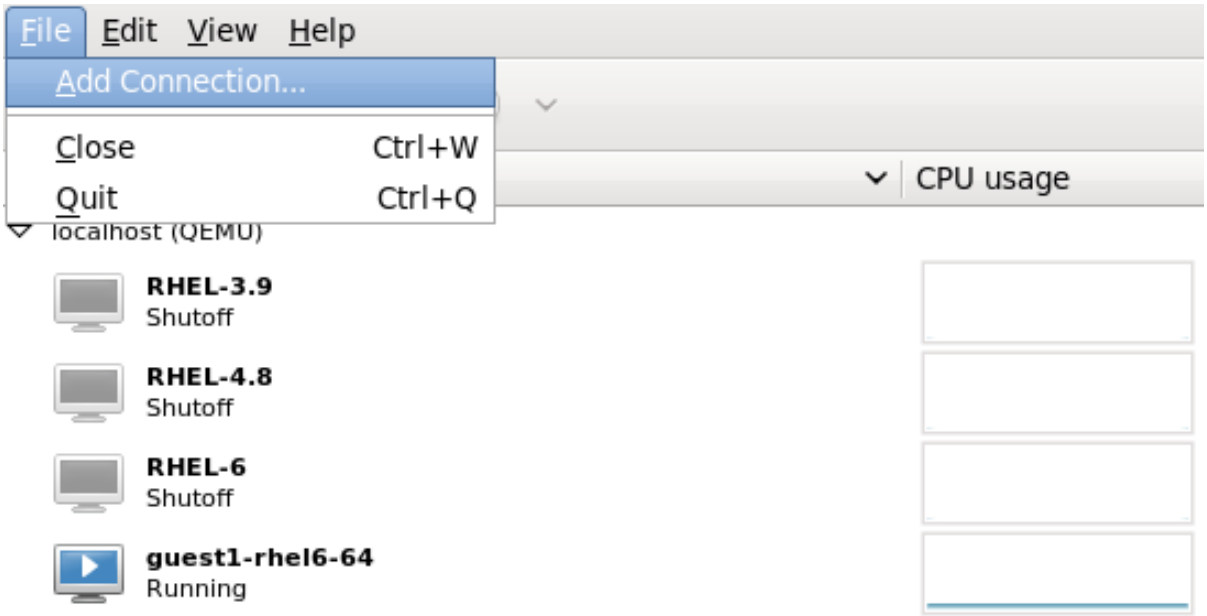
1. **virt-manager を開く**
   virt-manager を開きます。メインメニューバーから **Applications → System Tools → Virtual Machine Manager** を選択して、**virt-manager** を起動します。

図4.1 virt-Manager のメインメニュー



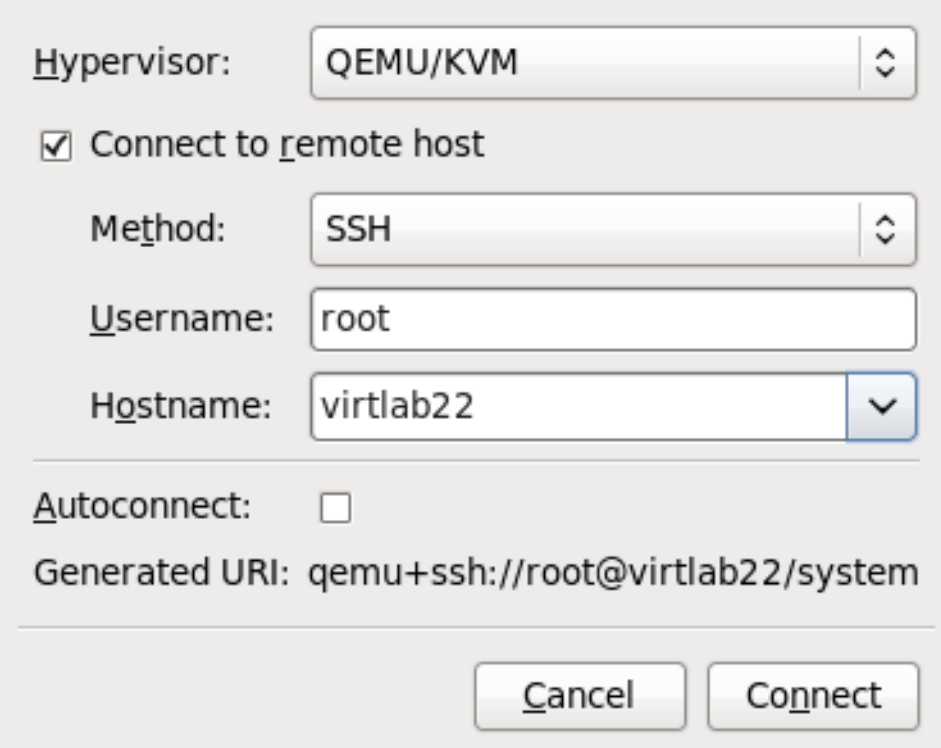
2. **ターゲットホストの物理マシンへの接続**
File メニューをクリックしてターゲットホストの物理マシンに接続し、Add Connection をクリックします。

図4.2 Add Connection ウィンドウの表示



3. **接続の追加**
Add Connection ウィンドウが表示されます。

図4.3 ターゲットホストの物理マシンへの接続の追加

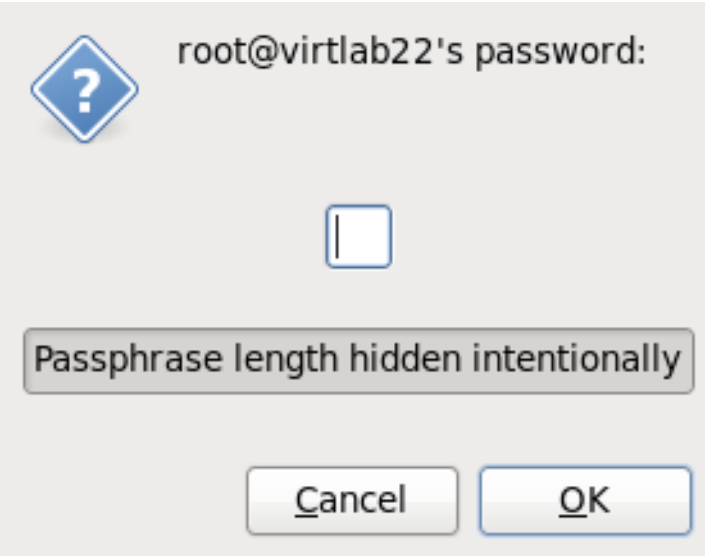


以下の詳細を入力します。

- **Hypervisor**: QEMU/KVM を選択します。
- **Method**: 接続方法を選択します。
- **Username**: リモートホストの物理マシンのユーザー名を入力します。
- **Hostname**: リモートホストの物理マシンのホスト名を入力します。

Connect ボタンをクリックします。この例では SSH 接続を使用しているため、次の手順で指定するユーザーのパスワードを入力する必要があります。

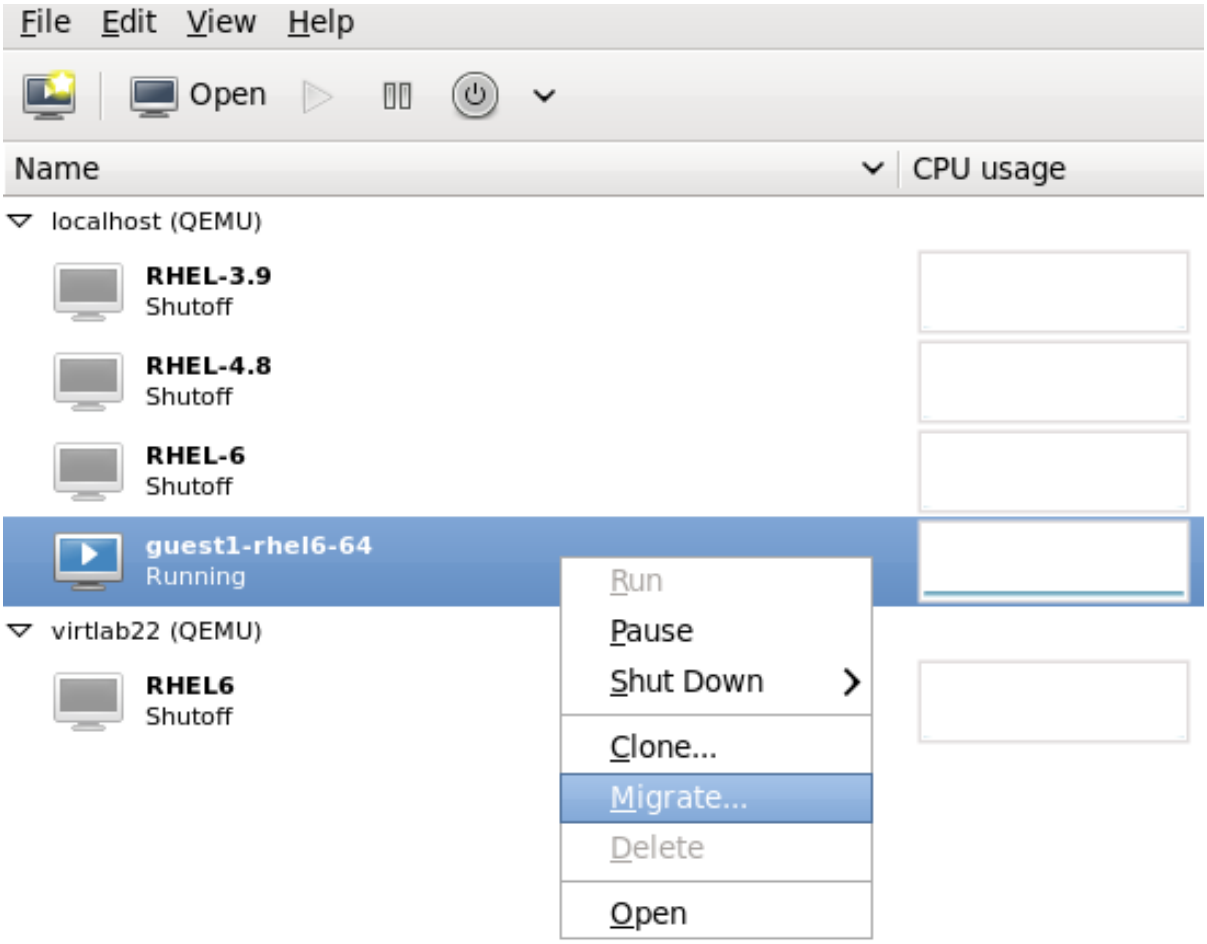
図4.4 パスワードを入力



4. **ゲスト仮想マシンの移行**

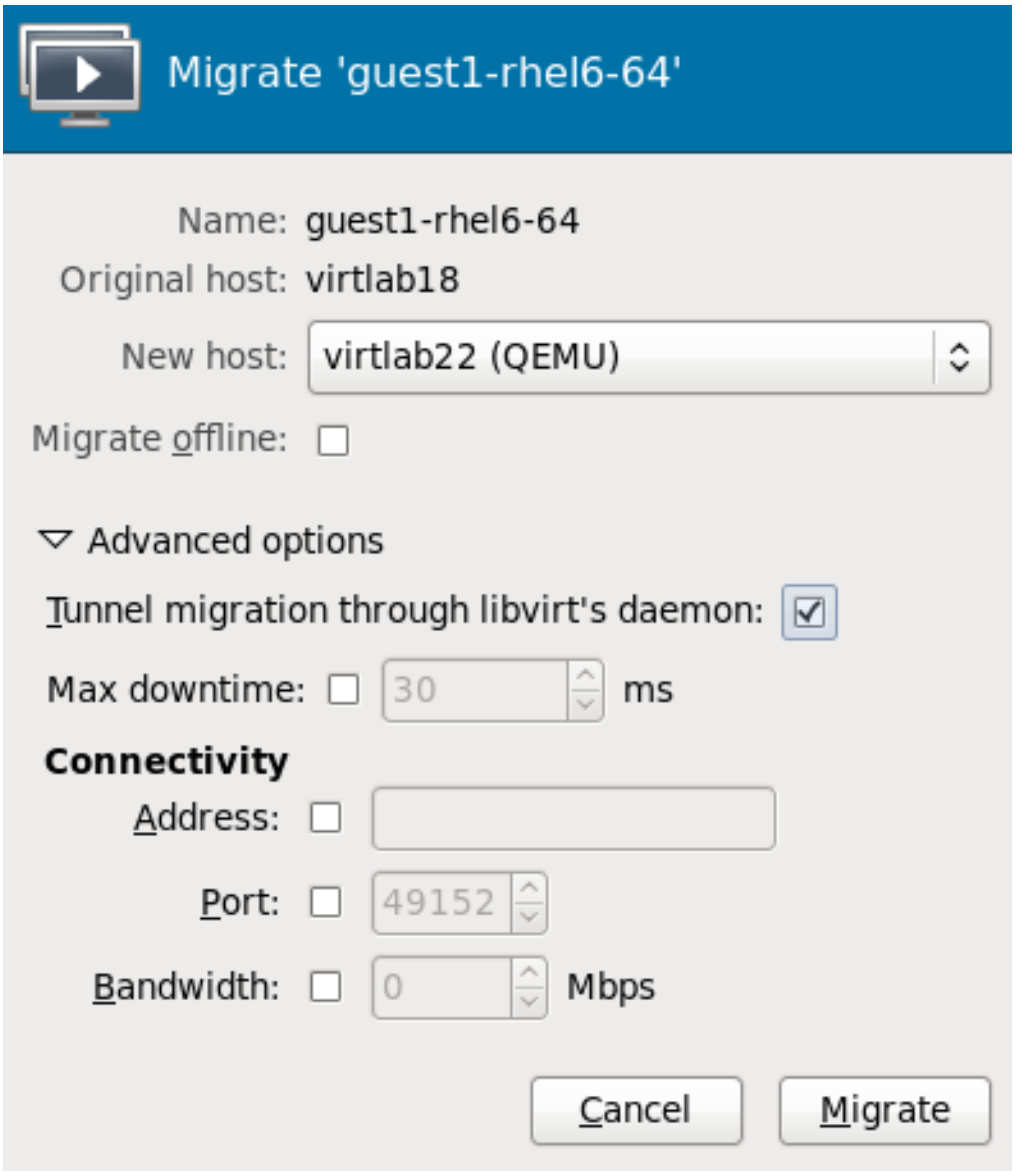
ソースホスト物理マシン内のゲストのリストを開き (ホスト名の左側にある小さな三角形をクリック)、移行するゲスト (この例では **guest1-rhel6-64**) を右クリックして、**Migrate** をクリックします。

図4.5 移行するゲストの選択



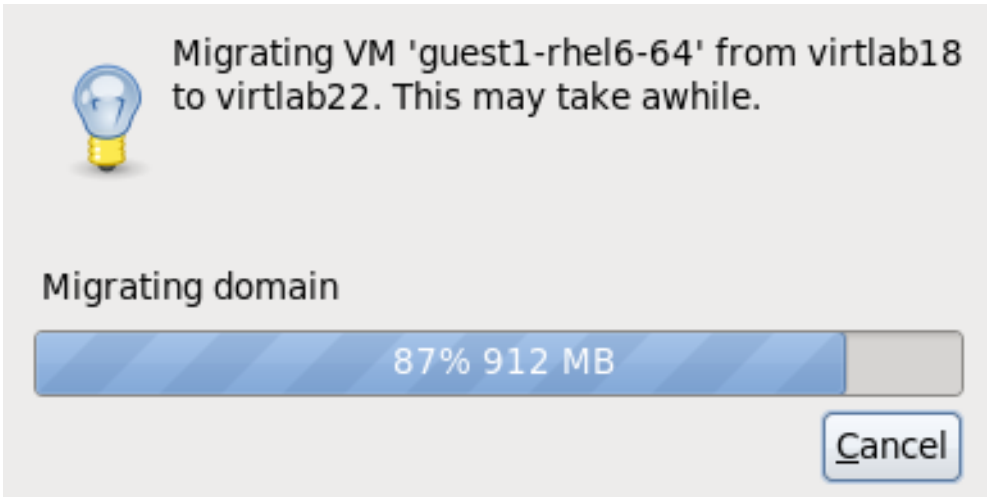
New Host フィールドで、ドロップダウンリストを使用して、ゲスト仮想マシンの移行先となるホスト物理マシンを選択し、**Migrate** をクリックします。

図4.6 移行先ホストの物理マシンの選択と、移行プロセスの開始



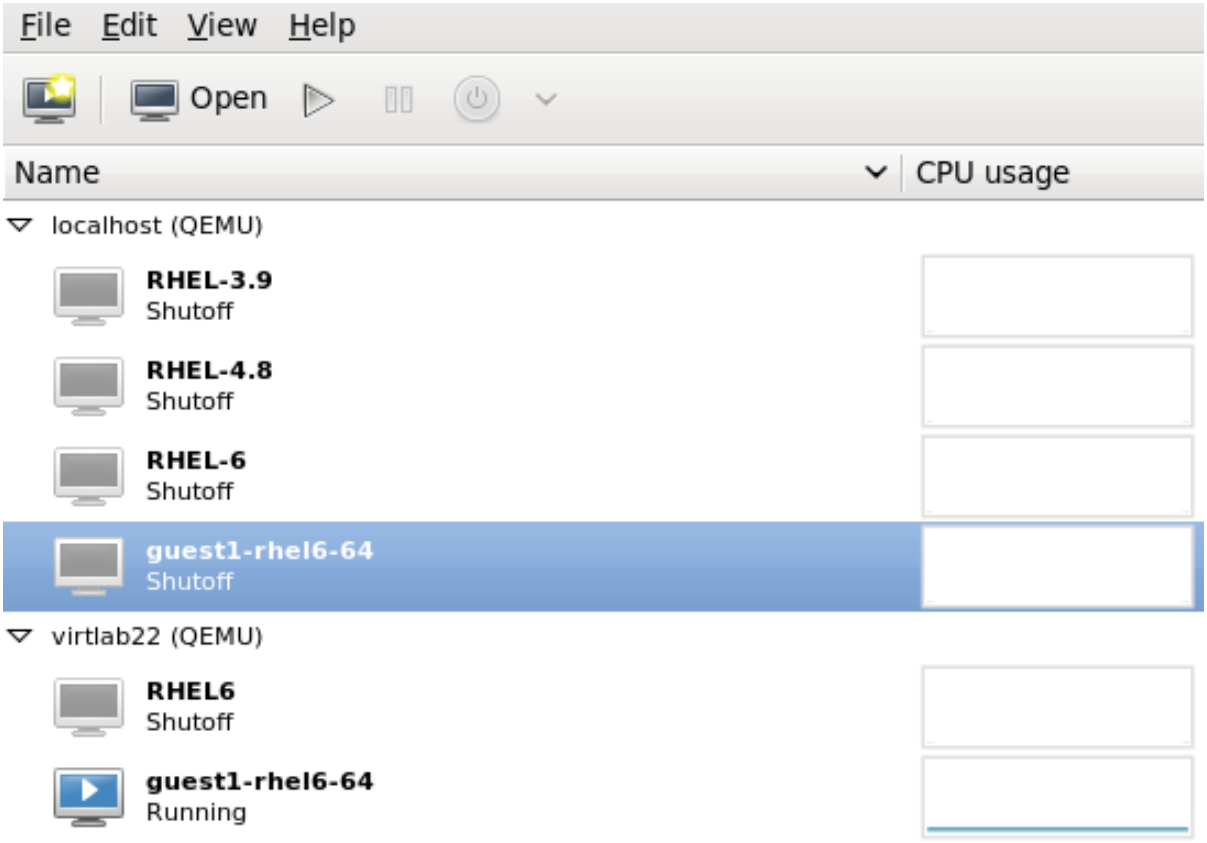
進捗ウィンドウが表示されます。

図4.7 進捗ウィンドウ



virt-manager は、移行先ホストで実行されている、新しく移行されたゲスト仮想マシンを表示するようになりました。これで、ソースホストの物理マシンで実行されていたゲスト仮想マシンがシャットオフ状態で一覧表示されます。

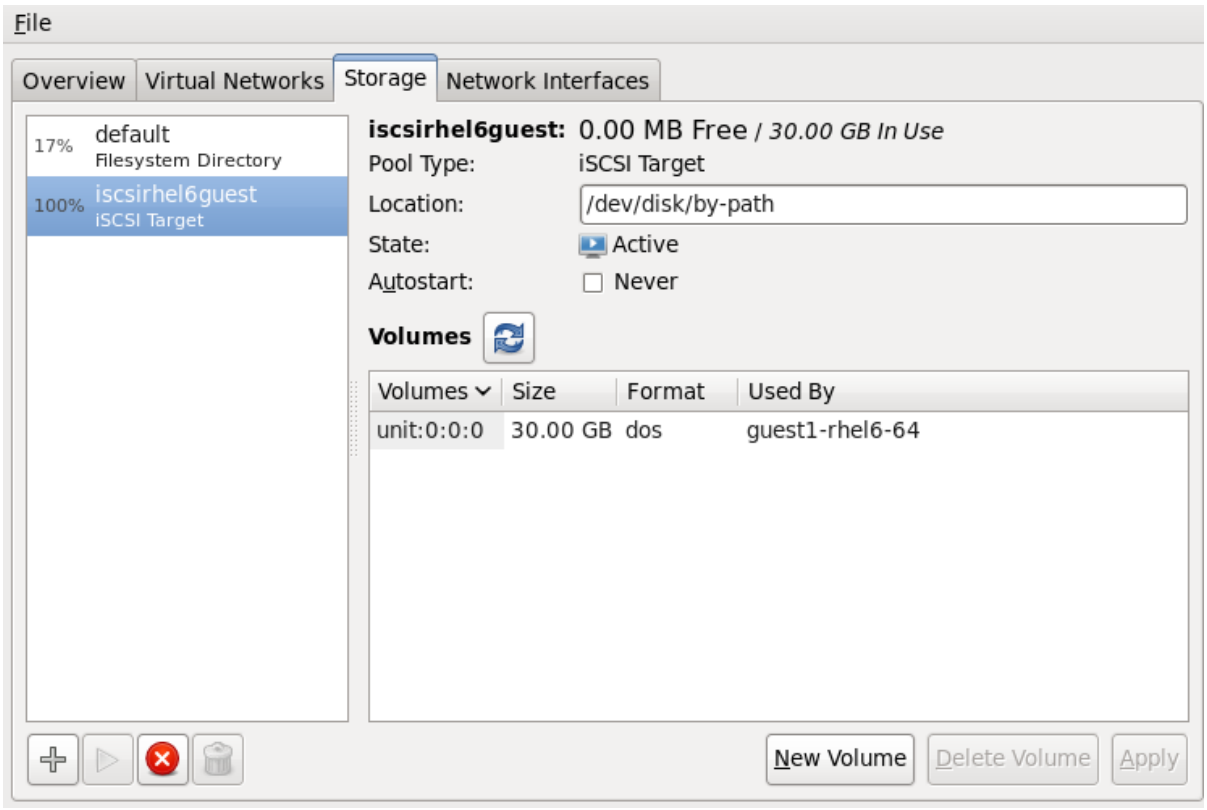
図4.8 移行先ホストの物理マシンで実行している移行ゲスト仮想マシン



5. **オプション: ホストの物理マシンのストレージの詳細を表示します。**
   Edit メニューで Connection Details をクリックすると、 **Connection Details** ウィンドウが表示されます。

   Storage タブをクリックします。移行先ホストの物理マシンの iSCSI ターゲットの詳細が表示されます。移行したゲスト仮想マシンがストレージを使用するものとして一覧表示されることに注意してください。

図4.9 ストレージの詳細



このホストは、以下の XML 設定で定義されています。

図4.10 宛先ホスト物理マシンの XML 設定

```
<pool type='iscsi'>
   <name>iscsirhel6guest</name>
   <source>
      <host name='virtlab22.example.com.'/>
      <device path='iqn.2001-05.com.iscsivendor:0-8a0906-fbab74a06-a700000017a4cc89-
rhevh'/>
   </source>
   <target>
      <path>/dev/disk/by-path</path>
   </target>
</pool>
  ...
```

第5章 ゲストのリモート管理

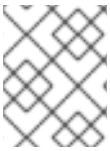
本セクションでは、**ssh** または TLS および SSL を使用してゲストをリモートで管理する方法を説明します。SSH の詳細は、『Red Hat Enterprise Linux デプロイメントガイド』 を参照してください。

5.1. SSH を使用したリモート管理

ssh パッケージは、暗号化したネットワークプロトコルを提供し、管理機能をリモート仮想サーバーに安全に送信できます。以下の方法は、**SSH** 接続上でセキュアにトンネリングされた **libvirt** 管理接続を使用して、リモートマシンを管理します。すべての認証は、**SSH**の公開鍵暗号と、ローカルの**SSH**エージェントが収集したパスワードまたはパスフレーズを使用して行われます。さらに、各ゲストの **VNC** コンソールは、**SSH** を介してトンネリングされます。

次のような、仮想マシンをリモート管理するために **SSH** を使用する際の問題に注意してください。

- 仮想マシンを管理するには、リモートマシンへの root ログインアクセスが必要です。
- 初期接続の設定プロセスが遅くなる可能性があります。
- すべてのホストまたはゲストでユーザーのキーを取り消す標準または簡単な方法はありません。
- SSH は、リモートマシンの数が多いと適切にスケーリングされません。



注記

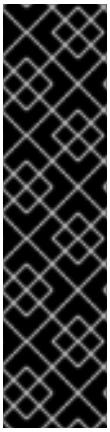
Red Hat Virtualization により、多数の仮想マシンのリモート管理が可能になります。詳細については、Red Hat Virtualization のドキュメントを参照してください。

ssh アクセスには、以下のパッケージが必要です。

- openssh
- openssh-askpass
- openssh-clients
- openssh-server

virt-manager 用にパスワードを使用しないまたはパスワードを使用したSSH アクセスの設定

以下の手順は、ゼロから開始しており、**SSH** キーが設定されていないことを前提としています。SSH 鍵を設定して別のシステムにコピーしている場合は、この手順をスキップできます。



重要

SSH 鍵はユーザーに依存するため、所有者のみが使用できます。キーの所有者は、キーを生成した人です。キーは共有できません。

リモートホストコンピューターに接続するには、キーを所有するユーザーが**virt-manager** を実行する必要があります。つまり、リモートシステムが root 以外のユーザーにより管理されている場合は、**virt-manager** を非特権モードで実行する必要があります。リモートシステムがローカルの root ユーザーで管理されている場合は、SSH 鍵を root が作成して所有する必要があります。

ローカルホストは、**virt-manager** を使用する非特権ユーザーとして管理できません。

1. **オプション: ユーザーの変更**
   必要に応じてユーザーを変更します。この例では、ローカルの root ユーザーを使用して、その他のホストとローカルホストをリモートで管理します。

   ```
   $ su -
   ```

2. **SSH 鍵ペアの生成**
   virt-manager が使用されているマシンで公開鍵ペアを生成します。この例では、~/**.ssh**/ ディレクトリーのデフォルトの鍵の場所を使用します。

   ```
   # ssh-keygen -t rsa
   ```

3. **リモートホストへの鍵のコピー**
   パスワードなしのリモートログイン、またはパスフレーズを使用したリモートログインでは、管理対象システムに SSH 鍵を配布する必要があります。**ssh-copy-id** コマンドを使用して、指定したシステムアドレス (この例では ***root@host2.example.com***) の root ユーザーにキーをコピーします。

   ```
   # ssh-copy-id -i ~/.ssh/id_rsa.pub root@host2.example.com
   root@host2.example.com's password:
   ```

   次に、**ssh root@host2.example.com** コマンドを使用してマシンにログインし、**.ssh/authorized_keys** ファイルをチェックインして予期しないキーが追加されていないことを確認します。

   必要に応じて、その他のシステムに対して繰り返します。

4. **オプション: パスフレーズを ssh-agent に追加します。**
   以下の手順では、既存の ssh-agent にパスフレーズを追加する方法について説明します。ssh-agent が実行されていない場合、実行に失敗します。エラーや競合を回避するには、SSH パラメーターが正しく設定されていることを確認してください。詳細は、『Red Hat Enterprise Linux 導入ガイド』 を参照してください。

   必要に応じて、SSH キーのパスフレーズを **ssh-agent** に追加します。ローカルホストで次のコマンドを使用して、パスフレーズ (存在する場合) を追加し、パスワードなしのログインを有効にします。

   ```
   # ssh-add ~/.ssh/id_rsa
   ```

   SSH キーがリモートシステムに追加されます。

libvirt デーモン (libvirtd)

libvirt デーモンは、仮想マシンを管理するインターフェイスを提供します。管理が必要なすべてのリモートホストに **libvirtd** デーモンをインストールして実行する必要があります。

```
$ ssh root@somehost
# chkconfig libvirtd on
# service libvirtd start
```

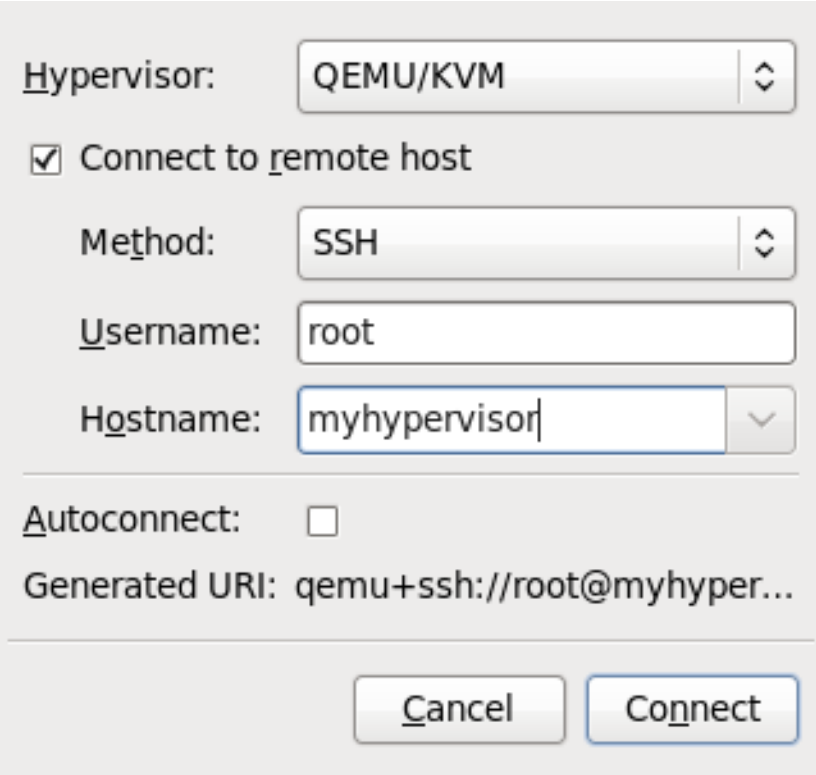
libvirtd と **SSH** を設定すると、仮想マシンにリモートでアクセスして管理できるようになります。この時点で、**VNC** でゲストにアクセスできるようになるはずです。

virt-manager を使用したリモートホストへのアクセス

リモートホストは、virt-manager GUI ツールで管理できます。SSH キーは、パスワードを使用しないログインを機能させるために、virt-manager を実行しているユーザーに属している必要があります。

1. virt-manager を起動します。
2. **File** ->**Add Connection** メニューを開きます。

   図5.1 Add Connection メニュー



3. ドロップダウンメニューを使用してハイパーバイザータイプを選択し、**Connect to remote host** チェックボックスをクリックして Connection **Method** (この場合は Remote tunnel over SSH) を開き、必要な **User name** と **Hostname** を入力して **Connect** をクリックします。

5.2. TLS および SSL でのリモート管理

TLS および SSL を使用して仮想マシンを管理できます。TLS および SSL はスケーラビリティーは高くなりますが、SSH よりも複雑になります (「SSH を使用したリモート管理」 を参照)。TLS と SSL は、Web ブラウザーでセキュアな接続に使用されるものと同じ技術です。**libvirt** 管理接続は受信接続の TCP ポートを開きます。これは、x509 証明書に基づいて安全に暗号化および認証されます。以下の手順では、TLS および SSL 管理用の認証証明書を作成してデプロイする方法を説明します。

手順5.1 TLS 管理用の認証局 (CA) 鍵の作成

1. 始める前に、**certtool** ユーティリティーがインストールされていることを確認してください。そうでない場合は、以下を行います。

```
# yum install gnutls-utils
```

2. 次のコマンドを使用して、秘密鍵を生成します。

```
# certtool --generate-privkey > cakey.pem
```

3. キーが生成されたら、次のステップは、キーが自己署名できるように署名ファイルを作成することです。作成するには、署名の詳細が含まれるファイルを作成し、**ca.info** という名前を付けます。このファイルには、以下の内容を記述する必要があります。

```
# vim ca.info
```

```
cn = Name of your organization
ca
cert_signing_key
```

4. 以下のコマンドを使用して自己署名キーを生成します。

```
# certtool --generate-self-signed --load-privkey cakey.pem --template ca.info --outfile
cacert.pem
```

ファイルが生成されたら、**rm** コマンドを使用して ca.info ファイルを削除できます。生成プロセスの結果として生成されるファイルの名前は **cacert.pem** です。このファイルは公開鍵 (証明書) です。読み込んだ**cakey.pem**は秘密鍵です。このファイルは共有スペースに保存しないでください。この鍵は秘密鍵を保持します。

5. **/etc/pki/CA/cacert.pem** ディレクトリー内のすべてのクライアントとサーバーに **cacert.pem** 認証局証明書ファイルをインストールして、CA によって発行された証明書が信頼できることを通知します。このファイルの内容を表示するには、次のコマンドを実行します。

```
# certtool -i --infile cacert.pem
```

これは、CA の設定に必要なものです。クライアントとサーバーの証明書を発行するために必要となるため、CA の秘密鍵を安全に保持します。

手順5.2 サーバー証明書の発行

この手順では、サーバーのホスト名に設定されている X.509 Common Name (CN) フィールドを使用して証明書を発行する方法を説明します。CN は、クライアントがサーバーへの接続に使用するホスト名と一致する必要があります。この例では、クライアントは URI: **qemu://mycommonname/system** を使用してサーバーに接続するため、CN フィールドは同一である必要があります (mycommoname)。

1. サーバーの秘密鍵を作成します。

```
# certtool --generate-privkey > serverkey.pem
```

2. 最初に **server.info** というテンプレートファイルを作成して、CA の秘密鍵の署名を生成します。CN がサーバーのホスト名と同じに設定されていることを確認します。

```
organization = Name of your organization
cn = mycommonname
tls_www_server
encryption_key
signing_key
```

3. 次のコマンドを使用して証明書を作成します。

```
# certtool --generate-certificate --load-privkey serverkey.pem --load-ca-certificate cacert.pem
--load-ca-privkey cakey.pem \ --template server.info --outfile servercert.pem
```

4. これにより、2 つのファイルが生成されます。

   - serverkey.pem - サーバーの秘密鍵
   - servercert.pem - サーバーの公開鍵

   秘密鍵の場所は、秘密にしておきます。ファイルの内容を表示するには、以下のコマンドを実行します。

```
# certtool -i --inifile servercert.pem
```

   このファイルを開く場合、**CN=** パラメーターは先に設定した CN と同じでなければなりません。(例:**mycommonname**)

5. 次の場所にある 2 つのファイルをインストールします。

   - **serverkey.pem** - サーバーの秘密鍵。このファイルは、**/etc/pki/libvirt/private/serverkey.pem** に置きます。
   - **servercert.pem** - サーバーの証明書。サーバーの **/etc/pki/libvirt/servercert.pem** にインストールします。

手順5.3 クライアント証明書の発行

1. すべてのクライアント (つまり、virt-manager などの libvirt にリンクされているプログラム) について、X.509 識別名 (DN) が適切な名前に設定された証明書を発行する必要があります。これは、企業レベルで決定する必要があります。

   たとえば、以下の情報が使用されます。

```
C=USA,ST=North Carolina,L=Raleigh,O=Red Hat,CN=name_of_client
```

   このプロセスは、以下の例外を除き、手順5.2「サーバー証明書の発行」 と非常に似ています。

2. 以下のコマンドを使用して秘密鍵を作成します。

```
# certtool --generate-privkey > clientkey.pem
```

3. 最初に **client.info** という名前のテンプレートファイルを作成して、CA の秘密鍵の署名を生成します。ファイルには以下の内容が含まれている必要があります (フィールドは、お住まいの地域/場所に合わせてカスタマイズする必要があります)。

```
country = USA
```

```
state = North Carolina
locality = Raleigh
organization = Red Hat
cn = client1
tls_www_client
encryption_key
signing_key
```

4. 次のコマンドを使用して、証明書に署名します。

```
# certtool --generate-certificate --load-privkey clientkey.pem --load-ca-certificate cacert.pem \
--load-ca-privkey cakey.pem --template client.info --outfile clientcert.pem
```

5. クライアントマシンに証明書をインストールします。

```
# cp clientkey.pem /etc/pki/libvirt/private/clientkey.pem
# cp clientcert.pem /etc/pki/libvirt/clientcert.pem
```

5.3. トランスポートモード

リモートマネジメントの場合、**libvirt** は以下のトランスポートモードに対応します。

Transport Layer Security (TLS)

Transport Layer Security TLS 1.0 (SSL 3.1) が認証され、暗号化された TCP/IP ソケット。通常はパブリックポート番号をリッスンします。これを使用するには、クライアント証明書とサーバー証明書を生成する必要があります。標準のポートは 16514 です。

UNIX ソケット

UNIX ドメインソケットは、ローカルマシンでのみアクセスできます。ソケットは暗号化されず、UNIX の権限または SELinux を使用して認証を行います。通常のソケット名は **/var/run/libvirt/libvirt-sock** および **/var/run/libvirt/libvirt-sock-ro** です (読み取り専用接続の場合)。

SSH

SSH (Secure Shell Protocol) 接続で転送されます。Netcat が必要です (ncパッケージ) がインストールされています。libvirt デーモン (**libvirtd**) は、リモートマシンで実行している必要があります。SSH アクセスには、ポート 22 を開いておく必要があります。ある種の SSH キー管理 (たとえば、**ssh-agent**ユーティリティー) を使用する必要があります。そうしないと、パスワードの入力を求められます。

ext

ext パラメーターは、libvirt のスコープ外の方法でリモートマシンに接続できる外部プログラムに使用されます。このパラメーターはサポートされていません。

TCP

暗号化されていない TCP/IP ソケット。実稼働環境での使用は推奨されていないため、これは通常無効になっていますが、管理者はこれをテストしたり、信頼できるネットワークで使用したりできます。デフォルトのポートは 16509 です。

デフォルトのトランスポートは、その他のトランスポートを指定しないと TLS になります。

リモート URI

URI (Uniform Resource Identifier) は、**virsh** および libvirt がリモートホストに接続するために使用します。URI は、**virsh** コマンドの **--connect** パラメーターとともに使用して、リモートホストで単一コマンドまたは移行を実行することもできます。リモート URI は、通常のローカル URI を取得し、ホスト名またはトランスポート名を追加することで形成されます。特別な場合として、remote の URI スキームを使用すると、リモートの libvirtd サーバーで、最適なハイパーバイザードライバーをプローブするように指示されます。これは、ローカル接続の NULL URI を渡すことに相当します。

libvirt の URI は一般的な形式をとります (角括弧内のコンテンツ "[]" は任意の関数を表します)。

```
driver[+transport]://[username@][hostname][:port]/path[?extraparameters]
```

ハイパーバイザー (ドライバー) が QEMU の場合は、パスが必須であることに注意してください。XEN の場合はオプションです。

以下は、有効なリモート URI の例になります。

- qemu://hostname/
- xen://hostname/
- xen+ssh://hostname/

外部の場所を対象とする場合は、トランスポート方法またはホスト名を指定する必要があります。詳細は、http://libvirt.org/guide/html/Application_Development_Guide-Architecture-Remote_URIs.html を参照してください。

リモート管理パラメーターの例

- SSH トランスポートと SSH ユーザー名 **virtuser** を使用して、**host2** という名前のリモート KVM ホストに接続します。それぞれの connect コマンドは **connect [<name>] [--readonly]** です。ここで、**<name>** はここで説明する有効な URI です。 **virsh connect** コマンドの詳細については、「connect」 を参照してください。

  ```
  qemu+ssh://virtuser@hot2/
  ```

- TLS を使用して、**host2** という名前のホスト上のリモートの KVM ハイパーバイザーに接続します。

  ```
  qemu://host2/
  ```

テスト例

- 標準以外の UNIX ソケットを使用して、ローカルの KVM ハイパーバイザーに接続します。この場合は、UNIX ソケットの完全パスが明示的に指定されています。

  ```
  qemu+unix:///system?socket=/opt/libvirt/run/libvirt/libvirt-sock
  ```

- 暗号化されていない TCP/IP 接続を使用して、ポート 5000 の IP アドレス 10.1.1.10 のサーバーに libvirt デーモンを接続します。これは、デフォルト設定でテストドライバーを使用します。

  ```
  test+tcp://10.1.1.10:5000/default
  ```

追加の URI パラメーター

リモートの URI には、追加のパラメーターを追加できます。以下の表 表5.1「追加の URI パラメーター」 では、認識されるパラメーターを説明します。その他のパラメーターはすべて無視されます。パラメーター値は URI エスケープする必要があります (つまり、パラメーターおよび特殊文字が URI 形式に変換される前に疑問符 (?) が追加されることを意味します)。

表5.1 追加の URI パラメーター

名前	トランスポートモード	説明	使用例
name	すべてのモード	リモートの virConnectOpen 関数に渡される名前。名前は、通常、リモートの URI から transport、hostname、port number、username、および追加のパラメーターを削除して生成されますが、非常に複雑なケースでは、名前を明示的に指定することをお勧めします。	name=qemu:///system
command	ssh と ext	外部コマンド。ext トランスポートの場合はこれが必要です。ssh の場合、デフォルトは ssh です。コマンド用に PATH が検索されます。	command=/opt/openssh/bin/ssh
socket	unix と ssh	UNIX ドメインソケットへのパスで、デフォルトを上書きします。ssh トランスポートの場合は、これがリモートの netcat コマンド (netcat を参照) に渡されます。	socket=/opt/libvirt/run/libvirt/libvirt-sock

名前	トランスポートモード	説明	使用例
netcat	ssh	**netcat** コマンドを使用して、リモートシステムに接続できます。デフォルトの netcat パラメーターは **nc** コマンドを使用します。SSH トランスポートの場合、libvirt は以下の形式を使用して SSH コマンドを構築します。 ***command*** -p ***port*** [-l ***username***] ***hostname*** ***netcat*** -U ソケット ***port***、***username***、および ***hostname*** パラメーターは、リモート URI の一部として指定できます。***command***、***netcat***、および ***socket*** は、他の追加パラメーターから取得されます。	netcat=/opt/netcat/bin/nc
no_verify	tls	ゼロ以外の値に設定すると、サーバーの証明書のクライアントチェックが無効になります。クライアントの証明書または IP アドレスのサーバーチェックを無効にするには、libvirtd 設定を変更する必要があります。	no_verify=1
no_tty	ssh	0 以外の値に設定すると、リモートマシンに自動的にログインできない場合に ssh がパスワードを要求できなくなります。ターミナル へのアクセスがない場合は、これを使用します。	no_tty=1

第6章 KVM でのオーバーコミット

KVM ハイパーバイザーは、CPU とメモリーを自動的にオーバーコミットします。つまり、仮想化した CPU とメモリーは、システムにある物理リソースよりも仮想マシンに割り当てることができます。これが可能なのは、ほとんどのプロセスが割り当てられたリソースの 100% に常にアクセスするわけではないためです。

その結果、十分に活用されていない仮想化サーバーまたはデスクトップをより少ないホストで実行できるため、多くのシステムリソースが節約され、サーバーハードウェアへの電力、冷却、および投資が削減されます。

6.1. メモリーのオーバーコミット

KVM ハイパーバイザーで実行しているゲスト仮想マシンには、物理 RAM の専用ブロックが割り当てられていません。代わりに、各ゲスト仮想マシンは、ホスト物理マシンの Linux カーネルが、要求された場合にのみメモリーを割り当てる Linux プロセスとして機能します。また、ホストのメモリーマネージャーは、ゲスト仮想マシンのメモリーを、自身の物理メモリーとスワップ領域の間で移動できます。

オーバーコミットを行うには、ホストの物理マシンに、すべてのゲスト仮想マシンに対応する十分なスワップ領域と、ホストの物理マシンのプロセスに十分なメモリーを割り当てる必要があります。基本的には、ホスト物理マシンのオペレーティングシステムには、最大 4GB のメモリーと、最小 4GB のスワップ領域が必要です。スワップパーティションに適したサイズを判断する方法は、Red Hat ナレッジベース を参照してください。

重要

オーバーコミットは、一般的なメモリー問題には理想的な解決策ではありません。メモリー不足に対処するための推奨される方法は、ゲストごとに割り当てるメモリーを減らすか、ホストに物理メモリーを追加するか、スワップスペースを利用することです。

仮想マシンを頻繁にスワップすると、仮想マシンの実行が遅くなります。また、オーバーコミットによりシステムのメモリーが不足 (OOM) する可能性があります。これにより、Linux カーネルが重要なシステムプロセスをシャットダウンする可能性があります。メモリーをオーバーコミットする場合は、十分なテストが行われていることを確認してください。オーバーコミットのサポートについては、Red Hat サポートまでご連絡ください。

オーバーコミットは、すべてのゲスト仮想マシンで機能するわけではありませんが、最小限の集中使用でデスクトップ仮想化セットアップで機能するか、カーネルの同じページのマージ (KSM) で複数の同一のゲスト仮想マシンを実行することがわかっています。

KSM とオーバーコミットの詳細については、*7章KSM* を参照してください。

重要

デバイスの割り当て が使用中の場合に、割り当てられたデバイスでダイレクトメモリーアクセス (DMA) を有効にするには、仮想マシンのすべてのメモリーを静的に事前に割り当てる必要があるためです。したがって、メモリーのオーバーコミットはデバイス割り当てではサポートされていません。

6.2. 仮想 CPU のオーバーコミット

KVM ハイパーバイザーは、仮想化 CPU のオーバーコミットに対応します。仮想化 CPU は、ゲスト仮想マシンの負荷制限が許す限り、オーバーコミットできます。負荷が 100% に近いと、要求が破棄され

たり、使用できない応答時間が発生する可能性があるため、VMSCPU をオーバーコミットする場合は注意してください。

仮想化 CPU (vCPU) は、単一のホスト物理マシンに同じ vCPU を共有しない複数のゲスト仮想マシンがある場合に最適にオーバーコミットされます。 KVM は、1 台の単一ホスト物理マシン上の 1 台の物理 CPU に対して 5 台の VCPU (5 台の仮想マシン上) の比率で、負荷が 100% 未満のゲスト仮想マシンを安全にサポートする必要があります。KVM はすべてのマシンを切り替え、負荷のバランスが取れていることを確認します。

処理コアの物理数を超えてゲスト仮想マシンをオーバーコミットしないでください。たとえば、4 つの vCPU を備えたゲスト仮想マシンは、デュアルコアプロセッサーを備えたホスト物理マシンではなく、クアッドコアホストで実行する必要があります。また、物理プロセッサーコアごとに、割り当てられた vCPU の合計が 10 を超えることは推奨されません。

重要

広範なテストを行わずに、実稼働環境で CPU をオーバーコミットしないでください。処理リソースを 100% 使用するアプリケーションは、オーバーコミットされた環境では不安定になる可能性があります。デプロイ前にテストします。

仮想マシンから最高のパフォーマンスを引き出す方法の詳細については、Red Hat Enterprise Linux 6 仮想化チューニングおよび最適化ガイド を参照してください。

第7章 KSM

共有メモリーの概念は、最新のオペレーティングシステムでは一般的です。たとえば、プログラムが最初に起動されると、そのプログラムはすべてのメモリーを親プログラムと共有します。子プログラムまたは親プログラムがこのメモリーを変更しようとすると、カーネルは新しいメモリー領域を割り当て、元のコンテンツをコピーして、プログラムがこの新しい領域を変更できるようにします。これは、コピーオンライトとして知られています。

KSM は、この概念を逆に使用する新しい Linux 機能です。KSM を使用すると、カーネルはすでに実行中の 2 つ以上のプログラムを検証し、それらのメモリーを比較できます。いずれかのメモリー領域またはページが同一である場合、KSM は複数の同一のメモリーページを 1 つのページに減らします。このページは、コピーオンライトとしてマークされます。ページのコンテンツがゲスト仮想マシンによって変更された場合、そのゲスト仮想マシン用に新しいページが作成されます。

これは、KVM を使用した仮想化に役立ちます。ゲスト仮想マシンが起動すると、親 **qemu-kvm** プロセスからメモリーのみが継承されます。ゲスト仮想マシンが実行されると、ゲストが同じオペレーティングシステムまたはアプリケーションを実行しているときに、ゲスト仮想マシンのオペレーティングシステムイメージのコンテンツを共有できます。

注記

ページ重複排除テクノロジー (KSM 実装でも使用) は、複数のゲスト間で情報を漏洩するために使用される可能性のあるサイドチャネルを導入する可能性があります。これが懸念される場合は、ゲストごとに KSM を無効にすることができます。

KSM は、メモリーの速度と使用率を強化します。KSM では、共通のプロセスデータがキャッシュまたはメインメモリーに保存されます。これにより、KVM ゲストのキャッシュミスが減少し、一部のアプリケーションとオペレーティングシステムのパフォーマンスが向上します。次に、メモリーを共有すると、ゲストの全体的なメモリー使用量が削減され、リソースの密度と使用率が向上します。

注記

Red Hat Enterprise Linux 6.5 以降、KSM は NUMA に対応しています。これにより、ページのコアレッシング中に NUMA の局所性を考慮できるため、ページがリモートノードに移動されることに関連するパフォーマンスの低下を防ぐことができます。Red Hat は、KSM が使用されている場合に、ノード間のメモリーマージを回避することを推奨します。KSM を使用している場合は、**/sys/kernel/mm/ksm/merge_across_nodes** を ***0*** に変更して、NUMA ノード間でページがマージされないようにします。カーネルメモリーが計算した統計情報は、ノード間での大量のマージ後にはそれぞれの間で相反する場合があります。そのため、KSM デーモンが大量のメモリーをマージすると、numad が混乱する可能性があります。システムに未使用のメモリーが大量にあると、KSM デーモンをオフにして無効にすることでパフォーマンスが高まる場合があります。NUMA の詳細は、『Red Hat Enterprise Linux パフォーマンスチューニングガイド』を参照してください。

Red Hat Enterprise Linux は、KSM を制御するために 2 つの異なる方法を使用します。

- **ksm** サービスは、KSM カーネルスレッドを開始および停止します。
- **ksmtuned** サービスは **ksm** を制御し、調整し、同じページのマージを動的に管理します。**ksmtuned** サービスは **ksm** を開始し、メモリー共有が必要ない場合は **ksm** サービスを停止します。**ksmtuned** サービスは、***retune***新しいゲストが作成または破棄されたときに実行するパラメーター。

これらのサービスは両方とも、標準のサービス管理ツールで制御されます。

KSM サービス

ksm サービスは qemu-kvm パッケージに含まれています。KSM は、Red Hat Enterprise Linux 6 ではデフォルトでオフになっています。ただし、Red Hat Enterprise Linux 6 を KVM ホスト物理マシンとして使用する場合は、**ksm**/**ksmtuned** サービスによってオンにされる可能性があります。

ksm サービスが開始されていない場合、KSM は 2000 ページのみを共有します。このデフォルトは低く、メモリー節約の利点は限られています。

ksm サービスが開始されると、KSM はホスト物理マシンシステムのメインメモリーの最大半分を共有します。**ksm** サービスを開始して、KSM がより多くのメモリーを共有できるようにしてください。

```
# service ksm start
Starting ksm:                                  [  OK  ]
```

ksm サービスは、デフォルトの起動シーケンスに追加できます。chkconfig コマンドを使用して、**ksm** サービスを永続化します。

```
# chkconfig ksm on
```

KSM チューニングサービス

ksmtuned サービスにはオプションがありません。**ksmtuned** サービスは、**ksm** をループして調整します。さらに、ゲスト仮想マシンが作成または破棄されると、**ksmtuned** サービスに libvirt から通知されます。

```
# service ksmtuned start
Starting ksmtuned:                             [  OK  ]
```

ksmtuned サービスは、***retune*** パラメーターを使用して調整できます。***retune*** パラメーターは、チューニング機能を手動で実行するように **ksmtuned** に指示します。

ファイル内のパラメーターを変更する前に、明確にする必要のあるいくつかの用語があります。

- **thres**: アクティベーションキーのしきい値 (kbytes)KSM サイクルは、全 ***thres*** プロセス RSZ の合計にシステムメモリーの合計に ***qemu-kvm*** 値が追加されるとトリガーされます。このパラメーターは、***KSM_THRES_COEF*** で定義されたパーセンテージの kbytes と同じです。

/etc/ksmtuned.conf ファイルは、**ksmtuned** サービスの設定ファイルです。以下のファイル出力は、デフォルトの **ksmtuned.conf** ファイルです。

```
# Configuration file for ksmtuned.

# How long ksmtuned should sleep between tuning adjustments
# KSM_MONITOR_INTERVAL=60

# Millisecond sleep between ksm scans for 16Gb server.
# Smaller servers sleep more, bigger sleep less.
# KSM_SLEEP_MSEC=10

# KSM_NPAGES_BOOST is added to the npages value, when free memory is less than thres.
# KSM_NPAGES_BOOST=300

# KSM_NPAGES_DECAY Value given is subtracted to the npages value, when free memory is
greater than thres.
```

```
# KSM_NPAGES_DECAY=-50

# KSM_NPAGES_MIN is the lower limit for the npages value.
# KSM_NPAGES_MIN=64

# KSM_NAGES_MAX is the upper limit for the npages value.
# KSM_NPAGES_MAX=1250

# KSM_TRES_COEF - is the RAM percentage to be calculated in parameter thres.
# KSM_THRES_COEF=20

# KSM_THRES_CONST - If this is a low memory system, and the thres value is less than
KSM_THRES_CONST, then reset thres value to KSM_THRES_CONST value.
# KSM_THRES_CONST=2048

# uncomment the following to enable ksmtuned debug information
# LOGFILE=/var/log/ksmtuned
# DEBUG=1
```

KSM 変数とモニターリング

KSM は、監視データを **/sys/kernel/mm/ksm/** ディレクトリーに保存します。このディレクトリー内のファイルは、カーネルによって更新される、KSM の使用状況と統計の正確な記録です。

以下のリストの変数は、下記のように、**/etc/ksmtuned.conf** ファイルの設定可能な変数でもあります。

/sys/kernel/mm/ksm/ ファイル

full_scans
フルスキャンが実行されます。

pages_shared
共有されたページの総数。

pages_sharing
現在共有されているページ。

pages_to_scan
スキャンされていないページ。

pages_unshared
共有されなくなったページ。

pages_volatile
揮発性ページの数。

run
KSM プロセスが実行されているかどうか。

sleep_millisecs

スリープのミリ秒。

DEBUG=1 行が **/etc/ksmtuned.conf** ファイルに追加された場合、KSM チューニングアクティビティーは **/var/log/ksmtuned** ログファイルに保存されます。ログファイルの場所は、***LOGFILE*** パラメーターを使用して変更することができます。ログファイルの場所を変更することはお勧めできません。SELinux 設定の特別な設定が必要になる場合があります。

KSM の非アクティブ化

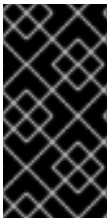
KSM にはパフォーマンスのオーバーヘッドがあり、特定の環境またはホストの物理マシンシステムには大きすぎる可能性があります。

KSM は、**ksmtuned** および **ksm** サービスを停止することで非アクティブ化できます。サービスを停止すると KSM が非アクティブになりますが、再起動後も持続しません。

```
# service ksmtuned stop
Stopping ksmtuned:                         [  OK  ]
# service ksm stop
Stopping ksm:                              [  OK  ]
```

chkconfig コマンドを使用して KSM を永続的に非アクティブ化します。サービスを無効にするには、以下のコマンドを実行します。

```
# chkconfig ksm off
# chkconfig ksmtuned off
```



重要

KSM であっても、コミットされた RAM にスワップサイズがあれば十分です。KSM は、同一または類似のゲストの RAM 使用量を削減します。十分な swap 領域のない KSM でゲストをオーバーコミットすると可能かもしれませんが、ゲスト仮想マシンのメモリー使用によりページが共有されない可能性があるためです。

第8章 高度なゲスト仮想マシン管理

この章では、ゲスト仮想マシンで利用できるようになるシステムリソースを微調整および制御するための高度な管理ツールについて説明します。

8.1. コントロールグループ (CGROUP)

Red Hat Enterprise Linux 6 は、新しいカーネル機能を提供します。*control groups* は、*cgroup* と呼ばれることがよくあります。Cgroup を使用すると、CPU 時間、システムメモリー、ネットワーク帯域幅、またはこれらのリソースの組み合わせなどのリソースを、システムで実行されているタスク (プロセス) のユーザー定義グループに割り当てることができます。設定した cgroup を監視したり、特定のリソースへの cgroup のアクセスを拒否したり、実行中のシステムで cgroup を動的に再設定したりすることもできます。

cgroup 機能は **libvirt** によって完全にサポートされています。デフォルトでは、**libvirt** は各ゲストをさまざまなコントローラー (メモリー、CPU、blkio、デバイスなど) の個別の制御グループに入れます。

ゲストが開始されると、ゲストはすでに cgroup に含まれています。必要になる可能性がある唯一の設定は、cgroups でのポリシーの設定です。cgroups の詳細は、『Red Hat Enterprise Linux リソース管理ガイド』 を参照してください。

8.2. HUGE PAGE のサポート

本セクションでは、Huge Page のサポートについて説明します。

はじめに

x86 CPU は通常、4kB ページのメモリーに対応しますが、*Huge Page* と呼ばれる大容量のページを使用できます。KVM ゲストは、*Transaction Lookaside Buffer (TLB)* に対する CPU キャッシュヒットを増やすことでパフォーマンスを向上させるために、Huge Page メモリーサポートを使用してデプロイできます。特に大容量メモリーとメモリーを大量に消費するワークロードなど、Huge Page によってパフォーマンスが大幅に向上する可能性があります。Red Hat Enterprise Linux 6 は、Huge Page を使用してページサイズを大きくすることにより、大量のメモリーをより効果的に管理できます。

KVM ゲストに巨大なページを使用することにより、ページテーブルに使用されるメモリーが少なくなり、TLB ミスが減少するため、特にメモリーを大量に消費する状況でパフォーマンスが大幅に向上します。

Transparent Huge Pages

Transparent Huge Page (THP) は、アプリケーションに必要な TLB エントリーを減らすカーネル機能です。また、すべての空きメモリーをキャッシュとして使用するようにすることで、パフォーマンスが向上します。

透過的な Huge Page を使用するには、**qemu.conf** ファイルに特別な設定は必要ありません。**/sys/kernel/mm/redhat_transparent_hugepage/enabled** が **always** に設定されている場合、Huge Page はデフォルトで使用されます。

透過的な Huge Page では、**hugetlbfs** 機能の使用は妨げられません。ただし、hugetlbfs が使用されていない場合、KVM は通常の 4kB ページサイズの代わりに透過的な巨大ページを使用します。

注記

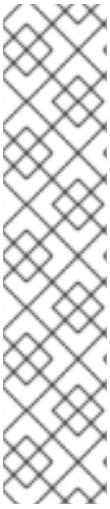
Huge Pages でメモリーパフォーマンスを調整する手順については、Red Hat Enterprise Linux 7 仮想化調整および最適化ガイド を参照してください。

8.3. HYPER-V ハイパーバイザーでのゲスト仮想マシンとしての RED HAT ENTERPRISE LINUX の実行

Microsoft Windows Hyper-V ハイパーバイザーを実行する Microsoft Windows ホストの物理マシンで Red Hat Enterprise Linux ゲスト仮想マシンを実行できます。特に、Red Hat Enterprise Linux ゲスト仮想マシンのデプロイおよび管理を容易にするために、以下の機能拡張が追加されました。

- VMBUS プロトコルのアップグレード - VMBUS プロトコルが Windows 8 レベルにアップグレードされました。この作業の一環として、ゲストで利用可能な全仮想 CPU で VMBUS 割り込みを処理できるようになりました。さらに、Red Hat Enterprise Linux ゲスト仮想マシンと Windows ホストの物理マシン間のシグナルプロトコルが最適化されています。
- 合成フレームバッファードライバー - Red Hat Enterprise Linux デスクトップユーザー向けのグラフィックパフォーマンスと優れた解決策を提供します。
- ライブ仮想マシンのバックアップサポート: ライブの Red Hat Enterprise Linux ゲスト仮想マシンの中断のないバックアップサポートをプロビジョニングします。
- 固定サイズの Linux VHD 動的拡張 - ライブマウントされた固定サイズ Red Hat Enterprise Linux VHD の拡張を可能にします。

詳細については、Windows Server 2012R2Hyper-V での Linux サポートの有効化 を参照してください。



注記

Hyper-V ハイパーバイザーは、ユーザーがディスクの未使用の最後の部分をドロップできるようにすることで、最後のパーティションの後に空き領域がある場合に、Red Hat Enterprise Linux ゲストの GPT パーティションディスクの縮小をサポートします。ただし、この操作はディスク上のセカンダリー GPT ヘッダーをサイレントに削除します。これにより、ゲストがパーティションテーブルを調べるときにエラーメッセージが生成される場合があります (たとえば、**parted** を使用してパーティションテーブルを印刷する場合)。これは Hyper-V の既知の制限です。回避策として、**gdisk** のエキスパートメニューと **e** コマンドを使用して、GPT ディスクを縮小した後にセカンダリー GPT ヘッダーを手動で復元することができます。さらに、Hyper-V マネージャーで展開オプションを使用すると、GPT セカンダリーヘッダーもディスクの端以外の場所に配置されますが、これは **parted** を使用して移動できます。これらのコマンドの詳細については、**gdisk** および **parted** の man ページを参照してください。

8.4. ゲスト仮想マシンのメモリー割り当て

次の手順は、ゲスト仮想マシンにメモリーを割り当てる方法を示しています。この割り当てと割り当ては起動時にのみ機能し、メモリー値への変更は次の再起動まで有効になりません。ゲストごとに割り当てることができる最大メモリーは 4TiB です。ただし、このメモリー割り当てがホストの物理マシンリソースが提供できる量を超えない場合に限ります。

有効なメモリーユニットは以下のとおりです。

- バイトの場合は ***b*** または ***bytes*** になります。
- キロバイトの場合は、***KB*** になります (10 ^{3} または 1,000 バイトのブロック)
- キビバイトの場合は ***k*** または ***KiB*** (2 ^{10} または 1024 バイトのブロック)
- ***MB*** メガバイトの場合 (10 ^{6} または 1,000,000 バイトのブロック)

- メビバイトの場合は ***M*** または ***MiB*** (2^{20} または 1,048,576 バイトのブロック)
- ***GB*** (ギガバイト) (10^9 または 1,000,000,000 バイトのブロック)
- ギビバイトの場合は ***G*** または ***GiB*** (^{30} または 1,073,741,824 バイトのブロック)
- エクサバイトの場合は ***TB*** (10^{12} または 1,000,000,000,000 バイト のブロック)
- テビバイトの場合は ***T*** または ***TiB*** (2^{40} または 1,099,511,627,776 のブロック)

すべての値は、libvirt により、最も近い kibibyte に丸められます。また、ハイパーバイザーが対応する粒度に丸めることもできます。一部のハイパーバイザーは、4000KiB(4000 x 2^{10} または 4,096,000 バイト) などの最小値も適用します。この値の単位は、オプションの属性 ***memory unit*** により決定されます。デフォルトは、測定単位としての KiB (キビバイト) になります。この値は、2^{10} または 1024 バイトのブロックで乗算されます。

ゲスト仮想マシンがクラッシュした場合に、オプションの属性***dumpCore*** を使用して、ゲスト仮想マシンのメモリーを、生成されるコアダンプ (***dumpCore='on'***) に含めるか (***dumpCore='off'***) 含まないかを制御できます。デフォルト設定は***on***したがって、パラメーターがに設定されていない場合***off***、ゲスト仮想マシンのメモリーはコアダンプファイルに含まれます。

currentMemory 属性は、ゲスト仮想マシンの実際のメモリー割り当てを決定します。この値は最大割り当てよりも低くなり、ゲスト仮想マシンのメモリーを即座にバルーンアップすることができます。これを省略すると、デフォルトでは memory 要素と同じ値に設定されます。unit 属性の動作は、メモリーと同じです。

このセクションはすべて、以下のようにドメイン XML を変更する必要があります。

```
<domain>

  <memory unit='KiB' dumpCore='off'>524288</memory>
  <!-- changes the memory unit to KiB and does not allow the guest virtual machine's memory to be
included in the generated coredump file -->
  <currentMemory unit='KiB'>524288</currentMemory>
  <!-- makes the current memory unit 524288 KiB -->
  ...
</domain>
```

8.5. ゲスト仮想マシンの自動起動

本セクションでは、ホストの物理マシンのブートフェーズでゲスト仮想マシンを自動的に起動する方法を説明します。

この例では、**virsh** を使用してゲスト仮想マシン ***TestServer*** を設定して、ホストの物理マシンのブート時に自動的に起動します。

```
# virsh autostart TestServer
Domain TestServer marked as autostarted
```

ゲスト仮想マシンがホストの物理マシンで自動的に起動するようになりました。

ゲスト仮想マシンの自動ブートを停止するには、***--disable*** パラメーターを使用します。

```
# virsh autostart --disable TestServer
Domain TestServer unmarked as autostarted
```

ゲスト仮想マシンは、ホストの物理マシンで自動的に起動しなくなりました。

8.6. ゲスト仮想マシンの SMART ディスク監視の無効化

SMART ディスクの監視は、仮想ディスクと物理ストレージデバイスがホストの物理マシンにより管理されているため、安全に無効にできます。

```
# service smartd stop
# chkconfig --del smartd
```

8.7. VNC サーバーの設定

VNC サーバーを設定するには、**System > Preferences** で **Remote Desktop** アプリケーションを使用します。または、**vino-preferences** コマンドを実行できます。

専用の VNC サーバーセッションを設定するには、以下の手順に従います。

必要に応じて、**~/.vnc/xstartup** ファイルを作成してから編集し、**vncserver** が起動するたびに GNOME セッションを開始します。**vncserver** スクリプトの初回実行時に、VNC セッションに使用するパスワードの入力が求められます。vnc サーバーファイルの詳細は、『Red Hat Enterprise Linux インストールガイド』 を参照してください。

8.8. 新しい一意の MAC アドレスの生成

ゲスト仮想マシン用に新しい、一意の MAC アドレスを生成する必要がある場合があります。書き込み時に新しい MAC アドレスを生成するためのコマンドラインツールはありません。以下のスクリプトは、ゲスト仮想マシンの新規の MAC アドレスを生成することができます。ゲスト仮想マシンの スクリプトを **macgen.py** として保存します。これで、そのディレクトリーから **./macgen.py** を使用してスクリプトを実行でき、新しい MAC アドレスが生成されます。出力の例を以下に示します。

```
$ ./macgen.py
00:16:3e:20:b0:11
```

```
#!/usr/bin/python
# macgen.py script to generate a MAC address for guest virtual machines
#
import random
#
def randomMAC():
 mac = [ 0x00, 0x16, 0x3e,
  random.randint(0x00, 0x7f),
  random.randint(0x00, 0xff),
  random.randint(0x00, 0xff) ]
 return ':'.join(map(lambda x: "%02x" % x, mac))
#
print randomMAC()
```

8.8.1. ゲスト仮想マシン用に新しい MAC を生成する別の方法

python-virtinst の組み込み モジュールを使用して、ゲスト仮想マシン設定ファイルで使用するための新しい MAC アドレスと **UUID** を生成することもできます。

```
# echo  'import virtinst.util ; print\
 virtinst.util.uuidToString(virtinst.util.randomUUID())' | python
# echo  'import virtinst.util ; print virtinst.util.randomMAC()' | python
```

上記のスクリプトは、以下のようにスクリプトファイルとして実装することもできます。

```
#!/usr/bin/env python
#  -*- mode: python; -*-
print ""
print "New UUID:"
import virtinst.util ; print virtinst.util.uuidToString(virtinst.util.randomUUID())
print "New MAC:"
import virtinst.util ; print virtinst.util.randomMAC()
print ""
```

8.9. ゲスト仮想マシンの応答時間の改善

ゲスト仮想マシンは、特定のワークロードや使用パターンで応答する可能性があります。低速または応答しないゲスト仮想マシンを引き起こす可能性のある状況の例:

- 深刻なオーバーコミットされたメモリー。
- プロセッサーの使用率が高いオーバーコミットメモリー
- その他の (**qemu-kvm** プロセスではない) は、ホストの物理マシンのプロセスがビジー状態または停止している。

KVM ゲスト仮想マシンは Linux プロセスとして機能します。Linux プロセスは、メインメモリー (物理 RAM) に永続的に保持されるわけではなく、特に使用されていない場合はスワップスペース (仮想メモリー) に配置されます。ゲスト仮想マシンが長期間非アクティブである場合、ホストの物理マシンはゲスト仮想マシンを swap に移動する可能性があります。swap は物理メモリーよりも遅くなるため、ゲストが応答していない可能性があります。ゲストがメインメモリーに読み込まれると、この変更が行われます。スワップに使用されるストレージのタイプとコンポーネントのパフォーマンスによっては、ゲスト仮想マシンをスワップからメインメモリーにロードするプロセスに、ゲスト仮想マシンに割り当てられた RAM のギガバイトあたり数秒かかる場合があることに注意してください。

KVM ゲスト仮想マシンプロセスは、メモリーがオーバーコミットされているか、全体的なメモリー使用量に関係なく、スワップに移動される場合があります。

安全でないオーバーコミットレベルを使用したり、スワップをオフにしてゲスト仮想マシンプロセスやその他の重要なプロセスをオーバーコミットしたりすることはお勧めしません。メモリーをオーバーコミットするときは、ホストの物理マシンに十分なスワップスペースがあることを常に確認してください。

KVM でのオーバーコミットの詳細については、*6章KVM でのオーバーコミット* を参照してください。

警告

仮想メモリーを使用すると、Linux システムはシステム上の物理 RAM よりも多くのメモリーを使用できます。十分に使用されていないプロセスがスワップアウトされるため、アクティブなプロセスがメモリーを使用できるようになり、メモリー使用率が向上します。swap を無効にすると、すべてのプロセスが物理 RAM に保存されるため、メモリー使用率が低下します。

swap がオフになっている場合は、ゲスト仮想マシンをオーバーコミットしないでください。swap を使用せずにゲスト仮想マシンをオーバーコミットすると、ゲスト仮想マシンまたはホストの物理マシンシステムがクラッシュする可能性があります。

8.10. LIBVIRT での仮想マシンタイマー管理

ゲスト仮想マシンでの正確な時間管理は、仮想プラットフォームにおける鍵となる課題です。さまざまなハイパーバイザーが、さまざまな方法で時間管理の問題を処理しようとします。libvirt は、ドメイン XML の <clock> 要素と <timer> 要素を使用して、時間管理のためのハイパーバイザーに依存しない設定設定を提供します。ドメイン XML は、**virsh edit** コマンドを使用して編集できます。詳しくは、「ゲスト仮想マシンの設定ファイルの編集」 を参照してください。

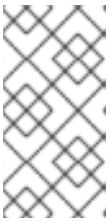
<clock> 要素は、ゲスト仮想マシンのクロックをホスト物理マシンのクロックと同期させる方法を決定するために使用されます。clock 要素には以下の属性があります。

- **offset** は、ゲスト仮想マシンのクロックがホストの物理マシンのクロックからどのようにオフセットされるかを決定します。offset 属性には、以下の値を使用できます。

表8.1 offset 属性値

値	説明
utc	ゲスト仮想マシンのクロックは、起動時に UTC に同期されます。
localtime	ゲスト仮想マシンのクロックは、起動時にホストの物理マシンの設定済みタイムゾーンに同期されます (存在する場合)。
timezone	ゲスト仮想マシンのクロックは、***timezone*** 属性で指定された特定のタイムゾーンに同期されます。

値	説明
variable	ゲスト仮想マシンクロックは UTC の任意のオフセットに同期されます。UTC との相対的なデルタは、***adjustment*** 属性を使用して秒単位で指定します。ゲスト仮想マシンは、時間の経過とともにリアルタイムクロック (RTC) を自由に調整でき、次の再起動後にそれが受け入れられることを期待しています。これは、***utc*** モードとは対照的に、リブートごとに RTC の調整が失われます。



注記

値 **utc** は、デフォルトで仮想マシンのクロックオフセットとして設定されます。ただし、ゲスト仮想マシンのクロックを **localtime** の値で実行している場合は、ゲスト仮想マシンのクロックをホスト物理マシンのクロックと同期させるため、クロックオフセットを変更して別の値にする必要があります。

- **timezone** 属性は、ゲスト仮想マシンクロックに使用されるタイムゾーンを決定します。
- **adjustment** 属性は、ゲスト仮想マシンのクロック同期にデルタを提供します。秒単位で、UTC との相対パスになります。

例8.1 常に UTC に同期する

```
<clock offset="utc" />
```

例8.2 常にホストの物理マシンのタイムゾーンに同期する

```
<clock offset="localtime" />
```

例8.3 任意のタイムゾーンへの同期

```
<clock offset="timezone" timezone="Europe/Paris" />
```

例8.4 UTC + 任意のオフセットに同期する

```
<clock offset="variable" adjustment="123456" />
```

8.10.1. クロックのタイマー子要素

clock 要素には、ゼロ以上の timer 要素を子として指定できます。timer 要素は、ゲスト仮想マシンのクロック同期に使用されるタイムソースを指定します。timer 要素には以下の属性があります。***name*** のみが必要で、他のすべての属性は任意です。

name 属性は使用する時間ソースの型を決定し、以下のいずれかになります。

表8.2 名前属性値

値	説明
pit	Programmable Interval Timer - 周期的な割り込みがあるタイマーです。
rtc	Real Time Clock - 周期的な割り込みがある継続的に実行するタイマー。
tsc	Time Stamp Counter: リセットしてからのティック数をカウント、割り込みなし。
kvmclock	KVM クロック: KVM ゲスト仮想マシンの推奨されるクロックソース。KVM pvclock または kvm-clock を使用すると、ゲスト仮想マシンがホストの物理マシンのウォールクロックタイムを読み取ることができます。

8.10.2. track

track 属性はタイマーによって追跡されるものを指定します。**rtc** の名前の値にのみ有効です。

表8.3 属性値を追跡する

値	説明
boot	古い **ホストの物理マシン** オプションに対応します。これはサポート対象外の追跡オプションです。
guest	RTC は常にゲスト仮想マシンの時間を追跡します。
wall	RTC は常にホスト時間を追跡します。

8.10.3. tickpolicy

tickpolicy 属性は、ゲスト仮想マシンにティックを渡すために使用されるポリシーを割り当てます。以下の値を使用できます。

表8.4 tickpolicy 属性値

値	説明
delay	通常のレートで配信を継続します (ティックが遅れます)。
catchup	キャッチアップするために、より高いレートで配信されます。

値	説明
merge	ティックが 1 つのティックにマージされます。
discard	不明なティックはすべて破棄されます。

8.10.4. 頻度、モード、および表示

frequency 属性は固定頻度を設定するために使用されます。Hz で測定されます。この属性は、**name** 要素に **tsc** の値がある場合にのみ関連します。他のすべてのタイマーは固定周波数 (**pit**、**rtc**) で動作します。

mode は、タイムソースがゲスト仮想マシンに公開される方法を決定します。この属性は、**tsc** の name 値にのみ関係します。その他のタイマーは常にエミュレートされます。コマンドは以下のようになります。**<timer name='tsc' frequency='NNN' mode='auto|native|emulate|smpsafe'/>**モードの定義は表に記載されます。

表8.5 モード属性値

値	説明
auto	TSC が不安定である場合はネイティブで、ネイティブの TSC アクセスを許可します。
native	常にネイティブ TSC アクセスを許可します。
エミュレート	常に TSC をエミュレートします。
smpsafe	常に TSC およびインロック SMP をエミュレートします。

present は、ゲスト仮想マシンに表示されるデフォルトのタイマーセットを上書きするために使用されます。

表8.6 present 属性値

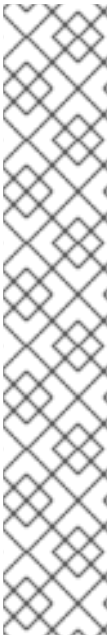
値	説明
はい	このタイマーがゲスト仮想マシンに表示されるよう強制します。
いいえ	このタイマーをゲスト仮想マシンに表示しないように強制します。

8.10.5. クロック同期の使用例

> **例8.5 RTC および PIT タイマーとローカルタイムに同期されるクロック**
>
> この例では、クロックは RTC および PIT タイマーのローカルタイムに同期されます。

```
<clock offset="localtime">
 <timer name="rtc" tickpolicy="catchup" track="guest virtual machine" />
 <timer name="pit" tickpolicy="delay" />

</clock>
```



注記

PIT クロックソースは、64 ビットホスト (PIT を使用できない) で実行されている 32 ビットゲストで、次の条件で使用できます。

- ゲスト仮想マシンには 1 つの CPU のみを指定可能
- APIC タイマーを無効にする必要があります (noapictimer コマンドラインオプションを使用)
- ゲストで NoHZ モードを無効にする必要があります (nohz = off コマンドラインオプションを使用)
- ゲストでは高解像度タイマーモードを無効にする必要があります (highres = off コマンドラインオプションを使用)
- PIT クロックソースは、高解像度タイマーモードまたは NoHz モードと互換性がありません。

8.11. PMU を使用したゲスト仮想マシンのパフォーマンスの監視

Red Hat Enterprise Linux 6.4 では、テクノロジープレビューとして vPMU(仮想 PMU) が導入されました。vPMU は、Intel の PMU (Performance Monitoring Units) に基づいており、Intel マシンでのみ使用できます。PMU により、ゲスト仮想マシンの動作を示す統計の追跡が可能になります。

パフォーマンス監視を使用すると、開発者はプロファイリングにパフォーマンスツールを使用する一方で、開発者は CPU の PMU カウンターを使用できます。仮想パフォーマンス監視ユニット機能により、仮想マシンユーザーはゲスト仮想マシンで発生する可能性のあるパフォーマンスの問題のソースを特定できるため、KVM ゲスト仮想マシンのプロファイルが改善します。

この機能を有効にするには、**-cpu host** フラグを設定する必要があります。

この機能は、Red Hat Enterprise Linux 6 を実行しているゲスト仮想マシンでのみサポートされており、デフォルトでは無効になっています。この機能は、Linux perf ツールの使用でのみ機能します。以下のコマンドを使用して、perf パッケージがインストールされていることを確認します。

```
# yum install perf.
```

perf コマンドの詳細は、**perf** の man ページを参照してください。

8.12. ゲスト仮想マシン電源管理

Libvirt のドメイン XML で以下のパラメーターを変更することで、ゲスト仮想マシンのオペレーティングシステムに対して BIOS アドバタイズメントを強制的に有効または無効にできます。

```
...
  <pm>
```

```
  <suspend-to-disk enabled='no'/>
  <suspend-to-mem enabled='yes'/>
</pm>
...
```

要素 ***pm*** S3 (ディスクへのサスペンド) および S4 (メモリーへのサスペンド) ACPI スリープ状態の BIOS サポートを有効 (はい) または無効 (いいえ) にします。何も指定しないと、ハイパーバイザーはデフォルト値のままになります。

第9章 ゲスト仮想マシンデバイスの設定

Red Hat Enterprise Linux 6 は、ゲスト仮想マシンの 3 つのクラスのデバイスに対応します。

- *Emulated devices* は、実際のハードウェアを模倣する純粋な仮想デバイスであり、未変更のゲストオペレーティングシステムは標準のインボックスドライバーを使用して動作できるようにします。Red Hat Enterprise Linux 6 は、216 virtio デバイスまで対応します。

- *VirtIO devices* は、仮想マシンで最適に動作するように設計された仮想デバイスです。VirtIO デバイスはエミュレートされたデバイスと似ていますが、Linux 以外の仮想マシンには、デフォルトで必要となるドライバーは含まれません。Virtual Machine Manager (**virt-manager**) や Red Hat Virtualization Hypervisor などの仮想化管理ソフトウェアは、サポートされている Linux 以外のゲストオペレーティングシステム用にこれらのドライバーを自動的にインストールします。Red Hat Enterprise Linux 6 は、最大 700 の scsi ディスクに対応します。

- *割り当てられたデバイス* は、仮想マシンに公開される物理デバイスです。このメソッドは passthrough としても知られています。デバイスの割り当てにより、仮想マシンはさまざまなタスクで PCI デバイスに排他的にアクセスできるようになり、PCI デバイスがゲストオペレーティングシステムに物理的に接続されているかのように見え、動作します。Red Hat Enterprise Linux 6 は、仮想マシンごとに割り当てられたデバイスを最大 32 個までサポートします。

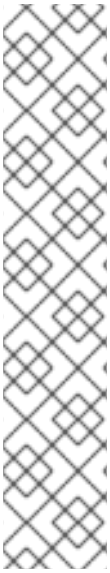
デバイスの割り当ては、グラフィックデバイスの選択など、PCIe デバイスでサポートされます。Red Hat Enterprise Linux 6 のデバイスの割り当てで、NVIDIA K シリーズ Quadro、GRID、および Tesla グラフィックカード GPU 機能がサポートされるようになりました。パラレル PCI デバイスは、割り当てられたデバイスとしてサポートされる場合がありますが、セキュリティーとシステム設定の競合のために厳しい制限があります。

注記

仮想マシンに接続できるデバイスの数は、いくつかの要因によって異なります。1 つの要因は、QEMU プロセスによって開かれるファイルの数です (**/etc/security/limits.conf** で設定され、**/etc/libvirt/qemu.conf** でオーバーライドできます)。他の制限要因には、仮想バスで利用可能なスロット数や、sysctl で設定されたオープンファイルのシステム全体の制限が含まれます。

特定のデバイスの詳細と制限に関する情報は、「Devices」 を参照してください。

Red Hat Enterprise Linux 6 は、仮想マシンに単一の機能スロットとして公開されるデバイスの PCI ホットプラグをサポートします。これを可能にするために、単一機能のホストデバイスや、多機能のホストデバイスの個別の機能を設定できます。デバイスを多機能 PCI スロットとして仮想マシンに公開する設定は、ホットプラグ以外のアプリケーションにのみ推奨されます。



注記

デバイスが割り当てられたゲストをホストから完全に分離するには、プラットフォームが割り込みの再マッピングをサポートしている必要があります。このようなサポートがないと、ホストは悪意のあるゲストからの割り込み注入攻撃に対して脆弱となる可能性があります。ゲストが信頼できる環境では、管理者は **allow_unsafe_interrupts** ***vfio_iommu_type1*** モジュールへのオプションを使用して引き続き PCI デバイスの割り当てを許可することを選択できます。これは、以下を含む **/etc/modprobe.d** に .conf ファイル (**local.conf** など) を追加することで、永続的に実行できます。

```
options vfio_iommu_type1 allow_unsafe_interrupts=1
```

または sysfs エントリーを使用して動的に同じことを行います。

```
# echo 1 > /sys/module/vfio_iommu_type1/parameters/allow_unsafe_interrupts
```

9.1. PCI デバイス

PCI デバイスの割り当ては、Intel VT-d または AMD IOMMU のいずれかに対応するハードウェアプラットフォームでのみ利用できます。PCI デバイスの割り当てを機能させるには、この Intel VT-d または AMD IOMMU の仕様が BIOS で有効になっている必要があります。

手順9.1 PCI デバイス割り当てのための Intel システムの準備

1. **Intel VT-d 仕様を有効にする**
   Intel VT-d 仕様では、物理デバイスを仮想マシンに直接割り当てるハードウェアサポートが提供されます。この仕様は、Red Hat Enterprise Linux で PCI デバイスの割り当てを使用するために必要です。

   Intel VT-d 仕様は、BIOS で有効にする必要があります。システムの製造元によっては、この仕様をデフォルトで無効にしている場合があります。これらの仕様を表示するのに使用される用語はメーカーにより異なります。適切な用語は、システムの製造元のドキュメントを参照してください。

2. **カーネルで Intel VT-d をアクティブにします。**
   /etc/sysconfig/grub ファイル内の GRUB_CMDLINX_LINUX 行の末尾に、引用符で囲んで ***intel_iommu=on*** パラメーターおよび iommu=pt パラメーターを追加して、カーネル内で Intel VT-d をアクティブにします。

   以下は、Intel VT-d を有効にした修正**grub**です。

   ```
   GRUB_CMDLINE_LINUX="rd.lvm.lv=vg_VolGroup00/LogVol01
   vconsole.font=latarcyrheb-sun16 rd.lvm.lv=vg_VolGroup_1/root
   vconsole.keymap=us $([ -x /usr/sbin/rhcrashkernel-param ] && /usr/sbin/
   rhcrashkernel-param || :) rhgb quiet intel_iommu=on"
   ```

3. **設定ファイルの再生成**
   以下を実行して /etc/grub2.cfg を再生成します。

   ```
   grub2-mkconfig -o /etc/grub2.cfg
   ```

   UEFI ベースのホストを使用している場合は、ターゲットファイルが**/etc/grub2-efi.cfg** であることに注意してください。

4. **使用準備完了**
   システムを再起動して、変更を有効にします。これで、システムで PCI デバイスの割り当てが可能になります。

手順9.2 PCI デバイス割り当て用の AMD システムの準備

1. **AMD IOMMU 仕様を有効にする**
   Red Hat Enterprise Linux で PCI デバイスの割り当てを使用するには、AMD IOMMU の仕様が必要です。この仕様は、BIOS で有効にする必要があります。システムの製造元によっては、この仕様をデフォルトで無効にしている場合があります。

2. **IOMMU カーネルサポートの有効化**
   システムの起動時に AMD IOMMU 仕様が有効になるように、**/etc/sysconfig/grub** の GRUB_CMDLINX_LINUX 行の末尾に引用符で囲って ***amd_iommu=on*** を追加します。

3. **設定ファイルの再生成**
   以下を実行して /etc/grub2.cfg を再生成します。

   ```
   grub2-mkconfig -o /etc/grub2.cfg
   ```

   UEFI ベースのホストを使用している場合は、ターゲットファイルが**/etc/grub2-efi.cfg** であることに注意してください。

4. **使用準備完了**
   システムを再起動して、変更を有効にします。これで、システムで PCI デバイスの割り当てが可能になります。

9.1.1. virsh を使用した PCI デバイスの割り当て

この手順では、KVM ハイパーバイザーの仮想マシンに PCI デバイスを割り当てる方法を説明します。

この例では、PCI 識別子コード、**pci_0000_01_00_0**、および完全に仮想化されたゲストマシン *guest1-rhel6-64* を持つ PCIe ネットワークコントローラーを使用します。

手順9.3 virsh を使用した PCI デバイスのゲスト仮想マシンへの割り当て

1. **デバイスの識別**
   まず、仮想マシンへのデバイス割り当てに指定されている PCI デバイスを特定します。使用可能な PCI デバイスの一覧を表示する場合は、**lspci** コマンドを実行します。**grep** を使用して、**lspci** の出力を絞り込むことができます。

   この例では、以下の出力で強調表示されているイーサネットコントローラーを使用します。

   ```
   # lspci | grep Ethernet
   00:19.0 Ethernet controller: Intel Corporation 82567LM-2 Gigabit Network Connection
   01:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
   01:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
   ```

   このイーサネットコントローラーは、短い識別子 **00:19.0** で表示されます。この PCI デバイスを仮想マシンに割り当てるには、**virsh** が使用する完全な ID を確認する必要があります。

   これを行うには、**virsh nodedev-list** コマンドを使用して、ホストマシンに接続されている特定タイプ (**pci**) のデバイスをすべて一覧表示します。次に、使用するデバイスの短い識別子にマップする文字列の出力を調べます。

この例では、短い識別子 **00:19.0** を使用して、イーサネットコントローラーにマップする文字列を示しています。この例では、**:** と **.** が、完全な識別子では、文字はアンダースコアに置き換えられます。

```
# virsh nodedev-list --cap pci
pci_0000_00_00_0
pci_0000_00_01_0
pci_0000_00_03_0
pci_0000_00_07_0
pci_0000_00_10_0
pci_0000_00_10_1
pci_0000_00_14_0
pci_0000_00_14_1
pci_0000_00_14_2
pci_0000_00_14_3
pci_0000_00_19_0
pci_0000_00_1a_0
pci_0000_00_1a_1
pci_0000_00_1a_2
pci_0000_00_1a_7
pci_0000_00_1b_0
pci_0000_00_1c_0
pci_0000_00_1c_1
pci_0000_00_1c_4
pci_0000_00_1d_0
pci_0000_00_1d_1
pci_0000_00_1d_2
pci_0000_00_1d_7
pci_0000_00_1e_0
pci_0000_00_1f_0
pci_0000_00_1f_2
pci_0000_00_1f_3
pci_0000_01_00_0
pci_0000_01_00_1
pci_0000_02_00_0
pci_0000_02_00_1
pci_0000_06_00_0
pci_0000_07_02_0
pci_0000_07_03_0
```

使用するデバイスにマップする PCI デバイス番号を記録します。これは別の手順で必要になります。

2. **デバイス情報の確認**
ドメイン、バス、および機能の情報は、**virsh nodedev-dumpxml** コマンドの出力から取得できます。

```
virsh nodedev-dumpxml pci_0000_00_19_0
<device>
  <name>pci_0000_00_19_0</name>
  <parent>computer</parent>
  <driver>
    <name>e1000e</name>
  </driver>
  <capability type='pci'>
```

```
    <domain>0</domain>
    <bus>0</bus>
    <slot>25</slot>
    <function>0</function>
    <product id='0x1502'>82579LM Gigabit Network Connection</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <iommuGroup number='7'>
      <address domain='0x0000' bus='0x00' slot='0x19' function='0x0'/>
    </iommuGroup>
  </capability>
</device>
```

注記

IOMMU グループは、IOMMU からの視認性とデバイスの分離に基づいて決定されます。各 IOMMU グループには、1 つ以上のデバイスを含めることができます。複数のデバイスが存在する場合は、グループ内のすべてのデバイスがゲストに割り当てられるため、IOMMU グループ内のすべてのエンドポイントが要求されます。これは、追加のエンドポイントをゲストに割り当てるか、**virsh nodedev-detach** を使用してホストドライバーから外すことで実行できます。1 つのグループに含まれるデバイスが複数のゲストに分割されたり、ホストとゲストが分割されたりすることはありません。PCIe root ポート、スイッチポート、ブリッジなどのエンドポイント以外のデバイスは、ホストドライバーから分離しないでください。また、エンドポイントの割り当てに影響を及ぼしません。

IOMMU グループ内のデバイスは、**virsh nodedev-dumpxml** 出力の IOMMU グループセクションを使用して決定できます。グループの各メンバーは、別のアドレスフィールドで提供されます。この情報は、sysfs で以下を使用して取得することもできます。

```
$ ls /sys/bus/pci/devices/0000:01:00.0/iommu_group/devices/
```

この出力の例を以下に示します。

```
0000:01:00.0  0000:01:00.1
```

ゲストに 0000.01.00.0 のみを割り当てるには、ゲストを起動する前に、使用されていないエンドポイントをホストから分離する必要があります。

```
$ virsh nodedev-detach pci_0000_01_00_1
```

3. **必要な設定の詳細を決定する**
   設定ファイルに必要な値は、**virsh nodedev-dumpxml pci_0000_00_19_0** コマンドの出力を参照してください。

   この例のデバイスは、bus = 0、slot = 25、および function = 0 の値を持ちます。10 進数の設定では、この 3 つの値が使用されます。

```
bus='0'
slot='25'
function='0'
```

4. **設定の詳細の追加**

virsh edit を実行し、仮想マシン名を指定し、**<source>** セクションにデバイスエントリーを追加して、PCI デバイスをゲスト仮想マシンに割り当てます。

```
# virsh edit guest1-rhel6-64
<hostdev mode='subsystem' type='pci' managed='yes'>
  <source>
    <address domain='0' bus='0' slot='25' function='0'/>
  </source>
</hostdev>
```

または、**virsh attach-device** を実行して、仮想マシンの名前とゲストの XML ファイルを指定します。

```
virsh attach-device guest1-rhel6-64 file.xml
```

5. **仮想マシンの起動**

```
# virsh start guest1-rhel6-64
```

これにより、PCI デバイスが仮想マシンに正常に割り当てられ、ゲストオペレーティングシステムにアクセスできるようになります。

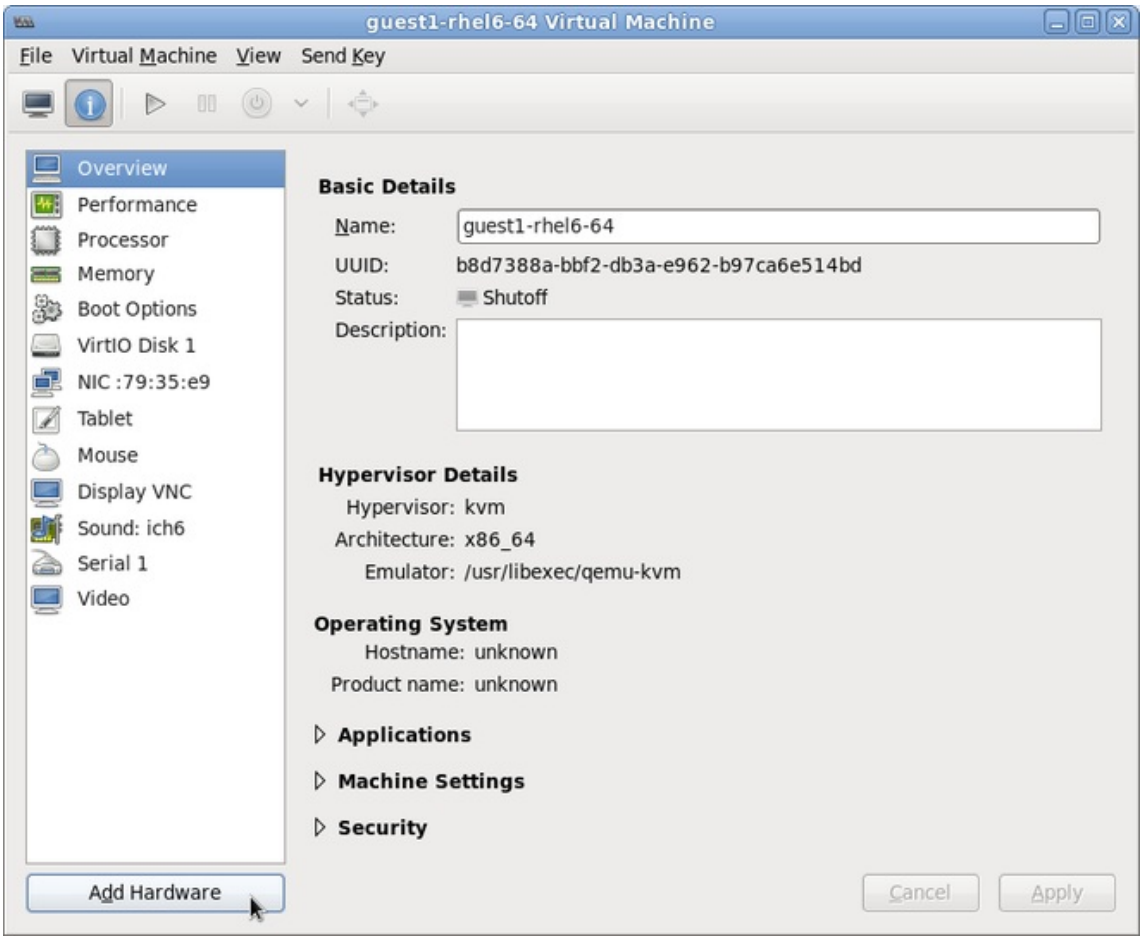
9.1.2. virt-manager を使用した PCI デバイスの割り当て

PCI デバイスは、グラフィカル**virt-manager**ツールを使用してゲスト仮想マシンに追加できます。次の手順では、ギガビットイーサネットコントローラーをゲスト仮想マシンに追加します。

手順9.4 virt-manager を使用した PCI デバイスのゲスト仮想マシンへの割り当て

1. **ハードウェア設定を開く**
   ゲスト仮想マシンを開き、**Add Hardware** をクリックして、仮想マシンに新しいデバイスを追加します。

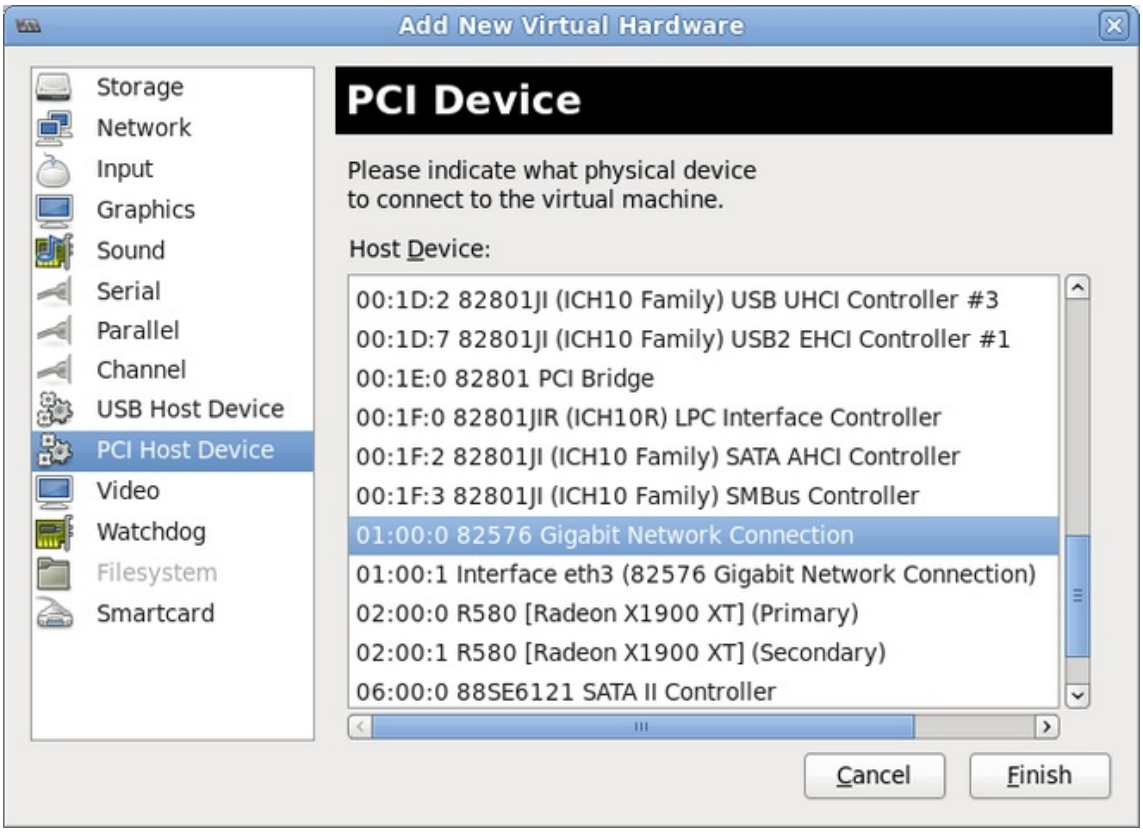
図9.1 仮想マシンのハードウェア情報ウィンドウ



2. **PCI デバイスの選択**
   左側の **Hardware** 一覧から **PCI Host Device** を選択します。

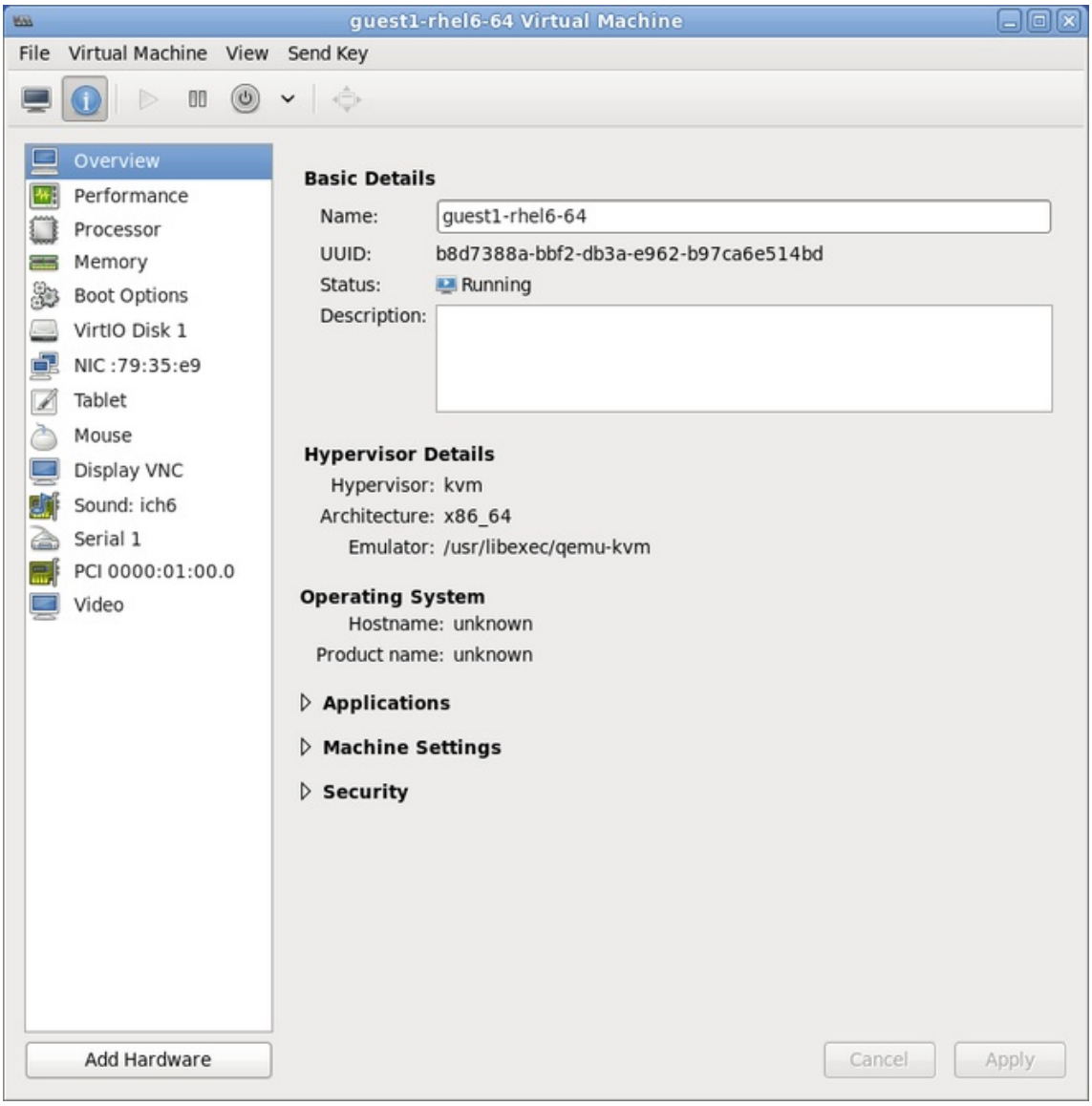
   未使用の PCI デバイスを選択します。別のゲストが使用している PCI デバイスを選択すると、エラーが発生する可能性があります。この例では、予備の 82576 ネットワークデバイスが使用されています。**Finish** を選択して設定を完了します。

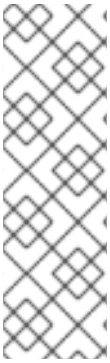
図9.2 Add new virtual hardware ウィザード



3. **新しいデバイスの追加**
   セットアップが完了し、ゲスト仮想マシンが PCI デバイスに直接アクセスできるようになりました。

図9.3 仮想マシンのハードウェア情報ウィンドウ





注記

デバイスの割り当てに失敗すると、同じ IOMMU グループに、ホストに依然として接続されているその他のエンドポイントが存在する可能性があります。virt-manager を使用してグループ情報を取得する方法はありませんが、virsh コマンドを使用すると、IOMMU グループの境界を分析したり、必要に応じてデバイスを隔離したりできます。

IOMMU グループの詳細と、virsh を使用してエンドポイントデバイスの割り当てを解除する方法は、「virsh を使用した PCI デバイスの割り当て」 の 注記 を参照してください。

9.1.3. virt-install を使用した PCI デバイスの割り当て

virt-install を使用して PCI デバイスを割り当てるには、 ***--host-device*** パラメーターを使用します。

手順9.5 virt-install を使用した、仮想マシンへの PCI デバイスの割り当て

1. **デバイスの識別**
   ゲスト仮想マシンにデバイス割り当てに指定されている PCI デバイスを特定します。

```
# lspci | grep Ethernet
00:19.0 Ethernet controller: Intel Corporation 82567LM-2 Gigabit Network Connection
01:00.0 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
01:00.1 Ethernet controller: Intel Corporation 82576 Gigabit Network Connection (rev 01)
```

virsh nodedev-list コマンドは、システムに接続されているすべてのデバイスの一覧を表示し、各 PCI デバイスを文字列で識別します。出力を PCI デバイスのみに制限するには、次のコマンドを実行します。

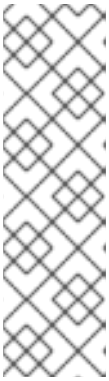
```
# virsh nodedev-list --cap pci
pci_0000_00_00_0
pci_0000_00_01_0
pci_0000_00_03_0
pci_0000_00_07_0
pci_0000_00_10_0
pci_0000_00_10_1
pci_0000_00_14_0
pci_0000_00_14_1
pci_0000_00_14_2
pci_0000_00_14_3
pci_0000_00_19_0
pci_0000_00_1a_0
pci_0000_00_1a_1
pci_0000_00_1a_2
pci_0000_00_1a_7
pci_0000_00_1b_0
pci_0000_00_1c_0
pci_0000_00_1c_1
pci_0000_00_1c_4
pci_0000_00_1d_0
pci_0000_00_1d_1
pci_0000_00_1d_2
pci_0000_00_1d_7
pci_0000_00_1e_0
pci_0000_00_1f_0
pci_0000_00_1f_2
pci_0000_00_1f_3
pci_0000_01_00_0
pci_0000_01_00_1
pci_0000_02_00_0
pci_0000_02_00_1
pci_0000_06_00_0
pci_0000_07_02_0
pci_0000_07_03_0
```

PCI デバイス番号を記録します。この番号は他の手順で確認する必要があります。

ドメイン、バス、および機能の情報は、**virsh nodedev-dumpxml** コマンドの出力から取得できます。

```
# virsh nodedev-dumpxml pci_0000_01_00_0
<device>
  <name>pci_0000_01_00_0</name>
  <parent>pci_0000_00_01_0</parent>
  <driver>
```

```
    <name>igb</name>
  </driver>
  <capability type='pci'>
    <domain>0</domain>
    <bus>1</bus>
    <slot>0</slot>
    <function>0</function>
    <product id='0x10c9'>82576 Gigabit Network Connection</product>
    <vendor id='0x8086'>Intel Corporation</vendor>
    <iommuGroup number='7'>
      <address domain='0x0000' bus='0x00' slot='0x19' function='0x0'/>
    </iommuGroup>
  </capability>
</device>
```



注記

IOMMU グループに複数のエンドポイントがあり、そのすべてがゲストに割り当てられていない場合は、ゲストを起動する前に次のコマンドを実行して、ホストからその他のエンドポイントの割り当てを手動で解除する必要があります。

```
$ virsh nodedev-detach pci_0000_00_19_1
```

IOMMU グループの詳細は、「virsh を使用した PCI デバイスの割り当て」 の 注記 を参照してください。

2. **デバイスの追加**
   virshnodedev コマンドから出力された PCI 識別子を ***--host-device*** パラメーターの値として使用します。

```
virt-install \
--name=guest1-rhel6-64 \
--disk path=/var/lib/libvirt/images/guest1-rhel6-64.img,size=8 \
--nonsparse --graphics spice \
--vcpus=2 --ram=2048 \
--location=http://example1.com/installation_tree/RHEL6.0-Server-x86_64/os \
--nonetworks \
--os-type=linux \
--os-variant=rhel6
--host-device=pci_0000_01_00_0
```

3. **インストールを完了する**
   ゲストのインストールを完了します。PCI デバイスはゲストに接続する必要があります。

9.1.4. 割り当てられた PCI デバイスの取り外し

ホストの PCI デバイスがゲストマシンに割り当てられると、ホストがそのデバイスを使用できなくなります。このセクションを読んで、**virsh** または **virt-manager** を使用してデバイスをゲストからデタッチし、ホストで使用できるようにする方法を学習してください。

手順9.6 virsh を使用したゲストからの PCI デバイスの切り離し

1. **デバイスの取り外し**

次のコマンドを使用して、ゲストの XML ファイルから PCI デバイスを削除し、ゲストから PCI デバイスを切り離します。

```
# virsh detach-device name_of_guest file.xml
```

2. **デバイスをホストに再接続します (オプション)。**
デバイスが ***managed*** モードにある場合は、この手順を省略します。デバイスは自動的にホストに戻ります。

デバイスが ***managed*** モードを使用していない場合は、以下のコマンドを使用して PCI デバイスをホストマシンに再接続します。

```
# virsh nodedev-reattach device
```

たとえば、**pci_0000_01_00_0** デバイスーをホストコンピューターに再接続するには、次のコマンドを実行します。

```
virsh nodedev-reattach pci_0000_01_00_0
```

これで、デバイスがホストで使用できるようになります。

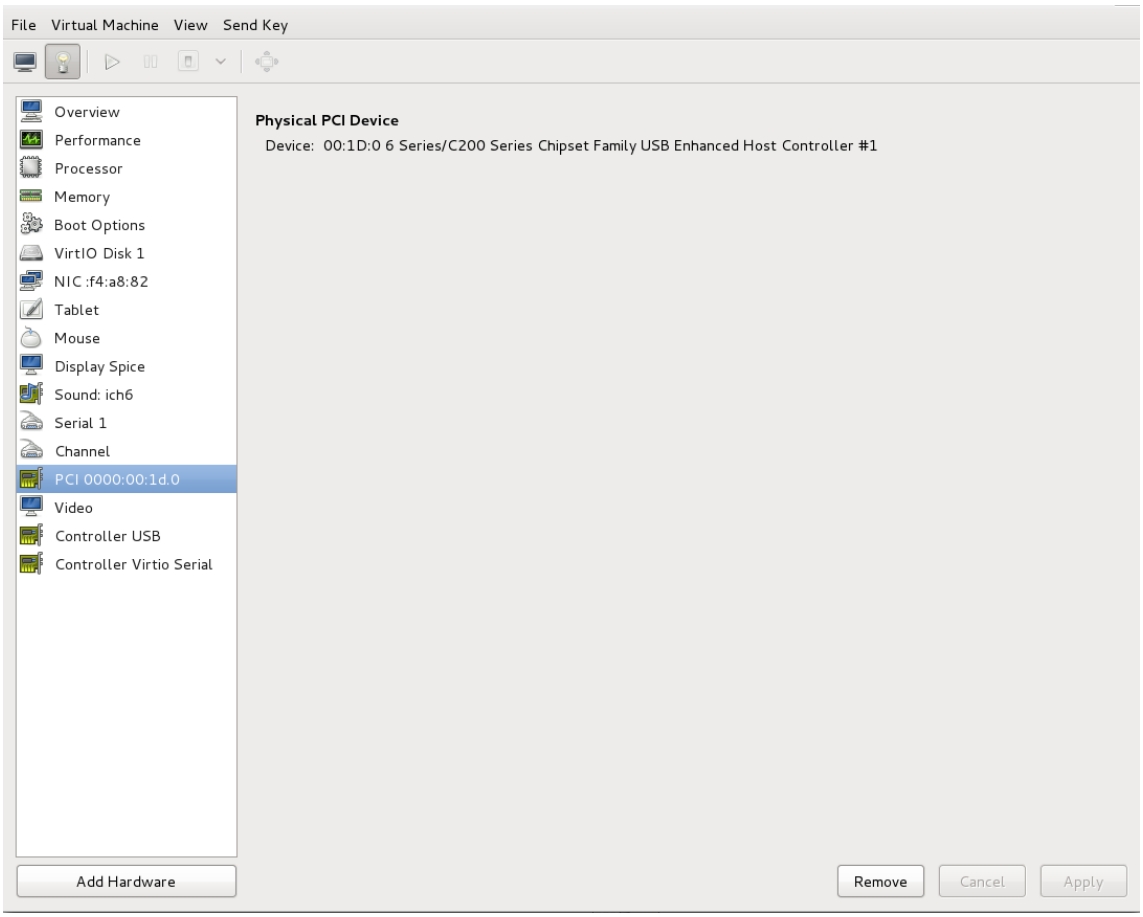
手順9.7 virt-manager を使用したゲストからの PCI デバイスの切り離し

1. **仮想ハードウェアの詳細画面を開きます。**
virt-manager で、デバイスを含む仮想マシンをダブルクリックします。**Show virtual hardware details** ボタンを選択すると、仮想ハードウェアの一覧が表示されます。

図9.4 仮想ハードウェアの詳細ボタン

2. **デバイスを選択して削除する**
左側のパネルにある仮想デバイスの一覧から、取り外す PCI デバイスを選択します。

図9.5 取り外す PCI デバイスの選択



Remove ボタンをクリックして確定します。これで、デバイスがホストで使用できるようになります。

9.1.5. PCI ブリッジの作成

PCI (Peripheral Component Interconnect) ブリッジは、ネットワークカード、モデム、サウンドカードなどのデバイスに接続するために使用されます。物理デバイスと同様に、仮想デバイスも PCI ブリッジに接続できます。以前は、31 個の PCI デバイスしかゲスト仮想マシンに追加できませんでした。現在、31 番目の PCI デバイスを追加すると、PCI ブリッジが 31 番目のスロットに自動的に配置され、追加した PCI デバイスを PCI ブリッジに移動します。各 PCI ブリッジには、31 の追加デバイス用に 31 個のスロットがあり、すべてがブリッジになることができます。この方法では、ゲスト仮想マシンで 900 を超えるデバイスを使用できます。

注記

ゲスト仮想マシンの実行中は、このアクションを実行できません。シャットダウンするゲスト仮想マシンに PCI デバイスを追加する必要があります。

9.1.6. PCI パススルー

<source> 要素で指定される PCI ネットワークデバイスは、ジェネリックデバイスパススルーを使用してゲストに直接割り当てられます。このデバイスの MAC アドレスは、最初にオプションで設定された値に設定され、そのデバイスの MAC アドレスがオプションで指定された *virtualport* 要素を使用して 802.1Qbh 対応スイッチに関連付けられます (**<type='direct'>** ネットワークデバイスの場合は、上記の仮想ポートの例を参照してください)。標準的なシングルポートの PCI イーサネットカードドライバー

設計の制限により、この方法で割り当てることができるのは Single Root I/O Virtualization (SR-IOV) virtual function (VF) デバイスのみとなります。標準的なシングルポートの PCI または PCIe イーサネットカードをゲストに割り当てる場合は、従来の **<hostdev>** デバイス定義を使用します。

従来の/レガシー KVM デバイス割り当てではなく VFIO デバイス割り当てを使用するために (VFIO は UEFI セキュアブートと互換性のあるデバイス割り当ての新しい方法です)、**<type='hostdev'>** インターフェイスは name 属性を持つオプションのドライバーサブ要素を持つことができます vfio に設定します。従来の KVM デバイス割り当てを使用するには、名前を kvm に設定できます (または、現在、**<driver ='kvm'>** がデフォルトであるため、単に **<driver>** 要素を省略します)。

注記

ネットワークデバイスにおけるインテリジェントパススルーは、標準の **<hostdev>** デバイスの機能と非常に似ています。相違点は、パススルーデバイスに MAC アドレスと**<virtualport>**を指定できることです。これらの機能が必要ない場合、SR-IOV をサポートしない標準のシングルポート PCI、PCIe、または USB ネットワークカードを使用している場合 (したがって、ゲストドメインに割り当てられた後、リセット中に設定された MAC アドレスが失われます)、または 0.9.11 より前のバージョンの libvirt を使用している場合は、**<interface type='hostdev'/>** の代わりに、標準の **<hostdev>** を使用してデバイスをゲストに割り当てる必要があります。

図9.6 PCI デバイスの割り当ての XML 例

```
 <devices>
<interface type='hostdev'>
 <driver name='vfio'/>
 <source>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
 </source>
 <mac address='52:54:00:6d:90:02'>
 <virtualport type='802.1Qbh'>
  <parameters profileid='finance'/>
 </virtualport>
</interface>
</devices>
```

9.1.7. SR-IOV デバイスを使用した PCI 割り当て (パススルー) の設定

このセクションは、SR-IOV デバイスのみを対象としています。SR-IOV ネットワークカードは、複数の *Virtual Functions* を提供します。各 VF は、PCI デバイスの割り当てを使用してゲスト仮想マシンに個別に割り当てることができます。割り当てが完了すると、各デバイスは完全な物理ネットワークデバイスとして機能します。これにより、多くのゲスト仮想マシンは、ホスト物理マシン上の単一のスロットのみを使用しながら、直接 PCI デバイス割り当てのパフォーマンス上の利点を得ることができます。

これらの VF は従来の方法でゲスト仮想マシンに **<hostdev>** 要素を使用して割り当てることができますが、SR-IOV VF ネットワークデバイスには永続的な一意の MAC アドレスがないため、ホストの物理マシンを再起動するたびにゲスト仮想マシンのネットワーク設定を再設定する必要があります。これを解決するには、VF をホスト物理マシンに割り当てる前に MAC アドレスを設定する必要があり、ゲスト仮想マシンが起動するたびにこれを設定する必要があります。この MAC アドレスやその他のオプションを割り当てるには、手順9.8「SR-IOV で PCI デバイスを割り当てる MAC アドレス、vLAN、および仮想ポートの設定」の手順を参照してください。

手順9.8 SR-IOV で PCI デバイスを割り当てる MAC アドレス、vLAN、および仮想ポートの設定

<mac>、**<vlan>**、および **<virtualport>** 要素は **<hostdev>** の有効な子ではないため、**<hostdev>** 要素は MAC アドレス割り当て、vLAN タグ ID 割り当て、仮想ポート割り当てなどの機能固有の項目には使用できないことに注意してください。これらは **<インターフェイス>** で有効であるため、新しいインターフェイスタイプのサポートが追加されました (**<interface type='hostdev'>**)。この新しいインターフェイスデバイスタイプは、**<インターフェイス>** と **<hostdev>** のハイブリッドとして動作します。そのため、libvirt は、ゲスト仮想マシンに PCI デバイスを割り当てる前に、ゲスト仮想マシンの XML 設定ファイルに示されているネットワーク固有のハードウェア/スイッチを初期化します (MAC アドレスの設定、vLAN タグの設定、802.1Qbh スイッチへの関連付けなど)。vLAN タグの設定に関する詳細は、「vLAN タグの設定」 を参照してください。

1. **ゲスト仮想マシンをシャットダウンします。**
   virsh shutdown コマンドの使用 (「ゲスト仮想マシンのシャットダウン」 を参照)、*guestVM* という名前のゲスト仮想マシンをシャットダウンします。

   ```
   # virsh shutdown guestVM
   ```

2. **情報の収集**
   <interface type='hostdev'> を使用するには、SR-IOV 対応のネットワークカード、Intel VT-d 拡張機能または AMD IOMMU 拡張機能に対応するホストの物理マシンハードウェアが必要で、割り当てる VF の PCI アドレスを把握している必要があります。

3. **編集する XML ファイルを開く**
   # **virsh save-image-edit** コマンドを実行し、XML ファイルを編集用に開きます (詳細は 「ドメイン XML 設定ファイルの編集」 を参照してください)。ゲスト仮想マシンを以前の実行状態に復元する必要があるため、この場合は **--running** が使用されます。この例の設定ファイルの名前は *guestVM.xml* です。ゲスト仮想マシンの名前は *guestVM* です。

   ```
   # virsh save-image-edit guestVM.xml --running
   ```

   *guestVM.xml*がデフォルトのエディターで開きます。

4. **XML ファイルの編集**
   設定ファイル (*guestVM.xml*) を更新して、以下のような**<devices>** エントリーが表示されるようにします。

図9.7 hostdev インターフェイスタイプのサンプルドメイン XML

```
<devices>
  ...
  <interface type='hostdev' managed='yes'>
    <source>
      <address type='pci' domain='0x0' bus='0x00' slot='0x07' function='0x0'/> <!--these values
can be decimal as well-->
    </source>
    <mac address='52:54:00:6d:90:02'/>                    <!--sets the mac address-->
    <virtualport type='802.1Qbh'>                         <!--sets the virtual port for the
802.1Qbh switch-->
      <parameters profileid='finance'/>
    </virtualport>
    <vlan>                                                <!--sets the vlan tag-->
     <tag id='42'/>
    </vlan>
  </interface>
  ...
</devices>
```

MAC アドレスを指定しないと、その他のタイプのインターフェイスデバイスと同様に、MAC アドレスが自動的に生成されることに注意してください。また、**<virtualport>** 要素は、802.11Qgh ハードウェアスイッチ(802.11Qbg)に接続する場合にのみ使用されます。"VEPA"）スイッチは現在サポートされていません。

5. **ゲスト仮想マシンの再起動**
   virsh start コマンドを実行して、最初の手順でシャットダウンしたゲスト仮想マシンを再起動します (例では、ゲスト仮想マシンのドメイン名として guestVM を使用します)。詳細は、「定義済みドメインの開始」 を参照してください。

   ```
   # virsh start guestVM
   ```

   ゲスト仮想マシンは起動時に、設定された MAC アドレスを持つ、物理ホストマシンのアダプターにより提供されたネットワークデバイスを認識します。この MAC アドレスは、ゲスト仮想マシンやホストの物理マシンを再起動しても変更されません。

9.1.8. SR-IOV 仮想機能のプールからの PCI デバイス割り当ての設定

特定の *Virtual Functions* (VF) の PCI アドレスをゲストの設定にハードコーディングすることには、2 つの重大な制限があります。

- 指定の VF は、ゲスト仮想マシンが起動したときにいつでも利用できる必要があります。つまり、管理者は、各 VF を 1 つのゲスト仮想マシンに永続的に割り当てる必要があります (または、各ゲスト仮想マシンの設定ファイルを変更して、ゲスト仮想マシンが起動するたびに現在使用されていない VF の PCI アドレスを指定します)。
- ゲスト仮想マシンを別のホスト物理マシンに移動する場合は、そのホスト物理マシンのハードウェアが PCI バス上の同じ場所にある必要があります (または、ゲスト仮想マシンの設定を起動前に変更する必要があります)。

SR-IOV デバイスのすべての VF を含むデバイスプールで libvirt ネットワークを作成することで、この両方の問題を回避できます。完了したら、ゲスト仮想マシンがこのネットワークを参照するように設定

します。ゲストを起動するたびに、プールから 1 つの VF が割り当てられ、ゲスト仮想マシンに割り当てられます。ゲスト仮想マシンが停止すると、VF は別のゲスト仮想マシンが使用するためにプールに戻されます。

手順9.9 デバイスプールの作成

1. **ゲスト仮想マシンをシャットダウンします。**
   virsh shutdown コマンドの使用 (「ゲスト仮想マシンのシャットダウン、再起動、および強制シャットダウン」 を参照)、*guestVM* という名前のゲスト仮想マシンをシャットダウンします。

   ```
   # virsh shutdown guestVM
   ```

2. **設定ファイルの作成**
   任意のエディターを使用して、**/tmp** ディレクトリーに XML ファイル (名前は *passthrough.xml* など) を作成します。**pf dev='eth3'** を、ご使用の SR-IOV デバイスの PF の netdev 名に置き換えてください。

   以下は、ホスト物理マシンの "eth3' にある *physical function (PF)* を持つ SR-IOV アダプターのすべての VF のプールを使用できるようにするネットワーク定義の例です。

   図9.8 サンプルのネットワーク定義ドメイン XML

   ```
   <network>
     <name>passthrough</name>                    <!--This is the name of the file you created-->
     <forward mode='hostdev' managed='yes'>
       <pf dev='myNetDevName'/>                  <!--Use the netdev name of your SR-IOV devices PF here-->
     </forward>
   </network>
   ```

3. **新しい XML ファイルの読み込み**
   /tmp/passthrough.xml を、前の手順で作成した XML ファイルの名前と場所に置き換え、次のコマンドを実行します。

   ```
   # virsh net-define /tmp/passthrough.xml
   ```

4. **ゲストの再起動**
   以下の *passthrough.xml* を、前の手順で作成した XML ファイルの名前に置き換えます。

   ```
   # virsh net-autostart passthrough # virsh net-start passthrough
   ```

5. **ゲスト仮想マシンの再起動**
   virsh start コマンドを実行して、最初の手順でシャットダウンしたゲスト仮想マシンを再起動します (例では、ゲスト仮想マシンのドメイン名として guestVM を使用します)。詳細は、「定義済みドメインの開始」 を参照してください。

   ```
   # virsh start guestVM
   ```

6. **デバイスのパススルーの開始**
表示されているデバイスは 1 つだけですが、libvirt は、ゲスト仮想マシンが次のようなドメイン XML のインターフェイス定義で初めて起動されたときに、その PF に関連付けられているすべての VF のリストを自動的に取得します。

図9.9 インターフェイスネットワーク定義のサンプルドメイン XML

```
<interface type='network'>
  <source network='passthrough'>
</interface>
```

7. **検証**
ネットワークを使用する最初のゲストを起動したら、**virsh net-dumpxml *passthrough*** コマンドを実行し、これを確認できます。以下のような出力が得られます。

図9.10 XML ダンプファイルの*passthrough*コンテンツ

```
<network connections='1'>
  <name>passthrough</name>
  <uuid>a6b49429-d353-d7ad-3185-4451cc786437</uuid>
  <forward mode='hostdev' managed='yes'>
    <pf dev='eth3'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x1'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x3'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x5'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x10' function='0x7'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x1'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x3'/>
    <address type='pci' domain='0x0000' bus='0x02' slot='0x11' function='0x5'/>
  </forward>
</network>
```

9.2. USB デバイス

本セクションでは、USB デバイスを扱うために必要なコマンドについて説明します。

9.2.1. ゲスト仮想マシンへの USB デバイスの割り当て

Web comeras、カードリーダー、キーボード、マウスなどのほとんどのデバイスは、USB ポートとケーブルを使用してコンピューターに接続されます。このようなデバイスをゲスト仮想マシンに渡すには、以下の 2 つの方法があります。

- USB パススルーを使用する - これにより、デバイスが、ゲスト仮想マシンをホストしているホストの物理マシンに物理的に接続されます。この場合は SPICE は必要ありません。ホストの USB デバイスは、コマンドラインまたは **virt-manager** を使用してゲストに渡すことができます。**virt manager** の方向については、「ゲスト仮想マシンへの USB デバイスの接続」 を参照してください。

注記

virt-manager は、ホットプラグまたはホットアンプラグデバイスに使用しないでください。USB デバイスをホットプラグまたはホットアンプラグする場合は、手順14.1「ゲスト仮想マシンが使用するホットプラグ USB デバイス」 を参照してください。

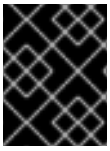
- USB リダイレクトの使用 - USB リダイレクトは、データセンターで実行されているホスト物理マシンがある場合に最適です。ユーザーは、ローカルマシンまたはシンクライアントからゲスト仮想マシンに接続します。このローカルマシンには SPICE クライアントがあります。ユーザーは、任意の USB デバイスをシンクライアントに接続できます。SPICE クライアントは、デバイスをデータセンターのホスト物理マシンにリダイレクトするため、シンクライアントで実行しているゲスト仮想マシンで使用できます。virt-manager を使用した USB リダイレクトの手順については、「ゲスト仮想マシンへの USB デバイスの接続」 を参照してください。TCP プロトコルを使用した USB リダイレクトはできないことに注意してください (BZ#1085318 を参照)。

9.2.2. USB デバイスリダイレクトの制限の設定

リダイレクトから特定のデバイスをフィルターリングするには、フィルタープロパティーを ***-device usb-redir*** に渡します。filter プロパティーは、フィルタールールで設定される文字列を取ります。ルールの形式は以下のとおりです。

```
<class>:<vendor>:<product>:<version>:<allow>
```

-1 の値を使用して、特定のフィールドに任意の値を受け入れるように指定します。同じコマンドラインで、| を区切り文字として使用し、複数のルールを使用できます。



重要

デバイスがどのルールフィルターにも一致しない場合、そのデバイスのリダイレクトは許可されません。

例9.1 Windows ゲスト仮想マシンを使用したリダイレクトを制限する例

1. Windows 7 ゲスト仮想マシンを準備します。
2. 以下のコード抜粋をゲスト仮想マシンのドメイン xml ファイルに追加します。

```
<redirdev bus='usb' type='spicevmc'>
  <alias name='redir0'/>
  <address type='usb' bus='0' port='3'/>
</redirdev>
<redirfilter>
  <usbdev class='0x08' vendor='0x1234' product='0xBEEF' version='2.0' allow='yes'/>
  <usbdev class='-1' vendor='-1' product='-1' version='-1' allow='no'/>
</redirfilter>
```

3. ゲスト仮想マシンを起動し、次のコマンドを実行して設定の変更を確認します。

```
#ps -ef | grep $guest_name
```

```
-device usb-redir,chardev=charredir0,id=redir0,/
filter=0x08:0x1234:0xBEEF:0x0200:1|-1:-1:-1:-1:0,bus=usb.0,port=3
```

4. USB デバイスをホストの物理マシンに接続し、**virt-manager** を使用してゲスト仮想マシンに接続します。

5. メニューで **Redirect USB Service** をクリックすると、Some USB devices are blocked by host policy というメッセージが表示されます。**OK** をクリックして確定し、続行します。

   フィルターが有効になります。

6. フィルターが正しくキャプチャーされていることを確認するには、USB デバイスのベンダーと製品を確認してから、USB リダイレクトを可能にするために、テストの仮想マシンのドメイン XML で次の変更を行います。

```
<redirfilter>
  <usbdev class='0x08' vendor='0x0951' product='0x1625' version='2.0' allow='yes'/>
  <usbdev allow='no'/>
 </redirfilter>
```

7. ゲスト仮想マシンを再起動してから、**virt-viewer** を使用してゲスト仮想マシンに接続します。これで、USB デバイスはトラフィックをゲスト仮想マシンにリダイレクトするようになります。

9.3. デバイスコントローラーの設定

ゲスト仮想マシンのアーキテクチャーによっては、一部のデバイスバスは複数回表示され、仮想デバイスのグループは仮想コントローラーに関連付けられています。通常、libvirt は、明示的な XML マークアップを必要とせずに、自動的にこのようなコントローラーを推測できますが、場合によっては仮想コントローラー要素を明示的に設定する方が良い場合があります。

図9.11 仮想コントローラーのドメイン XML の例

```
...
<devices>
  <controller type='ide' index='0'/>
  <controller type='virtio-serial' index='0' ports='16' vectors='4'/>
  <controller type='virtio-serial' index='1'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
  </controller>
  ...
</devices>
...
```

各コントローラーには必須の属性 **<controller type>** があり、これは次のいずれかになります。

- ide
- fdc
- scsi

- sata
- usb
- ccid
- virtio-serial
- pci

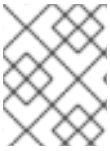
<controller> 要素には必須の属性 **<controller index>** があります。これは、バスコントローラーが発生した順序を表す 10 進数の整数です (**<address>** 要素のコントローラー属性で使用されます)。**<controller type ='virtio-serial'>** には、追加のオプション属性 (**ports** および **vectors**) が 2 つあります。これは、コントローラーを介して接続できるデバイスの数を制御します。Red Hat Enterprise Linux 6 は、デバイスあたり 32 を超えるベクターの使用をサポートしていないことに注意してください。よりベクトルを使用すると、ゲスト仮想マシンの移行でエラーが発生します。

<controller type ='scsi'> の場合、以下の値を持つことができる任意の属性**model** モデルがあります。

- auto
- buslogic
- ibmvscsi
- lsilogic
- lsisas1068
- lsisas1078
- virtio-scsi
- vmpvscsi

<controller type ='usb'> の場合、以下の値を持つことができる任意の属性**model** モデルがあります。

- piix3-uhci
- piix4-uhci
- ehci
- ich9-ehci1
- ich9-uhci1
- ich9-uhci2
- ich9-uhci3
- vt82c686b-uhci
- pci-ohci
- nec-xhci



注記

ゲスト仮想マシンに対して USB バスを明示的に無効にする必要がある場合は、**<model='none'>** を使用できます。 .

コントローラー自体が PCI バスまたは USB バスにある場合は、オプションのサブ要素 **<address>** は、「デバイスのアドレスの設定」 に示したセマンティクスを使用して、コントローラーとマスターバスの正確な関係を指定できます。

オプションの sub-element **<driver>** は、ドライバー固有のオプションを指定できます。現在、コントローラーのキューの数を指定する属性キューのみをサポートしています。パフォーマンスを最適化するには、vCPU の数に一致する値を指定することが推奨されます。

USB コンパニオンコントローラーには、コンパニオンとマスターコントローラーの完全なリレーションを指定するためのオプションのサブ要素 **<master>** があります。コンパニオンコントローラーはマスターと同じバスにあるため、コンパニオン **index** は等しい値になります。

使用できる XML の例を以下に示します。

図9.12 USB コントローラーのドメイン XML の例

```
  ...
<devices>
  <controller type='usb' index='0' model='ich9-ehci1'>
    <address type='pci' domain='0' bus='0' slot='4' function='7'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci1'>
    <master startport='0'/>
    <address type='pci' domain='0' bus='0' slot='4' function='0' multifunction='on'/>
  </controller>
  ...
</devices>
...
```

PCI コントローラーには、以下の値を持つオプションの **model** 属性があります。

- pci-root
- pcie-root
- pci-bridge
- dmi-to-pci-bridge

ルートコントローラー (**pci-root** および **pcie-root**) には、64 ビット PCI ホールの大きさ (キロバイト単位、または **pcihole64** の **unit** 属性で指定された単位) を指定するオプションの **pcihole64** 要素があります。一部のゲスト仮想マシン (Windows Server 2003) は、**unit** が無効になっていない限り (0 **unit='0'** に設定)、クラッシュを引き起こす可能性があります。

暗黙的な PCI バスを提供するマシンタイプの場合、**index='0'** を使用する pci-root コントローラーは自動追加され、PCI デバイスを使用する必要があります。pci-root にはアドレスがありません。PCI ブリッジは、**model='pci-root'** が提供する 1 つのバスに収まらないデバイスが多すぎる場合、または 0 を超える PCI バス番号が指定されている場合に自動追加されます。PCI ブリッジは手動で指定することも

できますが、そのアドレスには、指定されている PCI コントローラーが提供する PCI バスのみが表示されるようにしてください。PCI コントローラーインデックスにギャップがあると、設定が無効になる場合があります。以下の XML サンプルは、**<devices>** セクションに追加できます。

図9.13 PCI ブリッジのドメイン XML の例

```
...
<devices>
  <controller type='pci' index='0' model='pci-root'/>
  <controller type='pci' index='1' model='pci-bridge'>
    <address type='pci' domain='0' bus='0' slot='5' function='0' multifunction='off'/>
  </controller>
</devices>
...
```

暗黙的な PCI Express (PCIe) バスを提供するマシンタイプ (Q35 チップセットに基づくマシンタイプなど) の場合、**index='0'** を使用する pcie-root コントローラーはドメインの設定に自動的に追加されます。pcie-root にはアドレスはありませんが、31 スロット (1-31 の番号) を提供し、PCIe デバイスの接続にのみ使用できます。pcie-root コントローラーを持つシステムで標準的な PCI デバイスを接続するには、**model='dmi-to-pci-bridge'** を持つ pci コントローラーが自動的に追加されます。dmi-to-pci-bridge コントローラーは、(pcie-root が提供するように) PCIe スロットに接続し、それ自体で 31 個の標準的な PCI スロットを提供します (これはホットプラグできません)。ゲストシステムにホットプラグ可能な PCI スロットを確保するために、pci-bridge コントローラーも自動的に作成され、自動作成された dmi-to-pci-bridge コントローラーのスロットの 1 つに接続されます。PCI アドレスが libvirt により自動決定されるすべてのゲストデバイスは、この pci-bridge デバイスに配置されます。

図9.14 PCIe (PCI express) のようなドメイン XML

```
  ...
<devices>
  <controller type='pci' index='0' model='pcie-root'/>
  <controller type='pci' index='1' model='dmi-to-pci-bridge'>
    <address type='pci' domain='0' bus='0' slot='0xe' function='0'/>
  </controller>
  <controller type='pci' index='2' model='pci-bridge'>
    <address type='pci' domain='0' bus='1' slot='1' function='0'/>
  </controller>
</devices>
...
```

9.4. デバイスのアドレスの設定

多くのデバイスには、ゲスト仮想マシンに提示される仮想バス上のデバイスの場所を説明するのに使用されるオプションの **<address>** サブ要素があります。入力でアドレス (またはアドレス内の任意の属性) が省略された場合 libvirt は適切なアドレスを生成します。レイアウトをさらに制御する必要がある場合は明示的なアドレスが必要になります。**<address>** 要素を含むドメイン XML デバイスの例は、図 9.6「PCI デバイスの割り当ての XML 例」 を参照してください。

各アドレスには、デバイスが稼働しているバスを説明する必須の属性 **type** があります。特定のデバイスに使用するアドレスの選択は、デバイスやゲスト仮想マシンのアーキテクチャーによって一部が制約されます。たとえば、**<disk>** デバイスは **type='drive'** を使用し、**<console>** デバイスは i686 または

x86_64 ゲスト仮想マシンアーキテクチャー で **type='pci'** を使用します。各アドレスタイプには、表に記載されているように、バス上のどこにデバイスを配置するかを制御するオプションの属性がさらにあります。

表9.1 サポートされているデバイスのアドレスタイプ

アドレスの種類	説明
type='pci'	PCI アドレスには、以下の追加属性があります。 ● ドメイン (2 バイトの 16 進数の整数で、現在 qemu で使用されていません) ● bus (0 から 0xff までの 16 進数の値) ● slot (0x0 から 0x1f までの 16 進数の値) ● 関数 (0 から 7 までの値) ● PCI 制御レジスターの特定のスロット/機能の多機能ビットをオンにする多機能コントロールは、デフォルトではオフに設定されていますが、複数の機能が使用されるスロットの機能 0 ではオンに設定する必要があります。
type='drive'	ドライブアドレスには、以下の追加属性があります。 ● コントローラー (2 桁のコントローラー番号) ● バス (2 桁のバス番号) ● ターゲット (2 桁のバス番号) ● ユニット (バス上の 2 桁のユニット番号)
type='virtio-serial'	各 virtio-serial アドレスには、以下の追加属性があります。 ● コントローラー (2 桁のコントローラー番号) ● バス (2 桁のバス番号) ● スロット (バス内の 2 桁のスロット)
type='ccid'	スマートカード用の CCID アドレスには、以下の追加属性があります。 ● バス (2 桁のバス番号) ● スロット属性 (バス内の 2 桁のスロット)

アドレスの種類	説明
type='usb'	USB アドレスには、以下の追加属性があります。 ● bus (0 から 0xfff までの 16 進数の値) ● ポート (1.2 または 2.1.3.1 などの最大 4 オクテットのドット表記)
type='isa'	ISA アドレスには、以下の追加属性があります。 ● iobase ● irq

9.5. ゲスト仮想マシンでのストレージコントローラーの管理

Red Hat Enterprise Linux 6.4 以降、Red Hat Enterprise Linux6.4 以降を実行しているゲスト仮想マシンに SCSI および virtio-SCSI デバイスを追加することがサポートされています。virtio ディスクとは異なり、SCSI デバイスではゲスト仮想マシンにコントローラーが存在する必要があります。Virtio-SCSI は、SCSI LUN に直接接続する機能を提供し、virtio-blk と比較してスケーラビリティを大幅に向上させます。virtio-SCSI の利点は、28 台のデバイスしか処理できず PCI スロットを使い果たす virtio-blk と比較して、数百台のデバイスを処理できることです。Virtio-SCSI は、次の機能を備えたターゲットデバイスの機能セットを継承できるようになりました。

- virtio-scsi コントローラーを介して仮想ハードドライブまたは CD を接続します。
- QEMU scsi-block デバイスを介してホストからゲストに物理 SCSI デバイスをパススルーします。
- ゲストごとに数百台のデバイスを使用できるようにします。virtio-blk の 28 デバイスの制限からの改善。

本セクションでは、仮想 SCSI コントローラー ("Host Bus Adapter" または HBA とも呼ばれます) を作成し、SCSI ストレージをゲスト仮想マシンに追加するために必要な手順を詳細に説明します。

手順9.10 仮想 SCSI コントローラーの作成

1. ゲスト仮想マシン (**Guest1**) の設定を表示し、以前から存在している SCSI コントローラーを検索します。

   ```
   # virsh dumpxml Guest1 | grep controller.*scsi
   ```

   デバイスコントローラーが存在する場合は、このコマンドにより、次のような行が 1 つ以上出力されます。

   ```
   <controller type='scsi' model='virtio-scsi' index='0'/>
   ```

2. 前の手順でデバイスコントローラーが表示されなかった場合は、以下の手順に従って、新しいファイルにデバイスコントローラーの説明を作成し、仮想マシンに追加します。

   a. 新しいファイルに **<controller>** 要素を書き込んでデバイスコントローラーを作成し、このファイルに XML 拡張子で保存します。**virtio-scsi-controller.xml** など。

```
<controller type='scsi' model='virtio-scsi'/>
```

b. **virtio-scsi-controller.xml** で作成したデバイスコントローラーをゲスト仮想マシン (例: Guest1) に関連付けます。

```
# virsh attach-device --config Guest1 ~/virtio-scsi-controller.xml
```

この例では、**--config** オプションはディスクと同じように動作します。詳細は、手順 13.2「ゲストへの物理ブロックデバイスの追加」 を参照してください。

3. 新しい SCSI ディスクまたは CD-ROM を追加します。新しいディスクは、「 ゲストへのファイルベースストレージの追加」 セクションおよび 「ゲストへのハードドライブおよびその他のブロックデバイスの追加」 セクションの方法を使用して追加できます。SCSI ディスクを作成する場合は、*sd* で始まるターゲットデバイス名を指定します。

```
# virsh attach-disk Guest1 /var/lib/libvirt/images/FileName.img sdb --cache none
```

ゲスト仮想マシンのドライバーのバージョンによっては、実行中のゲスト仮想マシンで新しいディスクがすぐに検出されない場合があります。『Red Hat Enterprise Linux Storage Administration Guide』 の手順に従います。

9.6. 乱数ジェネレーター (RNG) デバイス

virtio-rng は、仮想 RNG (*random number generator*) デバイスであり、RNG データをゲスト仮想マシンのオペレーティングシステムにフィードします。これにより、要求に応じてゲスト仮想マシンに新しいエントロピーを提供します。

RNG の使用は、キーボード、マウス、その他の入力などのデバイスがゲスト仮想マシンで十分なエントロピーを生成するのに十分でない場合に特に役立ちます。virtio-rng デバイスは、Red Hat Enterprise Linux と Windows ゲスト仮想マシンの両方で利用できます。Windows 要件のインストール手順については、注記 を参照してください。特に明記されていない限り、以下の説明は Red Hat Enterprise Linux と Windows の両方のゲスト仮想マシンを対象としています。

Linux ゲスト仮想マシンで **virtio-rng** が有効になっている場合、chardev はゲスト仮想マシンの **/dev/hwrng/** の場所に作成されます。次に、この文字 dev を開いて読み取り、ホストの物理マシンからエントロピーをフェッチできます。ゲスト仮想マシンのアプリケーションが virtio-rng デバイスからのランダム性を透過的に使用することで利益を得るには、**/dev/hwrng/** からの入力をゲスト仮想マシンのカーネルエントロピープールに中継する必要があります。これは、この場所の情報が rgnd デーモン (rng-tools 内に含まれている) と結合されている場合に実行できます。

この結合により、エントロピーがゲスト仮想マシンの **/dev/random** ファイルにルーティングされます。このプロセスは、Red Hat Enterprise Linux 6 ゲスト仮想マシンで手動で行います。

Red Hat Enterprise Linux 6 ゲスト仮想マシンは、以下のコマンドを実行して組み合わせます。

```
# rngd -b -r /dev/hwrng/ -o /dev/random/
```

詳細は、ここで説明するコマンドオプションの説明は、**man rngd** コマンドを実行します。その他の例は、手順9.11「コマンドラインツールでの virtio-rng の実装」 で virtio-rng デバイスを設定します。

注記

Windows ゲスト仮想マシンでは、ドライバー **viorng** をインストールする必要があります。インストールが完了すると、仮想 RNG デバイスは Microsoft が提供する CNG(手動生成)API を使用して機能します。ドライバーがインストールされると、**virtrng** デバイスが RNG プロバイダーのリストに表示されます。

手順9.11 コマンドラインツールでの virtio-rng の実装

1. ゲスト仮想マシンをシャットダウンします。
2. ターミナルウィンドウで、**virsh edit *domain-name*** コマンドを使用して、目的のゲスト仮想マシンの XML ファイルを開きます。
3. **<devices>** 要素を編集して、以下の内容を追加します。

```
...
<devices>
  <rng model='virtio'>
    <rate period="2000" bytes="1234"/>
    <backend model='random'>/dev/random</backend>
        <source mode='bind' service='1234'>
        <source mode='connect' host='192.0.2.1' service='1234'>
    </backend>
  </rng>
</devices>
...
```

第10章 QEMU-IMG および QEMU ゲストエージェント

本章では、ゲスト仮想マシンで qemu-img パッケージを使用するための便利なヒントとヒントを紹介します。QEMU トレースイベントと引数に関する情報を探している場合は、次の場所にある README ファイルを参照してください。**/usr/share/doc/qemu-*/README.systemtap**。

10.1. QEMU-IMG の使用

qemu-img コマンドラインツールは、KVM が使用するさまざまなファイルシステムのフォーマット、修正、および検証に使用されます。**qemu-img** オプションと使用方法を以下に示します。

チェック

ディスクイメージの *filename* で整合性チェックを実行します。

```
#  qemu-img check -f qcow2 --output=qcow2 -r all filename-img.qcow2
```

注記

qcow2 および **vdi** 形式のみが整合性チェックに対応します。

-r を使用すると、チェック中に見つかった不整合を修復しようとしますが、***-r leaks*** クラスターのリークで使用されると修復され、***-r all*** ですべての種類のエラーが修正されます。これには、間違った修正を選択したり、すでに発生している可能性のある破損の問題を隠したりするリスクがあることに注意してください。

Commit

指定したイメージファイル (*filename*) に記録された変更を、**qemu-img commit** コマンドでファイルのベースイメージにコミットします。オプションで、ファイルのフォーマットタイプ (*format*) を指定します。

```
# qemu-img commit [-f format] [-t cache] filename
```

convert

convert オプションは、認識されているイメージ形式を別のイメージ形式に変換するために使用されます。

コマンド形式:

```
# qemu-img convert [-c] [-p] [-f format] [-t cache] [-O output_format] [-o options] [-S sparse_size]
filename output_filename
```

-p パラメーターはコマンドの進捗を示し (すべてのコマンドではなく任意)、***-S*** フラグは、ディスクイメージに含まれる *sparse file* の作成を許可します。ゼロのみを含む (何も含まない) 物理ブロックを除いて、あらゆる目的のスパースファイルは標準ファイルのように機能します。オペレーティングシステムがこのファイルを認識すると、たとえ実際にはディスクを使用していなくても、存在しているものとして扱われ、実際のディスク領域を消費します。ゲスト仮想マシン用のディスクを作成する場合に特に役立ちます。これにより、ディスクに必要なディスク領域よりもはるかに多くのディスク領域が使用されるようになります。たとえば、10Gb のディスクイメージで -S を 50Gb に設定すると、実際には 10Gb しか使用されていなくても、そのディスク領域の 10Gb は 60Gb に見えます。

filename 形式を使用して、ディスクイメージ ***output_filename*** をディスクイメージ ***output_format*** に

変換します。ディスクイメージは、***-c*** オプションで圧縮するか、**-o encryption** を設定して ***-o*** オプションで暗号化できます。***-o*** パラメーターで使用できるオプションは、選択した形式とは異なることに注意してください。

qcow2 形式のみが暗号化または圧縮をサポートします。**qcow2** 暗号化は、安全な 128 ビット鍵で AES 形式を使用します。**qcow2** 圧縮は読み取り専用であるため、圧縮したセクターが **qcow2** 形式から変換されると、非圧縮データとして新しい形式に書き込まれます。

イメージの変換は、**qcow** または **cow** など、サイズを大きくできる形式を使用する場合に、小さなイメージを取得する場合にも役立ちます。空のセクターは、宛先イメージから検出され、抑制されます。

Create

サイズ***size***、フォーマット***format***、新しいディスクイメージの *ファイル名*を作成します。

```
# qemu-img create [-f format] [-o options] filename [size][preallocation]
```

ベースイメージが **-o backing_file=*filename*** で指定されている場合は、イメージ自体とベースイメージの違いのみが記録されます。バッキングファイルは、**commit** コマンドを使用しない限り変更されません。この場合、サイズの指定は必要ありません。

事前割り当ては、qcow2 イメージの作成でのみ使用できるオプションです。許可される値には **-o preallocation=*off*|*meta*|*full*|*falloc*** が含まれます。事前に割り当てられたメタデータを持つイメージは、イメージよりも大きくなります。ただし、イメージサイズが大きくなると、イメージのサイズが大きくなるとパフォーマンスが向上します。

full 割り当ての使用には、イメージのサイズが大きい場合に時間がかかる可能性があることに注意してください。完全な割り当てと時間が経つ必要がある場合、フラックを使用 **falloc** 時間を節約できます。

Info

info パラメーターは、ディスクイメージの*filename*に関する情報を表示します。**info** オプションの形式は、以下のとおりです。

```
# qemu-img info [-f format] filename
```

このコマンドは、多くの場合、表示されているサイズとは異なるディスクに予約されているサイズを検出するために使用されます。スナップショットがディスクイメージに保存されている場合は、そのスナップショットも表示されます。このコマンドは、たとえば、ブロックデバイスの **qcow2** イメージが使用している領域を表示します。これは、**qemu-img** を実行して行います。使用中のイメージが、**qemu-img check** コマンドで **qemu-img info** コマンドの出力と一致するイメージであることを確認できます。「qemu-img の使用」 を参照してください。

```
# qemu-img info /dev/vg-90.100-sluo/lv-90-100-sluo
image: /dev/vg-90.100-sluo/lv-90-100-sluo
file format: qcow2
virtual size: 20G (21474836480 bytes)
disk size: 0
cluster_size: 65536
```

マップ

 # qemu-img map [-f *format*] [--output=*output_format*] filename コマンドは、イメージファイル名のメタデータとそのバッキングファイルチェーンをダンプします。具体的には、このコマンドは、指定されたファイルのすべてのセクターの割り当て状態を、それをバッキングファイルチェーンに割り当てる最上位のファイルとともにダンプします。たとえば、c.qcow2 → b.qcow2 → a.qcow2 などのチェーンが

ある場合、a.qcow2 が元のファイルである場合、b.qcow2 は a.qcow2 に加えられた変更で、c.qcow2 は b.qcow2 のデルタファイルになります。このチェーンが作成されると、イメージファイルは通常のイメージデータを保存し、どのファイルの場所とそれがファイルに保存されるかに関する情報も保存されます。この情報は、イメージのメタデータと呼ばれます。**-f** フォーマットオプションは、指定されたイメージファイルのフォーマットです。raw、qcow2、vhdx、vmdk などの形式を使用できます。可能な出力オプションには、**human** と **json** の 2 つがあります。

- **human** がデフォルト設定です。人間の目に読みやすくなるように設計されているため、この形式は解析しないでください。明確さと単純さのために、デフォルトの **human** フォーマットは、ファイルの既知のゼロ以外の領域のみをダンプします。ファイルの既知のゼロの部分は完全に省略され、同様にチェーン全体に割り当てられていない部分も省略されます。コマンドが実行されると、qemu-img 出力は、データを読み取ることができるファイルと、ファイル内のオフセットを識別します。出力は、4 列のテーブルとして表示されます。最初の 3 つは 16 進数です。

  ```
  # qemu-img map -f qcow2 --output=human /tmp/test.qcow2
  Offset          Length          Mapped to       File
  0               0x20000         0x50000         /tmp/test.qcow2
  0x100000        0x80000         0x70000         /tmp/test.qcow2
  0x200000        0x1f0000        0xf0000         /tmp/test.qcow2
  0x3c00000       0x20000         0x2e0000        /tmp/test.qcow2
  0x3fd0000       0x10000         0x300000        /tmp/test.qcow2
  ```

- **json**、つまり *JSON* (JavaScript Object Notation) は、人間が読むこともできますが、プログラミング言語であるため、解析することも前提に設計されています。たとえば、パーサーで qemu-img map の出力を解析する場合は、オプション **--output=json** を使用する必要があります。

  ```
  # qemu-img map -f qcow2 --output=json /tmp/test.qcow2
  [{ "start": 0, "length": 131072, "depth": 0, "zero": false, "data": true, "offset": 327680},
  { "start": 131072, "length": 917504, "depth": 0, "zero": true, "data": false},
  ```

  JSON 形式の詳細については、qemu-img (1) の man ページを参照してください。

リベース

イメージのバッキングファイルを変更します。

```
# qemu-img rebase [-f format] [-t cache] [-p] [-u] -b backing_file [-F backing_format] filename
```

バッキングファイルの形式が *backing_file* に変更され (*filename* の形式がこの機能に対応している場合)、バッキングファイルの形式が *backing_format* に変更されます。

注記

バッキングファイル (リベース) の変更に対応するのは、**qcow2** 形式のみです。

rebase が動作できるモードには、**Safe** と **Unsafe** の 2 つがあります。

Safe モードは、デフォルトで使用され、実際のリベース操作を実行します。新しいバッキングファイルは古いファイルとは異なる場合があり、**qemu-img rebase** は、ゲストの仮想マシンで認識される *filename* の内容を変更せずに保持します。これを実現するため、*filename* の *backing_file* と古いバッキングファイルとで異なるクラスターは、すべて *filename* にマージされてから、バッキングファイルを変更します。

safe モードはイメージの変換に相当する費用のかかる動作であることに注意してください。古いバッキングファイルは、正常に完了するために必要です。

Unsafe モードは、*-u* オプションが **qemu-img rebase** に渡される場合に使用されます。このモードでは、ファイルの内容を確認せずに、バッキングファイルの名前と *filename* の形式のみが変更されます。新しいバッキングファイルが正しく指定されていることを確認してください。指定されていない場合、イメージのゲストに表示されるコンテンツが破損します。

このモードは、バッキングファイルの名前変更や移動に役立ちます。アクセス可能な古いバッキングファイルがなくても使用できます。たとえば、バッキングファイルがすでに移動または名前変更されているイメージを修正するために使用できます。

サイズの変更

サイズ *size*で作成されたかのように、ディスクイメージの*filename*を変更します。バージョンに関係なく、サイズ変更できるのは raw 形式のイメージのみです。Red Hat Enterprise Linux 6.1 以降では、**qcow2** フォーマットでイメージを拡大 (縮小はしない) する機能が追加されています。

次のコマンドを使用して、ディスクイメージの*filename*のサイズを *size* バイトに設定します。

```
# qemu-img resize filename size
```

また、ディスクイメージの現在のサイズを基準にしてサイズを変更することもできます。現在のサイズに相対的なサイズを指定するには、バイト数の前に **+** を付けて拡大するか、**-** を付けてディスクイメージのサイズをそのバイト数だけ縮小します。ユニットの接尾辞を追加すると、イメージのサイズをキロバイト (K)、メガバイト (M)、ギガバイト (G)、またはテラバイト (T) で設定できます。

```
# qemu-img resize filename [+|-]size[K|M|G|T]
```

警告

このコマンドを使用してディスクイメージを縮小する前に、仮想マシン自体の内部でファイルシステムとパーティションツールを使用して、割り当てられたファイルシステムとパーティションサイズを適宜減らす**必要**があります。そうしないと、データが失われることになります。

このコマンドを使用してディスクイメージを拡張したら、ファイルシステムと仮想マシン内のパーティションツールを使用して、デバイス上の新しい領域の使用を実際に開始する必要があります。

スナップショット

イメージ (*ファイル名*) の既存のスナップショット (*スナップショット*) を一覧表示、適用、作成、または削除します。

```
# qemu-img snapshot [ -l | -a snapshot | -c snapshot | -d snapshot ] filename
```

-l は、指定したディスクイメージに関連付けられているすべてのスナップショットを一覧表示します。apply オプション **-a** は、ディスクイメージ (*filename*) を、以前保存した*スナップショット*の状態に戻します。**-c** は、イメージ (*filename*) のスナップショット (*snapshot*) を作成します。**-d** は、指定したス

ナップショットを削除します。

サポート対象の形式

qemu-img は、ファイルを次のいずれかのフォーマットに変換するように設計されています。

raw

Raw ディスクイメージ形式 (デフォルト)これは、ファイルベースで最も高速な形式になります。ファイルシステムがホールをサポートしている場合 (たとえば、Linux の ext2 または ext3、Windows の NTFS)、書き込まれたセクターのみがスペースを予約します。**qemu-img info** を使用して、イメージが使用する実際のサイズを取得するか、Unix/Linux の **ls -ls** を取得します。Raw イメージは最適なパフォーマンスを提供しますが、Raw イメージで使用できるのは非常に基本的な機能のみです (たとえば、スナップショットは使用できません)。

qcow2

QEMU イメージ形式。最も汎用性が高く、機能セットが最適な形式です。これを使用して、オプションの AES 暗号化、zlib ベースの圧縮、複数の VM スナップショットのサポート、およびホールをサポートしないファイルシステム (Windows 上の非 NTFS ファイルシステム) で役立つ小さなイメージを使用します。この拡張機能セットはパフォーマンスに影響を与えることに注意してください。

ゲスト仮想マシンまたはホスト物理マシンでの実行には上記の形式のみを使用できますが、**qemu-img** は、**raw**、または **qcow2** 形式に変換するために、以下の形式も認識してサポートします。イメージの形式は、通常、自動的に検出されます。この形式を **raw** または **qcow2** に変換することに加えて、**raw** または **qcow2** から元の形式に戻すことができます。

bochs

Bochs ディスクイメージ形式。

cloop

Linux Compressed Loop イメージ。Knoppix CD-ROM などにある直接圧縮 CD-ROM イメージを再利用する場合にのみ役立ちます。

cow

User Mode Linux Copy On Write イメージ形式。**cow** 形式は、以前のバージョンとの互換性のためにのみ同梱されています。Windows では動作しません。

dmg

Mac ディスクイメージフォーマット。

nbd

ネットワークブロックデバイス。

parallels

Parallels 仮想化ディスクイメージフォーマット。

qcow

古い QEMU イメージフォーマット。古いバージョンとの互換性にのみ含まれます。

vdi

Oracle VM VirtualBox ハードディスクイメージ形式。

vmdk

VMware 互換のイメージフォーマット (バージョン 1 および 2 の読み取り/書き込みサポート、およびバージョン 3 の読み取り専用サポート)。

vpc

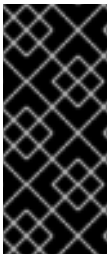
WindowsVirtualPC のディスクイメージフォーマット。**vhd**、または Microsoft 仮想ハードディスクイメージフォーマットとも呼ばれます。

vvfat

仮想 VFAT ディスクイメージフォーマット。

10.2. QEMU ゲストエージェント

QEMU ゲストエージェントはゲスト内で実行され、ホストマシンが libvirt を使用してゲストオペレーティングシステムにコマンドを発行できるようにします。ゲストオペレーティングシステムは、これらのコマンドに非同期に応答します。本章では、ゲストエージェントが使用できる libvirt コマンドとオプションを説明します。



重要

信頼できるゲストによって実行される場合にのみ、ゲストエージェントに依存することが安全であることに注意してください。信頼できないゲストは、ゲストエージェントプロトコルを悪意を持って無視または悪用する可能性があります。ホストに対するサービス拒否攻撃を防ぐための保護機能が組み込まれていますが、ホストは、操作を期待どおりに実行するためにゲストの協力を必要とします。

QEMU ゲストエージェントを使用して、ゲストの実行中に仮想 CPU(vCPU) を有効または無効にできるため、ホットプラグおよびホットアンプラグ機能を使用せずに vCPU の数を調整できることに注意してください。詳細は、「仮想 CPU 数の設定」 を参照してください。

10.2.1. ゲストエージェントをインストールして有効にする

ゲスト仮想マシンに **yum install qemu-guest-agent** コマンドでqemu-guest-agentをインストールし、サービス (qemu-guest-agent.service) として起動毎に自動実行されるようにします。

10.2.2. ゲストエージェントとホスト間の通信の設定

ホストマシンは、ホストとゲストマシン間の VirtIO シリアル接続を介してゲストエージェントと通信します。VirtIO シリアルチャネルはキャラクターデバイスドライバー (通常は Unix ソケット) を介してホストに接続し、ゲストはこのシリアルチャネルをリッスンします。次の手順は、ゲストエージェントが使用できるようにホストマシンとゲストマシンをセットアップする方法を示しています。

注記

Windows ゲストで QEMU ゲストエージェントを設定する方法については、こちら の手順を参照してください。。

手順10.1 ゲストエージェントとホスト間の通信の設定

1. **ゲスト XML を開きます**
   QEMU ゲストエージェント設定でゲスト XML を開きます。ファイルを開くにはゲスト名が必要です。ホストマシンでコマンド **#virsh list** を使用して、認識できるゲストを一覧表示します。この例では、ゲストの名前は *rhel6* です。

   ```
   # virsh edit rhel6
   ```

2. **ゲスト XML ファイルを編集します**
   次の要素を XML ファイルに追加し、変更を保存します。

   図10.1 ゲスト XML を編集して QEMU ゲストエージェントを設定する

   ```
   <channel type='unix'>
     <source mode='bind' path='/var/lib/libvirt/qemu/rhel6.agent'/>
     <target type='virtio' name='org.qemu.guest_agent.0'/>
   </channel>
   ```

3. **ゲストでの QEMU ゲストエージェントの起動**
   まだ行っていない場合は、**yum install qemu-guest-agent** を使用してゲスト仮想マシンにゲストエージェントをダウンロードしてインストールします。インストールしたら、次のようにサービスを開始します。

   ```
   # service start qemu-guest-agent
   ```

これで、有効な送信によってゲストと通信できます確立されたキャラクターデバイスドライバーに対する libvirt コマンド。

10.2.3. QEMU ゲストエージェントの使用

QEMU ゲストエージェントプロトコル (QEMU GA) パッケージ、qemu-guest-agent は、Red Hat Enterprise Linux 以降で完全にサポートされています。ただし、isa-serial/virtio-serial トランスポートに関しては、次の制限があります。

- qemu-guest-agentは、クライアントがチャネルに接続したかどうかを検出することはできません。
- クライアントが、qemu-guest-agent がバックエンドに切断または再接続したかどうかを検出する方法はありません。
- virtio-serial デバイスがリセットされ、qemu-guest-agentチャネルに接続されていない場合 (通常、再起動またはホットプラグが原因)、クライアントからのデータはドロップされます。
- virtio-serial デバイスのリセット後に qemu-guest-agent がチャネルに接続した場合、qemu-guest-agent がまだ実行中か接続中かにかかわらず、クライアントからのデータはキューに入れられます (そして利用可能なバッファーを使い切ると最終的にスロットルに入れられます)。

10.2.4. libvirt での QEMU ゲストエージェントの使用

QEMU ゲストエージェントをインストールすると、他のさまざまな libvirt コマンドのパフォーマンスが向上します。ゲストエージェントは、次の**virsh** コマンドを拡張します。

- **virsh shutdown --mode=agent** - このシャットダウン方法は、**virsh shutdown --mode=acpi** よりも信頼性が高くなります。QEMU ゲストエージェントで使用する **virsh shutdown** は、ク

リーンな状態で協調ゲストをシャットダウンすることが保証されます。エージェントが存在しない場合、libvirt は代わりに ACPI シャットダウンイベントの挿入に依存する必要がありますが、一部のゲストはそのイベントを無視するため、シャットダウンしません。

virsh reboot と同じ構文で使用できます。

- **virsh snapshot-create --quiesce** - スナップショットが作成される前に、ゲストが I/O を安定した状態にフラッシュできるようにします。これにより、fsck を実行したり、データベーストランザクションの一部を失うことなくスナップショットを使用できます。ゲストエージェントは、ゲストの相互作用を提供することで、高レベルのディスクコンテンツの安定性を実現します。

- **virsh setvcpus --guest** - CPU をオフラインにするようにゲストに指示します。

- **virsh dompmsuspend** - ゲストオペレーティングシステムの電源管理機能を使用して、実行中のゲストを優雅に一時停止します。

10.2.5. ゲスト仮想マシンのディスクバックアップの作成

libvirt は、qemu-ga と通信して、ゲスト仮想マシンのファイルシステムのスナップショットが内部で一貫性を保ち、必要に応じてすぐに使用できるようにします。Red Hat Enterprise Linux 6 の改善により、ファイルレベルとアプリケーションレベルの両方の Synchronization (フラッシュ) が確実に実行されるようになりました。ゲストシステム管理者は、アプリケーション固有のフリーズ/フリーズ解除フックスクリプトを作成してインストールできます。ファイルシステムをフリーズする前に、qemu-gaはメインフックスクリプト (qemu-gaパッケージに含まれています) を起動します。フリーズプロセスでは、ゲスト仮想マシンのアプリケーションがすべて一時的に非アクティブになります。

ファイルシステムがフリーズする直前に、次のアクションが発生します。

- ファイルシステムアプリケーション/データベースは、作業バッファーを仮想ディスクにフラッシュし、クライアント接続の受け入れを停止します。

- アプリケーションがデータファイルを一貫した状態にします。

- メインフックスクリプトが返されます。

- qemu-ga ファイルシステムをフリーズし、管理スタックがスナップショットを取得します

- スナップショットが確認されました。

- ファイルシステムの機能が再開します。

フリーズ解除は逆の順序で行われます。

snapshot-create-as コマンドを使用して、ゲストディスクのスナップショットを作成します。このコマンドの詳細については、「現在のドメインのスナップショットを作成する」 を参照してください。

注記

アプリケーション固有のフックスクリプトでは、データベースと通信するためにスクリプトをソケットに接続する必要がある場合と同様に、正しく実行するために様々な SELinux のパーミッションが必要になることがあります。一般に、ローカル SELinux ポリシーは、そのような目的のために開発およびインストールする必要があります。**/etc/qemu-ga/fsfreeze-hook.d/** というラベルが付いた表の行の 表10.1「QEMU ゲストエージェントパッケージの内容」 に一覧表示されている **restorecon -FvvR** コマンドを実行した後、ファイルシステムノードへのアクセスがすぐに機能するようになります。

qemu-guest-agent バイナリー RPM には、以下のファイルが含まれています。

表10.1 QEMU ゲストエージェントパッケージの内容

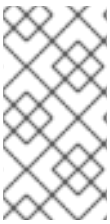
File name	説明
/etc/rc.d/init.d/qemu-ga	QEMU ゲストエージェントのサービス制御スクリプト (start/stop)
/etc/sysconfig/qemu-ga	**/etc/rc.d/init.d/qemu-ga** 制御スクリプトによって読み取られる QEMU ゲストエージェントの設定ファイル。設定は、シェルスクリプトのコメントが記載されたファイルで説明されています。
/usr/bin/qemu-ga	QEMU ゲストエージェントのバイナリーファイル。
/usr/libexec/qemu-ga/	フックスクリプトのルートディレクトリー。
/usr/libexec/qemu-ga/fsfreeze-hook	メインフックスクリプト。ここでは変更は必要ありません。
/usr/libexec/qemu-ga/fsfreeze-hook.d/	個々のアプリケーション固有のフックスクリプトのディレクトリー。ゲストシステム管理者は、フックスクリプトを手動でこのディレクトリーにコピーし、適切なファイルモードビットを確認してから、このディレクトリーで **restorecon -FvvR** を実行する必要があります。
/usr/share/qemu-kvm/qemu-ga/	サンプルスクリプトを含むディレクトリー (サンプル目的のみ)ここに含まれるスクリプトは実行されません。

メインフックスクリプトの **/usr/libexec/qemu-ga/fsfreeze-hook** は、独自のメッセージと、アプリケーション固有のスクリプトの標準出力およびエラーメッセージを、ログファイル **/var/log/qemu-ga/fsfreeze-hook.log** に記録します。詳細は、wiki.qemu.org または libvirt.org にある qemu-guest-agentwiki ページを参照してください。

10.3. WINDOWS ゲストでの QEMU ゲストエージェントの実行

Red Hat Enterprise Linux ホストマシンは、ゲストで QEMU ゲストエージェントを実行することにより、Windows ゲストにコマンドを発行できます。これは、Red Hat Enterprise Linux 6.5 以降を実行しているホスト、および次の Windows ゲストオペレーティングシステムでサポートされています。

- Windows XP Service Pack 3 (VSS はサポートされていません)
- Windows Server 2003 R2 - x86 および AMD64 (VSS はサポートされていません)
- Windows Server 2008
- Windows Server 2008 R2
- Windows7 - x86 および AMD64
- Windows Server 2012
- Windows Server 2012 R2
- Windows8 - x86 および AMD64
- Windows8.1 - x86 および AMD64



注記

Windows ゲスト仮想マシンには、Windows 用の QEMU ゲストエージェントパッケージが必要です。qemu-guest-agent-win。このエージェントは、Red Hat Enterprise Linux で実行されている Windows ゲスト仮想マシンの VSS(ボリュームシャドウコピーサービス) サポートに必要です。詳細はこちらの記事を参照してください:

手順10.2 Windows ゲストでの QEMU ゲストエージェントの設定

Red Hat Enterprise Linux ホストマシンで実行されている Windows ゲストの場合は、次の手順に従います。

1. **Red Hat Enterprise Linux ホストマシンを準備します**
   次のパッケージが Red Hat Enterprise Linux ホスト物理マシンにインストールされていることを確認してください。

   - virtio-win**usr/share/virtio-win/** にあります。

   Windows ゲストのドライバーをコピーするには、次のコマンドを使用して **qxl** ドライバーの ***.iso** ファイルを作成します。

   ```
   # mkisofs -o /var/lib/libvirt/images/virtiowin.iso /usr/share/virtio-win/drivers
   ```

2. **Windows ゲストを準備する**
   virtio-serial をインストールしますドライバーを更新するために Windows ゲストに ***.iso** をマウントすることにより、ゲストのドライバー。ゲストを起動し、次に示すようにドライバーの.iso ファイルをゲストに添付します (*hdb* という名前のディスクを使用)。

   ```
   # virsh attach-disk guest /var/lib/libvirt/images/virtiowin.iso hdb
   ```

   Windows の **コントロールパネル** を使用してドライバーをインストールするには、次のメニューに移動します。

- virtio-win ドライバーをインストールするには - **ハードウェアとサウンド** > **デバイスマネージャー** > **virtio- シリアルドライバー** を選択します。

3. **Windows ゲスト XML 設定ファイルを更新します**
Windows ゲストのゲスト XML ファイルは、Red Hat Enterprise Linux ホストマシンにあります。このファイルにアクセスするには、Windows ゲスト名が必要です。ホストマシンで **#virsh list** コマンドを使用して、認識できるゲストを一覧表示します。この例では、ゲストの名前は win7x86 です。

#virsh edit *win7x86* コマンドを使用して次の要素を XML ファイルに追加し、変更を保存します。ソースソケット名は、この例では *win7x86.agent* という名前で、ホスト内で一意である必要があることに注意してください。

図10.2 Windows ゲスト XML を編集して QEMU ゲストエージェントを設定する

```
...
<channel type='unix'>
   <source mode='bind' path='/var/lib/libvirt/qemu/win7x86.agent'/>
   <target type='virtio' name='org.qemu.guest_agent.0'/>
   <address type='virtio-serial' controller='0' bus='0' port='1'/>
</channel>
<channel type='spicevmc'>
   <target type='virtio' name='com.redhat.spice.0'/>
   <address type='virtio-serial' controller='0' bus='0' port='2'/>
</channel>
...
```

4. **Windows ゲストを再起動します**
Windows ゲストを再起動して、変更を適用します。

```
# virsh reboot win7x86
```

5. **Windows ゲストで QEMU ゲストエージェントを準備します**
Windows ゲストでゲストエージェントを準備するには:

   a. **最新のものをインストールするvirtio-winパッケージ**
   Red Hat Enterprise Linux ホストの物理マシンのターミナルウィンドウで次のコマンドを実行して、インストールするファイルを見つけます。以下に示すファイルは、システムが検出したファイルと完全に同じではない場合がありますが、最新の公式バージョンである必要があることに注意してください。

```
# rpm -qa|grep virtio-win
virtio-win-1.6.8-5.el6.noarch

# rpm -iv virtio-win-1.6.8-5.el6.noarch
```

   b. **インストールが完了したことを確認します**
   後に virtio-win パッケージのインストールが完了したら、**/usr/share/virtio-win/guest-agent/** フォルダーを確認すると、次のような qemu-ga-x64.msi または qemu-ga-x86.msi という名前のファイルが見つかります。

```
# ls -l /usr/share/virtio-win/guest-agent/
```

```
total 1544

-rw-r--r--. 1 root root 856064 Oct 23 04:58 qemu-ga-x64.msi

-rw-r--r--. 1 root root 724992 Oct 23 04:58 qemu-ga-x86.msi
```

c. **.msi ファイルをインストールします**
Windows ゲスト (たとえば、win7x86) から、ファイルをダブルクリックして qemu-ga-x64.msi または qemu-ga-x86.msi をインストールします。インストールされると、システムマネージャー内の Windows ゲストに qemu-ga サービスとして表示されます。この同じマネージャーを使用して、サービスのステータスを監視できます。

10.3.1. Windows ゲストでの QEMUGuestAgent での libvirt コマンドの使用

QEMU ゲストエージェントは、Windows ゲストで次の **virsh** コマンドを使用できます。

- **virsh shutdown --mode=agent** - このシャットダウン方法は、**virsh shutdown --mode=acpi** よりも信頼性が高くなります。QEMU ゲストエージェントで使用する **virsh shutdown** は、クリーンな状態で協調ゲストをシャットダウンすることが保証されます。エージェントが存在しない場合、libvirt は代わりに ACPI シャットダウンイベントの挿入に依存する必要がありますが、一部のゲストはそのイベントを無視するため、シャットダウンしません。

  virsh reboot と同じ構文で使用できます。

- **virsh snapshot-create --quiesce** - スナップショットが作成される前に、ゲストが I/O を安定した状態にフラッシュできるようにします。これにより、fsck を実行したり、データベーストランザクションの一部を失うことなくスナップショットを使用できます。ゲストエージェントは、ゲストの相互作用を提供することで、高レベルのディスクコンテンツの安定性を実現します。

- **virsh dompmsuspend** - ゲストオペレーティングシステムの電源管理機能を使用して、実行中のゲストを優雅に一時停止します。

10.4. デバイスリダイレクトの制限の設定

リダイレクトから特定のデバイスをフィルターリングするには、フィルタープロパティーを ***-device usb-redir*** に渡します。filter プロパティーは、フィルタールールで設定される文字列を取ります。ルールの形式は次のとおりです。

```
<class>:<vendor>:<product>:<version>:<allow>
```

-1 の値を使用して、特定のフィールドに任意の値を受け入れるように指定します。同じコマンドラインで、| を区切り文字として使用し、複数のルールを使用できます。デバイスがどのフィルタールールにも一致しない場合、リダイレクトは許可されないことに注意してください。

例10.1 Windows ゲスト仮想マシンによるリダイレクトの制限

1. Windows 7 ゲスト仮想マシンを準備します。

2. 以下のコード抜粋をゲスト仮想マシンの XML ファイルに追加します。

```
<redirdev bus='usb' type='spicevmc'>
```

```
  <alias name='redir0'/>
  <address type='usb' bus='0' port='3'/>
</redirdev>
<redirfilter>
  <usbdev class='0x08' vendor='0x1234' product='0xBEEF' version='2.0' allow='yes'/>
  <usbdev class='-1' vendor='-1' product='-1' version='-1' allow='no'/>
</redirfilter>
```

3. ゲスト仮想マシンを起動し、次のコマンドを実行して設定の変更を確認します。

```
# ps -ef | grep $guest_name
```

```
-device usb-redir,chardev=charredir0,id=redir0,/
filter=0x08:0x1234:0xBEEF:0x0200:1|-1:-1:-1:-1:0,bus=usb.0,port=3
```

4. USB デバイスをホストの物理マシンに接続し、**virt-viewer** を使用してゲスト仮想マシンに接続します。

5. メニューで **USB デバイスの選択** をクリックすると、一部の USB デバイスはホストポリシーによってブロックされていますというメッセージが表示されます。**OK** をクリックして確定し、続行します。

   フィルターが有効になります。

6. フィルターが正しくキャプチャーされていることを確認するには、USB デバイスのベンダーと製品を確認してから、USB リダイレクトを可能にするために、ホストの物理マシンのドメイン XML で次の変更を行います。

```
<redirfilter>
  <usbdev class='0x08' vendor='0x0951' product='0x1625' version='2.0' allow='yes'/>
  <usbdev allow='no'/>
 </redirfilter>
```

7. ゲスト仮想マシンを再起動してから、**virt-viewer** を使用してゲスト仮想マシンに接続します。これで、USB デバイスはトラフィックをゲスト仮想マシンにリダイレクトするようになります。

10.5. 仮想 NIC に接続しているホスト物理マシンまたはネットワークブリッジの動的な変更

本セクションでは、ゲスト仮想マシンを危険にさらすことなく、ゲスト仮想マシンの実行中にゲスト仮想マシンの vNIC をあるブリッジから別のブリッジに移動する方法を説明します。

1. 次のような設定で、ゲスト仮想マシンを準備します。

```
<interface type='bridge'>
    <mac address='52:54:00:4a:c9:5e'/>
    <source bridge='virbr0'/>
    <model type='virtio'/>
</interface>
```

2. インターフェイスを更新するために XML ファイルを準備します。

```
# cat br1.xml
```

```
<interface type='bridge'>
      <mac address='52:54:00:4a:c9:5e'/>
      <source bridge='virbr1'/>
      <model type='virtio'/>
</interface>
```

3. ゲスト仮想マシンを起動し、ゲスト仮想マシンのネットワーク機能を確認して、ゲスト仮想マシンの vnetX が指定したブリッジに接続されていることを確認します。

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.5254007da9f2       yes                virbr0-nic

vnet0
virbr1          8000.525400682996       yes                virbr1-nic
```

4. 次のコマンドを使用して、新しいインターフェイスパラメーターでゲスト仮想マシンのネットワークを更新します。

```
# virsh update-device test1 br1.xml

Device updated successfully
```

5. ゲスト仮想マシンで **service network restart** を実行します。ゲスト仮想マシンは、virbr1 の新しい IP アドレスを取得します。ゲスト仮想マシンの vnet0 が新しいブリッジ (virbr1) に接続されていることを確認します。

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.5254007da9f2       yes             virbr0-nic
virbr1          8000.525400682996       yes             virbr1-nic      vnet0
```

第11章 ストレージの概念

この章では、ストレージデバイスの説明と管理に使用される概念を紹介します。ストレージプールやボリュームなどの用語については、次のセクションで説明します。

11.1. ストレージプール

ストレージプール は、ゲスト仮想マシンにストレージを提供する目的で libvirt によって管理されるファイル、ディレクトリー、またはストレージデバイスです。ストレージプールはローカルにすることも、ネットワーク経由で共有することもできます。ストレージプールは、ゲスト仮想マシンで使用するために、管理者 (多くの場合は専用のストレージ管理者) によって確保されるストレージの量です。ストレージプールは、ストレージ管理者またはシステム管理者によってストレージボリュームに分割され、ボリュームはブロックデバイスとしてゲスト仮想マシンに割り当てられます。要するに、ストレージボリュームはパーティションになり、ストレージプールはディスクになります。ストレージプールは仮想コンテナーですが、次の 2 つの要因で制限されます。つまり、qemu-kvm が許可する最大サイズと、ホストの物理マシンのディスクのサイズです。ストレージプールは、ホストの物理マシン上のディスクのサイズを超えてはなりません。最大サイズは、以下のとおりです。

- virtio-blk = 2^63 バイトまたは 8 エクサバイト (raw ファイルまたはディスクを使用)
- Ext4 = ~ 16TB (4KB のブロックサイズを使用)
- XFS = ~8 エクサバイト
- qcow2 およびホストのファイルシステムでは、非常に大きなイメージサイズを試行する際に、独自のメタデータとスケーラビリティーを評価/調整する必要があります。raw ディスクを使用すると、スケーラビリティーまたは最大サイズに影響を与えるレイヤーが減ります。

libvirt は、ディレクトリーベースのストレージプールである **/var/lib/libvirt/images/** ディレクトリーをデフォルトのストレージプールとして使用します。デフォルトのストレージプールは、別のストレージプールに変更できます。

- **ローカルストレージプール** - ローカルストレージプールは、ホストの物理マシンサーバーに直接接続されています。ローカルストレージプールには、ローカルディレクトリー、直接接続されたディスク、物理パーティション、および LVM ボリュームグループが含まれます。これらのストレージボリュームは、ゲスト仮想マシンイメージを格納するか、追加のストレージとしてゲスト仮想マシンに接続されます。ローカルストレージプールはホストの物理マシンサーバーに直接接続されているため、移行や多数のゲスト仮想マシンを必要としない開発、テスト、および小規模な展開に役立ちます。ローカルストレージプールはライブマイグレーションをサポートしていないため、ローカルストレージプールは多くの本番環境には適していません。
- **ネットワーク型 (共有) ストレージプール** - ネットワーク型ストレージプールには、標準プロトコルを使用してネットワーク上で共有されるストレージデバイスが含まれます。virt-manager を使用してホスト物理マシン間で仮想マシンを移行する場合はネットワークストレージが必要ですが、virsh を使用して移行する場合は任意です。ネットワークストレージプールは libvirt によって管理されます。ネットワークストレージプールでサポートされているプロトコルは次のとおりです。
  - ファイバーチャネルベースの LUN
  - iSCSI
  - NFS
  - GFS2

- SCSI RDMA プロトコル (SCSI RCP) - InfiniBand アダプターおよび 10GbE iWARP アダプターで使用されるブロックエクスポートプロトコル

注記

マルチパスストレージプールは完全にはサポートされていないため、作成または使用しないでください。

11.2. ボリューム

ストレージプールは、ストレージボリュームに分類されます。ストレージボリュームは、libvirt が処理する物理パーティション、LVM 論理ボリューム、ファイルベースのディスクイメージ、その他のストレージタイプの抽象化です。ストレージボリュームは、基本となるハードウェアに関係なく、ゲスト仮想マシンにローカルストレージデバイスとして提示されます。

ボリュームの参照

特定のボリュームを参照するには、次の 3 つのアプローチが可能です。

ボリュームとストレージプールの名前

ボリュームは、それが属するストレージプールの識別子とともに名前で参照される場合があります。virsh コマンドラインでは、***--pool*** *storage_pool volume_name* の形式を取ります。

たとえば、*guest_images* プールの *firstimage* という名前のボリュームの場合は以下のようになります。

```
# virsh vol-info --pool guest_images firstimage
Name:           firstimage
Type:           block
Capacity:       20.00 GB
Allocation:     20.00 GB

virsh #
```

ホスト物理マシンシステム上のストレージへのフルパス

ボリュームは、ファイルシステム上のフルパスによって参照される場合もあります。このアプローチを使用する場合、プール識別子を含める必要はありません。

たとえば、*secondimage.img* という名前のボリュームは、ホスト物理マシンシステムに */images/secondimage.img* として表示されます。このイメージは */images/secondimage.img* として参照できます。

```
# virsh vol-info /images/secondimage.img
Name:           secondimage.img
Type:           file
Capacity:       20.00 GB
Allocation:     136.00 kB
```

ユニークなボリュームキー

仮想化システムでボリュームが最初に作成されると、一意の識別子が生成され、ボリュームに割り当てられます。一意の識別子は *ボリュームキー* と呼ばれます。このボリュームキーの形式は、使用するストレージによって異なります。

LVM などのブロックベースのストレージで使用する場合、ボリュームキーは次の形式に従う場合があります。

```
c3pKz4-qPVc-Xf7M-7WNM-WJc8-qSiz-mtvpGn
```

ファイルベースのストレージで使用する場合、ボリュームキーは代わりにボリュームストレージへのフルパスのコピーである可能性があります。

```
/images/secondimage.img
```

たとえば、ボリュームキーが *Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr*:

```
# virsh vol-info Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr
Name:           firstimage
Type:           block
Capacity:       20.00 GB
Allocation:     20.00 GB
```

virsh は、ボリューム名、ボリュームパス、またはボリュームキーを変換するためのコマンドを提供します。

vol-name

ボリュームパスまたはボリュームキーが指定されている場合は、ボリューム名を返します。

```
# virsh vol-name /dev/guest_images/firstimage
firstimage
# virsh vol-name Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr
```

vol-path

ボリュームキー、またはストレージプール識別子とボリューム名が指定されている場合は、ボリュームパスを返します。

```
# virsh vol-path Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr
/dev/guest_images/firstimage
# virsh vol-path --pool guest_images firstimage
/dev/guest_images/firstimage
```

vol-key コマンド

ボリュームパス、またはストレージプール識別子とボリューム名が指定されている場合は、ボリュームキーを返します。

```
# virsh vol-key /dev/guest_images/firstimage
Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr
# virsh vol-key --pool guest_images firstimage
Wlvnf7-a4a3-Tlje-lJDa-9eak-PZBv-LoZuUr
```

第12章 ストレージプール

この章には、さまざまなタイプのストレージプールを作成する手順が含まれています。*ストレージプール* は、仮想マシンで使用するために、管理者 (多くの場合は専用のストレージ管理者) によって確保されるストレージの量です。多くの場合、ストレージプールは、ストレージ管理者またはシステム管理者によってストレージボリュームに分割され、ボリュームはブロックデバイスとしてゲスト仮想マシンに割り当てられます。

例12.1 NFS ストレージプール

NFS サーバーを担当するストレージ管理者が、ゲスト仮想マシンのデータを格納するための共有を作成するとします。システム管理者は、共有の詳細を使用してホスト物理マシン上のプールを定義します (nfs.example.com:/**path/to/share** は **/vm_data** にマウントする必要があります)。プールが開始すると、libvirt は、システム管理者がログインして **mount nfs.example.com:/path/to/share /vmdata** を実行した場合と同じように、指定したディレクトリーに共有をマウントします。プールが自動開始するように設定されている場合、libvirt は、libvirt の開始時に指定されたディレクトリーに NFS 共有がマウントされていることを確認します。

プールが開始されると、NFS が共有するファイルがボリュームとして報告され、libvirtAPI を使用してストレージボリュームのパスが照会されます。ストレージボリュームのパスは、ゲスト仮想マシンの XML 定義ファイルセクションにコピーできます。このセクションは、ゲスト仮想マシンのブロックデバイスのソースストレージを記述します。NFS を使用すると、libvirt API を使用するアプリケーションは、プールのサイズ (共有の最大ストレージ容量) の制限まで、プール内のボリューム (NFS 共有内のファイル) を作成および削除できます。すべてのプールタイプがボリュームの作成と削除をサポートしているわけではありません。プールを停止すると、開始操作が無効になります。この場合、NFS 共有がアンマウントされます。名前にもかかわらず、共有のデータは破棄操作によって変更されません。詳細については、manvirsh を参照してください。

注記

ゲスト仮想マシンを適切に動作させるために、ストレージプールとボリュームは必要ありません。プールとボリュームは、libvirt が特定のストレージをゲスト仮想マシンで使用できるようにする方法を提供しますが、一部の管理者は独自のストレージを管理することを好み、ゲスト仮想マシンはプールやボリュームを定義しなくても適切に動作します。プールを使用しないシステムでは、システム管理者は、起動時に共有がマウントされるように、ホスト物理マシンの fstab に NFS 共有を追加するなど、好みのツールを使用してゲスト仮想マシンのストレージの可用性を確保する必要があります。

警告

ゲストにストレージプールを作成するときは、セキュリティー上の考慮事項に必ず従ってください。この情報については、『Red Hat Enterprise Linux 仮想化セキュリティーガイド』 を参照してください。https://access.redhat.com/site/documentation/。

12.1. ディスクベースのストレージプール

このセクションでは、ゲスト仮想マシン用のディスクベースのストレージデバイスの作成について説明します。

警告

ゲストには、ディスク全体またはブロックデバイス (**/dev/sdb** など) への書き込みアクセス権を付与しないでください。パーティション (**/dev/sdb1** など) または LVM ボリュームを使用します。

ブロックデバイス全体をゲストに渡すと、ゲストはブロックデバイスをパーティションに分割するか、ブロックデバイスに独自の LVM グループを作成します。これにより、ホストの物理マシンがこのようなパーティションや LVM グループを検出し、エラーが発生する場合があります。

12.1.1. virsh を使用したディスクベースのストレージプールの作成

この手順では、**virsh** コマンドでディスクデバイスを使用して新しいストレージプールを作成します。

警告

ディスクをストレージプール専用にすると、現在ディスクデバイスに保存されているすべてのデータが再フォーマットおよび消去されます。次の手順を開始する前に、ストレージデバイスをバックアップすることを強くお勧めします。

1. **ディスクに GPT ディスクラベルを作成します**
   ディスクには、*GUID パーティションテーブル* (GPT) ディスクラベルを付け直す必要があります。GPT ディスクラベルを使用すると、各デバイスに最大 128 個のパーティションを作成できます。GPT パーティションテーブルは、MS-DOS パーティションテーブルよりもはるかに多くのパーティションのパーティションデータを格納できます。

   ```
   # parted /dev/sdb
   GNU Parted 2.1
   Using /dev/sdb
   Welcome to GNU Parted! Type 'help' to view a list of commands.
   (parted) mklabel
   New disk label type? gpt
   (parted) quit
   Information: You may need to update /etc/fstab.
   #
   ```

2. **ストレージプール設定ファイルを作成します**
   新規デバイスに必要なストレージプール情報を含む一時的な XML テキストファイルを作成します。

   ファイルは以下に示す形式で、次のフィールドが含まれている必要があります。

   ```
   <name>guest_images_disk</name>
   ```

nameパラメーターは、ストレージプールの名前を決定します。この例では、以下の例で *guest_images_disk* という名前を使用しています。

<device path='*/dev/sdb*'/>

deviceパラメーターに***path***属性を指定すると、ストレージデバイスのデバイスパスが指定されます。この例では、デバイス */dev/sdb* を使用しています。

<target> <path>*/dev*</path></target>

ファイルシステム ***target*** パラメーターと ***path*** サブパラメーターは、このストレージプールで作成されたボリュームを接続するホスト物理マシンファイルシステム上の場所を決定します。

たとえば、sdb1、sdb2、sdb3 です。以下の例のように */dev/* を使用すると、このストレージプールから作成されたボリュームに */dev*/sdb1、*/dev*/sdb2、*/dev*/sdb3 としてアクセスできることを意味します。

<format type='*gpt*'/>

formatパラメーターは、パーティションテーブルの種類を指定します。この例では、次の例の *gpt* を使用して、前の手順で作成した GPT ディスクラベルタイプと一致させます。

テキストエディターを使用して、ストレージプールデバイスの XML ファイルを作成します。

例12.2 ディスクベースのストレージデバイスストレージプール

```
<pool type='disk'>
  <name>guest_images_disk</name>
  <source>
    <device path='/dev/sdb'/>
    <format type='gpt'/>
  </source>
  <target>
    <path>/dev</path>
  </target>
</pool>
```

3. **デバイスを接続します**
前の手順で作成した XML 設定ファイルで **virshpool-define** コマンドを使用して、ストレージプール定義を追加します。

```
# virsh pool-define ~/guest_images_disk.xml
Pool guest_images_disk defined from /root/guest_images_disk.xml
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_disk    inactive   no
```

4. **ストレージプールを起動します。**
virshpool-start コマンドを使用してストレージプールを開始します。**virshpool-list--all** コマンドを使用してプールが開始されていることを確認します。

```
# virsh pool-start guest_images_disk
Pool guest_images_disk started
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_disk    active     no
```

5. **自動起動をオンにします**
   オンにする***autostart***ストレージプール用。Autostart は、サービスの開始時にストレージプールを開始するように **libvirtd** サービスを設定します。

```
# virsh pool-autostart guest_images_disk
Pool guest_images_disk marked as autostarted
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_disk    active     yes
```

6. **ストレージプールの設定を確認する**
   ストレージプールが正しく作成され、サイズが正しく報告され、状態が次のように報告されることを確認します**running**。

```
# virsh pool-info guest_images_disk
Name:           guest_images_disk
UUID:           551a67c8-5f2a-012c-3844-df29b167431c
State:          running
Capacity:       465.76 GB
Allocation:     0.00
Available:      465.76 GB
# ls -la /dev/sdb
brw-rw----. 1 root disk 8, 16 May 30 14:08 /dev/sdb
# virsh vol-list guest_images_disk
Name                 Path
-----------------------------------------
```

7. **オプション: 一時設定ファイルを削除します**
   必要がない場合は、一時ストレージプールの XML 設定ファイルを削除します。

```
# rm ~/guest_images_disk.xml
```

ディスクベースのストレージプールが利用可能になりました。

12.1.2. virsh を使用したストレージプールの削除

次に、virsh を使用してストレージプールを削除する方法を示します。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。

```
# virsh pool-destroy guest_images_disk
```

2. ストレージプールの定義を削除します。

```
# virsh pool-undefine guest_images_disk
```

12.2. パーティションベースのストレージプール

このセクションでは、事前にフォーマットされたブロックデバイスであるパーティションをストレージプールとして使用する方法について説明します。

次の例では、ホスト物理マシンに 500GB のハードドライブ (**/dev/sdc**) が 1 つの 500GB の ext4 フォーマット済みパーティション (**/dev/sdc1**) にパーティション化されています。以下の手順でストレージプールを設定します。

12.2.1. virt-manager を使用したパーティションベースのストレージプールの作成

この手順では、ストレージデバイスのパーティションを使用して新しいストレージプールを作成します。

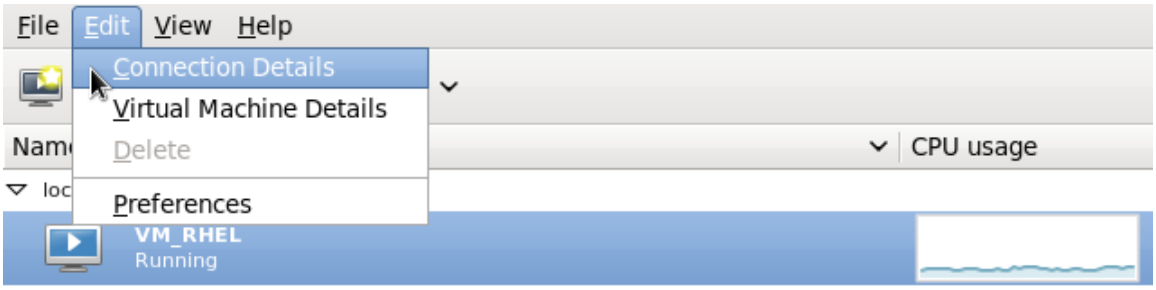
手順12.1 virt-manager を使用したパーティションベースのストレージプールの作成

1. **ストレージプールの設定を開きます**

   a. **virt-manager** のグラフィカルインターフェイスで、メインウィンドウからホストの物理マシンを選択します。

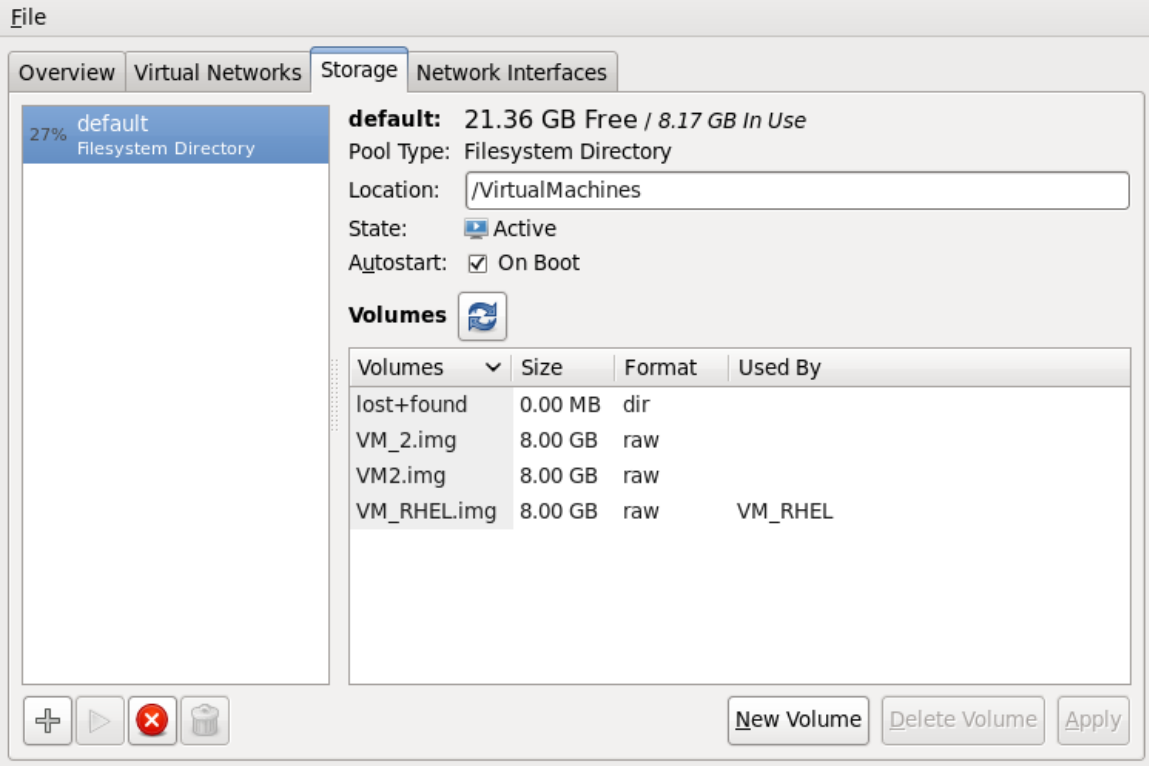
   Edit メニューを開き、**Connection Details** を選択します。

   図12.1 接続の詳細



   b. **Connection Details** ウィンドウの **Storage** タブをクリックします。

図12.2 ストレージタブ

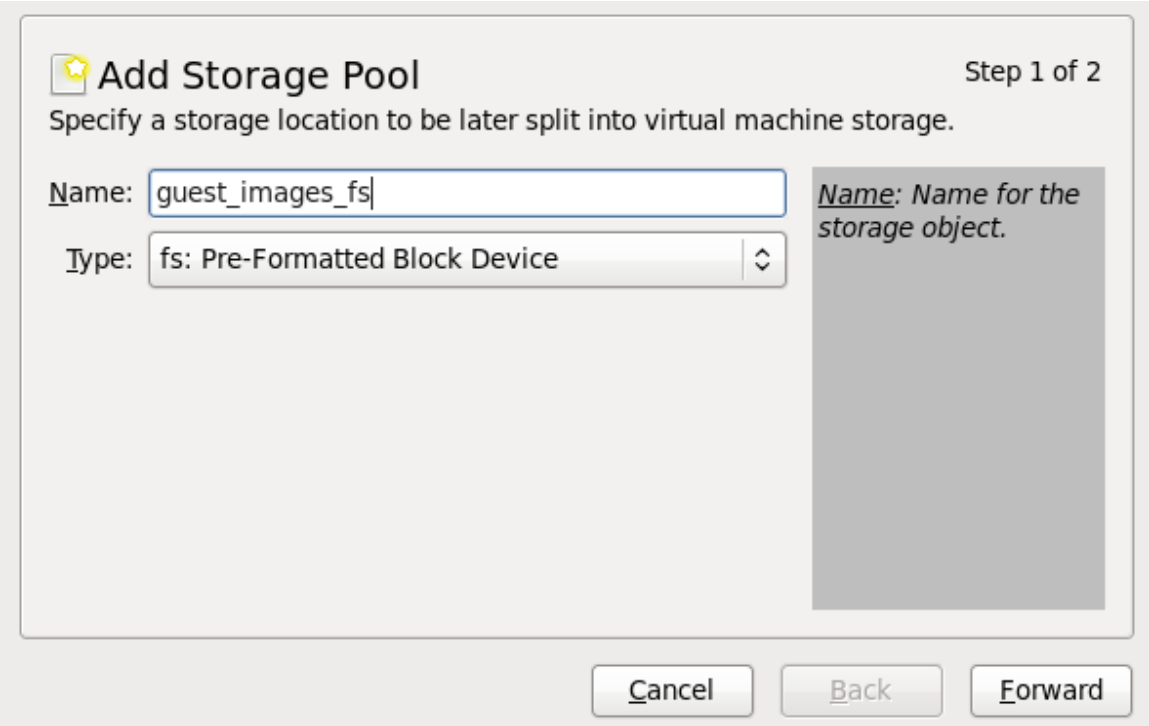


2. **新しいストレージプールを作成します**

   a. **新しいプールの追加 (パート 1)**
   + ボタン (プールの追加ボタン) を押します。**Add a New Storage Pool** ウィザードが表示されます。

   ストレージプールの **Name** を選択します。この例では、*guest_images_fs* という名前を使用します。**Type** を **fs: Pre-Formatted Block Device** に変更します。

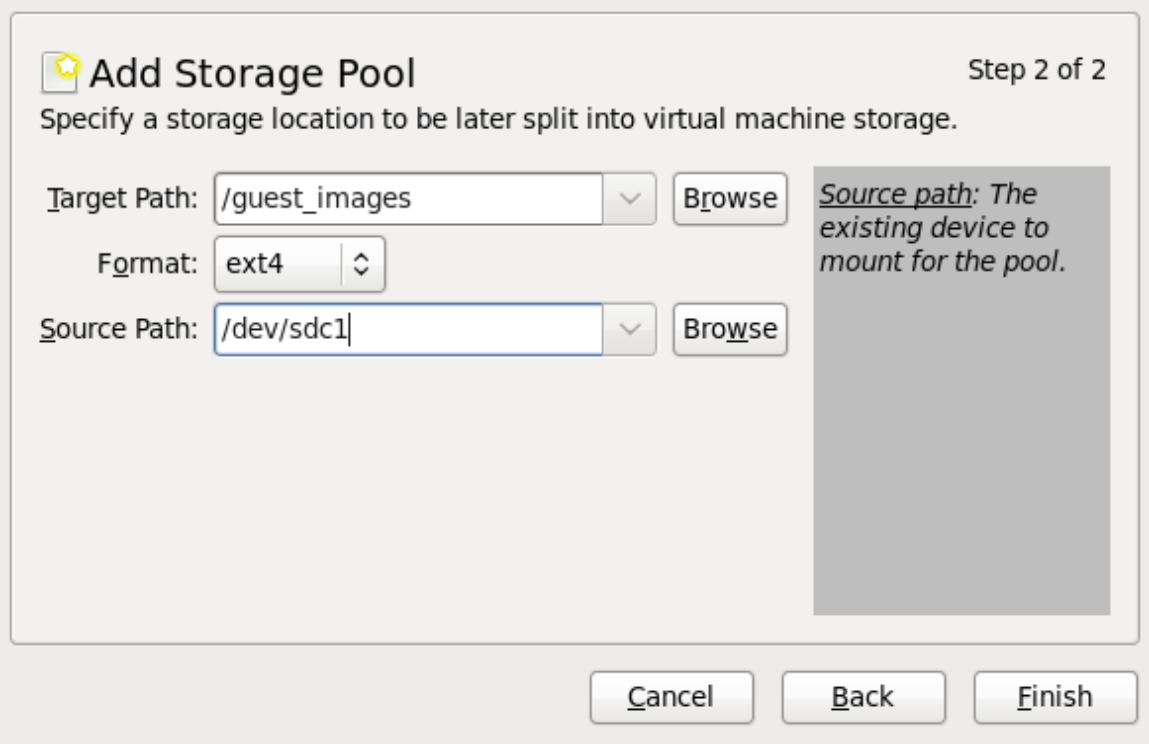
図12.3 ストレージプールの名前とタイプ



Forward ボタンを押して続行します。

b. **新しいプールの追加 (パート 2)**
Target Path、**Format**、および **Source Path** フィールドを変更します。

図12.4 ストレージプールのパスと形式



ターゲットパス

Target Path フィールドに、ストレージプールのソースデバイスをマウントする場所を入力します。場所がまだ存在しない場合は、**virt-manager** がディレクトリーを作成します。

形式

Format リストからフォーマットを選択します。デバイスは選択したフォーマットでフォーマットされています。

この例では、デフォルトの Red Hat Enterprise Linux ファイルシステムである *ext4* ファイルシステムを使用しています。

ソースパス

Source Path フィールドにデバイスを入力します。

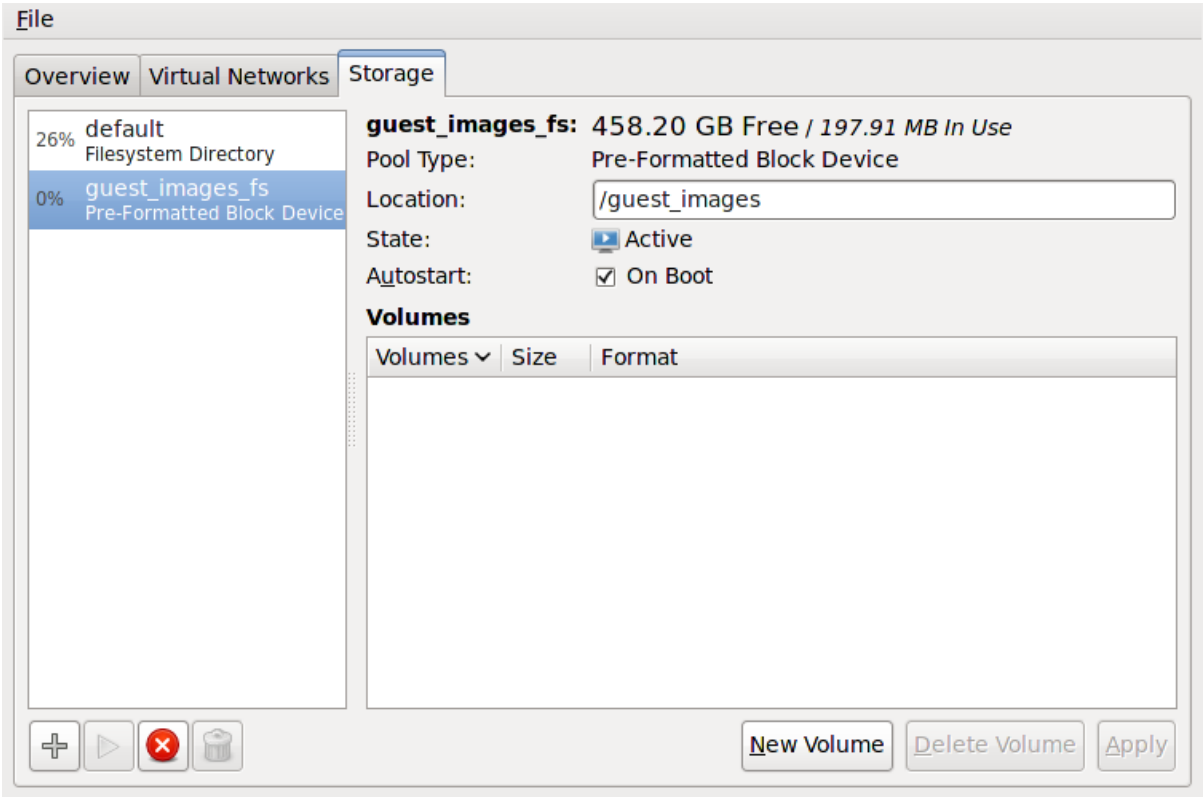
この例では、*/dev/sdc1* デバイスを使用しています。

詳細を確認し、**Finish** ボタンを押してストレージプールを作成します。

3. **新しいストレージプールを確認します**
新しいストレージプールは、数秒後に左側のストレージリストに表示されます。サイズが期待どおりに報告されていることを確認します。この例では *458.20 GB が空き* です。**State** フィールドが新しいストレージプールを *Active* としてレポートしていることを確認します。

ストレージプールを選択します。**Autostart** フィールドで、**On Boot** 時チェックボックスをクリックします。これにより、**libvirtd** サービスが開始するたびにストレージデバイスが確実に開始されます。

図12.5 ストレージリストの確認



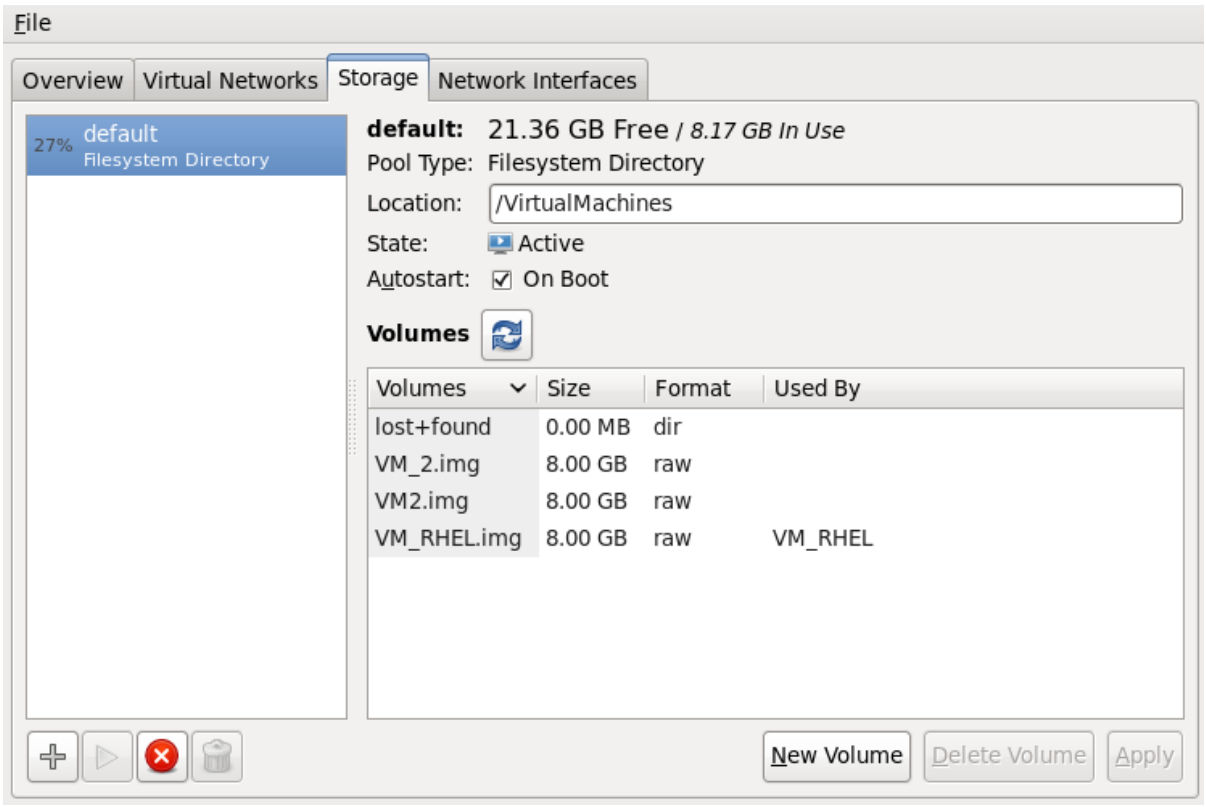
これでストレージプールが作成され、**Connection Details** ウィンドウを閉じます。

12.2.2. virt-manager を使用したストレージプールの削除

この手順は、ストレージプールを削除する方法を示しています。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。これを行うには、停止するストレージプールを選択し、ストレージウィンドウの下部にある赤い X アイコンをクリックします。

図12.6 停止アイコン



2. ごみ箱アイコンをクリックして、ストレージプールを削除します。このアイコンは、最初にストレージプールを停止した場合にのみ有効になります。

12.2.3. virsh を使用したパーティションベースのストレージプールの作成

このセクションでは、**virsh** コマンドを使用したパーティションベースのストレージプールの作成について説明します。

警告

この手順は、ディスク全体をストレージプール (**/dev/sdb** など) として割り当てるのに使用しないでください。ゲストには、ディスク全体またはブロックデバイスへの書き込みアクセス権を付与しないでください。この方法は、パーティション (たとえば、**/dev/sdb1**) をストレージプールに割り当てる場合にのみ使用してください。

手順12.2 virsh を使用して事前にフォーマットされたブロックデバイスストレージプールを作成する

1. **ストレージプール定義を作成します**
   virsh **pool-define-as** コマンドを使用して、新しいストレージプール定義を作成します。事前にフォーマットされたディスクをストレージプールとして定義するには、次の 3 つのオプションを提供する必要があります。

   パーティション名

nameパラメーターは、ストレージプールの名前を決定します。この例では、以下の例で *guest_images_fs* という名前を使用しています。

device

deviceパラメーターに***path***属性を指定すると、ストレージデバイスのデバイスパスが指定されます。この例では、パーティション */dev/sdc1* を使用しています。

mountpoint

フォーマットされたデバイスがマウントされるローカルファイルシステム上の***mountpoint***。マウントポイントディレクトリーが存在しない場合、**virsh** コマンドでディレクトリーを作成できます。

この例では、ディレクトリー */guest_images* が使用されています。

```
# virsh pool-define-as guest_images_fs fs - - /dev/sdc1 - "/guest_images"
Pool guest_images_fs defined
```

これで、新しいプールとマウントポイントが作成されました。

2. **新しいプールを確認する**
現在のストレージプールを一覧表示します。

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_fs      inactive   no
```

3. **マウントポイントを作成します。**
virsh pool-build コマンドを使用して、事前にフォーマットされたファイルシステムストレージプールのマウントポイントを作成します。

```
# virsh pool-build guest_images_fs
Pool guest_images_fs built
# ls -la /guest_images
total 8
drwx------.  2 root root 4096 May 31 19:38 .
dr-xr-xr-x. 25 root root 4096 May 31 19:38 ..
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_fs      inactive   no
```

4. **ストレージプールを起動します。**
virsh pool-start コマンドを使用して、ファイルシステムをマウントポイントにマウントし、プールを使用できるようにします。

```
# virsh pool-start guest_images_fs
Pool guest_images_fs started
# virsh pool-list --all
Name                 State      Autostart
```

```
----------------------------------------
default              active     yes
guest_images_fs      active     no
```

5. **自動起動をオンにします**
   デフォルトでは、**virsh** で定義されたストレージプールは、**libvirtd** が起動するたびに自動的に起動するように設定されていません。これを修正するには、**virsh pool-autostart** コマンドで自動開始を有効にします。ストレージプールは、**libvirtd** が起動するたびに自動的に起動するようになりました。

```
# virsh pool-autostart guest_images_fs
Pool guest_images_fs marked as autostarted

# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_fs      active     yes
```

6. **ストレージプールを確認します。**
   ストレージプールが正しく作成され、報告されたサイズが期待どおりで、状態が **running** と報告されたことを確認します。ファイルシステムのマウントポイントに lost+found ディレクトリーがあり、デバイスがマウントされていることを確認します。

```
# virsh pool-info guest_images_fs
Name:           guest_images_fs
UUID:           c7466869-e82a-a66c-2187-dc9d6f0877d0
State:          running
Persistent:     yes
Autostart:      yes
Capacity:       458.39 GB
Allocation:     197.91 MB
Available:      458.20 GB
# mount | grep /guest_images
/dev/sdc1 on /guest_images type ext4 (rw)
# ls -la /guest_images
total 24
drwxr-xr-x.  3 root root  4096 May 31 19:47 .
dr-xr-xr-x. 25 root root  4096 May 31 19:38 ..
drwx------.  2 root root 16384 May 31 14:18 lost+found
```

12.2.4. virsh を使用したストレージプールの削除

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。

```
# virsh pool-destroy guest_images_disk
```

2. オプションで、ストレージプールが存在するディレクトリーを削除する場合は、次のコマンドを使用します。

```
# virsh pool-delete guest_images_disk
```

3. ストレージプールの定義を削除します。

```
# virsh pool-undefine guest_images_disk
```

12.3. ディレクトリーベースのストレージプール

このセクションでは、ゲスト仮想マシンをホスト物理マシン上のディレクトリーに保存する方法について説明します。

ディレクトリーベースのストレージプールは、**virt-manager** または **virsh** コマンドラインツールを使用して作成できます。

12.3.1. virt-manager を使用したディレクトリーベースのストレージプールの作成

1. **ローカルディレクトリーを作成します**

   a. **オプション: ストレージプール用の新しいディレクトリーを作成します**
   ストレージプール用のディレクトリーをホスト物理マシンに作成します。この例では、*/guest virtual machine_images* という名前のディレクトリーを使用しています。

   ```
   # mkdir /guest_images
   ```

   b. **ディレクトリーの所有権を設定する**
   ディレクトリーのユーザーとグループの所有権を変更します。ディレクトリーは root ユーザーが所有している必要があります。

   ```
   # chown root:root /guest_images
   ```

   c. **ディレクトリーのアクセス許可を設定する**
   ディレクトリーのファイル権限を変更します。

   ```
   # chmod 700 /guest_images
   ```

   d. **変更を確認します。**
   権限が変更されたことを確認します。出力には、正しく設定された空のディレクトリーが表示されます。

   ```
   # ls -la /guest_images
   total 8
   drwx------.  2 root root 4096 May 28 13:57 .
   dr-xr-xr-x. 26 root root 4096 May 28 13:57 ..
   ```

2. **SELinux ファイルコンテキストを設定する**
   新しいディレクトリーの正しい SELinux コンテキストを設定します。プールの名前とディレクトリーは一致している必要はないことに注意してください。ただし、ゲスト仮想マシンをシャットダウンすると、libvirt はコンテキストをデフォルト値に戻す必要があります。ディレクトリーのコンテキストによって、このデフォルト値が決まります。ディレクトリー virt_image_t に明示的にラベルを付けることは価値があります。これにより、ゲスト仮想マシンがシャットダウンされると、イメージに virt_image_t というラベルが付けられ、ホスト物理マシンで実行されている他のプロセスから分離されます。

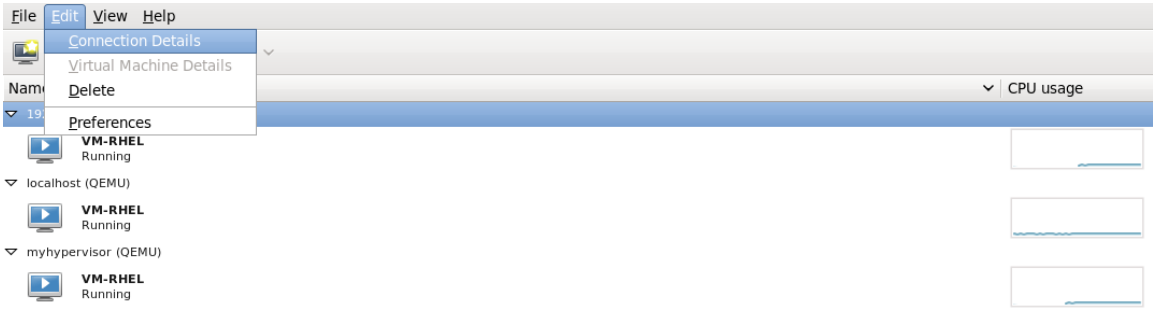
```
# semanage fcontext -a -t virt_image_t '/guest_images(/.*)?'
# restorecon -R /guest_images
```

3. **ストレージプールの設定を開きます**

   a. **virt-manager** のグラフィカルインターフェイスで、メインウィンドウからホストの物理マシンを選択します。

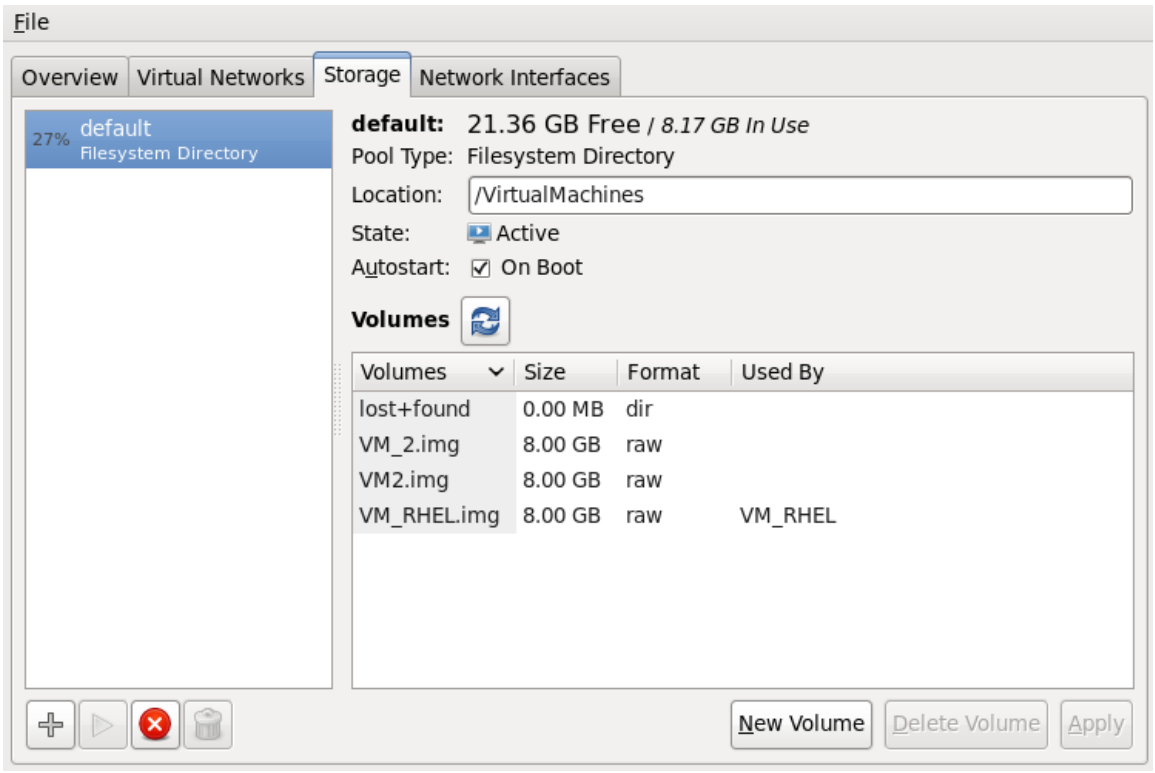
   Edit メニューを開き、**Connection Details** を選択します。

   図12.7 接続詳細ウィンドウ



   b. **Connection Details** ウィンドウの **Storage** タブをクリックします。

   図12.8 ストレージタブ

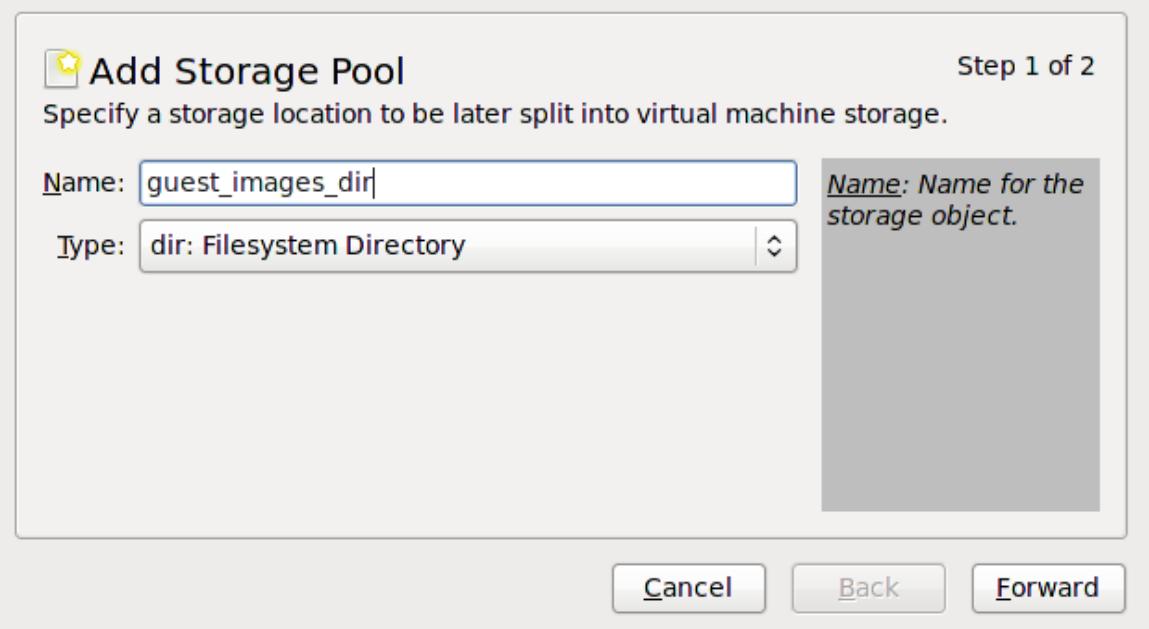


4. **新しいストレージプールを作成します**

a. **新しいプールの追加 (パート 1)**
+ ボタン (プールの追加ボタン) を押します。**Add a New Storage Pool** ウィザードが表示されます。

ストレージプールの **Name** を選択します。この例では、*guest_images* という名前を使用します。**Type** を **dir: Filesystem Directory** に変更します。

図12.9 ストレージプールに名前を付ける



Forward ボタンを押して続行します。

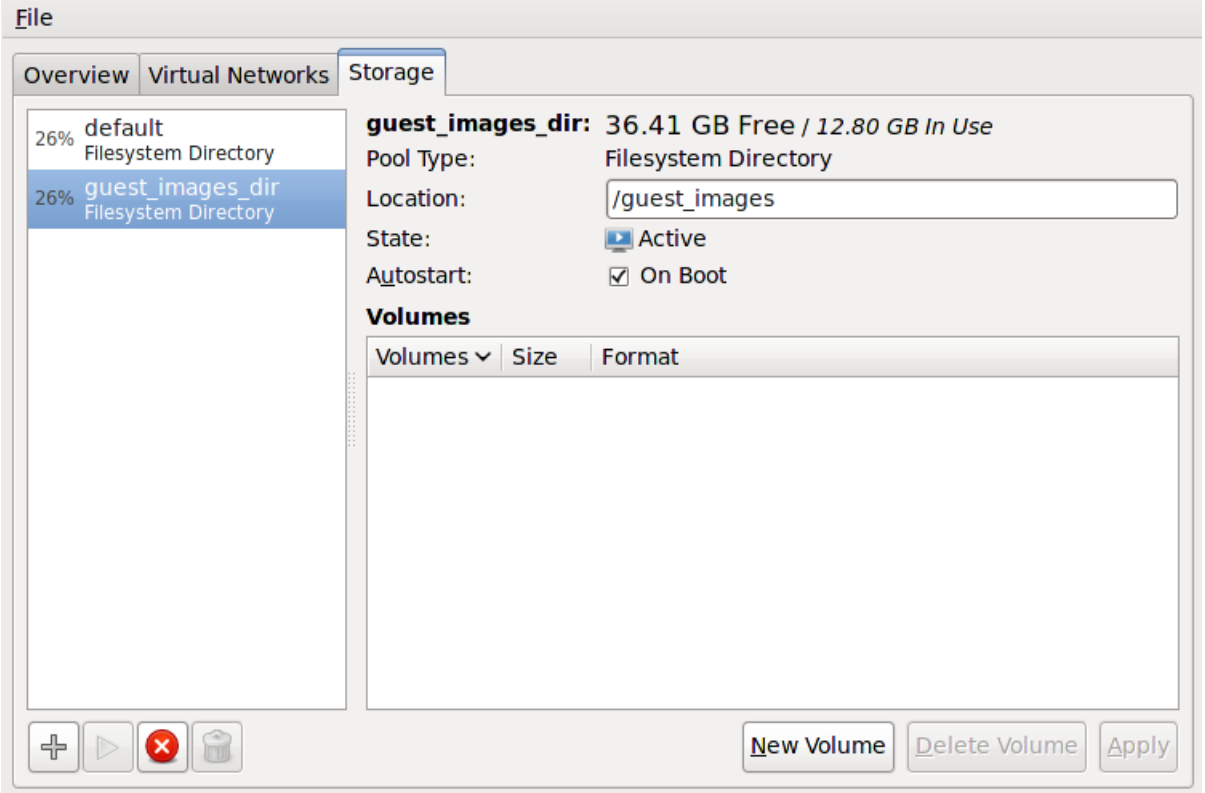
b. **新しいプールの追加 (パート 2)**
Target Path フィールドを変更します。たとえば、*/guest_images*。

詳細を確認し、**Finish** ボタンを押してストレージプールを作成します。

5. **新しいストレージプールを確認します**
新しいストレージプールは、数秒後に左側のストレージリストに表示されます。サイズが期待どおりに報告されていることを確認します。この例では *36.41 GB の空き* 容量です。**State** フィールドが新しいストレージプールを *Active* としてレポートしていることを確認します。

ストレージプールを選択します。**Autostart** フィールドで、**On Boot** チェックボックスがオンになっていることを確認します。これにより、**libvirtd** サービスが開始するたびにストレージプールが確実に開始されます。

図12.10 ストレージプール情報を確認します



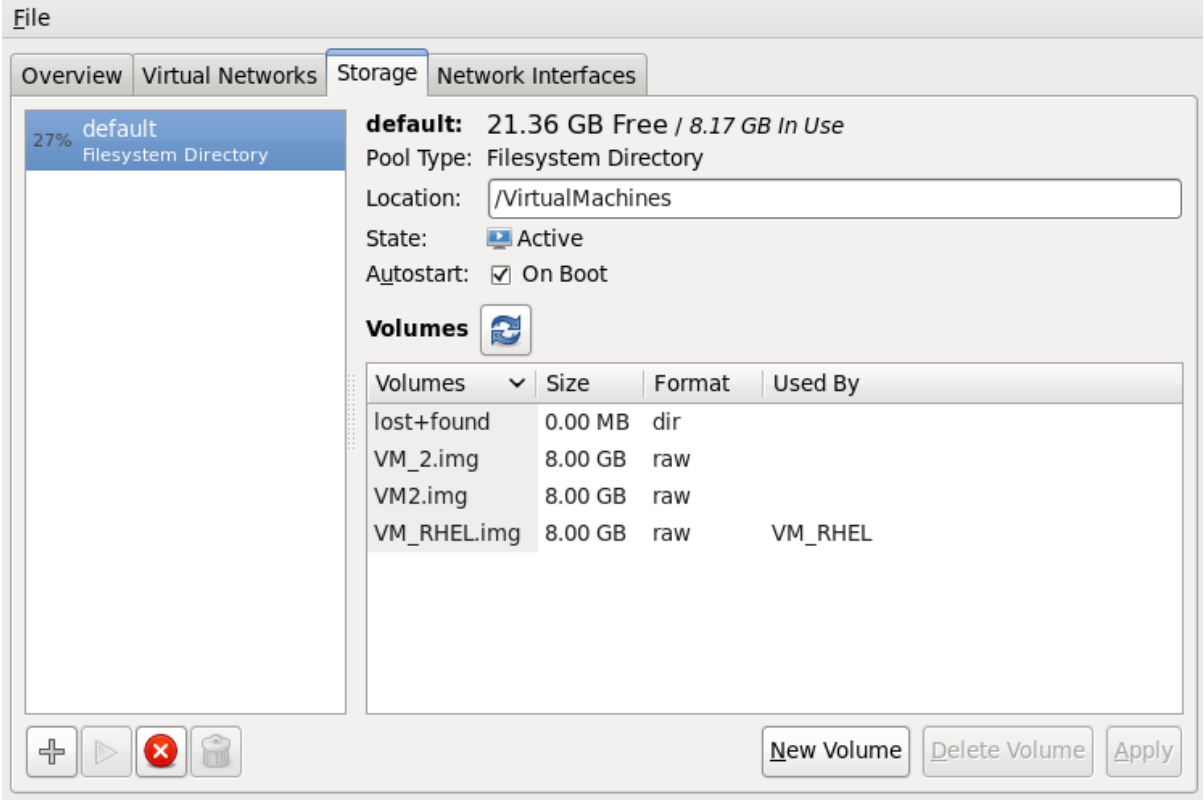
これでストレージプールが作成され、**Connection Details** ウィンドウを閉じます。

12.3.2. virt-manager を使用したストレージプールの削除

この手順は、ストレージプールを削除する方法を示しています。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。これを行うには、停止するストレージプールを選択し、ストレージウィンドウの下部にある赤い X アイコンをクリックします。

図12.11 停止アイコン



2. ごみ箱アイコンをクリックして、ストレージプールを削除します。このアイコンは、最初にストレージプールを停止した場合にのみ有効になります。

12.3.3. virsh を使用したディレクトリーベースのストレージプールの作成

1. **ストレージプール定義を作成します**
   virsh pool-define-as コマンドを使用して、新しいストレージプールを定義します。ディレクトリーベースのストレージプールを作成するには、次の 2 つのオプションが必要です。

   - ストレージプールの **name**

     この例では、*guest_images* という名前を使用します。この例で使用されている以降のすべての **virsh** コマンドは、この名前を使用します。

   - ゲストイメージファイルを保存するためのファイルシステムディレクトリーへの **path**。このディレクトリーが存在しない場合は、**virsh** が作成します。

     この例では、*/guest_images* ディレクトリーを使用しています。

```
# virsh pool-define-as guest_images dir - - - - "/guest_images"
Pool guest_images defined
```

2. **ストレージプールがリストされていることを確認します**
   ストレージプールオブジェクトが正しく作成され、状態が次のようにレポートすることを確認します **inactive**。

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
```

```
default              active     yes
guest_images     inactive   no
```

3. **ローカルディレクトリーを作成します**
   次に示すように、**virsh pool-build** コマンドを使用して、ディレクトリー *guest_images* のディレクトリーベースのストレージプールを構築します (たとえば)。

```
# virsh pool-build guest_images
Pool guest_images built
# ls -la /guest_images
total 8
drwx------.  2 root root 4096 May 30 02:44 .
dr-xr-xr-x. 26 root root 4096 May 30 02:44 ..
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images     inactive   no
```

4. **ストレージプールを起動します。**
   virsh コマンド **pool-start** を使用して、ディレクトリーストレージプールを有効にします。これにより、プールのボリュームをゲストディスクイメージとして使用できるようになります。

```
# virsh pool-start guest_images
Pool guest_images started
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default            active     yes
guest_images    active     no
```

5. **自動起動をオンにします**
   オンにする***autostart***ストレージプール用。Autostart は、サービスの開始時にストレージプールを開始するように **libvirtd** サービスを設定します。

```
# virsh pool-autostart guest_images
Pool guest_images marked as autostarted
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images         active     yes
```

6. **ストレージプールの設定を確認する**
   ストレージプールが正しく作成され、サイズが正しく報告され、状態が次のようにレポートされることを確認します**running**。ゲスト仮想マシンが実行されていない場合でもプールにアクセスできるようにするには、**Persistent** が **yes** として報告されていることを確認します。サービス開始時にプールを自動的に起動させたい場合は、**Autostart** が **yes** と報告されていることを確認します。

```
# virsh pool-info guest_images
Name:           guest_images
UUID:           779081bf-7a82-107b-2874-a19a9c51d24c
State:          running
```

```
Persistent:     yes
Autostart:      yes
Capacity:       49.22 GB
Allocation:     12.80 GB
Available:      36.41 GB

# ls -la /guest_images
total 8
drwx------.  2 root root 4096 May 30 02:44 .
dr-xr-xr-x. 26 root root 4096 May 30 02:44 ..
#
```

ディレクトリーベースのストレージプールが利用可能になりました。

12.3.4. virsh を使用したストレージプールの削除

次に、virsh を使用してストレージプールを削除する方法を示します。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。

```
# virsh pool-destroy guest_images_disk
```

2. オプションで、ストレージプールが存在するディレクトリーを削除する場合は、次のコマンドを使用します。

```
# virsh pool-delete guest_images_disk
```

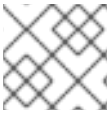
3. ストレージプールの定義を削除します。

```
# virsh pool-undefine guest_images_disk
```

12.4. LVM ベースのストレージプール

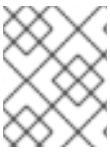
本章では、LVM ボリュームグループをストレージプールとして使用する方法について説明します。

LVM ベースのストレージグループは、LVM の完全な柔軟性を提供します。



注記

現在、LVM ベースのストレージプールではシンプロビジョニングはできません。



注記

LVM の詳細は、『Red Hat Enterprise Linux ストレージ管理ガイド』 を参照してください。

警告

LVM ベースのストレージプールには、完全なディスクパーティションが必要です。この手順で新しいパーティション/デバイスをアクティベートすると、パーティションはフォーマットされ、すべてのデータが削除されます。ホストの既存のボリュームグループ (VG) を使用すると、何も消去されません。以下の手順を開始する前に、ストレージデバイスのバックアップを作成することが推奨されます。

12.4.1. LVM ベースのストレージプールの作成 virt-manager を使用します。

LVM ベースのストレージプールは、既存の LVM ボリュームグループを使用することも、空のパーティションに新しい LVM ボリュームグループを作成することもできます。

1. **オプション:LVM ボリューム用の新しいパーティションを作成します**
   これらの手順では、新しいハードディスクドライブに新しいパーティションと LVM ボリュームグループを作成する方法について説明します。

警告

この手順により、選択したストレージデバイスからすべてのデータが削除されます。

   a. **新しいパーティションの作成**
   fdisk コマンドを使用して、コマンドラインから新しいディスクパーティションを作成します。次の例では、ストレージデバイス **/dev/sdb** にディスク全体を使用する新しいパーティションを作成しています。

```
# fdisk /dev/sdb
Command (m for help):
```

   nを押して、新しいパーティションを作成します。

   b. ***p***を押して、プライマリーパーティションにします。

```
Command action
   e   extended
   p   primary partition (1-4)
```

   c. 使用可能なパーティション番号を選択します。この例では、最初のパーティションは次のように入力して選択されます ***1***。

```
Partition number (1-4): 1
```

   d. ***Enter***を押して、デフォルトの最初のシリンダーを入力します。

```
First cylinder (1-400, default 1):
```

e. パーティションのサイズを選択します。この例では、***Enter*** を押してディスク全体を割り当てています。

```
Last cylinder or +size or +sizeM or +sizeK (2-400, default 400):
```

f. ***t*** を押して、パーティションの種類を設定します。

```
Command (m for help): t
```

g. 前の手順で作成したパーティションを選択します。この例では、パーティション番号は ***1***。

```
Partition number (1-4): 1
```

h. Linux LVM パーティションの場合、***8e*** を入力します。

```
Hex code (type L to list codes): 8e
```

i. 変更をディスクに書き込んで終了します。

```
Command (m for help): w
Command (m for help): q
```

j. **新しい LVM ボリュームグループを作成する**
vgcreate コマンドを使用して新しい LVM ボリュームグループを作成します。この例では、*guest_images_lvm* という名前のボリュームグループを作成します。

```
# vgcreate guest_images_lvm /dev/sdb1
  Physical volume "/dev/vdb1" successfully created
  Volume group "guest_images_lvm" successfully created
```

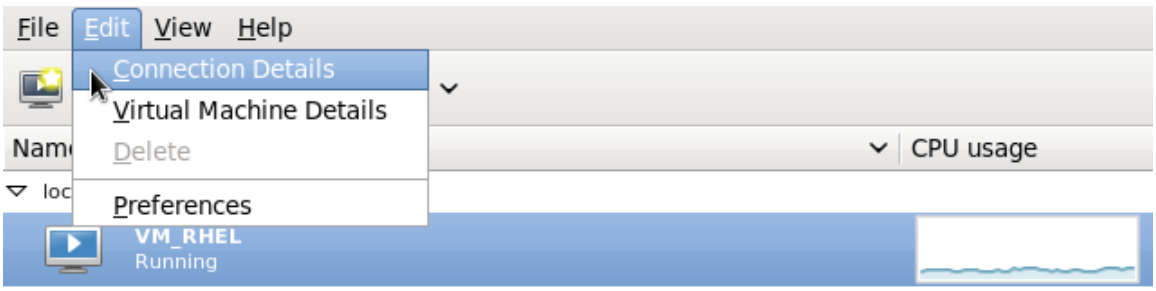
新しい LVM ボリュームグループ *guest_images_lvm* を、LVM ベースのストレージプールに使用できるようになりました。

2. **ストレージプールの設定を開きます**

a. **virt-manager** のグラフィカルインターフェイスで、メインウィンドウからホストを選択します。

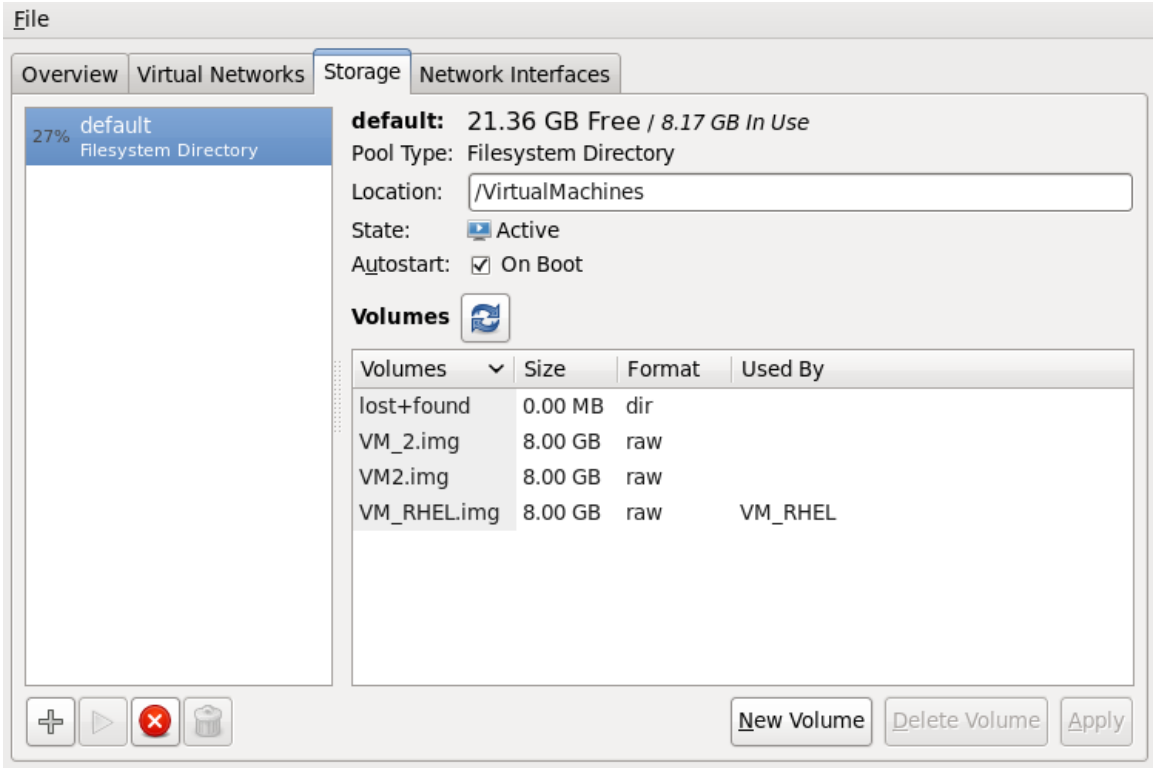
Edit メニューを開き、**Connection Details** を選択します。

図12.12 接続の詳細



b. **Storage** タブをクリックします。

図12.13 ストレージタブ

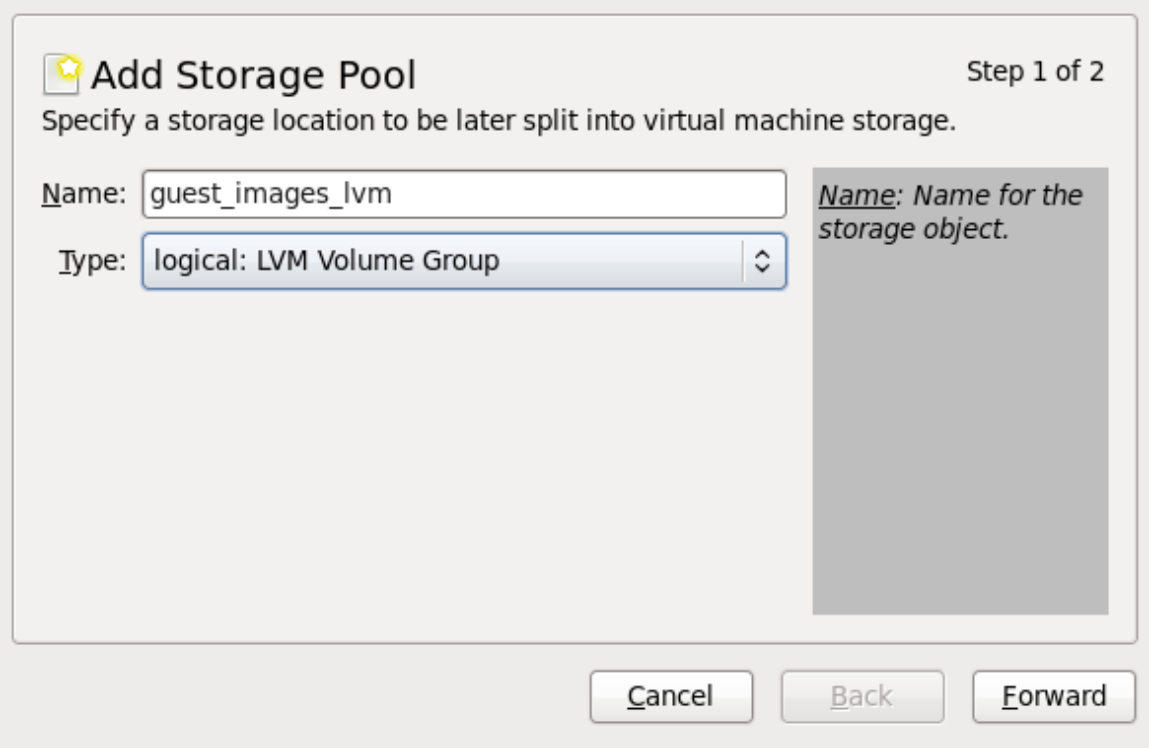


3. **新しいストレージプールを作成します**

a. **ウィザードを開始します**
+ ボタン (プールの追加ボタン) を押します。**Add a New Storage Pool** ウィザードが表示されます。

ストレージプールの **Name** を選択します。この例では、*guest_images_lvm* を使用します。次に、**Type** を **logical: LVM Volume Group** に変更します。また、

図12.14 LVM ストレージプールを追加する



Forward ボタンを押して続行します。

b. **新しいプールの追加 (パート 2)**
Target Path フィールドを変更します。この例では */guest_images* を使用しています。

次に、**Target Path** フィールドと **Source Path** フィールドに入力し、**Build Pool** チェックボックスをオンにします。

- **Target Path** フィールドを使用して、既存の LVM ボリュームグループを選択する **か**、新しいボリュームグループの名前として使用します。デフォルトの形式は /**dev**/*storage_pool_name*。

  この例では、*/dev/guest_images_lvm* という名前の新しいボリュームグループを使用しています。

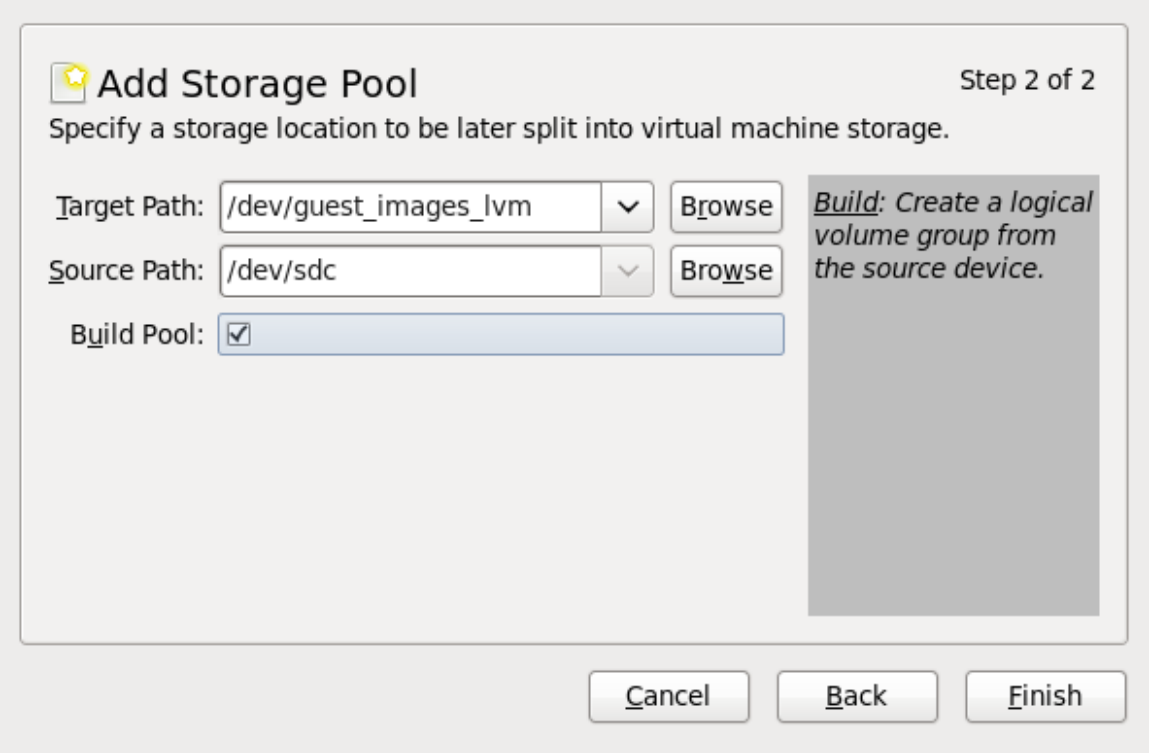
- 既存の LVM ボリュームグループが **Target Path** で使用されている場合、**Source Path** フィールドはオプションです。

  新しい LVM ボリュームグループの場合は、**Source Path** フィールドにストレージデバイスの場所を入力します。この例では、空白のパーティション */dev/sdc* を使用しています。

- **Build Pool** チェックボックスは、**virt-manager** に新しい LVM ボリュームグループを作成するように指示します。既存のボリュームグループを使用している場合は、**Build Pool** チェックボックスを選択しないでください。

  この例では、空白のパーティションを使用して新しいボリュームグループを作成しているため、**Build Pool** チェックボックスをオンにする必要があります。

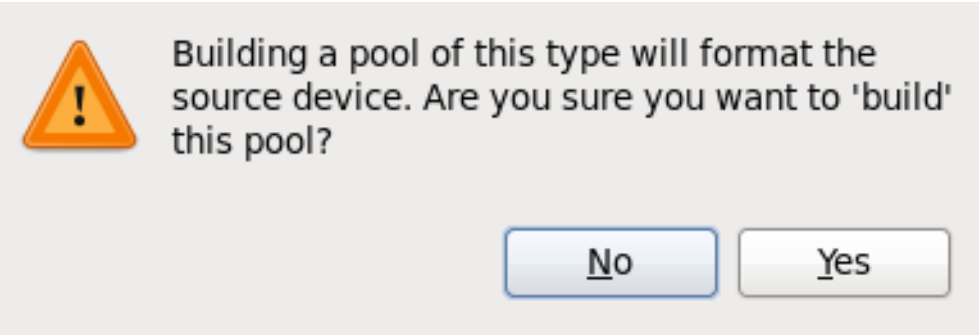
図12.15 ターゲットとソースを追加する



詳細を確認し、**Finish** ボタンを押して LVM ボリュームグループをフォーマットし、ストレージプールを作成します。

c. **フォーマットするデバイスを確認します**
警告メッセージが表示されます。

図12.16 警告メッセージ

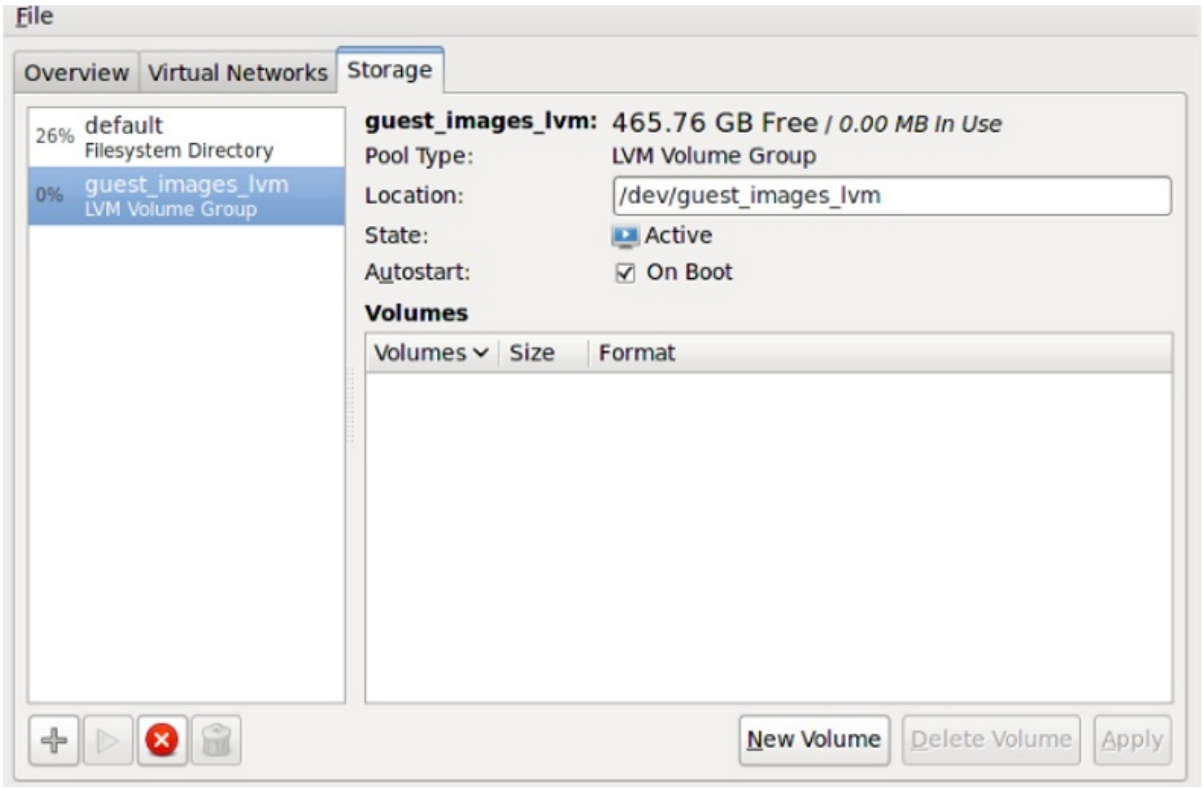


Yes ボタンを押して、ストレージデバイス上のすべてのデータを消去し、ストレージプールを作成します。

4. **新しいストレージプールを確認します**
新しいストレージプールは、数秒後に左側のリストに表示されます。詳細が期待どおりであることを確認します。この例では、*465.76 GB が空き*です。また、**State** フィールドが新しいストレージプールを*Active* としてレポートしていることを確認します。

ストレージプールが libvirtd で自動的に開始されるようにするには、通常、**Autostart** チェックボックスを有効にすることをお勧めします。

図12.17 LVM ストレージプールの詳細を確認する



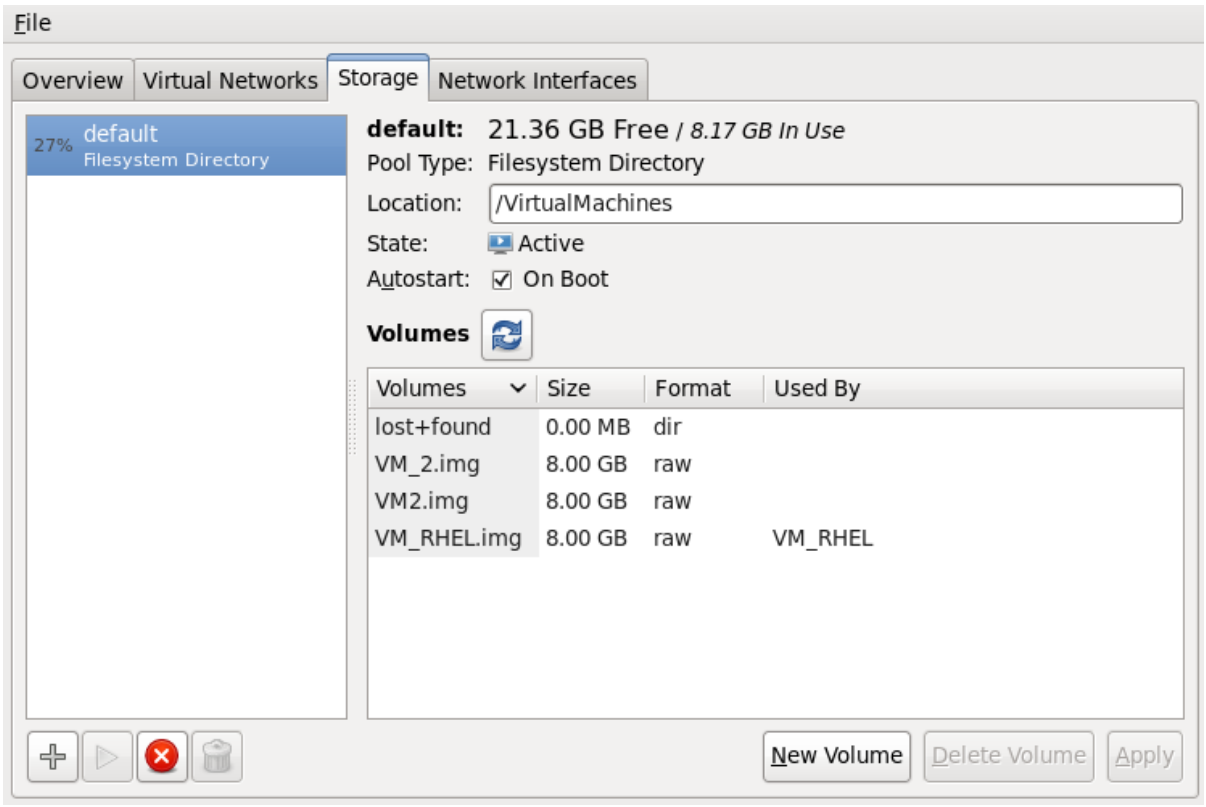
タスクが完了したら、ホストの詳細ダイアログを閉じます。

12.4.2. virt-manager を使用したストレージプールの削除

この手順は、ストレージプールを削除する方法を示しています。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。これを行うには、停止するストレージプールを選択し、ストレージウィンドウの下部にある赤い X アイコンをクリックします。

図12.18 停止アイコン



2. ごみ箱アイコンをクリックして、ストレージプールを削除します。このアイコンは、最初にストレージプールを停止した場合にのみ有効になります。

12.4.3. virsh を使用した LVM ベースのストレージプールの作成

このセクションでは、**virsh** コマンドを使用して LVM ベースのストレージプールを作成するために必要な手順の概要を説明します。単一のドライブ (**/dev/sdc**) からの **guest_images_lvm** という名前のプールの例を使用します。これは単なる例であり、設定は必要に応じて置き換える必要があります。

手順12.3 virsh を使用した LVM ベースのストレージプールの作成

1. プール名 **guest_images_lvm** を定義します。

```
# virsh pool-define-as guest_images_lvm logical - - /dev/sdc libvirt_lvm \ /dev/libvirt_lvm
Pool guest_images_lvm defined
```

2. 指定された名前に従ってプールを構築します。既存のボリュームグループを使用している場合は、この手順をスキップしてください。

```
# virsh pool-build guest_images_lvm

Pool guest_images_lvm built
```

3. 新しいプールを初期化します。

```
# virsh pool-start guest_images_lvm

Pool guest_images_lvm started
```

4. **vgs** コマンドを使用してボリュームグループ情報を表示します。

```
# vgs
VG          #PV #LV #SN Attr   VSize   VFree
libvirt_lvm   1   0   0 wz--n- 465.76g 465.76g
```

5. 自動的に開始するようにプールを設定します。

```
# virsh pool-autostart guest_images_lvm
Pool guest_images_lvm marked as autostarted
```

6. **virsh** コマンドを使用して、使用可能なプールを一覧表示します。

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
guest_images_lvm     active     yes
```

7. 次のコマンドは、このプール内に 3 つのボリューム (**volume1**、**volume2**、および **volume3**) を作成する方法を示しています。

```
# virsh vol-create-as guest_images_lvm volume1 8G
Vol volume1 created

# virsh vol-create-as guest_images_lvm volume2 8G
Vol volume2 created

# virsh vol-create-as guest_images_lvm volume3 8G
Vol volume3 created
```

8. **virsh** コマンドを使用して、このプールで使用可能なボリュームをリストします。

```
# virsh vol-list guest_images_lvm
Name                 Path
-----------------------------------------
volume1              /dev/libvirt_lvm/volume1
volume2              /dev/libvirt_lvm/volume2
volume3              /dev/libvirt_lvm/volume3
```

9. 次の 2 つのコマンド (**lvscan** および **lvs**) は、新しく作成されたボリュームに関する詳細情報を表示します。

```
# lvscan
ACTIVE            '/dev/libvirt_lvm/volume1' [8.00 GiB] inherit
ACTIVE            '/dev/libvirt_lvm/volume2' [8.00 GiB] inherit
ACTIVE            '/dev/libvirt_lvm/volume3' [8.00 GiB] inherit

# lvs
LV      VG          Attr     LSize   Pool Origin Data%  Move Log Copy%  Convert
volume1 libvirt_lvm -wi-a-   8.00g
volume2 libvirt_lvm -wi-a-   8.00g
volume3 libvirt_lvm -wi-a-   8.00g
```

12.4.4. virsh を使用したストレージプールの削除

次に、virsh を使用してストレージプールを削除する方法を示します。

1. 同じプールを使用している他のゲストとの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。

```
# virsh pool-destroy guest_images_disk
```

2. オプションで、ストレージプールが存在するディレクトリーを削除する場合は、次のコマンドを使用します。

```
# virsh pool-delete guest_images_disk
```

3. ストレージプールの定義を削除します。

```
# virsh pool-undefine guest_images_disk
```

12.5. ISCSI ベースのストレージプール

このセクションでは、iSCSI ベースのデバイスを使用してゲスト仮想マシンを格納する方法について説明します。

iSCSI (Internet Small Computer System Interface) は、ストレージデバイスを共有するネットワークプロトコルです。iSCSI は、IP 層で SCSI 命令を使用してイニシエーター (ストレージクライアント) をターゲット (ストレージサーバー) に接続します。

12.5.1. ソフトウェア iSCSI ターゲットの設定

scsi-target-utils パッケージは、ソフトウェアでバックアップされた iSCSI ターゲットを作成するためのツールです。

手順12.4 iSCSI ターゲットの作成

1. **必要なパッケージのインストール**
scsi-target-utils パッケージと依存するすべてのパッケージをインストールします。

```
# yum install scsi-target-utils
```

2. **tgtd サービスを開始します**
tgtd サービスは物理マシンの SCSI ターゲットをホストし、iSCSI プロトコルを使用して物理マシンターゲットをホストします。**tgtd** サービスを開始し、**chkconfig** コマンドで再起動した後、サービスを永続化します。

```
# service tgtd start
# chkconfig tgtd on
```

3. **オプション:LVM ボリュームを作成する**
LVM ボリュームは、iSCSI バッキングイメージに役立ちます。LVM スナップショットとサイズ変更は、ゲスト仮想マシンにとって有益な場合があります。この例では、iSCSI を使用してゲスト仮想マシンをホストするために、RAID5 アレイ上の *virtstore* という名前の新しいボリュームグループに *virtimage1* という名前の LVM イメージを作成します。

a. **RAID アレイを作成する**
ソフトウェア RAID5 アレイの作成については、『Red Hat Enterprise Linux デプロイメントガイド』 で説明されています。

b. **LVM ボリュームグループを作成する**
vgcreate コマンドを使用して、*virtstore* という名前のボリュームグループを作成します。

```
# vgcreate virtstore /dev/md1
```

c. **LVM 論理ボリュームを作成する**
lvcreate コマンドを使用して、サイズが 20GB の *virtstore* ボリュームグループに *virtimage1* という名前の論理ボリュームグループを作成します。

```
# lvcreate --size 20G -n virtimage1 virtstore
```

新しい論理ボリューム *virtimage1* は、iSCSI で使用する準備ができています。

4. **オプション: ファイルベースのイメージを作成します**
テストにはファイルベースのストレージで十分ですが、実稼働環境や重要な I/O アクティビティーにはお勧めしません。このオプションの手順では、iSCSI ターゲット用に *virtimage2.img* という名前のファイルベースのイメージを作成します。

a. **イメージの新しいディレクトリーを作成します**
イメージを保存するための新しいディレクトリーを作成します。ディレクトリーには正しい SELinux コンテキストが必要です。

```
# mkdir -p /var/lib/tgtd/virtualization
```

b. **イメージファイルを作成する**
サイズが 10GB の *virtimage2.img* という名前のイメージを作成します。

```
# dd if=/dev/zero of=/var/lib/tgtd/virtualization/virtimage2.img bs=1M seek=10000 count=0
```

c. **SELinux ファイルコンテキストを設定する**
新しいイメージとディレクトリーの正しい SELinux コンテキストを設定します。

```
# restorecon -R /var/lib/tgtd
```

新しいファイルベースのイメージ *virtimage2.img* は、iSCSI で使用する準備ができています。

5. **ターゲットを作成する**
ターゲットは、XML エントリーを **/etc/tgt/targets.conf** ファイルに追加することで作成できます。**target** 属性には、iSCSI Qualified Name (IQN) が必要です。IQN の形式は次のとおりです。

```
iqn.yyyy-mm.reversed domain name:optional identifier text
```

詳細は以下のようになります。

- *yyyy-mm*は、デバイスが開始された年と月を表します (例: *2010-05*)。

- *逆ドメイン名* は、逆のホスト物理マシンのドメイン名です (たとえば、IQN の *server1.example.com* は *com.example.server1* になります)。と
- *オプションの識別子テキスト* は、スペースを含まない任意のテキスト文字列であり、管理者がデバイスまたはハードウェアを識別するのに役立ちます。

この例では、*server1.example.com* のオプションの手順で作成された 2 種類のイメージの iSCSI ターゲットを、オプションの識別子 *トライアル* を使用して作成します。**/etc/tgt/targets.conf** ファイルに以下を追加します。

```
<target iqn.2010-05.com.example.server1:iscsirhel6guest>
  backing-store /dev/virtstore/virtimage1  #LUN 1
  backing-store /var/lib/tgtd/virtualization/virtimage2.img  #LUN 2
  write-cache off
</target>
```

/etc/tgt/targets.conf ファイルに**default-driver iscsi**ドライバータイプを iSCSI として設定する行。ドライバーはデフォルトで iSCSI を使用します。

重要

この例では、アクセス制御なしでグローバルにアクセス可能なターゲットを作成します。安全なアクセスの実装については、scsi-target-utils を参照してください。

6. **tgtd サービスを再起動します**
   tgtd サービスを再起動して、設定の変更を再ロードします。

```
# service tgtd restart
```

7. **iptable の設定**
   iptables を使用した iSCSI アクセス用にポート 3260 を開きます。

```
# iptables -I INPUT -p tcp -m tcp --dport 3260 -j ACCEPT
# service iptables save
# service iptables restart
```

8. **新しいターゲットを確認します**
   新しいターゲットを表示して、**tgt-admin--show** コマンドでセットアップが成功したことを確認します。

```
# tgt-admin --show
Target 1: iqn.2010-05.com.example.server1:iscsirhel6guest
System information:
Driver: iscsi
State: ready
I_T nexus information:
LUN information:
LUN: 0
    Type: controller
    SCSI ID: IET     00010000
    SCSI SN: beaf10
    Size: 0 MB
    Online: Yes
```

```
    Removable media: No
    Backing store type: rdwr
    Backing store path: None
LUN: 1
    Type: disk
    SCSI ID: IET     00010001
    SCSI SN: beaf11
    Size: 20000 MB
    Online: Yes
    Removable media: No
    Backing store type: rdwr
    Backing store path: /dev/virtstore/virtimage1
LUN: 2
    Type: disk
    SCSI ID: IET     00010002
    SCSI SN: beaf12
    Size: 10000 MB
    Online: Yes
    Removable media: No
    Backing store type: rdwr
    Backing store path: /var/lib/tgtd/virtualization/virtimage2.img
Account information:
ACL information:
ALL
```

警告

ACL リストは all に設定されています。これにより、ローカルネットワーク上のすべてのシステムがこのデバイスにアクセスできるようになります。実稼働環境用にホスト物理マシンアクセス ACL を設定することをお勧めします。

9. **オプション: テスト検出**
   新しい iSCSI デバイスが検出可能かどうかをテストします。

```
# iscsiadm --mode discovery --type sendtargets --portal server1.example.com
127.0.0.1:3260,1 iqn.2010-05.com.example.server1:iscsirhel6guest
```

10. **オプション: デバイスの接続をテストします**
    新しいデバイス (*iqn.2010-05.com.example.server1:iscsirhel6guest*) を接続して、デバイスを接続できるかどうかを判断します。

```
# iscsiadm -d2 -m node --login
scsiadm: Max file limits 1024 1024

Logging in to [iface: default, target: iqn.2010-05.com.example.server1:iscsirhel6guest, portal:
10.0.0.1,3260]
Login to [iface: default, target: iqn.2010-05.com.example.server1:iscsirhel6guest, portal:
10.0.0.1,3260] successful.
```

デバイスの取り外し

```
# iscsiadm -d2 -m node --logout
scsiadm: Max file limits 1024 1024

Logging out of session [sid: 2, target: iqn.2010-05.com.example.server1:iscsirhel6guest,
portal: 10.0.0.1,3260
Logout of [sid: 2, target: iqn.2010-05.com.example.server1:iscsirhel6guest, portal:
10.0.0.1,3260] successful.
```

これで、iSCSI デバイスを仮想化に使用する準備が整いました。

12.5.2. virt-manager への iSCSI ターゲットの追加

この手順では、**virt-manager** で iSCSI ターゲットを使用してストレージプールを作成する方法について説明します。

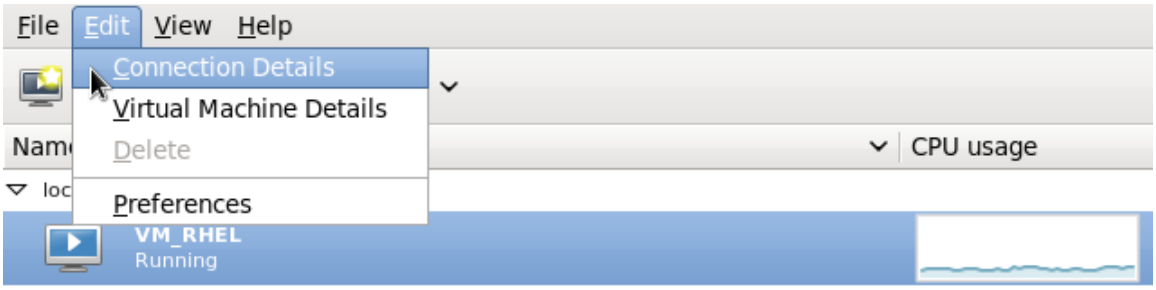
手順12.5 virt-manager への iSCSI デバイスの追加

1. **ホスト物理マシンのストレージタブを開きます**
   Connection Details ウィンドウの **Storage** タブを開きます。

   a. **virt-manager** を開きます。

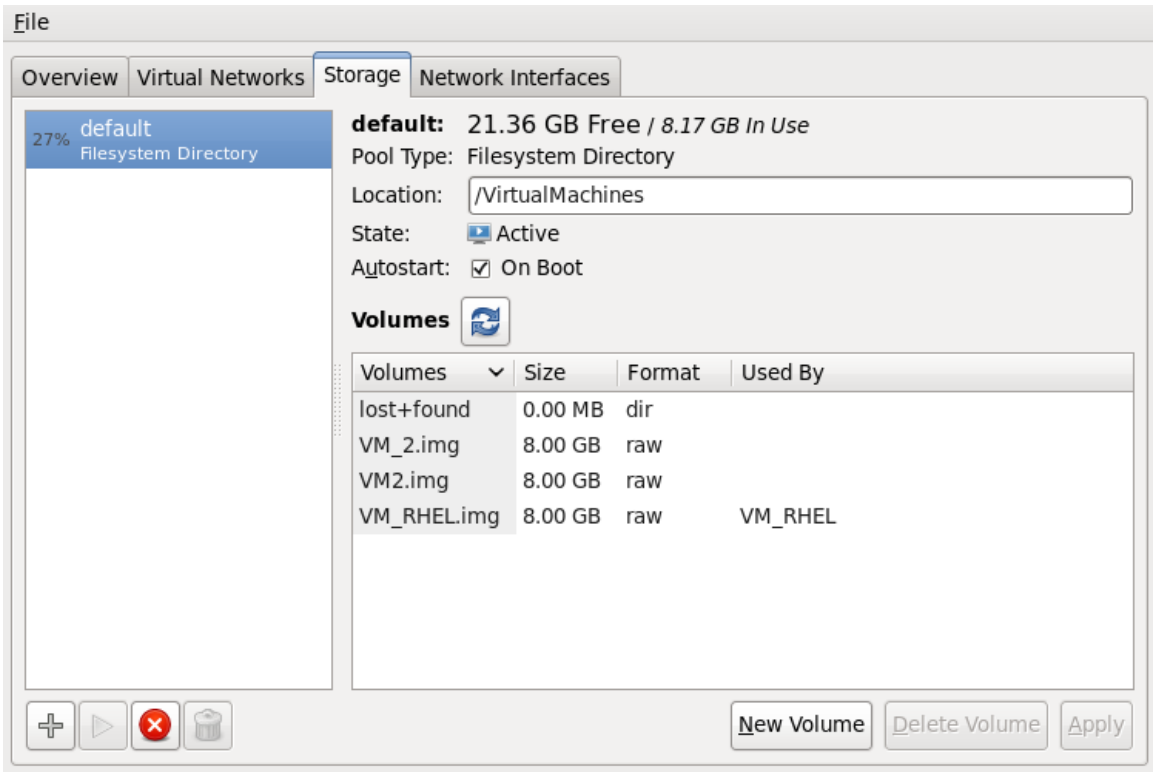
   b. メインの **virt-manager** ウィンドウからホスト物理マシンを選択します。**Edit menu** をクリックして、**Connection Details** を選択します。

   図12.19 接続の詳細



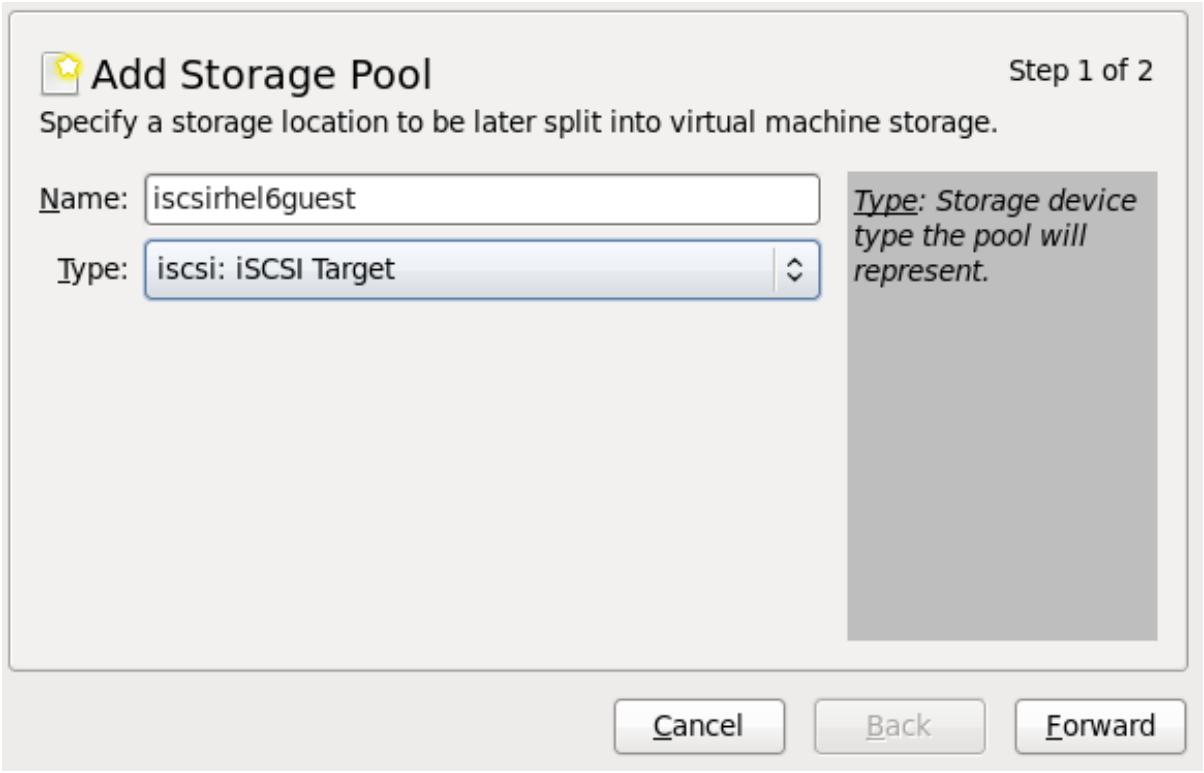
   c. **Storage** タブをクリックします。

図12.20 ストレージメニュー



2. **新しいプールの追加 (パート 1)**
   + ボタン (プールの追加ボタン) を押します。**Add a New Storage Pool** ウィザードが表示されます。

   図12.21 iscsi ストレージプールの名前とタイプを追加します



   ストレージプールの名前を選択し、タイプを iscsi に変更し、**Forward** を押して続行します。

3. **新しいプールの追加 (パート 2)**

このメニューのフィールドを完成させるには、「iSCSI ベースのストレージプール」 と 手順12.4「iSCSI ターゲットの作成」 で使用した情報が必要です。

a. iSCSI ソースとターゲットを入力します。フォーマットはゲスト仮想マシンによって処理されるため、**Format** オプションは使用できません。**Target Path** を編集することはお勧めしません。デフォルトのターゲットパス値 **/dev/disk/by-path/** は、そのディレクトリーにドライブパスを追加します。ターゲットパスは、移行するすべてのホスト物理マシンで同じである必要があります。

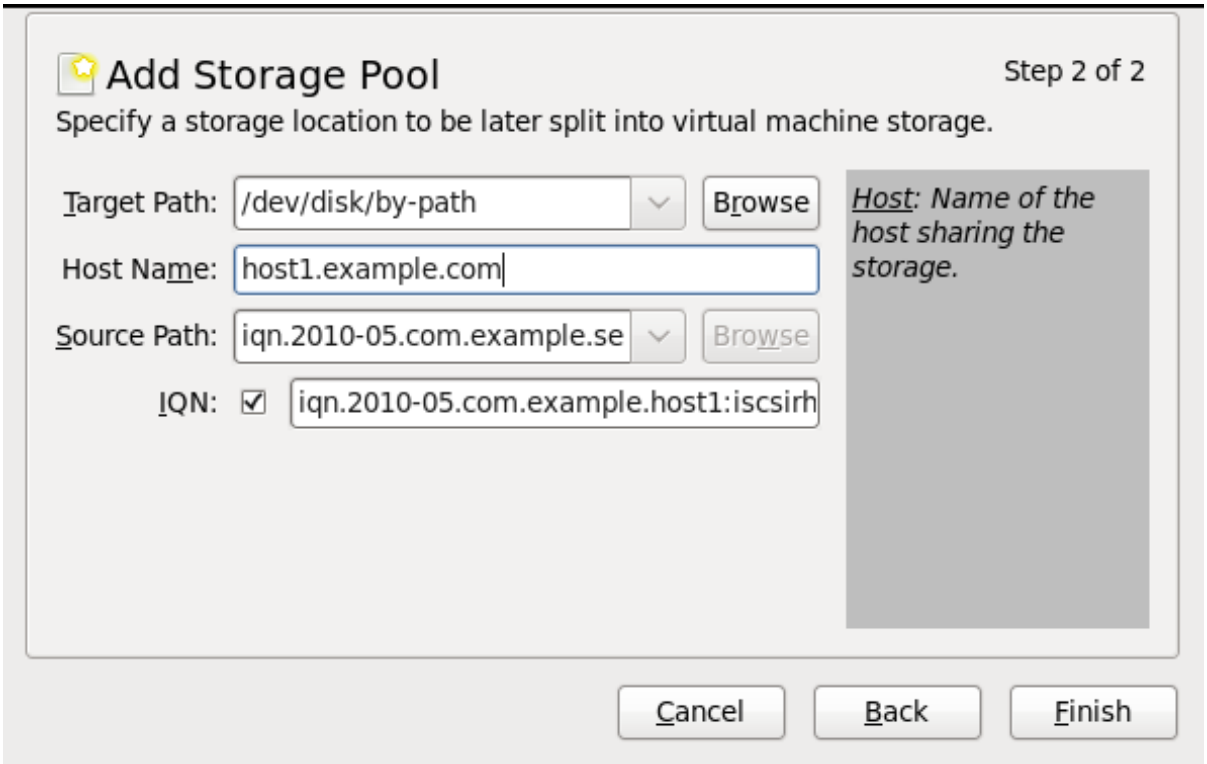
b. iSCSI ターゲットのホスト名または IP アドレスを入力します。この例では **host1.example.com** を使用しています。

c. **Source Path** フィールドに、iSCSI ターゲット IQN を入力します。「iSCSI ベースのストレージプール」 の 手順12.4「iSCSI ターゲットの作成」 を見ると、これは **/etc/tgt/targets.conf ファイル**で追加した情報であることがわかります。この例では **iqn.2010-05.com.example.server1:iscsirhel6guest** を使用しています。

d. **IQN** チェックボックスをオンにして、イニシエータの IQN を入力します。この例では **iqn.2010-05.com.example.host1:iscsirhel6** を使用しています。

e. **Finish** をクリックして、新しいストレージプールを作成します。

図12.22 iscsi ストレージプールを作成する

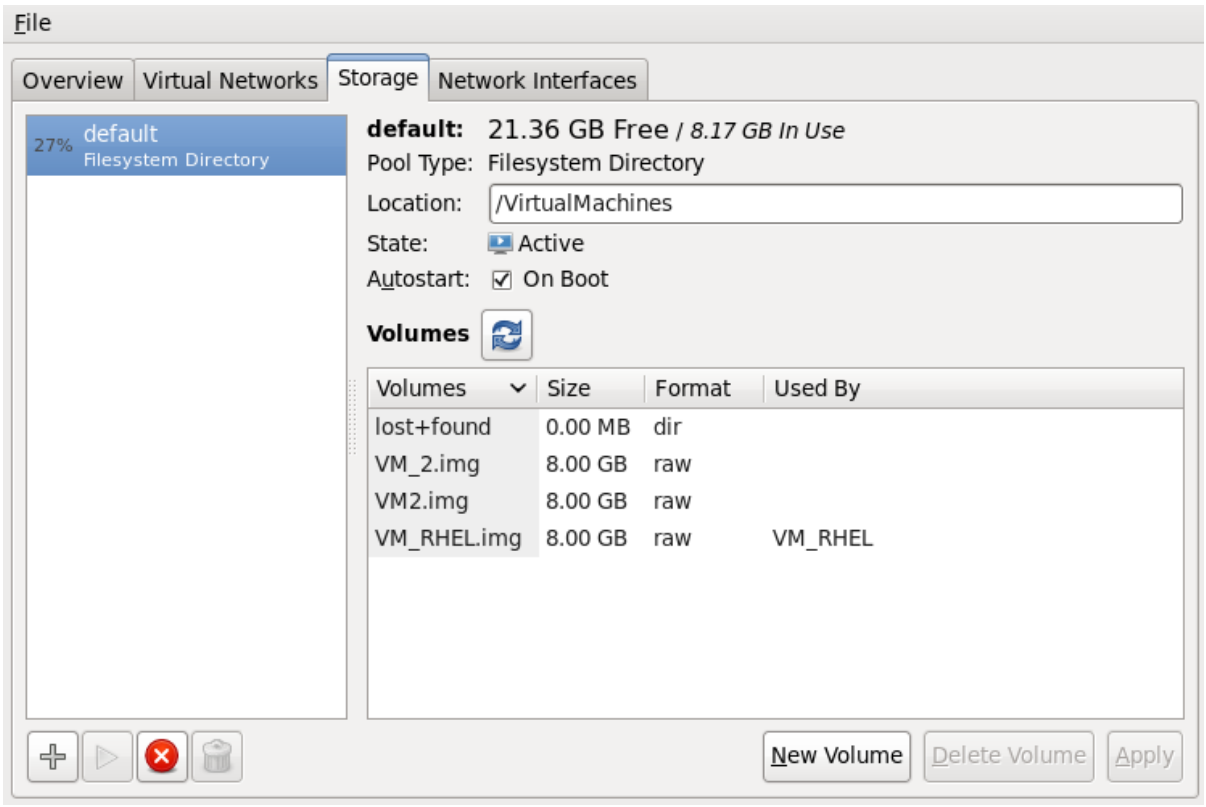


12.5.3. virt-manager を使用したストレージプールの削除

この手順は、ストレージプールを削除する方法を示しています。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。これを行うには、停止するストレージプールを選択し、ストレージウィンドウの下部にある赤い X アイコンをクリックします。

図12.23 停止アイコン



2. ごみ箱アイコンをクリックして、ストレージプールを削除します。このアイコンは、最初にストレージプールを停止した場合にのみ有効になります。

12.5.4. virsh を使用した iSCSI ベースのストレージプールの作成

1. **pool-define-as を使用して、コマンドラインからプールを定義します**
ストレージプールの定義は、**virsh** コマンドラインツールを使用して作成できます。**virsh** を使用してストレージプールを作成すると、システム管理者がスクリプトを使用して複数のストレージプールを作成する場合に役立ちます。

virsh pool-define-as コマンドには、次の形式で受け入れられるいくつかのパラメーターがあります。

```
virsh pool-define-as name type source-host source-path source-dev source-name target
```

パラメーターの説明は次のとおりです。

type
このプールを特定のタイプ、たとえば iscsi として定義します

name
一意である必要があり、ストレージプールの名前を設定します

source-host および source-path
それぞれホスト名と iSCSIIQN

source-dev および source-name

これらのパラメーターは iSCSI ベースのプールには必要ありません。フィールドを空白のままにするには、**-** 文字を使用します。

target

ホスト物理マシンに iSCSI デバイスをマウントする場所を定義します

以下の例では、前の手順と同じ iSCSI ベースのストレージプールを作成します。

```
#   virsh pool-define-as --name scsirhel6guest --type iscsi \
    --source-host server1.example.com \
    --source-dev iqn.2010-05.com.example.server1:iscsirhel6guest
    --target /dev/disk/by-path
Pool iscsirhel6guest defined
```

2. **ストレージプールがリストされていることを確認します**
ストレージプールオブジェクトが正しく作成され、状態が次のように報告されることを確認します**inactive**。

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
iscsirhel6guest      inactive   no
```

3. **ストレージプールを起動します。**
これには、virsh コマンド **pool-start** を使用します。**pool-start** は、ディレクトリーストレージプールを有効にし、ボリュームとゲスト仮想マシンに使用できるようにします。

```
# virsh pool-start guest_images_disk
Pool guest_images_disk started
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
iscsirhel6guest      active     no
```

4. **自動起動をオンにします**
オンにする***autostart***ストレージプール用。Autostart は、サービスの開始時にストレージプールを開始するように **libvirtd** サービスを設定します。

```
# virsh pool-autostart iscsirhel6guest
Pool iscsirhel6guest marked as autostarted
```

iscsirhel6guest プールに自動開始が設定されていることを確認します。

```
# virsh pool-list --all
Name                 State      Autostart
-----------------------------------------
default              active     yes
iscsirhel6guest      active     yes
```

5. **ストレージプールの設定を確認する**

ストレージプールが正しく作成され、サイズが正しく報告され、状態が次のように報告されることを確認します**running**。

```
# virsh pool-info iscsirhel6guest
Name:           iscsirhel6guest
UUID:           afcc5367-6770-e151-bcb3-847bc36c5e28
State:          running
Persistent:     unknown
Autostart:      yes
Capacity:       100.31 GB
Allocation:     0.00
Available:      100.31 GB
```

iSCSI ベースのストレージプールが利用可能になりました。

12.5.5. virsh を使用したストレージプールの削除

次に、virsh を使用してストレージプールを削除する方法を示します。

1. 同じプールを使用している他のゲスト仮想マシンでの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。

```
# virsh pool-destroy guest_images_disk
```

2. ストレージプールの定義を削除します。

```
# virsh pool-undefine guest_images_disk
```

12.6. NFS ベースのストレージプール

この手順では、**virt-manager** で NFS マウントポイントを使用してストレージプールを作成する方法について説明します。

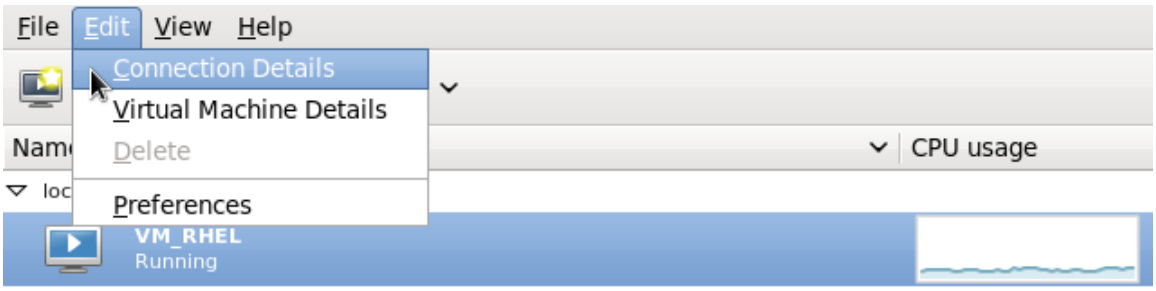
12.6.1. virt-manager を使用した NFS ベースのストレージプールの作成

1. **ホスト物理マシンのストレージタブを開きます**
   Host Details ウィンドウで **Storage** タブを開きます。

   a. **virt-manager** を開きます。

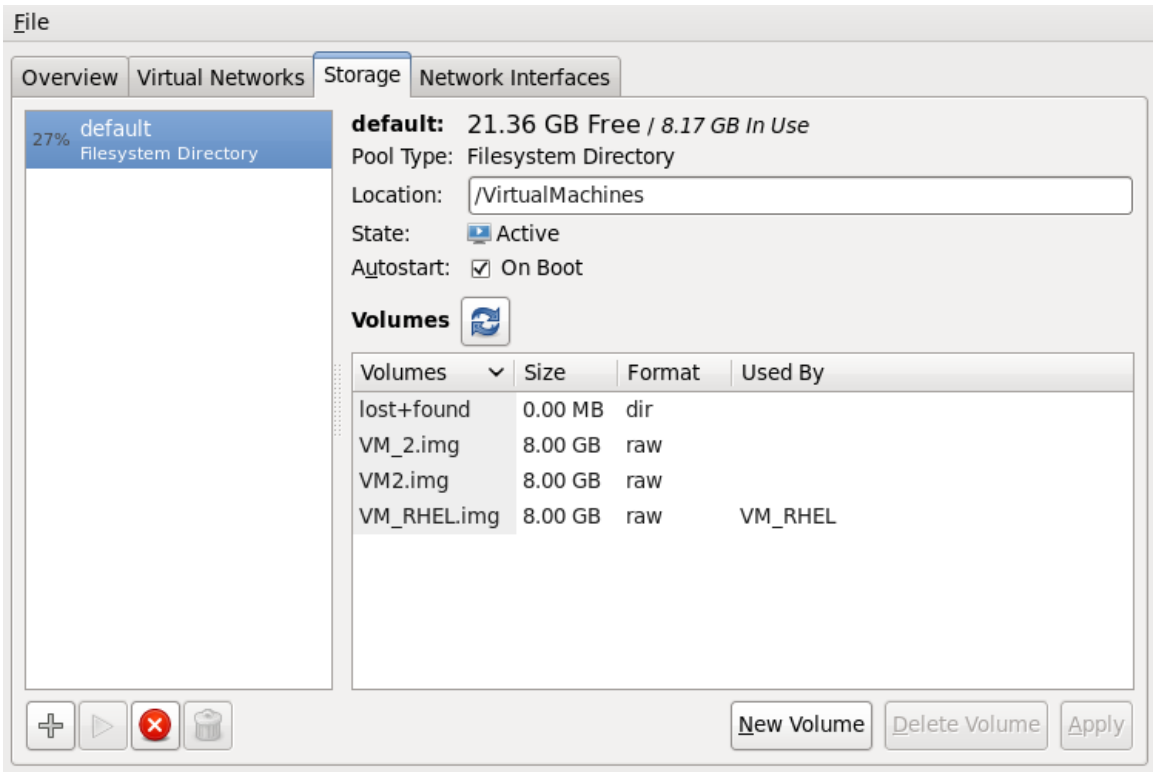
   b. メインの **virt-manager** ウィンドウからホスト物理マシンを選択します。**Edit menu** をクリックして、**Connection Details** を選択します。

図12.24 接続の詳細



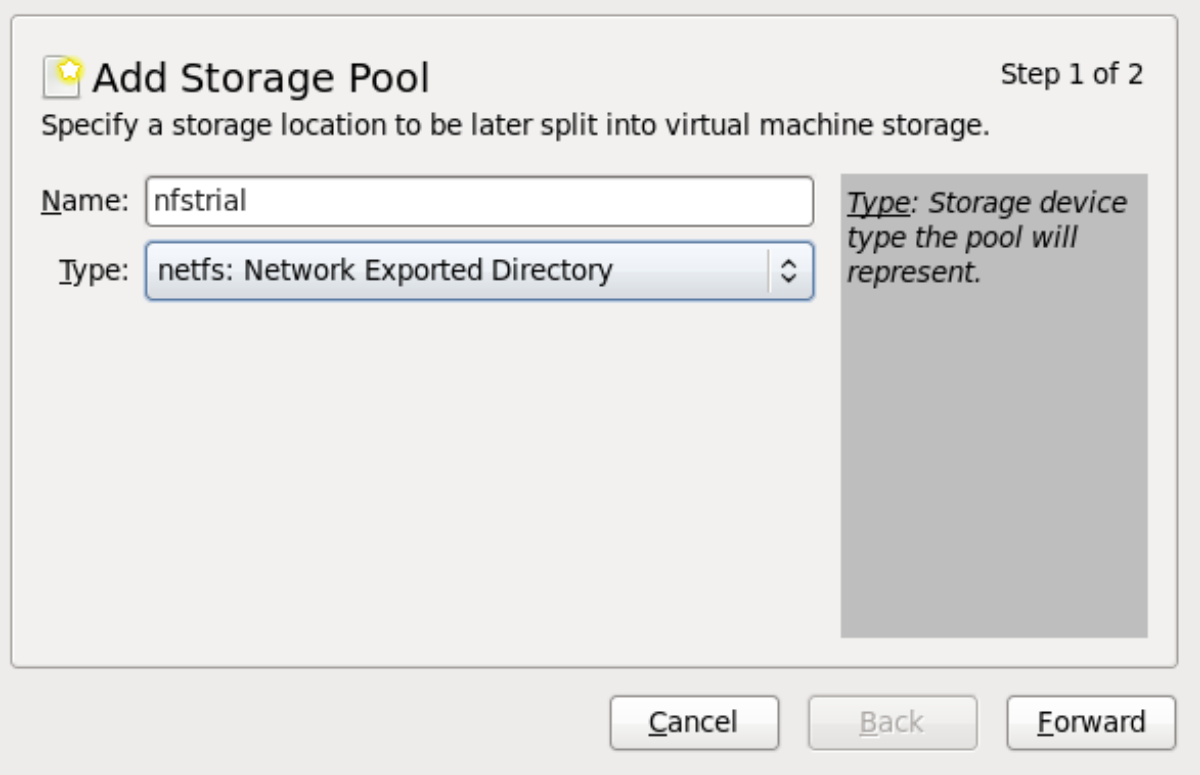
c. Storage タブをクリックします。

図12.25 ストレージタブ



2. **新しいプールを作成する (パート 1)**
+ ボタン (プールの追加ボタン) を押します。**Add a New Storage Pool** ウィザードが表示されます。

図12.26 NFS 名とタイプを追加します



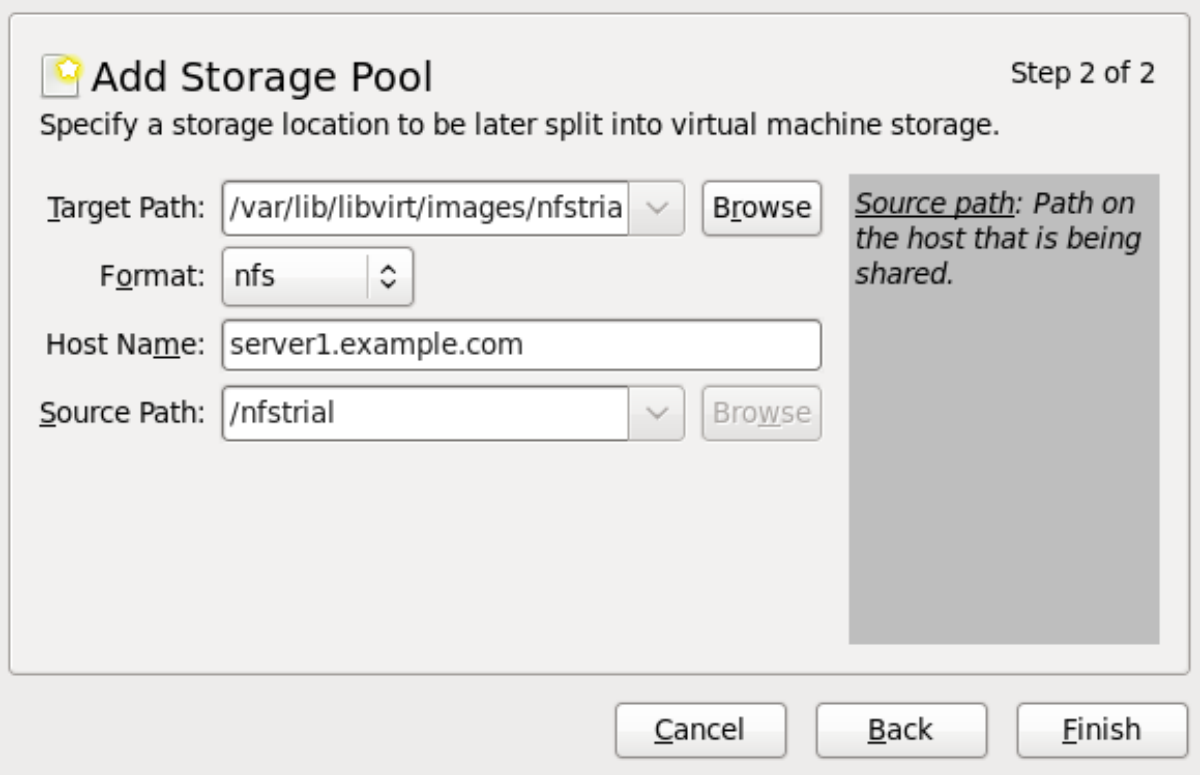
ストレージプールの名前を選択し、**Forward** を押して続行します。

3. **新しいプールを作成する (パート 2)**
デバイスのターゲットパス、ホスト名、および NFS 共有パスを入力します。**Format** オプションを **NFS** または **auto** (タイプを検出するため) に設定します。ターゲットパスは、移行するすべてのホスト物理マシンで同一である必要があります。

NFS サーバーのホスト名または IP アドレスを入力します。この例では **server1.example.com** を使用しています。

NFS パスを入力します。この例では **/nfstrial** を使用しています。

図12.27 NFS ストレージプールを作成する



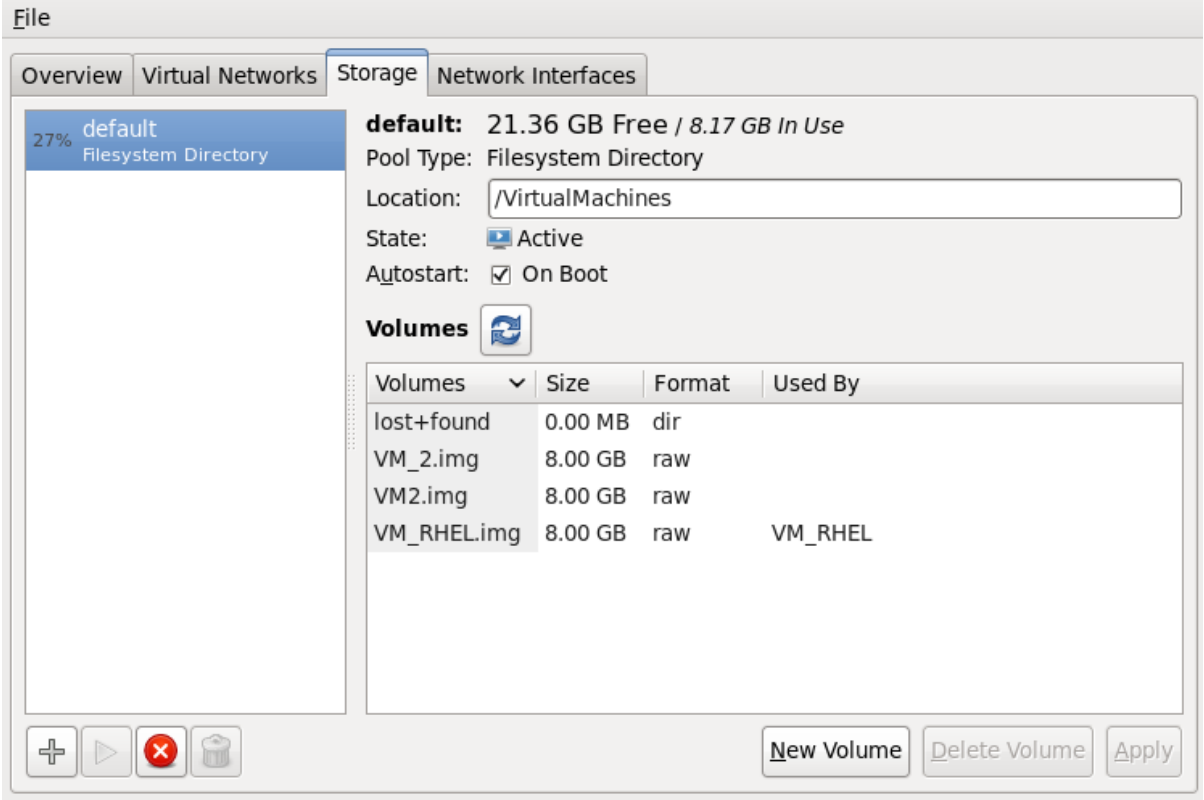
Finish を押して、新しいストレージプールを作成します。

12.6.2. virt-manager を使用したストレージプールの削除

この手順は、ストレージプールを削除する方法を示しています。

1. 同じプールを使用している他のゲストとの問題を回避するには、ストレージプールを停止し、使用中のリソースを解放することをお勧めします。これを行うには、停止するストレージプールを選択し、ストレージウィンドウの下部にある赤い X アイコンをクリックします。

図12.28 停止アイコン



2. ごみ箱アイコンをクリックして、ストレージプールを削除します。このアイコンは、最初にストレージプールを停止した場合にのみ有効になります。

12.7. GLUSTERFS ストレージプール

GlusterFS は、FUSE を使用するユーザースペースファイルシステムです。ゲスト仮想マシンで有効にすると、KVM ホスト物理マシンが 1 つ以上の GlusterFS ストレージボリュームからゲスト仮想マシンイメージを起動し、GlusterFS ストレージボリュームからのイメージをゲスト仮想マシンのデータディスクとして使用できるようになります。

重要

Red Hat Red Hat Enterprise Linux 6 は、ストレージプールでの GlusterFS の使用をサポートしていません。ただし、Red Hat Enterprise Linux 6.5 以降には、libgfapi ライブラリーを使用して GlusterFS で仮想マシンを作成するためのネイティブサポートが含まれています。

12.8. SCSI デバイスでの NPIV 仮想アダプター (VHBA) の使用

NPIV (N_Port ID Virtualization) は、1 つの物理ファイバーチャンネルのホストバスアダプター (HBA) の共有を可能にするソフトウェアテクノロジーです。

これにより、複数のゲストが複数の物理ホストから同じストレージを認識できるため、ストレージの移行パスが容易になります。そのため、正しいストレージパスが指定されていれば、移行を使用してストレージを作成またはコピーする必要はありません。

仮想化では、*仮想ホストバスアダプター* (*vHBA*) が仮想マシンの LUN を制御します。各 vHBA は、独自の WWNN(ワールドワイドノード名) と WWPN(ワールドワイドポート名) によって識別されます。ストレージのパスは、WWNN および WWPN の値で決定します。

このセクションでは、仮想マシン上で vHBA を設定するための手順を説明します。Red Hat Enterprise Linux 6 は、ホストの再起動後の永続的な vHBA 設定をサポートしていないことに注意してください。ホストの再起動後に vHBA 関連の設定を確認します。

12.8.1. vHBA の作成

手順12.6 vHBA の作成

1. **ホストシステムで HBA の場所を特定します。**
   ホストシステム上の HBA を見つけるには、ホストシステム上の SCSI デバイスを調べ、**vport** 機能を持つ **scsi_host** を見つけます。

   以下のコマンドを実行し、**scsi_host** リストを取得します。

   ```
   # virsh nodedev-list --cap scsi_host
   scsi_host0
   scsi_host1
   scsi_host2
   scsi_host3
   scsi_host4
   ```

   各**scsi_host**について、次のコマンドを実行して、デバイス XML の行 **<capability type='vport_ops'>** を調べます。これは、**vport** ケーパビリティを持つ **scsi_host** を示しています。

   ```
   # virsh nodedev-dumpxml scsi_hostN
   ```

2. **HBA の詳細を確認します。**
   virsh nodedev-dumpxml *HBA_device* コマンドを実行して、HBA の詳細を表示します。

   virsh nodedev-dumpxml コマンドからの XML 出力には、フィールドが一覧表示されます **<name>**、**<wwnn>**、と**<wwpn>**、vHBA の作成に使用されます。**<max_vports>** の値は、サポートされる vHBA の最大数を示しています。

   ```
    # virsh nodedev-dumpxml scsi_host3
   <device>
     <name>scsi_host3</name>
     <path>/sys/devices/pci0000:00/0000:00:04.0/0000:10:00.0/host3</path>
     <parent>pci_0000_10_00_0</parent>
     <capability type='scsi_host'>
       <host>3</host>
       <capability type='fc_host'>
         <wwnn>20000000c9848140</wwnn>
         <wwpn>10000000c9848140</wwpn>
         <fabric_wwn>2002000573de9a81</fabric_wwn>
       </capability>
       <capability type='vport_ops'>
         <max_vports>127</max_vports>
         <vports>0</vports>
       </capability>
     </capability>
   </device>
   ```

この例では、**<max_vports>** には、HBA 設定で使用できる仮想ポートが 127 個あることを示しています。**<vports>** 値は、現在使用されている仮想ポートの数を示します。この値は、vHBA の作成後に更新されます。

3. **vHBA ホストデバイスの作成**
vHBA ホスト用に次のような XML ファイル (この例では *vhba_host3.xml* という名前) を作成します。

```
# cat vhba_host3.xml
  <device>
    <parent>scsi_host3</parent>
    <capability type='scsi_host'>
      <capability type='fc_host'>
      </capability>
    </capability>
  </device>
```

<parent> フィールドは、この vHBA デバイスに関連付ける HBA デバイスを指定します。**<device>**タグの詳細は、ホスト用の新しい vHBA デバイスを作成するために、次の手順で使用されます。**nodedev**XML フォーマットの詳細は、http://libvirt.org/formatnode.htmlを参照してください。

4. **vHBA ホストデバイスに新しい vHBA を作成します。**
vhba_host3 に vHBA を作成するには、**virshnodedev-create** コマンドを使用します。

```
# virsh nodedev-create vhba_host3.xml
Node device scsi_host5 created from vhba_host3.xml
```

5. **vHBA の確認**
virsh nodedev-dumpxml コマンドを使用して、新しい vHBA の詳細 (**scsi_host5**) を確認します。

```
# virsh nodedev-dumpxml scsi_host5
<device>
  <name>scsi_host5</name>
  <path>/sys/devices/pci0000:00/0000:00:04.0/0000:10:00.0/host3/vport-3:0-0/host5</path>
  <parent>scsi_host3</parent>
  <capability type='scsi_host'>
    <host>5</host>
    <capability type='fc_host'>
      <wwnn>5001a4a93526d0a1</wwnn>
      <wwpn>5001a4ace3ee047d</wwpn>
      <fabric_wwn>2002000573de9a81</fabric_wwn>
    </capability>
  </capability>
</device>
```

12.8.2. vHBA を使用したストレージプールの作成

vHBA 設定を保持するために、vHBA に基づいて libvirt ストレージプールを定義することをお勧めします。

ストレージプールを使用することには、2 つの主な利点があります。

- libvirt コードは、**virsh** コマンドの出力を使用すると、LUN のパスを簡単に見つけることができます。また、

- 仮想マシンの移行には、ターゲットマシンで同じ vHBA 名を持つストレージプールの定義と起動のみが必要です。これを行うには、仮想マシンの XML 設定で、vHBA LUN、libvirt ストレージプール、およびボリューム名を指定する必要があります。例は、「vHBALUN を使用するように仮想マシンを設定する」 を参照してください。

1. **SCSI ストレージプールを作成する**
vHBA 設定を作成するには、まず vHBA に基づいて libvirt **'scsi'** ストレージプール XML ファイルを以下のフォーマットで作成します。

注記

手順12.6「vHBA の作成」 で作成した vHBA をホスト名として使用することを確認し、vHBA 名 *scsi_hostN* を *hostN* に修正してストレージプール設定に使用します。この例では、vHBA の名前は *scsi_host5* であり、Red Hat Enterprise Linux6 libvirt ストレージプールで **<adaptername='host5'/>** として指定されています。

<path>の値には、システム上の **/dev/disk/by-{path|id|uuid|label} の** いずれかの場所など、安定した場所を使用することをお勧めします。**<path>**および**<target>**内の要素の詳細は、http://libvirt.org/formatstorage.htmlを参照してください。

この例では、**'scsi'**ストレージプールの名前は *vhbapool_host3.xml* です。

```
<pool type='scsi'>
  <name>vhbapool_host3</name>
  <uuid>e9392370-2917-565e-692b-d057f46512d6</uuid>
  <capacity unit='bytes'>0</capacity>
  <allocation unit='bytes'>0</allocation>
  <available unit='bytes'>0</available>
  <source>
    <adapter name='host5'/>
  </source>
   <target>
     <path>/dev/disk/by-path</path>
     <permissions>
       <mode>0700</mode>
       <owner>0</owner>
       <group>0</group>
     </permissions>
   </target>
 </pool>
```

2. **プールを定義する**
ストレージプール (この例では *vhbapool_host3* という名前) を定義するには、**virshpool-define** コマンドを使用します。

```
# virsh pool-define vhbapool_host3.xml
Pool vhbapool_host3 defined from vhbapool_host3.xml
```

3. **プールを開始します**
次のコマンドでストレージプールを開始します。

```
# virsh pool-start vhbapool_host3
Pool vhbapool_host3 started
```

4. **自動起動を有効にする**
   最後に、後続のホストの再起動で仮想マシンで使用する vHBA が自動的に定義されるようにするには、ストレージプールの自動開始機能を設定します (この例では、*vhbapool_host3* という名前のプールに対して)。

```
# virsh pool-autostart vhbapool_host3
```

12.8.3. vHBALUN を使用するように仮想マシンを設定する

vHBA のストレージプールが作成されたら、vHBALUN を仮想マシン設定に追加します。

1. **利用可能な LUN を見つける**
   まず、**virsh vol-list** コマンドを使用して、vHBA で使用可能な LUN のリストを生成します。以下に例を示します。

```
# virsh vol-list vhbapool_host3
 Name                 Path
-----------------------------------------------------------------------------
 unit:0:4:0           /dev/disk/by-path/pci-0000:10:00.0-fc-0x5006016844602198-lun-0
 unit:0:5:0           /dev/disk/by-path/pci-0000:10:00.0-fc-0x5006016044602198-lun-0
```

   表示される LUN 名のリストは、仮想マシン設定でディスクボリュームとして使用できます。

2. **vHBALUN を仮想マシンに追加します**
   仮想マシンの XML で指定して、仮想マシン に vHBA LUN を追加します。

   - **<disk>**パラメーターに、**lun** または **disk** としてデバイスタイプを指定し
   - は、**<source>** パラメーターでソースデバイスを指定します。これは、**/dev/sdaN** として入力することも、udev によって生成されたシンボリックリンクとして **/dev/disk/by-path|by-id|by-uuid|by-label** として入力することもできます。これは、**virsh vol-list *pool*** コマンドを実行することで見つけることができます。

   以下に例を示します。

```
<disk type='block' device='lun'>
  <driver name='qemu' type='raw'/>
  <source dev='/dev/disk/by-path/pci-0000\:04\:00.1-fc-0x203400a0b85ad1d7-lun-0'/>
  <target dev='sda' bus='scsi'/>
</disk>
```

12.8.4. vHBA ストレージプールを破棄する

vHBA ストレージプールは、**virshpool-destroy** コマンドで破棄できます。

```
# virsh pool-destroy vhbapool_host3
```

次のコマンドで vHBA を削除します

```
# virsh nodedev-destroy scsi_host5
```

プールと vHBA が破棄されたことを確認するには、次のコマンドを実行します。

```
# virsh nodedev-list --cap scsi_host
```

scsi_host5 結果のリストに表示されなくなります。

第13章 ボリューム

13.1. ボリュームの作成

このセクションでは、ブロックベースのストレージプール内にディスクボリュームを作成する方法を示します。以下の例では、**virsh vol-create-as** コマンドは、*guest_images_disk* ストレージプール内に GB 単位の特定のサイズのストレージボリュームを作成します。このコマンドは必要なボリュームごとに繰り返されるため、例に示すように 3 つのボリュームが作成されます。

```
# virsh vol-create-as guest_images_disk volume1 8G
Vol volume1 created

# virsh vol-create-as guest_images_disk volume2 8G
Vol volume2 created

# virsh vol-create-as guest_images_disk volume3 8G
Vol volume3 created

# virsh vol-list guest_images_disk
Name                 Path
-----------------------------------------
volume1              /dev/sdb1
volume2              /dev/sdb2
volume3              /dev/sdb3

# parted -s /dev/sdb print
Model: ATA ST3500418AS (scsi)
Disk /dev/sdb: 500GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End     Size    File system  Name     Flags
2      17.4kB  8590MB  8590MB               primary
3      8590MB  17.2GB  8590MB               primary
1      21.5GB  30.1GB  8590MB               primary
```

13.2. クローンボリューム

新しいボリュームは、複製されるボリュームと同じストレージプール内のストレージから割り当てられます。**virsh vol-clone** には、クローンを作成するボリュームを含むストレージプールの名前を指定する **--pool** 引数が必要です。コマンドの残りの部分では、クローンを作成するボリューム (volume3) と、クローンを作成した新しいボリュームの名前 (clone1) を指定します。**virsh vol-list** コマンドは、ストレージプール (guest_images_disk) に存在するボリュームを一覧表示します。

```
# virsh vol-clone --pool guest_images_disk volume3 clone1
Vol clone1 cloned from volume3

# virsh vol-list guest_images_disk
Name                 Path
-----------------------------------------
volume1              /dev/sdb1
volume2              /dev/sdb2
```

```
volume3             /dev/sdb3
clone1              /dev/sdb4


# parted -s /dev/sdb print
Model: ATA ST3500418AS (scsi)
Disk /dev/sdb: 500GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    File system  Name     Flags
1      4211MB  12.8GB  8595MB  primary
2      12.8GB  21.4GB  8595MB  primary
3      21.4GB  30.0GB  8595MB  primary
4      30.0GB  38.6GB  8595MB  primary
```

13.3. ゲストへのストレージデバイスの追加

このセクションでは、ゲストへのストレージデバイスの追加について説明します。追加のストレージは、必要な場合にのみ追加できます。

13.3.1. ゲストへのファイルベースストレージの追加

ファイルベースのストレージは、ゲストの仮想化ハードドライブとして機能するホスト物理マシンのファイルシステムに保存されるファイルのコレクションです。ファイルベースのストレージを追加するには、次の手順を実行します。

手順13.1 ファイルベースのストレージの追加

1. ストレージファイルを作成するか、既存のファイル (IMG ファイルなど) を使用します。次のコマンドは両方とも、ゲストの追加ストレージとして使用できる 4GB のファイルを作成することに注意してください。

   - ファイルベースのストレージイメージには、事前に割り当てられたファイルをお勧めします。次に示すように、次の **dd** コマンドを使用して、事前に割り当てられたファイルを作成します。

     ```
     # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M count=4096
     ```

   - または、事前に割り当てられたファイルの代わりにスパースファイルを作成します。スパースファイルははるかに高速に作成され、テストに使用できますが、データの整合性とパフォーマンスの問題があるため、実稼働環境にはお勧めしません。

     ```
     # dd if=/dev/zero of=/var/lib/libvirt/images/FileName.img bs=1M seek=4096 count=0
     ```

2. 新しいファイルに *<disk>* 要素を書き込んで、追加のストレージを作成します。この例では、このファイルは **NewStorage.xml** と呼ばれます。

   <disk> 要素は、ディスクのソースと仮想ブロックデバイスのデバイス名を記述します。デバイス名は、ゲスト内のすべてのデバイスで一意である必要があり、ゲストが仮想ブロックデバイスを見つけるバスを識別します。次の例では、ソースが **FileName.img** という名前のファイルベースのストレージコンテナーである virtio ブロックデバイスを定義しています。

   ■

```
<disk type='file' device='disk'>
  <driver name='qemu' type='raw' cache='none'/>
  <source file='/var/lib/libvirt/images/FileName.img'/>
  <target dev='vdb'/>
</disk>
```

デバイス名は hd または sd で始めることもでき、それぞれ IDE と SCSI ディスクを識別します。設定ファイルには、新しいデバイスのバス上の位置を指定する **<address>** サブ要素を含めることもできます。virtio ブロックデバイスの場合、これは PCI アドレスである必要があります。**<address>** サブ要素を省略すると、libvirt は次に使用可能な PCI スロットを見つけて割り当てることができます。

3. 次のように CD-ROM を接続します。

```
<disk type='file' device='cdrom'>
  <driver name='qemu' type='raw' cache='none'/>
  <source file='/var/lib/libvirt/images/FileName.img'/>
  <readonly/>
  <target dev='hdc'/>
</disk >
```

4. **NewStorage.xml** で定義されたデバイスをゲスト (**Guest1**) とともに追加します。

```
# virsh attach-device --config Guest1 ~/NewStorage.xml
```

注記

この変更は、ゲストが破棄されて再起動された後にのみ適用されます。さらに、永続デバイスは、永続ドメイン、つまり **virsh define** コマンドで設定が保存されたドメインにのみ追加できます。

ゲストが実行中で、ゲストが破棄されるまで新しいデバイスを一時的に追加する場合は、**--config** オプションを省略します。

```
# virsh attach-device Guest1 ~/NewStorage.xml
```

注記

virsh コマンドを使用すると、XML ファイルを作成しなくても、より単純な構文で限られた数のパラメーターを設定できる **attach-disk** コマンドを使用できます。**attach-disk** コマンドは、次に示すように、前述の **attach-device** コマンドと同様の方法で使用されます。

```
# virsh attach-disk Guest1 /var/lib/libvirt/images/FileName.img vdb --cache
none --driver qemu --subdriver raw
```

virsh attach-disk コマンドは **--config** オプションも受け入れることに注意してください。

5. ゲストマシンを起動します (現在実行されていない場合)。

```
# virsh start Guest1
```

注記

次の手順は Linux ゲスト固有です。他のオペレーティングシステムは、さまざまな方法で新しいストレージデバイスを処理します。その他のシステムについては、そのオペレーティングシステムのドキュメントを参照してください。

6. **ディスクドライブのパーティション分割**
これで、ゲストには **/dev/vdb** というハードディスクデバイスがあります。。必要に応じて、このディスクドライブをパーティションに分割し、パーティションをフォーマットします。追加したデバイスが表示されない場合は、ゲストのオペレーティングシステムのディスクホットプラグに問題があることを示しています。

   a. 新しいデバイスの **fdisk** を起動します。

   ```
   # fdisk /dev/vdb
   Command (m for help):
   ```

   b. 新しいパーティションのために、***n*** と入力します。

   c. 次のように表示されます。

   ```
   Command action
   e   extended
   p   primary partition (1-4)
   ```

   プライマリーパーティションには、***p*** と入力します。

   d. 使用可能なパーティション番号を選択します。この例では、***1***を入力して、最初のパーティションを選択します。

   ```
   Partition number (1-4): 1
   ```

   e. ***Enter***を押して、デフォルトの最初のシリンダーを入力します。

   ```
   First cylinder (1-400, default 1):
   ```

   f. パーティションのサイズを選択します。この例では、**Enter** を押してディスク全体を割り当てます。

   ```
   Last cylinder or +size or +sizeM or +sizeK (2-400, default 400):
   ```

   g. **t** を入力して、パーティションタイプを設定します。

   ```
   Command (m for help): t
   ```

   h. 前の手順で作成したパーティションを選択します。この例では、パーティションが 1 つしか作成されておらず、fdisk がパーティション 1 を自動的に選択したため、パーティション番号は **1** です。

   ```
   Partition number (1-4): 1
   ```

   i. Linux パーティションの場合は **83** と入力します。

```
Hex code (type L to list codes): 83
```

j. **w** と入力して変更を書き込み、終了します。

```
Command (m for help): w
```

k. **ext3** ファイルシステムで新しいパーティションをフォーマットします。

```
# mke2fs -j /dev/vdb1
```

7. マウントディレクトリーを作成し、ゲストにディスクをマウントします。この例では、ディレクトリーは *myfiles* にあります。

```
# mkdir /myfiles
# mount /dev/vdb1 /myfiles
```

これで、ゲストは追加の仮想化されたファイルベースのストレージデバイスを使用できます。ただし、ゲストの **/etc/fstab** ファイルで定義されていない限り、このストレージは再起動後に永続的にマウントされないことに注意してください。

```
/dev/vdb1    /myfiles    ext3    defaults    0 0
```

13.3.2. ゲストへのハードドライブおよびその他のブロックデバイスの追加

システム管理者は、追加のハードドライブを使用して、ゲストのストレージスペースを増やすか、システムデータをユーザーデータから分離するかを選択できます。

手順13.2 ゲストへの物理ブロックデバイスの追加

1. この手順では、ホスト物理マシン上のハードドライブをゲストに追加する方法について説明します。これは、CD-ROM、DVD、フロッピーデバイスを含むすべての物理ブロックデバイスに適用されます。

   ハードディスクデバイスをホストの物理マシンに物理的に接続します。ドライブにデフォルトでアクセスできない場合は、ホスト物理マシンを設定します。

2. 次のいずれかを行います。

   a. 新しいファイルに **disk** 要素を書き込んで、追加のストレージを作成します。この例では、このファイルは **NewStorage.xml** と呼ばれます。次の例は、ホスト物理マシンパーティション **/dev/sr0** 用の追加のデバイスベースのストレージコンテナーを含む設定ファイルセクションです。

   ```
   <disk type='block' device='disk'>
       <driver name='qemu' type='raw' cache='none'/>
       <source dev='/dev/sr0'/>
       <target dev='vdc' bus='virtio'/>
   </disk>
   ```

   b. 前のセクションの手順に従って、デバイスをゲスト仮想マシンに接続します。または、***virsh attach-disk***示されているように、コマンド:

```
# virsh attach-disk Guest1 /dev/sr0 vdc
```

次のオプションが使用可能であることに注意してください。

- **virsh attach-disk** コマンドは、図のように ***--config***、***--type***、および ***--mode*** オプションも受け付けます。

```
# virsh attach-disk Guest1 /dev/sr0 vdc --config --type cdrom --mode readonly
```

- さらに、 ***--type***、デバイスがハードディスクの場合は、***--type disk*** も受け付けます。

3. ゲスト仮想マシンには、Linux では **/dev/vdc** (またはゲスト仮想マシン OS が選択するものに応じて同様のもの) または Windows では **D: drive** (たとえば) と呼ばれる新しいハードディスクデバイスがあります。これで、ゲスト仮想マシンのオペレーティングシステムの標準手順に従って、ゲスト仮想マシンからディスクを初期化できます。例は、手順13.1「ファイルベースのストレージの追加」 を参照してください。

警告

ゲストにブロックデバイスを追加するときは、セキュリティー上の考慮事項に必ず従ってください。この情報については、『Red Hat Enterprise Linux 仮想化セキュリティーガイド』 を参照してください。https://access.redhat.com/site/documentation/。

重要

ゲスト仮想マシンには、ディスク全域、またはブロックデバイス全域 (例: **/dev/sdb**) への書き込みアクセス権を付与しないでください。ブロックデバイス全域にアクセスを持つゲスト仮想マシンは、ボリュームラベルを修正できる場合があり、これがホスト物理マシンシステムの攻撃に使用される可能性があります。パーティション (例: **/dev/sdb1**) または LVM ボリュームを使用して、この問題を回避してください。

13.4. ボリュームの削除と削除

このセクションでは、**virsh vol-delete** コマンドを使用してブロックベースのストレージプールからディスクボリュームを削除する方法を示します。この例では、ボリュームは *volume 1* であり、ストレージプールは *guest_images* です。

```
# virsh vol-delete --pool guest_images volume1
Vol volume1 deleted
```

第14章 VIRSH を使用したゲスト仮想マシンの管理

virsh は、ゲスト仮想マシンとハイパーバイザーを管理するためのコマンドラインインターフェイスツールです。**virsh** コマンドラインツールは **libvirt** 管理 API に基づいて構築されており、**qemu-kvm** コマンドおよびグラフィカルな **virt-manager** アプリケーションの代替として機能します。**virsh** コマンドは、特権のないユーザーが読み取り専用モードで使用することも、root アクセスで完全な管理機能を使用することもできます。**virsh** コマンドは、仮想化管理のスクリプトを作成するのに理想的です。

14.1. 一般的なコマンド

このセクションのコマンドは、どのドメインにも固有ではないため、一般的なものです。

14.1.1. help

$ virsh help command | group help コマンドは、オプションの有無にかかわらず使用できます。オプションなしで使用すると、すべてのコマンドが 1 行に 1 つずつ一覧表示されます。オプションとともに使用すると、カテゴリーにグループ化され、各グループのキーワードが表示されます。

特定のオプション専用のコマンドを表示するには、そのグループのキーワードをオプションとして指定する必要があります。以下に例を示します。

```
$ virsh help pool
 Storage Pool (help keyword 'pool'):
   find-storage-pool-sources-as   find potential storage pool sources
   find-storage-pool-sources      discover potential storage pool sources
   pool-autostart               autostart a pool
   pool-build                   build a pool
   pool-create-as               create a pool from a set of args
   pool-create                  create a pool from an XML file
   pool-define-as               define a pool from a set of args
   pool-define                  define (but don't start) a pool from an XML file
   pool-delete                  delete a pool
   pool-destroy                 destroy (stop) a pool
   pool-dumpxml                 pool information in XML
   pool-edit                    edit XML configuration for a storage pool
   pool-info                    storage pool information
   pool-list                    list pools
   pool-name                    convert a pool UUID to pool name
   pool-refresh                 refresh a pool
   pool-start                   start a (previously defined) inactive pool
   pool-undefine                undefine an inactive pool
   pool-uuid                    convert a pool name to pool UUID
```

同じコマンドをコマンドオプションとともに使用すると、その 1 つの特定のコマンドに関するヘルプ情報が得られます。以下に例を示します。

```
$ virsh help vol-path
  NAME
    vol-path - returns the volume path for a given volume name or key

  SYNOPSIS
    vol-path <vol> [--pool <string>]
```

```
OPTIONS
  [--vol] <string>  volume name or key
  --pool <string>  pool name or uuid
```

14.1.2. 終了して終了します

quit コマンドと exit コマンドは、ターミナルを閉じます。以下に例を示します。

```
$ virsh exit
```

```
$ virsh quit
```

14.1.3. version

version コマンドは、現在の libvirt バージョンを表示し、ビルドの作成元に関する情報を表示します。以下に例を示します。

```
$ virsh version
Compiled against library: libvirt 1.1.1
Using library: libvirt 1.1.1
Using API: QEMU 1.1.1
Running hypervisor: QEMU 1.5.3
```

14.1.4. 引数の表示

virsh echo [--shell][--xml][arg] コマンドは、指定された引数をエコーまたは表示します。エコーされた各引数はスペースで区切られます。**--shell** オプションを使用すると、出力は必要に応じて一重引用符で囲まれるため、シェルコマンドでの再利用に適しています。**--xml** オプションを使用すると、出力は XML ファイルでの使用に適したものになります。たとえば、**virsh echo --shell "hello world"** というコマンドは、出力 **'hello world'** を送信します。

14.1.5. connect

ハイパーバイザーセッションに接続します。シェルが最初に起動されたとき、このコマンドは、URI パラメーターが **-c** コマンドによって要求されたときに自動的に実行されます。URI は、ハイパーバイザーへの接続方法を指定します。最も一般的に使用される URI は以下のとおりです。

- **xen:**/// - ローカルの Xen ハイパーバイザーに接続します。
- **qemu:///system-** ルートとしてローカルで QEMU および KVM ドメインを監視するデーモンに接続します。
- **xen:///session-** ユーザーとしてローカルでユーザーの QEMU および KVM ドメインのセットに接続します。
- **lxc:**/// - ローカルの Linux コンテナーに接続します。

追加の値は、libvirt の Web サイト http://libvirt.org/uri.html で入手できます。

コマンドは次のように実行できます。

```
$ virsh connect {name|URI}
```

ここで、***{name}*** は、ハイパーバイザーのマシン名 (ホスト名) または URL (**virsh uri** コマンドの出力) です。読み取り専用接続を開始するには、上記のコマンドに **--readonly** を追加します。URI の詳細については、リモート URI を参照してください。URI がわからない場合は、**virsh uri** コマンドを実行すると以下のようなメッセージが表示されます。

```
$ virsh uri
qemu:///session
```

14.1.6. 基本情報の表示

次のコマンドを使用して、基本情報を表示できます。

- $ **hostname-** ハイパーバイザーのホスト名を表示します
- $ **sysinfo-** 利用可能な場合、ハイパーバイザーのシステム情報の XML 表現を表示します

14.1.7. NMI の挿入

$ **virsh inject-nmi [domain]** は、NMI (マスク不可割り込み) メッセージをゲスト仮想マシンに挿入します。これは、回復不能なハードウェアエラーなど、応答時間が重要な場合に使用されます。このコマンドを実行するには:

```
$ virsh inject-nmi guest-1
```

14.2. VIRSH を使用したデバイスの接続および更新

ストレージデバイスの取り付けについては、以下を参照してください。「ゲストへのファイルベースストレージの追加」

手順14.1 ゲスト仮想マシンが使用するホットプラグ USB デバイス

次の手順は、USB デバイスをゲスト仮想マシンに接続する方法を示しています。これは、ゲスト仮想マシンがホットプラグ手順として実行されているときに実行することも、ゲストがシャットオフされているときに実行することもできます。エミュレートするデバイスは、ホストの物理マシンに接続する必要があります。

1. 次のコマンドを使用して、接続する USB デバイスを見つけます。

```
# lsusb -v

idVendor           0x17ef Lenovo
idProduct          0x480f Integrated Webcam [R5U877]
```

2. XML ファイルを作成し、論理名 (**usb_device.xml** など) を付けます。検索で表示されたとおりにベンダー ID と製品 ID をコピーしてください。

図14.1 USB デバイスの XML スニペット

```
<hostdev mode='subsystem' type='usb' managed='yes'>
  <source>
    <vendor id='0x17ef'/>
    <product id='0x480f'/>
  </source>
 </hostdev>
...
```

3. 次のコマンドでデバイスを接続します。

```
# virsh attach-device rhel6 --file usb_device.xml --config
```

この例では、rhel6 はゲスト仮想マシンの名前であり、usb_device.xml は前の手順で作成したファイルです。次回の再起動時に変更を有効にする場合は、**--config** オプションを使用します。この変更を永続的にする場合は、**--persistent** オプションを使用します。変更を現在のドメインで有効にする場合は、**--current** オプションを使用します。詳細については、Virsh の man ページを参照してください。

4. デバイスを切り離す (ホットアンプラグ) 場合は、次のコマンドを実行します。

```
# virsh detach-device rhel6 --file usb_device.xml
```

この例では、rhel6 はゲスト仮想マシンの名前であり、usb_device.xml は前の手順で添付したファイルです。

14.3. インターフェイスデバイスの接続

virsh attach-interface*domain type source* コマンドは、次のオプションを使用できます。

- **--live-** - 実行中のドメインから値を取得します
- **--config** - システムの次回起動時に使用する値を取得します。
- **--current-** - 現在のドメインの状態に応じて値を取得します
- **--persistent-** - オフラインドメインの場合は **--config** のように動作し、実行中のドメインの場合は **--live** のように動作します。
- **--target** - ゲスト仮想マシンのターゲットデバイスを示します。
- **--mac-** - これを使用して、ネットワークインターフェイスの MAC アドレスを指定します
- **--script** - これは、デフォルトのパスの代わりに、ブリッジを処理するスクリプトファイルへのパスを指定します。
- **--model-** - これを使用してモデルタイプを指定します。
- **--inbound** - インターフェイスの着信帯域幅を制御します。指定できる値は、**average**、**peak**、および **burst** です。

- **--outbound** - インターフェイスのアウトバウンド帯域幅を制御します。指定できる値は、**average**、**peak**、および **burst** です。

type は、物理ネットワークデバイスを示す **network**、またはデバイスへの **bridge** を示すブリッジのいずれかです。*source* はデバイスのソースです。接続したデバイスを削除する場合は、**virsh detach-device** を実行します。

14.4. CDROM のメディアの変更

CDROM のメディアを別のソースまたはフォーマットに変更する

```
# change-media domain path source --eject --insert --update --current --live --config --force
```

- **--path** - ディスクデバイスの完全修飾パスまたはターゲットが含まれる文字列
- **--source** - メディアのソースが含まれる文字列
- **--eject-** - メディアを排出します
- **--insert** - メディアを挿入します
- **--update-** - メディアを更新します
- **--current** - **--live** および **--config** のいずれかまたは両方を指定できます。これは、ハイパーバイザードライバーの実装に依存します。
- **--live-** - 実行中のドメインのライブ設定を変更します
- **--config-** - 永続的な設定を変更し、次回の起動時に影響が観察されます
- **--force-** - メディアの変更を強制します

14.5. ドメインコマンド

これらのコマンドのほとんどは、指定されたドメインを直接操作するため、ドメイン名が必要です。ドメインは、短整数 (0,1,2 ...)、名前、または完全な UUID として指定できます。

14.5.1. 起動時に自動的に開始されるドメインの設定

$ virsh autostart [--disable] domain は、起動時に指定されたドメインを自動的に開始します。**--disable** オプションを使用すると、自動起動が無効になります。

```
# virsh autostart rhel6
```

上記の例では、ホストの物理マシンが起動すると、rhel6 ゲスト仮想マシンが自動的に起動します。

```
# virsh autostart rhel6 --disable
```

上記の例では、自動起動機能が無効になっており、ホスト物理マシンの起動時にゲスト仮想マシンが自動的に起動しなくなりました。

14.5.2. ゲスト仮想マシンのシリアルコンソールの接続

$ virsh console <domain> [--devname <string>] [--force] [--safe] コマンドは、ゲスト仮想マシンの

仮想シリアルコンソールを接続します。オプションの --devname <string> パラメーターは、ゲスト仮想マシンに設定された代替コンソール、シリアル、またはパラレルデバイスのデバイスエイリアスを参照します。このパラメーターを省略すると、プライマリーコンソールが開きます。**--force** オプションは、コンソール接続を強制するか、disconnect と一緒に使用すると、接続を切断します。**--safe** オプションを使用すると、安全なコンソール処理がサポートされている場合にのみゲストが接続できます。

```
$ virsh console virtual_machine --safe
```

14.5.3. XML ファイルを使用したドメインの定義

define <FILE> コマンドは、XML ファイルからドメインを定義します。この場合のドメイン定義は登録されていますが、開始されていません。ドメインがすでに実行されている場合、変更は次回の起動時に有効になります。

14.5.4. ドメインの説明とタイトルの編集と表示

次のコマンドは、ドメインの説明とタイトルを表示または変更するために使用されますが、設定はしません。

```
# virsh desc [domain-name] [[--live] [--config] | [--current]] [--title] [--edit] [--new-desc New description or title message]
```

これらの値は、ドメインを簡単に識別できるように任意のテキストデータを保存できるユーザーフィールドです。理想的には、タイトルは短くする必要がありますが、これは libvirt によって強制されません。

オプション **--live** または **--config** は、このコマンドがドメインのライブ定義または永続定義のどちらで機能するかを選択します。**--live** と **--config** の両方が指定されている場合、最初に **--config** オプションが実装され、コマンドに入力された説明が、ライブ設定と永続設定設定の両方に適用される新しい設定設定になります。**--current** オプションは、現在の状態の設定を変更または取得し、永続的ではありません。**--live**、**--config**、**--current** のいずれも指定されていない場合、**--current** オプションが使用されます。**--edit** オプションは、現在の説明またはタイトルの内容を含むエディターを開き、内容を後で保存することを指定します。**--title** オプションを使用すると、ドメインのタイトルフィールドのみが表示または変更され、説明は含まれません。また、コマンドで **--edit** も **--new-desc** も使用されていない場合は、説明のみが表示され、変更できません。

たとえば、次のコマンドは、ゲスト仮想マシンのタイトルを *testvm* から *TestVM-4F* に変更し、説明を *4 階の GuestVM* に変更します。

```
$ virsh desc testvm --current --title TestVM-4F --new-desc Guest VM on fourth floor
```

14.5.5. デバイスブロック統計の表示

このコマンドは、実行中のドメインのブロック統計を表示します。ドメイン名とデバイス名の両方が必要です (デバイスを一覧表示するには、**virsh domblklist** を使用します)。この場合、ブロックデバイスは一意のターゲット名 (<target dev='name'/>) またはソースファイル (< source file ='name'/>) です。すべてのハイパーバイザーがすべてのフィールドを表示できるわけではないことに注意してください。出力が最も読みやすい形式で表示されるようにするには、次のように **--human** オプションを使用します。

```
# virsh domblklist rhel6
Target     Source
------------------------------------------------
```

```
vda        /VirtualMachines/rhel6.img
hdc        -

# virsh domblkstat --human rhel6 vda
Device: vda
 number of read operations:      174670
 number of bytes read:           3219440128
 number of write operations:     23897
 number of bytes written:        164849664
 number of flush operations:     11577
 total duration of reads (ns):   1005410244506
 total duration of writes (ns):  1085306686457
 total duration of flushes (ns): 340645193294
```

14.5.6. ネットワーク統計の取得

domnetstat [domain][interface-device] コマンドは、特定のドメインで実行されている特定のデバイスのネットワークインターフェイス統計を表示します。

```
# domifstat rhel6 eth0
```

14.5.7. ドメインの仮想インターフェイスのリンク状態の変更

次のコマンドは、指定されたインターフェイスをアップまたはダウンとして設定できます。

```
# domif-setlink [domain][interface-device][state]{--config}
```

これを使用すると、指定されたドメインの指定されたインターフェイスのステータスが変更されます。ドメインの永続的な設定のみを変更する場合は、**--config** オプションを使用する必要があることに注意してください。互換性の理由から、**--persistent** は **--config** のエイリアスであることに注意してください。interface device は、インターフェイスのターゲット名または MAC アドレスにすることができます。

```
# domif-setlink rhel6 eth0 up
```

14.5.8. ドメインの仮想インターフェイスのリンク状態の一覧表示

このコマンドを使用して、特定のドメイン上の指定されたインターフェイスの状態をクエリーできます。ドメインの永続的な設定のみを変更する場合は、**--config** オプションを使用する必要があることに注意してください。互換性の理由から、**--persistent** は **--config** のエイリアスであることに注意してください。interface device は、インターフェイスのターゲット名または MAC アドレスにすることができます。

```
# domif-getlink rhel6 eth0 up
```

14.5.9. ネットワークインターフェイスの帯域幅パラメーターの設定

domiftune は、ゲスト仮想マシンのネットワークインターフェイス帯域幅パラメーターを設定します。以下の形式を使用する必要があります。

```
#virsh domiftune domain interface-device [[--config] [--live] | [--current]] [--inbound
average,peak,burst] [--outbound average,peak,burst]
```

-

必須パラメーターはゲスト仮想マシンのドメイン名とインターフェイスデバイスのみで、**--config**、**--live**、**--current** の機能は 「スケジュールパラメーターの設定」 と同じです。制限が指定されていない場合は、現在のネットワークインターフェイス設定をクエリーします。それ以外の場合は、以下のオプションで制限を変更します。

- <interface-device> これは必須であり、ドメインのネットワークインターフェイスの帯域幅パラメーターを設定またはクエリーします。**interface-device** は、インターフェイスのターゲット名 (<target dev='name'/>) または MAC アドレスにすることができます。
- **--inbound** または **--outbound** が指定されていない場合、このコマンドは帯域幅設定をクエリーして表示します。それ以外の場合は、インバウンドまたはアウトバウンドの帯域幅を設定します。average、peak、burst は、**attach-interface** コマンドの場合と同じです。「インターフェイスデバイスの接続」 を参照してください。

14.5.10. 実行中のドメインのメモリー統計の取得

このコマンドは、使用しているハイパーバイザーに応じてさまざまな結果を返す場合があります。

dommemstat [domain] [--period (sec)][[--config][--live]|[--current]] は、実行中のドメインのメモリー統計情報を表示します。**--period** オプションを使用するには、秒単位の期間が必要です。このオプションを 0 より大きい値に設定すると、バルーンドライバーは追加の統計を返します。これは、後続の **domemstat** コマンドにより表示されます。**--period** オプションを 0 に設定すると、バルーンドライバーの収集は停止しますが、バルーンドライバーの統計情報はクリアされません。バルーンドライバーの収集期間を設定するために **--period** オプションも設定しないと、**--live**、**--config**、**--current** オプションを使用することはできません。**--live** オプションが指定されている場合、実行中のゲストの収集期間のみが影響を受けます。**--config** オプションを使用すると、永続ゲストの次回の起動に影響します。**--current** オプションを使用すると、現在のゲストの状態に影響します

--live オプションおよび **--config** オプションの両方を使用できますが、**--current** は使用できません。オプションが指定されていない場合、ゲストの状態によって動作が異なります。

```
#virsh domemstat rhel6 --current
```

14.5.11. ブロックデバイスのエラーの表示

このコマンドは、I/O エラーのためにドメインが一時停止していることをレポートする **domstate** の後に使用するのが最適です。**domblkerror *domain*** コマンドは、特定のドメインでエラー状態にあるすべてのブロックデバイスを表示し、デバイスが報告しているエラーメッセージを表示します。

```
# virsh domblkerror rhel6
```

14.5.12. ブロックデバイスのサイズの表示

この場合、ブロックデバイスは一意のターゲット名 (<target dev='name'/>) またはソースファイル (<source file ='name'/>) です。リストを取得するには、**domblklist** を実行します。この **domblkinfo** には *ドメイン* 名が必要です。

```
# virsh domblkinfo rhel6
```

14.5.13. ドメインに関連付けられたブロックデバイスの表示

domblklist *domain* --inactive --details は、指定されたドメインに関連付けられているすべてのブロックデバイスのテーブルを表示します。

--inactive を指定すると、次回のシステムの起動時に使用されるデバイスが結果に表示されます。また、ドメインで現在使用中のデバイスは表示されません。**--details** を指定すると、ディスクのタイプとデバイス値がテーブルに含まれます。この表に表示される情報は、 **domblkinfo** および **snapshot-create** で使用できます。

```
#domblklist rhel6 --details
```

14.5.14. ドメインに関連付けられた仮想インターフェイスの表示

domiflist コマンドを実行すると、指定したドメインに関連付けられているすべての仮想インターフェイスの情報を表示するテーブルが表示されます。**domiflist** には *ドメイン* 名が必要であり、オプションで **--inactive** オプションを使用できます。

--inactive を指定すると、次回のシステムの起動時に使用されるデバイスが結果に表示されます。また、ドメインで現在使用中のデバイスは表示されません。

仮想インターフェイスの MAC アドレスを必要とするコマンド (**detach-interface** や **domif-setlink** など) は、このコマンドによって表示される出力を受け入れます。

14.5.15. blockcommit を使用してバッキングチェーンを短縮する

このセクションでは、**virsh blockcommit** を使用してバッキングチェーンを短縮する方法を示します。バッキングチェーンの背景については、「ライブブロックコピーを使用したディスクイメージ管理」を参照してください。

blockcommit は、チェーンの一部からバッキングファイルにデータをコピーし、コミットされた部分をバイパスするためにチェーンの残りの部分をピボットできるようにします。たとえば、これが現在の状態であるとします。

```
    base ← snap1 ← snap2 ← active.
```

blockcommit を使用すると、snap2 のコンテンツが snap1 に移動します。これにより、チェーンから snap2 を削除できるため、バックアップが非常に速くなります。

手順14.2 virsh blockcommit

- 以下のコマンドを実行します。

  ```
  # virsh blockcommit $dom $disk -base snap1 -top snap2 -wait -verbose
  ```

  snap2 の内容が snap1 に移動し、以下のような結果になります。

  base ← **snap1** ← **active**.Snap2 が無効になり、削除できるようになりました。

警告

blockcommit は***-base***オプションに依存するすべてのファイルを破壊します (***-top***オプションに依存するファイル以外は、それらのファイルがベースを指すようになったため)。これを回避するには、複数のゲストが共有するファイルに変更をコミットしないでください。***-verbose***オプションにより、進捗状況を画面上に出力することができます。

14.5.16. ブロックプルを使用してバッキングチェーンを短縮する

blockpull は、以下のアプリケーションで使用できます。

- バッキングイメージチェーンからのデータをイメージに入力して、イメージをフラット化します。これにより、イメージファイルは自己完結型になり、背面イメージに依存しなくなり、以下のようになります。
  - 前: base.img ← Active
  - 後: base.img がゲストで使用されなくなり、Active にすべてのデータが含まれるようになりました。
- バッキングイメージチェーンの一部を平坦化します。これを使用すると、スナップショットをトップレベルイメージに平坦化し、以下のようになります。
  - 前: base ← sn1 ←sn2 ← active
  - 後: base.img ← active.アクティブには、sn1 および sn2 のすべてのデータが含まれるようになりました。また、ゲストでは sn1 も sn2 も使用されないことに注意してください。
- ディスクイメージを、ホストの新しいファイルシステムに移動します。これにより、ゲストの実行中にイメージファイルを移動できます。以下のようになります。
  - 前 (元のイメージファイル) - **/fs1/base.vm.img**
  - 後: **/fs2/active.vm.qcow2** が新しいファイルシステムになり、**/fs1/base.vm.img** が使用されなくなります。
- コピー後のストレージ移行を使用したライブマイグレーションに役立ちます。ディスクイメージは、ライブマイグレーションの完了後に、移行元ホストから移行先ホストにコピーされます。

  要するに、こういうことです: 前: **/source-host/base.vm.img** 後: **/destination-host/active.vm.qcow2**。**/source-host/base.vm.img** は使用されなくなりました。

手順14.3 ブロックプルを使用してバッキングチェーンを短縮する

1. **blockpull** を実行する前にこのコマンドを実行すると役立つ場合があります。

   ```
   # virsh snapshot-create-as $dom $name - disk-only
   ```

2. チェーンが次のようになっている場合: **base** ← **snap1** ← **snap2** ← **active** 次を実行します。

```
# virsh blockpull $dom $disk snap1
```

このコマンドは、snap2 からデータをアクティブにプルすることで、snap1 のバッキングファイルをアクティブな状態にします。その結果、base ← snap1 ← active になります。

3. **blockpull**が完了すると、チェーンに追加イメージを作成したスナップショットの**libvirt**追跡は役に立ちなくなります。次のコマンドを使用して、古いスナップショットの追跡を削除します。

```
# virsh snapshot-delete $dom $name - metadata
```

blockpull 追加アプリケーションは、以下のように実行できます。

- 単一のイメージをフラット化し、そのバッキングイメージチェーンからのデータを取り込むには: **# virsh blockpull *example-domain* vda - wait**
- バッキングイメージチェーンの一部をフラット化するには: **# virsh blockpull *example-domain* vda - base /path/to/base.img - wait**
- ディスクイメージをホスト上の新しいファイルシステムに移動するには: **# virsh snapshot-create *example-domain*e - xmlfile /path/to/new.xml - disk-only** に続いて **# virsh blockpull *example-domain* vda - wait**
- コピー後のストレージ移行でライブ移行を使用する
  - destination 実行時:

    ```
    # qemu-img create -f qcow2 -o backing_file=/source-host/vm.img /destination-host/vm.qcow2
    ```

  - ソース実行時:

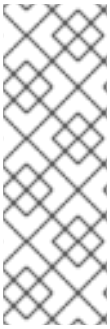
    ```
    # virsh migrate example-domain
    ```

  - destination 実行時:

    ```
    # virsh blockpull example-domain vda - wait
    ```

14.5.17. blockresize を使用してドメインパスのサイズを変更する

blockresize は、ドメインの実行中にドメインのブロックデバイスのサイズを変更するために使用できます。これには、一意のターゲット名 (**<target dev="name"/>**) またはソースファイルにも対応するブロックデバイスの絶対パスが使用されます。(**<source file="name"/>**)。これは、ドメインに接続されているディスクデバイスの 1 つに適用できます (コマンド **domblklist** を使用して、特定のドメインに関連付けられているすべてのブロックデバイスの簡単な情報を示すテーブルを出力できます)。

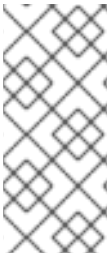


注記

ライブイメージのサイズを変更すると、イメージのサイズは常に変更されますが、ゲストはすぐにはサイズを変更しない場合があります。最近のゲストカーネルでは、virtio-blk デバイスのサイズが自動的に更新されます (古いカーネルではゲストを再起動する必要があります)。SCSI デバイスの場合は、**echo > /sys/class/scsi_device/0:0:0:0/device/rescan** コマンドを使用して、ゲストの再スキャンを手動でトリガーする必要があります。さらに、IDE では、新しいサイズを取得する前にゲストを再起動する必要があります。

- 次のコマンドを実行します: **blockresize [domain] [path size]**。ここで、
  - Domain は、サイズを変更するドメインの一意のターゲット名またはソースファイルです
  - Path size はスケーリングされた整数であり、接尾辞がない場合、デフォルトで KiB (1024 バイトのブロック) になります。バイトには B の接尾辞を使用する必要があります。

14.5.18. ライブブロックコピーを使用したディスクイメージ管理



注記

ライブブロックコピーは、Red Hat EnterpriseLinux で提供されているバージョンの KVM ではサポートされていない機能です。ライブブロックコピーは、RedHat Virtualization に付属しているバージョンの KVM で利用できます。この機能をサポートするには、このバージョンの KVM が物理ホストマシンで実行されている必要があります。詳細については、Red Hat の担当者にお問い合わせください。

ライブブロックコピーを使用すると、使用中のゲストディスクイメージを宛先イメージにコピーし、ゲストの実行中にゲストディスクイメージを宛先ゲストイメージに切り替えることができます。ライブマイグレーションがホストのメモリーとレジストリーの状態を移動する間、ゲストは共有ストレージに保持されます。ライブブロックコピーを使用すると、ゲストの実行中にゲストコンテンツ全体をその場で別のホストに移動できます。ライブブロックコピーは、永続的な共有ストレージを必要とせずにライブ移行に使用することもできます。この方法では、移行後、ゲストの実行中にディスクイメージが移行先ホストにコピーされます。

ライブブロックコピーは、次のアプリケーションで特に役立ちます。

- ゲストイメージをローカルストレージから中央の場所に移動する
- メンテナーンスが必要な場合、パフォーマンスを損なうことなく、ゲストを別の場所に移動できます
- 速度と効率のためにゲストイメージの管理を可能にします
- ゲストをシャットダウンせずにイメージ形式の変換を行うことができます

例14.1 ライブブロックコピーを使用した例

この例は、ライブブロックコピーが実行されたときに何が起こるかを示しています。この例には、ソースと宛先の間で共有されるバッキングファイル (ベース) があります。また、ソースにのみ存在し、コピーする必要がある 2 つのオーバーレイ (sn1 と sn2) があります。

1. 最初のバッキングファイルチェーンは次のようになります。

   base ← sn1 ← sn2

コンポーネントは以下のとおりです。

- ベース - 元のディスクイメージ
- sn1 - ベースディスクイメージから取得された最初のスナップショット
- sn2 - 最新のスナップショット
- アクティブ - ディスクのコピー

2. イメージのコピーが sn2 の上に新しいイメージとして作成されると、結果は次のようになります。

   base ← sn1 ← sn2 ← active

3. この時点で、読み取り権限はすべて正しい順序であり、自動的に設定されます。書き込み権限が適切に設定されていることを確認するために、ミラーメカニズムはすべての書き込みを sn2 とアクティブの両方にリダイレクトし、sn2 とアクティブがいつでも同じように読み取るようにします (このミラーメカニズムは、ライブブロックコピーとイメージストリーミングの本質的な違いです)。

4. すべてのディスククラスターをループするバックグラウンドタスクが実行されます。クラスターごとに、次のケースとアクションが考えられます。

   - クラスターはすでにアクティブに割り当てられており、何もする必要はありません。
   - **bdrv_is_allocated()** を使用して、バッキングファイルチェーンを追跡します。クラスターが (共有されている) ベースから読み取られる場合、何もする必要はありません。
   - **bdrv_is_allocated()** バリアントが実行可能でない場合は、イメージをリベースし、読み取りデータをベースの書き込みデータと比較して、コピーが必要かどうかを判断します。
   - 他のすべての場合は、クラスターを **active** にコピーします

5. コピーが完了すると、アクティブのバッキングファイルがベースに切り替えられます (リベースと同様)

一連のスナップショットの後でバッキングチェーンの長さを短くするには、**blockcommit** および **blockpull** のコマンドが役立ちます。詳細は、「blockcommit を使用してバッキングチェーンを短縮する」 を参照してください。

14.5.19. グラフィック表示への接続の URI の表示

virsh domdisplay コマンドを実行すると、URI が出力されます。この URI を使用して、VNC、SPICE、または RDP を介してドメインのグラフィック表示に接続できます。**--include-password** オプションを使用すると、SPICE チャネルのパスワードが URI に含まれます。

14.5.20. ドメイン取得コマンド

次のコマンドは、特定のドメインに関するさまざまな情報を表示します

- **virsh domhostname *domain*** は、ハイパーバイザーが公開できる場合、指定されたドメインのホスト名を表示します。

- **virsh dominfo *domain*** は、指定されたドメインに関する基本情報を表示します。
- **virsh domuid *domain*|*ID*** は、指定されたドメイン名または ID を UUID に変換します。
- **virsh domid *domain*|*ID*** は、指定されたドメイン名または UUID を ID に変換します。
- **virsh domjobabort *domain*** は、指定されたドメインで現在実行中のジョブを中止します。
- **virsh domjobinfo *domain*** は、移行統計情報など、指定されたドメインで実行されているジョブに関する情報を表示します
- **virsh domname *domain ID*|*UUID*** は、指定されたドメイン ID または UUID をドメイン名に変換します。
- **virsh domstate *domain*** は、指定されたドメインの状態を表示します。**--reason** オプションを使用すると、表示された状態の理由も表示されます。
- **virsh domcontrol *domain*** は、ドメインの制御に使用された VMM へのインターフェイスの状態を表示します。OK でも Error でもない状態の場合は、制御インターフェイスが表示状態に入ってから経過した秒数も出力されます。

例14.2 統計的フィードバックの例

ドメインに関する情報を取得するには、次のコマンドを実行します。

```
# virsh domjobinfo rhel6
Job type:         Unbounded
Time elapsed:     1603        ms
Data processed:   47.004 MiB
Data remaining:   658.633 MiB
Data total:       1.125 GiB
Memory processed: 47.004 MiB
Memory remaining: 658.633 MiB
Memory total:     1.125 GiB
Constant pages:   114382
Normal pages:     12005
Normal data:      46.895 MiB
Expected downtime: 0          ms
Compression cache: 64.000 MiB
Compressed data:  0.000 B
Compressed pages: 0
Compression cache misses: 12005
Compression overflows: 0
```

14.5.21. QEMU 引数のドメイン XML への変換

virsh domxml-from-native は、libvirt が使用できる libvirt ドメイン XML を使用して、既存の QEMU 引数のセットをゲスト description に変換する方法を提供します。このコマンドは、libvirt で管理できるように、コマンドラインから起動した既存の qemu ゲストを変換する場合にのみ使用することを目的としています。ここで説明する方法を使用して、新規ゲストを作成することはできません。新しいゲストは、virsh または virt-manager のいずれかを使用して作成する必要があります。追加情報は ここ にあります。

次の args ファイルを持つ QEMU ゲストがいるとします。

```
 $ cat demo.args
LC_ALL=C
PATH=/bin
HOME=/home/test
USER=test
LOGNAME=test /usr/bin/qemu -S -M pc -m 214 -smp 1 -nographic -monitor pty -no-acpi -boot c -hda
/dev/HostVG/QEMUGuest1 -net none -serial none -parallel none -usb
```

これをドメイン XML ファイルに変換して、ゲストを libvirt で管理できるようにするには、次のコマンドを実行します。

```
$ virsh domxml-from-native qemu-argv demo.args
```

このコマンドは、上記の args ファイルを次のドメイン XML ファイルに変換します。

```
<domain type='qemu'>
  <uuid>00000000-0000-0000-0000-000000000000</uuid>
  <memory>219136</memory>
  <currentMemory>219136</currentMemory>
  <vcpu>1</vcpu>
  <os>
    <type arch='i686' machine='pc'>hvm</type>
    <boot dev='hd'/>
  </os>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/bin/qemu</emulator>
    <disk type='block' device='disk'>
      <source dev='/dev/HostVG/QEMUGuest1'/>
      <target dev='hda' bus='ide'/>
    </disk>
  </devices>
</domain>
```

14.5.22. ドメインのコアのダンプファイルの作成

分析できるように、ドメインのコアを含むダンプファイルを作成する必要がある場合があります (特にトラブルシューティングの場合)。この場合、**virsh dump *domain corefilepath* --bypass-cache --live |--crash |--reset --verbose --memory-only** を実行すると、ドメインコアが *corefilepath* で指定されたファイルにダンプされます。一部のハイパーバイザーが提供する場合があることに注意してください。このアクションに対する制限があり、*corefilepath* パラメーターで指定されたファイルとパスに対する適切なアクセス許可を手動で確認する必要がある場合があります。このコマンドは、SR-IOV デバイスやその他のパススルーデバイスでサポートされます。次のオプションがサポートされており、次の効果があります。

- **--bypass-cache** 保存されたファイルには、ファイルシステムキャッシュは含まれません。このオプションを選択すると、ダンプ操作が遅くなる可能性があることに注意してください。
- **--live** は、ドメインの実行を継続するときにファイルを保存し、ドメインを一時停止または停止しません。

- **--crash** は、ダンプファイルの保存中にドメインを一時停止状態のままにするのではなく、クラッシュ状態にします。
- **--reset** ダンプファイルが正常に保存されると、ドメインがリセットされます。
- **--verbose** は、ダンププロセスの進捗を表示します。
- **--memory-only** ダンプファイルに保存される情報は、ドメインのメモリーと CPU 共通レジスタファイルのみです。

プロセス全体が **domjobinfo** コマンドを使用してモニターでき、**domjobabort** コマンドを使用してキャンセルできることに注意してください。

14.5.23. 仮想マシンの XML ダンプの作成 (設定ファイル)

virsh を使用してゲスト仮想マシンの XML 設定ファイルを出力します。

```
# virsh dumpxml {guest-id, guestname or uuid}
```

このコマンドは、ゲスト仮想マシンの XML 設定ファイルを標準出力 (**stdout**) に出力します。出力をファイルにパイプすることでデータを保存できます。出力を *guest.xml* というファイルにパイプする例:

```
# virsh dumpxml GuestID > guest.xml
```

このファイル **guest.xml** は、ゲスト仮想マシンを再作成できます (参照 「ゲスト仮想マシンの設定ファイルの編集」。この XML 設定ファイルを編集して、追加のデバイスを設定したり、追加のゲスト仮想マシンをデプロイしたりできます。

virsh dumpxml 出力の例:

```
# virsh dumpxml guest1-rhel6-64
<domain type='kvm'>
  <name>guest1-rhel6-64</name>
  <uuid>b8d7388a-bbf2-db3a-e962-b97ca6e514bd</uuid>
  <memory>2097152</memory>
  <currentMemory>2097152</currentMemory>
  <vcpu>2</vcpu>
  <os>
    <type arch='x86_64' machine='rhel6.2.0'>hvm</type>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
    <pae/>
  </features>
  <clock offset='utc'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>restart</on_crash>
  <devices>
    <emulator>/usr/libexec/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw' cache='none' io='threads'/>
      <source file='/home/guest-images/guest1-rhel6-64.img'/>
```

```
    <target dev='vda' bus='virtio'/>
    <shareable/<
    <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'/>
  </disk>
  <interface type='bridge'>
    <mac address='52:54:00:b9:35:a9'/>
    <source bridge='br0'/>
    <model type='virtio'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
  </interface>
  <serial type='pty'>
    <target port='0'/>
  </serial>
  <console type='pty'>
    <target type='serial' port='0'/>
  </console>
  <input type='tablet' bus='usb'/>
  <input type='mouse' bus='ps2'/>
  <graphics type='vnc' port='-1' autoport='yes'/>
  <sound model='ich6'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
  </sound>
  <video>
    <model type='cirrus' vram='9216' heads='1'/>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
  </video>
  <memballoon model='virtio'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
  </memballoon>
 </devices>
</domain>
```

<shareable/> フラグが設定されていることに注意してください。これは、デバイスがドメイン間で共有されることが期待されていることを示します (ハイパーバイザーと OS がこれをサポートしていると仮定)。つまり、そのデバイスのキャッシュを非アクティブ化する必要があります。

14.5.24. 設定ファイルからのゲスト仮想マシンの作成

ゲスト仮想マシンは、XML 設定ファイルから作成できます。以前に作成したゲスト仮想マシンから既存の XML をコピーするか、**dumpxml** オプションを使用できます (参照 「仮想マシンの XML ダンプの作成 (設定ファイル)」)。XML ファイルから **virsh** を使用してゲスト仮想マシンを作成するには:

```
# virsh create configuration_file.xml
```

14.6. ゲスト仮想マシンの設定ファイルの編集

dumpxml オプションを使用する代わりに (参照 「仮想マシンの XML ダンプの作成 (設定ファイル)」)、ゲスト仮想マシンは、実行中またはオフライン中に編集できます。**virsh edit** コマンドは、この機能を提供します。たとえば、***rhel6*** という名前のゲスト仮想マシンを編集するには、以下を実行します。

```
# virsh edit rhel6
```

これにより、テキストエディターが開きます。デフォルトのテキストエディターは **$EDITOR** シェルパラメーターです (デフォルトでは **vi** に設定されています)。

14.6.1. KVM ゲスト仮想マシンへの複合 PCI デバイスの追加

このセクションでは、多機能 PCI デバイスを KVM ゲスト仮想マシンに追加する方法を示します。

1. **virsh edit *[guestname]*** コマンドを実行して、ゲスト仮想マシンの XML 設定ファイルを編集します。
2. アドレスタイプタグに、**function='0x0'** の **multifunction='on'** エントリーを追加します。

   これにより、ゲスト仮想マシンで多機能 PCI デバイスを使用できるようになります。

   ```
   <disk type='file' device='disk'>
   <driver name='qemu' type='raw' cache='none'/>
   <source file='/var/lib/libvirt/images/rhel62-1.img'/>
   <target dev='vda' bus='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' multifunction='on'/
   </disk>
   ```

   2 つの機能を持つ PCI デバイスの場合は、XML 設定ファイルを修正して、最初のデバイスと同じスロット番号と、別の機能番号 (**function='0x1'** など) を持つ 2 番目のデバイスを追加します。

   以下に例を示します。

   ```
   <disk type='file' device='disk'>
   <driver name='qemu' type='raw' cache='none'/>
   <source file='/var/lib/libvirt/images/rhel62-1.img'/>
   <target dev='vda' bus='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0'
   multifunction='on'/>
   </disk>
   <disk type='file' device='disk'>
   <driver name='qemu' type='raw' cache='none'/>
   <source file='/var/lib/libvirt/images/rhel62-2.img'/>
   <target dev='vdb' bus='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x1'/>
   </disk>
   ```

3. KVM ゲスト仮想マシンからの **lspci** 出力は次のことを示しています。

   ```
   $ lspci

   00:05.0 SCSI storage controller: Red Hat, Inc Virtio block device
   00:05.1 SCSI storage controller: Red Hat, Inc Virtio block device
   ```

14.6.2. 実行中のドメインを停止して後で再起動する

virsh managedsave *domain* --bypass-cache --running | --paused | --verbose は、実行中のドメインを保存および破棄 (停止) して、後で同じ状態から再起動できるようにします。**virsh start** コマンドと併用すると、この保存ポイントから自動的に開始されます。**--bypass-cache** オプションとともに使用

すると、保存によってファイルシステムのキャッシュが回避されます。このオプションを使用すると、保存プロセスの速度が低下する可能性があることに注意してください。

--verbose は、ダンププロセスの進捗を表示します。

通常の条件では、管理保存は、保存が完了したときにドメインが置かれている状態によって決定される実行状態と一時停止状態のどちらを使用するかを決定します。ただし、これは、**--running** オプションを使用して、実行状態のままにする必要があることを示したり、**--paused** オプションを使用して、一時停止状態のままにすることを示したりして上書きできます。

管理対象の保存状態を削除するには、**virsh managedsave-remove** コマンドを使用します。これにより、ドメインが次に起動したときに、ドメインが強制的にフルブートします。

管理保存プロセス全体は、**domjobinfo** コマンドを使用して監視できます。**domjobabort** コマンドを使用して監視を中止することもできます。

14.6.3. 指定されたドメインの CPU 統計情報の表示

virsh cpu-stats *domain* --total start count コマンドは、指定されたドメインの CPU 統計情報を提供します。デフォルトでは、すべての CPU の統計および合計が表示されます。**--total** オプションは、合計統計のみを表示します。

14.6.4. スクリーンショットの保存

virsh screenshot コマンドは現在のドメインコンソールのスクリーンショットを取り、それをファイルに保存します。ただし、ハイパーバイザーがドメインに対してより多くの表示をサポートしている場合は、**--screen** を使用して画面 ID を指定すると、キャプチャする画面を指定できます。複数のグラフィックカードがあり、デバイスの前にヘッドが番号付けされている場合、画面 ID5 は 2 番目のカードの 2 番目のヘッドをアドレス指定します。

14.6.5. 指定されたドメインへのキーストロークの組み合わせの送信

virsh send-key *domain* --codeset --holdtime *keycode* を使用すると、シーケンスを *keycode* として特定のドメインに送信できます。

各 *キーコード* は、数値または対応する *コードセット* のシンボリック名になります。複数の *キーコード* を指定すると、すべてがゲスト仮想マシンに同時に送信されるため、順不同で受信されることがあります。個別のキーコードが必要な場合は、**send-key** コマンドを複数回送信する必要があります。

```
# virsh send-key rhel6 --holdtime 1000 0xf
```

--holdtime を指定すると、各キーストロークは指定した時間 (ミリ秒単位) 保持されます。**--codeset** ではコードセットを指定できます。デフォルトは Linux ですが、以下のオプションを使用できます。

- **linux** - このオプションを選択すると、シンボリック名が対応する Linux キー定数マクロ名に一致するようになります。数値は、Linux 汎用入力イベントサブシステムが提供するものになります。
- **xt** - XT キーボードコントローラーで定義する値を送信します。シンボリック名は記載されていません。
- **atset1** - 数値は、AT キーボードコントローラー、set1 (XT 互換セット) で定義されるものです。atset1 から拡張されたキーコードは、XT コードセットの拡張キーコードとは異なる場合があります。シンボリック名は記載されていません。

- **atset2** - 数値は、AT キーボードコントローラー、set 2 で定義されるものです。シンボリック名は記載されていません。
- **atset3** - 数値は、AT キーボードコントローラー、set 3 (PS/2 互換) で定義されるものです。シンボリック名は記載されていません。
- **os_x** - 数値は、OS-X キーボード入力サブシステムで定義されるものです。シンボリック名は、対応する OS-X の鍵定数マクロ名と一致します。
- **xt_kbd** - 数値は、Linux KBD デバイスーで定義されるものです。これは、元の XT コードセットのバリアントですが、拡張キーコードのエンコードが異なります。シンボリック名は記載されていません。
- **win32** - 数値は、Win32 キーボード入力サブシステムで定義されるものです。シンボリック名は、対応する Win32 鍵定数マクロ名と一致します。
- **usb** - 数値は、キーボード入力の USB HID で定義されるものです。シンボリック名は記載されていません。
- **rfb** - 数値は、raw キーコードを送信するために RFB 拡張で定義されるものです。これは XT コードセットのバリアントですが、拡張キーコードでは、最初のバイトの上位ビットではなく、2 番目のビットセットの下位ビットが使用されます。シンボリック名は記載されていません。

14.6.6. プロセス信号名を仮想プロセスに送信する

virsh send-process-signal *domain-ID PID signame* コマンドを使用すると、指定されたシグナル (*signame* で識別) が仮想ドメイン (*ドメイン ID* で指定) で実行され、プロセス ID(*PID*) で識別されるプロセスに送信されます。

整数信号定数またはシンボリック信号名のいずれかをこの方法で送信できます。したがって、たとえば、次の両方のコマンドは、**rhel6** ドメインのプロセス ID **187** に **kill** シグナルを送信します。

```
# virsh send-process-signal rhel6 187 kill
# virsh send-process-signal rhel6 187 9
```

使用可能なシグナルとその使用法の完全なリストについては、virsh(1) および signal(7) のマニュアルページを参照してください。

14.6.7. VNC ディスプレイの IP アドレスとポート番号の表示

virsh vncdisplay は、指定されたドメインの VNC ディスプレイの IP アドレスとポート番号を出力します。情報が利用できない場合は、終了コード 1 が表示されます。

```
# virsh vncdisplay rhel6
127.0.0.1:0
```

14.7. NUMA ノード管理

このセクションには、NUMA ノード管理に必要なコマンドが含まれています。

14.7.1. ノード情報の表示

nodeinfo コマンドは、モデル番号、CPU 数、CPU の種類、物理メモリーのサイズなど、ノードに関する基本情報を表示します。出力は **virNodeInfo** 構造に対応します。具体的には、"CPU socket(s)" フィールドは NUMA セルごとの CPU ソケット数を示します。

```
$ virsh nodeinfo
CPU model:           x86_64
CPU(s):              4
CPU frequency:       1199 MHz
CPU socket(s):       1
Core(s) per socket:  2
Thread(s) per core:  2
NUMA cell(s):        1
Memory size:         3715908 KiB
```

14.7.2. NUMA パラメーターの設定

virsh numatune は、指定されたドメインの NUMA パラメーターを設定または取得できます。ドメイン XML ファイル内では、これらのパラメーターは **<numatune>** 要素内にネストされています。オプションを使用しない場合は、現在の設定のみが表示されます。**numatune** ***domain*** コマンドには指定されたドメインが必要であり、次のオプションを選択できます。

- **--mode** - モードは、**strict**、**インターリーブ**、または **推奨** のいずれかに設定できます。実行中のドメインは、ドメインが **strict** モードで開始されていない限り、ライブ中にモードを変更することはできません。
- **--nodeset** には、ドメインを実行するためにホスト物理マシンによって使用される NUMA ノードのリストが含まれています。この一覧には、それぞれがコンマで区切られたノードが含まれ、ノード範囲に使用されるダッシュ**-**と、ノードの除外に使用されるカレット**^** も含まれています。
- インスタンスごとに使用できるオプションは、次の 3 つのうち 1 つだけです。
  - **--config** は、永続ゲスト仮想マシンの次回の起動時に有効になります。
  - **--live** は、実行中のゲスト仮想マシンのスケジューラー情報を設定します。
  - **--current** は、ゲスト仮想マシンの現在の状態に影響を及ぼします。

14.7.3. NUMA セルの空きメモリー量の表示

virsh freecell は、指定された NUMA セル内に、マシンで利用可能なメモリー容量を表示します。このコマンドは、指定したオプションに応じて、マシンで利用可能なメモリーを、3 つの異なるディスプレイのいずれかに表示します。オプションを使用しない場合は、マシンの空きメモリーの合計が表示されます。**--all** オプションを使用すると、各セルの空きメモリーとマシンの合計空きメモリーが表示されます。数値引数を使用するか、セル番号とともに **--cellno** オプションを使用すると、指定したセルの空きメモリーが表示されます。

14.7.4. CPU リストの表示

nodecpumap コマンドは、ノードがオンラインであるかどうかに関係なく、ノードで使用可能な CPU の数を表示し、現在オンラインである CPU の数も一覧表示します。

```
$ virsh nodecpumap
  CPUs present: 4
  CPUs online: 1
```

```
CPU map: y
```

14.7.5. CPU 統計の表示

nodecpustats コマンドは、CPU が指定されている場合、指定された CPU に関する統計情報を表示します。そうでない場合は、ノードの CPU ステータスが表示されます。パーセントを指定すると、1秒間隔で記録された各タイプの CPU 統計のパーセンテージが表示されます。

この例は、CPU が指定されていないことを示しています。

```
$ virsh nodecpustats
user:                1056442260000000
system:               401675280000000
idle:                7549613380000000
iowait:                94593570000000
```

この例は、CPU 番号 2 の統計パーセンテージを示しています。

```
$ virsh nodecpustats 2 --percent
usage:          2.0%
user:           1.0%
system:         1.0%
idle:          98.0%
iowait:         0.0%
```

ゲスト仮想マシンの設定ファイルの **on_reboot** 要素を変更することで、再起動中のゲスト仮想マシンの動作を制御できます。

14.7.6. ホスト物理マシンの一時停止

nodesuspend コマンドは、ホスト物理マシンを、Suspend-to-RAM(s3)、Suspend-to-Disk (s4)、または Hybrid-Suspend と同様のシステム全体のスリープ状態にし、Real-Time-Clock をセットアップします。設定された期間が経過した後にノードをウェイクアップします。**--target** オプションは、次のいずれかに設定できます。***mem***、***disk***、または***hybrid***。これらのオプションは、メモリー、ディスク、または 2 つの組み合わせを一時停止するように設定することを示します。**--duration** を設定すると、設定した期間が終了した後に起動するようにホスト物理マシンに指示します。秒単位で設定されます。継続時間は 60 秒より長くすることをお勧めします。

```
$ virsh nodesuspend disk 60
```

14.7.7. ノードメモリーパラメーターの設定および表示

node-memory-tune [shm-pages-to-scan] [shm-sleep-milisecs] [shm-merge-across-nodes] コマンドが表示され、ノードメモリーパラメーターを設定できるようになります。このコマンドで設定できるパラメーターは 3 つあります。

- **shm-pages-to-scan** - 共有メモリーサービスがスリープ状態になる前にスキャンするページ数を設定します。
- **shm-sleep-milisecs** - 次のスキャンの前に共有メモリーサービスがスリープするミリ秒数を設定します
- **shm-merge-across-nodes** - 別の NUMA ノードのページをマージできるかどうかを指定します。許可される値は ***0*** と ***1***です。***0*** に設定すると、マージ可能なページのみが、同じ NUMA

ノードのメモリー領域に存在するページのみです。***1*** に設定すると、全 NUMA ノードのページをマージすることができます。デフォルト設定は次のとおりです***1***。

14.7.8. ホストノードでのデバイスの作成

virsh nodedev-create *file* コマンドを使用すると、ホストの物理マシンにデバイスを作成し、ゲスト仮想マシンに割り当てることができます。**libvirt** は通常、使用可能なホストノードを自動的に検出しますが、このコマンドを使用すると、**libvirt** が検出しなかったホストハードウェアを登録できます。 *ファイル* には、ノードデバイスの最上位 **<デバイス>** の XML の説明が含まれている必要があります。

このデバイスを停止するには、**nodedev-destroy *device*** コマンドを使用します。

14.7.9. ノードデバイスの切り離し

virsh nodedev-detach は、nodedev をホストから切り離して、**<hostdev>** パススルーを介してゲストが安全に使用できるようにします。このアクションは **nodedev-reattach** コマンドで元に戻すことができますが、マネージドサービスでは自動的に実行されます。このコマンドは、**nodedev-dettach** も受け入れます。

異なるドライバーは、デバイスが異なるダミーデバイスにバインドされることを想定していることに注意してください。**--driver** オプションを使用すると、目的のバックエンドドライバーを指定できます。

14.7.10. デバイスの設定設定の取得

virsh nodedev-dumpxml *[device]* コマンドは、指定されたノード **<デバイス>**の XML 設定ファイルをダンプします。XML 設定には、デバイス名、たとえばデバイスを所有するバス、ベンダー、製品 ID などの情報が含まれます。引数の *device* は、デバイス名にすることも、WWNN | WWPN 形式の WWN の組み合わせにすることもできます (HBA のみ)。

14.7.11. ノード上のデバイスの一覧表示

virsh nodedev-list *cap* --tree コマンドは、**libvirt** によって認識されているノードで使用可能なすべてのデバイスを一覧表示します。*cap* は、機能タイプでリストをフィルターリングするために使用されます。各タイプはコンマで区切られ、**--tree** と一緒に使用することはできません。**--tree** オプションを使用すると、次のように出力がツリー構造になります。

```
 #  virsh nodedev-list --tree
 computer
|
+- net_lo_00_00_00_00_00_00
+- net_macvtap0_52_54_00_12_fe_50
+- net_tun0
+- net_virbr0_nic_52_54_00_03_7d_cb
+- pci_0000_00_00_0
+- pci_0000_00_02_0
+- pci_0000_00_16_0
+- pci_0000_00_19_0
|  |
|  +- net_eth0_f0_de_f1_3a_35_4f

(this is a partial screen)
```

14.7.12. ノードのリセットのトリガー

nodedev-reset *nodedev* コマンドは、指定された *nodedev* のデバイスリセットをトリガーします。このコマンドを実行すると、ゲスト仮想マシンパススルーとホスト物理マシンの間でノードデバイスを転送する前に役立ちます。libvirt 必要に応じてこのアクションを暗黙的に実行しますが、このコマンドを使用すると、必要に応じて明示的にリセットできます。

14.8. ゲスト仮想マシンの起動、一時停止、再開、保存、および復元

このセクションでは、ゲスト仮想マシンの起動、一時停止、再開、保存、および復元に関する情報を提供します。

14.8.1. 定義済みドメインの開始

virsh start *domain* --console --paused --autodestroy --bypass-cache --force-boot --pass-fds コマンドは、定義済みで非アクティブのドメインを起動します。この仮想マシンのステータスは、最後の管理保存状態または起動直後から非アクティブになっています。このコマンドは、次のオプションを選択できます。

- **--console** - コンソールに接続しているドメインを起動します
- **--console** - これがドライバーによってサポートされている場合、ドメインを起動してから一時停止状態にします
- **--autodestroy** - virsh セッションが閉じるか、libvirt への接続が閉じるか、それ以外の場合は終了すると、ゲスト仮想マシンは自動的に破棄されます
- **--bypass-cache** - ドメインが管理された保存状態にある場合に使用されます。これを使用すると、システムキャッシュを回避して、ゲスト仮想マシンが復元されます。これにより、復元プロセスが遅くなることに注意してください。
- **--force-boot** - managedsave オプションをすべて破棄し、新規ブートが実行されます。
- **--pass-fds** - は、ゲスト仮想マシンに渡される、コンマで区切られた追加オプションのリストです。

14.8.2. ゲスト仮想マシンの一時停止

virsh を使用してゲスト仮想マシンを一時停止します。

```
# virsh suspend {domain-id, domain-name or domain-uuid}
```

ゲスト仮想マシンが一時停止状態の場合、システム RAM は消費されますが、プロセッサーリソースは消費されません。ゲスト仮想マシンが一時停止されている間は、ディスクとネットワークの I/O は発生しません。この操作はすぐに実行され、ゲスト仮想マシンは **resume** で再起動できます (「ゲスト仮想マシンの再開」) オプション。

14.8.3. 実行中のドメインの一時停止

virsh dompmsuspend *domain* --duration --target コマンドは、実行中のドメインを取得して一時停止し、3 つの可能な状態 (S3、S4、または 2 つのハイブリッド) のいずれかに配置できるようにします。

```
# virsh dompmsuspend rhel6 --duration 100 --target mem
```

このコマンドは、次のオプションを使用できます。

- **--duration** - 状態変更の期間を秒単位で設定します
- **--target** - **mem (suspend to RAM (S3))disk (suspend to disk (S4))** または **hybrid (hybrid suspend)** のいずれかを使用できます。

14.8.4. pmsuspend 状態からドメインをウェイクアップする

このコマンドは、設定された期間の期限が切れるのを待つのではなく、pmsuspend 状態にあるゲストにウェイクアップアラートを挿入します。ドメインが実行されている場合、この操作は失敗しません。

```
# dompmwakeup rhel6
```

このコマンドには、ドメインの名前 (たとえば、次に示すように *rhel6*) が必要です。

14.8.5. ドメインの定義を解除する

```
# virsh undefine domain --managed-save --snapshots-metadata --storage --remove-all-storage --
wipe-storage
```

このコマンドは、ドメインの定義を解除します。実行中のドメインで動作することはできますが、実行中のドメインを停止せずに一時的なドメインに変換します。ドメインが非アクティブの場合、ドメイン設定は削除されます。

このコマンドは、次のオプションを選択できます。

- **--managed-save** - このオプションは、管理対象の保存イメージもクリーンアップされることを保証します。このオプションを使用しない場合、管理対象の保存イメージを使用してドメインの定義を解除しようとすると失敗します。
- **--snapshots-metadata** - このオプションは、非アクティブなドメインの定義を解除するときに、スナップショット (**snapshot-list** で示される) もクリーンアップされることを保証します。設定ファイルにスナップショットメタデータが含まれている非アクティブなドメインの定義を解除しようとすると失敗することに注意してください。このオプションが使用され、ドメインがアクティブである場合、それは無視されます。
- **--storage** - このオプションを使用する場合は、ボリュームターゲット名またはストレージボリュームのソースパスをコンマで区切って指定し、未定義のドメインと一緒に削除する必要があります。このアクションでは、ストレージボリュームを削除する前にそのストレージボリュームの定義が解除されます。これは、非アクティブなドメインでのみ実行できることに注意してください。これは、libvirt で管理されるストレージボリュームでのみ機能することに注意してください。
- **--remove-all-storage** - ドメインの定義を解除することに加えて、関連するすべてのストレージボリュームが削除されます。
- **--wipe-storage** - ストレージボリュームを削除する以外にも、コンテンツをワイプします。

14.8.6. ゲスト仮想マシンの再開

resume オプションを使用して、中断されたゲスト仮想マシンを **virsh** で復元します。

```
# virsh resume {domain-id, domain-name or domain-uuid}
```

この操作は即時であり、ゲスト仮想マシンのパラメーターは **suspend** および **resume** 操作のために保持されます。

14.8.7. ゲスト仮想マシンを保存する

virsh コマンドを使用して、ゲスト仮想マシンの現在の状態をファイルに保存します。

```
# virsh save {domain-name|domain-id|domain-uuid} state-file --bypass-cache --xml --running --
paused --verbose
```

これにより、指定したゲスト仮想マシンが停止し、データがファイルに保存されます。ゲスト仮想マシンで使用されているメモリーの量を考えると、時間がかかる場合があります。復元を使用して、ゲスト仮想マシンの状態を **restore** できます (「ゲスト仮想マシンを復元する」) オプション。保存は一時停止に似ていますが、ゲスト仮想マシンを一時停止するだけでなく、ゲスト仮想マシンの現在の状態が保存されます。

virsh save コマンドは、次のオプションを使用できます。

- **--bypass-cache** - 復元を実行してファイルシステムキャッシュを回避しますが、このオプションを使用すると復元動作が遅くなる可能性があることに注意してください。
- **--xml** - このオプションは、XML ファイル名とともに使用する必要があります。通常、このオプションは省略されますが、ドメイン XML のホスト固有の部分のみを変更し、復元したゲスト仮想マシンで使用するための代替 XML ファイルを提供するために使用できます。たとえば、ゲストの保存後に取得されたディスクスナップショットによる、基になるストレージのファイル名の違いを説明するために使用できます。
- **--running** - は、保存イメージに記録された状態をオーバーライドして、ドメインを実行中として開始します。
- **--paused** - 保存イメージに記録された状態をオーバーライドして、ドメインを一時停止として開始します。
- **--verbose** - 保存の進捗を表示します。

ゲスト仮想マシンを XML ファイルから直接復元する場合は、**virsh restore** コマンドがそれを実行します。**domjobinfo** でプロセスをモニターし、**domjobabort** でこれをキャンセルできます。

14.8.8. ゲストの復元に使用されるドメイン XML ファイルの更新

virsh save-image-define *file xml* --running|--paused コマンドは、指定されたファイルが後で **virsh restore** コマンドで使用されるときに使用されるドメイン XML ファイルを更新します。*xml* 引数は、ドメイン XML のホスト物理マシン固有の部分のみが変更された代替 XML を含む XML ファイル名である必要があります。たとえば、ゲストを保存した後に、基となるストレージのディスクスナップショットを作成することで生じるファイルの命名の相違点を説明するために使用できます。ドメインを実行状態または一時停止状態に復元する必要があるかどうかを、イメージの保存で記録します。オプション **--running** または **--paused** を使用すると、使用する状態が決まります。

14.8.9. ドメイン XML ファイルの抽出

save-image-dumpxml *file* --security-info コマンドは、保存された状態ファイル (**virsh save** コマンドで使用) が参照されたときに有効だったドメイン XML ファイルを抽出します。**--security-info** オプションを使用すると、ファイルにセキュリティー上の機密情報が含まれます。

14.8.10. ドメイン XML 設定ファイルの編集

save-image-edit *file* --running --paused コマンドは、**virsh save** コマンドによって作成された保存済み *ファイル*に関連付けられている XML 設定ファイルを編集します。

保存イメージには、ドメインを **--running** 状態または **--paused** 状態のどちらに復元する必要があるかが記録されることに注意してください。これらのオプションを使用しない場合、状態はファイル自体によって決定されます。**--running** または **--paused** を選択すると、**virsh restore** が使用する必要のある状態を上書きできます。

14.8.11. ゲスト仮想マシンを復元する

virsh save コマンドで以前に保存したゲスト仮想マシンを復元します (「ゲスト仮想マシンを保存する」) **virsh**を使用する:

```
# virsh restore state-file
```

これにより、保存されたゲスト仮想マシンが再起動します。これには時間がかかる場合があります。ゲスト仮想マシンの名前と UUID は保持されますが、新しい ID に割り当てられます。

virsh restore *state-file* コマンドは、次のオプションを使用できます。

- **--bypass-cache** - 復元を実行してファイルシステムキャッシュを回避しますが、このオプションを使用すると復元動作が遅くなる可能性があることに注意してください。
- **--xml** - このオプションは、XML ファイル名とともに使用する必要があります。通常、このオプションは省略されますが、ドメイン XML のホスト固有の部分のみを変更し、復元したゲスト仮想マシンで使用するための代替 XML ファイルを提供するために使用できます。たとえば、ゲストの保存後に取得されたディスクスナップショットによる、基になるストレージのファイル名の違いを説明するために使用できます。
- **--running** - は、保存イメージに記録された状態をオーバーライドして、ドメインを実行中として開始します。
- **--paused** - 保存イメージに記録された状態をオーバーライドして、ドメインを一時停止として開始します。

14.9. ゲスト仮想マシンのシャットダウン、再起動、および強制シャットダウン

このセクションでは、ゲスト仮想マシンのシャットダウン、リブート、および強制シャットダウンに関する情報を提供します。

14.9.1. ゲスト仮想マシンのシャットダウン

virsh shutdown コマンドを使用して、ゲスト仮想マシンをシャットダウンします。

```
# virsh shutdown {domain-id, domain-name or domain-uuid} [--mode method]
```

ゲスト仮想マシンの設定ファイルの **on_shutdown** パラメーターを変更することで、ゲスト仮想マシンの再起動の動作を制御できます。

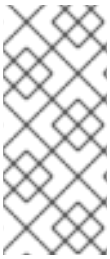
14.9.2. Red Hat Enterprise Linux 7 ホストでの Red Hat Enterprise Linux 6 ゲストのシャットダウン

Minimal installation を使用して Red Hat Enterprise Linux 6 ゲスト仮想マシンをインストールして

も、acpid パッケージ はインストールされません。Red Hat Enterprise Linux 7 は、 **systemd** に引き継がれたため、このパッケージを必要としなくなりました。ただし、Red Hat Enterprise Linux 7 ホストで実行されている Red Hat Enterprise Linux 6 ゲスト仮想マシンには引き続き必要です。

acpid パッケージがないと、**virsh shutdown** コマンドの実行時に Red Hat Enterprise Linux 6 ゲスト仮想マシンはシャットダウンしません。**virsh shutdown** は、ゲスト仮想マシンを適切にシャットダウンするように設計されています。

virsh shutdown を使用すると、システム管理がより簡単かつ安全になります。**virsh shutdown** コマンドを使用して正常にシャットダウンしない場合は、システム管理者がゲスト仮想マシンに手動でログインするか、**Ctrl**-**Alt**-**Del** のキーの組み合わせを各ゲスト仮想マシンに送信する必要があります。



注記

その他の仮想化オペレーティングシステムは、この問題の影響を受ける可能性があります。**virsh shutdown** コマンドでは、ACPI のシャットダウン要求を処理するようにゲスト仮想マシンのオペレーティングシステムを設定する必要があります。多くのオペレーティングシステムでは、ACPI のシャットダウン要求を受け入れるために、ゲスト仮想マシンのオペレーティングシステムで追加の設定が必要です。

手順14.4 Red Hat Enterprise Linux 6 ゲストの回避策

1. **acpid パッケージのインストール**
   acpid サービスは、ACPI 要求をリッスンして処理します。

   ゲスト仮想マシンにログインし、ゲスト仮想マシンに acpid パッケージをインストールします。

   ```
   # yum install acpid
   ```

2. **acpid サービスを有効にする**
   ゲスト仮想マシンの起動シーケンス中に**acpid**サービスを開始するように設定し、サービスを開始します。

   ```
   # chkconfig acpid on
   # service acpid start
   ```

3. **ゲストドメイン xml の準備**
   ドメインの XML ファイルを編集して、次の要素を追加します。virtio シリアルポートを **org.qemu.guest_agent.0** に置き換え、*$guestname* の代わりにゲストの名前を使用します。

   図14.2 ゲスト XML の置き換え

   ```
   <channel type='unix'>
     <source mode='bind' path='/var/lib/libvirt/qemu/{$guestname}.agent'/>
     <target type='virtio' name='org.qemu.guest_agent.0'/>
   </channel>
   ```

4. **QEMU ゲストエージェントのインストール**

QEMU ゲストエージェント (QEMU-GA) をインストールし、指示に従ってサービスを開始します。10章*qemu-img および QEMU ゲストエージェント*。Windows ゲストを実行している場合は、本章にもそのための手順があります。

5. **ゲストをシャットダウンします**

   a. 以下のコマンドを実行します。

```
# virsh list --all  - this command lists all of the known domains
  Id Name             State
----------------------------------
  rhel6              running
```

   b. ゲスト仮想マシンのシャットダウン

```
# virsh shutdown rhel6

Domain rhel6 is being shutdown
```

   c. ゲスト仮想マシンがシャットダウンするまで数秒待ちます。

```
# virsh list --all
 Id Name               State
----------------------------------
  . rhel6              shut off
```

   d. 編集した XML ファイルを使用して、*rhel6* という名前のドメインを開始します。

```
# virsh start rhel6
```

   e. *rhel6* ゲスト仮想マシンの acpi をシャットダウンします。

```
# virsh shutdown --mode acpi rhel6
```

   f. すべてのドメインを再度一覧表示し、*rhel6* が一覧に含まれていることを確認し、シャットダウンしていることを示します。

```
# virsh list --all
  Id Name               State
----------------------------------
  rhel6              shut off
```

   g. 編集した XML ファイルを使用して、*rhel6* という名前のドメインを開始します。

```
# virsh start rhel6
```

   h. *rhel6* ゲスト仮想マシンゲストエージェントをシャットダウンします。

```
# virsh shutdown --mode agent rhel6
```

   i. ドメインを一覧表示します。*rhel6* はまだリストに含まれており、シャットオフされていることを示しているはずです。

```
# virsh list --all
 Id Name                 State
----------------------------------
 rhel6                shut off
```

ゲスト仮想マシンは、上記の回避策を使用せずに、連続したシャットダウンの **virsh shutdown** コマンドを使用してシャットダウンします。

上記の方法の他に、**libvirt-guests** サービスを停止することで、ゲストを自動的にシャットダウンできます。この方法に関する詳細は、「libvirt-guests 設定設定の操作」 を参照してください。

14.9.3. libvirt-guests 設定設定の操作

libvirt-guests サービスには、ゲストが適切にシャットダウンするように設定できるパラメーターがあります。これは、libvirt インストールの一部で、デフォルトでインストールされるパッケージです。このサービスは、ホストのシャットダウン時にゲストをディスクに自動的に保存し、ホストの再起動時にゲストをシャットダウン前の状態に復元します。デフォルトでは、この設定はゲストを一時停止するように設定されています。ゲストをシャットオフする場合は、**libvirt-guests** 設定ファイルのパラメーターの 1 つを変更する必要があります。

手順14.5 ゲストの正常なシャットダウンを可能にするための libvirt-guests サービスパラメーターの変更

ここで説明する手順では、ホストの物理マシンが停止している場合、電源が切れている場合、または再起動が必要な場合に、ゲスト仮想マシンを正常にシャットダウンできるようにします。

1. **設定ファイルを開きます。**
   設定ファイルは **/etc/sysconfig/libvirt-guests** にあります。ファイルを編集し、コメントマーク (#) を削除し、 ***ON_SHUTDOWN=suspend*** を ***ON_SHUTDOWN=shutdown*** に変更します。変更を保存することを忘れないでください。

```
$ vi /etc/sysconfig/libvirt-guests

# URIs to check for running guests
# example: URIS='default xen:/// vbox+tcp://host/system lxc:///'
#URIS=default

# action taken on host boot
# - start   all guests which were running on shutdown are started on boot
#           regardless on their autostart settings
# - ignore  libvirt-guests init script won't start any guest on boot, however,
#           guests marked as autostart will still be automatically started by
#           libvirtd
#ON_BOOT=start

# Number of seconds to wait between each guest start. Set to 0 to allow
# parallel startup.
#START_DELAY=0

# action taken on host shutdown
# - suspend   all running guests are suspended using virsh managedsave
# - shutdown  all running guests are asked to shutdown. Please be careful with
#             this settings since there is no way to distinguish between a
#             guest which is stuck or ignores shutdown requests and a guest
#             which just needs a long time to shutdown. When setting
```

```
#           ON_SHUTDOWN=shutdown, you must also set SHUTDOWN_TIMEOUT to a
#           value suitable for your guests.
ON_SHUTDOWN=shutdown

# If set to non-zero, shutdown will suspend guests concurrently. Number of
# guests on shutdown at any time will not exceed number set in this variable.
#PARALLEL_SHUTDOWN=0

# Number of seconds we're willing to wait for a guest to shut down. If parallel
# shutdown is enabled, this timeout applies as a timeout for shutting down all
# guests on a single URI defined in the variable URIS. If this is 0, then there
# is no time out (use with caution, as guests might not respond to a shutdown
# request). The default value is 300 seconds (5 minutes).
#SHUTDOWN_TIMEOUT=300

# If non-zero, try to bypass the file system cache when saving and
# restoring guests, even though this may give slower operation for
# some file systems.
#BYPASS_CACHE=0
```

???

URIS - checks the specified connections for a running guest. The ***Default*** setting functions in the same manner as **virsh** does when no explicit URI is set In addition, one can explicitly set the URI from **/etc/libvirt/libvirt.conf**. It should be noted that when using the libvirt configuration file default setting, no probing will be used.

???

ON_BOOT - specifies the action to be done to / on the guests when the host boots. The ***start*** option starts all guests that were running prior to shutdown regardless on their autostart settings. The ***ignore*** option will not start the formally running guest on boot, however, any guest marked as autostart will still be automatically started by libvirtd.

???

The ***START_DELAY*** - sets a delay interval in between starting up the guests. This time period is set in seconds. Use the 0 time setting to make sure there is no delay and that all guests are started simultaneously.

???

ON_SHUTDOWN - specifies the action taken when a host shuts down. Options that can be set include: ***suspend*** which suspends all running guests using **virsh managedsave** and ***shutdown*** which shuts down all running guests. It is best to be careful with using the **shutdown** option as there is no way to distinguish between a guest which is stuck or ignores shutdown requests and a guest that just needs a longer time to shutdown. When setting the ***ON_SHUTDOWN=shutdown***, you must also set ***SHUTDOWN_TIMEOUT*** to a value suitable for the guests.

???

PARALLEL_SHUTDOWN Dictates that the number of guests on shutdown at any time will not exceed number set in this variable and the guests will be suspended concurrently. If set to **0**, then guests are not shutdown concurrently.

???

Number of seconds to wait for a guest to shut down. If ***SHUTDOWN_TIMEOUT*** is enabled, this timeout applies as a timeout for shutting down all guests on a single URI defined in the variable URIS. If ***SHUTDOWN_TIMEOUT*** is set to ***0***, then there is no time out (use with

caution, as guests might not respond to a shutdown request). The default value is 300 seconds (5 minutes).

???

BYPASS_CACHE can have 2 values, 0 to disable and 1 to enable. If enabled it will by-pass the file system cache when guests are restored. Note that setting this may effect performance and may cause slower operation for some file systems.

2. **libvirt-guests サービスを開始します。**
   サービスを起動していない場合は、libvirt-guests サービスを起動します。実行中のすべてのドメインがシャットダウンするため、サービスを再起動しないでください。

14.9.4. ゲスト仮想マシンの再起動

virsh restart コマンドを使用して、ゲスト仮想マシンを再起動します。再起動が実行されると、プロンプトが返されます。ゲスト仮想マシンが戻るまでに時間がかかる場合があることに注意してください。

```
#virsh reboot {domain-id, domain-name or domain-uuid} [--mode method]
```

ゲスト仮想マシンの設定ファイルの **<on_reboot>** 要素を変更することで、再起動中のゲスト仮想マシンの動作を制御できます。詳細は、「イベントの設定」 を参照してください。

デフォルトでは、ハイパーバイザーは適切なシャットダウン方法を選択しようとします。別の方法を指定するには、**--mode** オプションに、**initctl**、**acpi**、**agent**、**signal** などのコンマ区切りリストを指定します。ドライバーが各モードを試行する順序は、コマンドで指定された順序とは関係ありません。順序を厳格に制御するには、一度に 1 つのモードを使用し、コマンドを繰り返します。

14.9.5. ゲスト仮想マシンの強制停止

virsh destroy コマンドを使用して、ゲスト仮想マシンを強制的に停止します。

```
# virsh destroy {domain-id, domain-name or domain-uuid} [--graceful]
```

このコマンドは、即座に不正なシャットダウンを実行し、指定されたゲスト仮想マシンを停止します。**virsh destroy** を使用すると、ゲスト仮想マシンのファイルシステムが破損する可能性があります。ゲスト仮想マシンが応答しない場合にのみ、**destroy** オプションを使用してください。正常シャットダウンを開始する場合は、**virsh destroy --graceful** コマンドを使用します。

14.9.6. 仮想マシンのリセット

virsh reset *domain* は、ゲストをシャットダウンせずにドメインをすぐにリセットします。リセットは、マシンの電源リセットボタンをエミュレートします。すべてのゲストハードウェアが RST 行を認識し、内部状態を再初期化します。ゲストの仮想マシン OS がシャットダウンしないと、データが失われる可能性があることに注意してください。

14.10. ゲストの仮想マシン情報の取得

このセクションでは、ゲスト仮想マシン情報の取得に関する情報を提供します。

14.10.1. ゲスト仮想マシンのドメイン ID の取得

ゲスト仮想マシンのドメイン ID の取得

-

```
# virsh domid {domain-name or domain-uuid}
```

14.10.2. ゲスト仮想マシンのドメイン名の取得

ゲスト仮想マシンのドメイン名の取得

```
# virsh domname {domain-id or domain-uuid}
```

14.10.3. ゲスト仮想マシンの UUID の取得

ゲスト仮想マシンの Universally Unique Identifier (UUID) を取得するには、次のようにします。

```
# virsh domuuid {domain-id or domain-name}
```

virsh domuuid 出力の例:

```
# virsh domuuid r5b2-mySQL01
4a4c59a7-ee3f-c781-96e4-288f2862f011
```

14.10.4. ゲスト仮想マシン情報の表示

ゲスト仮想マシンのドメイン ID、ドメイン名、または UUID で **virsh**を使用すると、指定したゲスト仮想マシンに関する情報を表示できます。

```
# virsh dominfo {domain-id, domain-name or domain-uuid}
```

これは、**virsh dominfo** 出力の例です。

```
# virsh dominfo vr-rhel6u1-x86_64-kvm
Id:             9
Name:           vr-rhel6u1-x86_64-kvm
UUID:           a03093a1-5da6-a2a2-3baf-a845db2f10b9
OS Type:        hvm
State:          running
CPU(s):         1
CPU time:       21.6s
Max memory:     2097152 kB
Used memory:    1025000 kB
Persistent:     yes
Autostart:      disable
Security model: selinux
Security DOI:   0
Security label: system_u:system_r:svirt_t:s0:c612,c921 (permissive)
```

14.11. ストレージプールコマンド

次のコマンドは、ストレージプールを操作します。libvirt を使用すると、仮想マシン内のデバイスとして表示されるストレージボリュームを提供するために使用されるファイル、raw パーティション、ドメイン固有の形式など、さまざまなストレージソリューションを管理できます。この機能の詳細は、libvirt.org を参照してください。ストレージプールのコマンドの多くは、ドメインに使用されるコマンドと似ています。

14.11.1. ストレージプール XML の検索

find-storage-pool-sources *type srcSpec* コマンドは、検出された可能性のある特定の *タイプ*のすべてのストレージプールを説明する XML を表示します。*srcSpec* が提供されている場合、それはプールのクエリーをさらに制限するための XML を含むファイルです。

find-storage-pool-sources-as *type host port initiator* は、検出された可能性のある特定の *タイプ*のすべてのストレージプールを説明する XML を表示します。*ホスト*、*ポート*、または *イニシエーター*が提供されている場合、それらはクエリーが実行される場所を制御します。

pool-info *pool-or-uuid* コマンドは、指定されたストレージプールオブジェクトに関する基本情報を一覧表示します。このコマンドには、ストレージプールの名前または UUID が必要です。この情報を取得するには、次のコマンドを使用します。

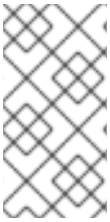
```
pool-list [--inactive] [--all] [--persistent] [--transient] [--autostart] [--no-autostart] [--details] type
```

これは、libvirt に認識されるストレージプールオブジェクトの一覧を表示します。デフォルトでは、アクティブなプールのみが一覧表示されますが、**--inactive** オプションを使用すると非アクティブプールのみが表示され、**--all** オプションを使用するとすべてのストレージプールが一覧表示されます。

これらのオプションに加えて、リストのコンテンツをフィルターリングするために使用できるフィルターリングオプションのセットがいくつかあります。**--persistent** は、一覧を永続プールに制限します。**--transient** はリストを一時的なプールに制限し、**--autostart** はリストをプールの自動起動に制限し、最後に **--no-autostart** は、自動起動が無効になっているストレージプールにリストを制限します。

type を必要とするすべてのストレージプールコマンドでは、プールタイプをコンマで区切る必要があります。有効なプールタイプには、**dir**、**fs**、**netfs**、**logical**、**disk**、**iscsi**、**scsi**、**mpath**、**rbd**、および **sheepdog**が含まれます。

--details オプションは、**virsh** に対して、プール永続性と利用可能な容量関連の情報を追加で表示するように指示します。



注記

このコマンドを古いサーバーで使用すると、固有の競合で一連の API 呼び出しの使用が強制されます。ここでは、リストの収集中にプールの状態が呼び出し間で変更した場合に、プールが一覧に表示されなかったり、複数回表示されたりすることがあります。ただし、新しいサーバーにはこの問題はありません。

pool-refresh *pool-or-uuid* は、プールに含まれるボリュームのリストを更新します。

14.11.2. ストレージプールの作成、定義、および起動

このセクションでは、ストレージプールの作成、定義、および開始に関する情報を提供します。

14.11.2.1. ストレージプールの構築

pool-build *pool-or-uuid* --overwrite --no-overwrite コマンドは、指定された *プール名または UUID*でプールを構築します。オプション **--overwrite** および **--no-overwrite**は、タイプがファイルシステムであるプールにのみ使用できます。どちらのオプションも指定されておらず、プールがファイルシステムタイプのプールである場合、結果のビルドはディレクトリーのみを作成します。

--no-overwrite を指定すると、プローブにより、ファイルシステムがターゲットデバイスにすでに存在するかどうかを判断したり、存在する場合はエラーを返すか、存在しない場合は **mkfs** を使用してターゲットデバイスをフォーマットします。**--overwrite** を指定した場合は、**mkfs** コマンドが実行され、

ターゲットデバイスに存在するデータがすべて上書きされます。

14.11.2.2. XML ファイルからのストレージプールの作成と定義

pool-create *file* は、関連付けられた XML ファイルからストレージプールを作成して開始します。

pool-define *file* は、XML ファイルからストレージプールオブジェクトを作成しますが、開始しません。

14.11.2.3. 生のパラメーターからストレージプールを作成して開始する

```
# pool-create-as name --print-xml type source-host source-path source-dev source-name <target> --source-format format
```

このコマンドは、指定された生のパラメーターからプールオブジェクト名を作成して開始します。

--print-xml が指定されている場合、プールを作成せずにストレージプールオブジェクトの XML を出力します。それ以外の場合、プールを構築するにはタイプが必要です。*type* を必要とするすべてのストレージプールコマンドでは、プールタイプをコンマで区切る必要があります。有効なプールタイプには、**dir**、**fs**、**netfs**、**logical**、**disk**、**iscsi**、**scsi**、**mpath**、**rbd**、および **sheepdog**が含まれます。

対照的に、次のコマンドは、指定された生のパラメーターからプールオブジェクト名を作成しますが、開始しません。

```
# pool-define-as name --print-xml type source-host source-path source-dev source-name <target> --source-format format
```

--print-xml が指定されている場合、プールを定義せずにプールオブジェクトの XML を出力します。それ以外の場合、プールは指定されたタイプである必要があります。*type* を必要とするすべてのストレージプールコマンドでは、プールタイプをコンマで区切る必要があります。有効なプールタイプには、**dir**、**fs**、**netfs**、**logical**、**disk**、**iscsi**、**scsi**、**mpath**、**rbd**、および **sheepdog**が含まれます。

pool-start *pool-or-uuid* は、以前に定義されたが非アクティブである、指定されたストレージプールを開始します。

14.11.2.4. ストレージプールの自動起動

pool-autostart *pool-or-uuid* --disable コマンドは、ストレージプールが起動時に自動的に開始することを有効または無効にします。このコマンドには、プール名または UUID が必要です。**pool-autostart** コマンドを無効にするには、**--disable** オプションを使用します。

14.11.3. ストレージプールの停止および削除

pool-destroy *pool-or-uuid* は、ストレージプールを停止します。停止すると、libvirt はプールを管理しなくなりますが、プールに含まれる生データは変更されず、後で **pool-create** コマンドを使用して復元できます。

pool-delete *pool-or-uuid* は、指定されたストレージプールによって使用されているリソースを破棄します。この操作は復元できず、元に戻せないことに注意してください。ただし、プール構造はこのコマンドの後に依然として存在し、新しいストレージボリュームの作成を受け入れる準備ができています。

pool-undefine *pool-or-uuid* コマンドは、非アクティブなプールの設定を定義解除します。

14.11.4. ストレージプール用の XML ダンプファイルの作成

pool-dumpxml --inactive *pool-or-uuid* コマンドは、指定されたストレージプールオブジェクトに関する XML 情報を返します。**--inactive** を使用すると、現在のプール設定ではなく、プールの次回の開始時に使用される設定がダンプされます。

14.11.5. ストレージプールの設定ファイルの編集

pool-edit *pool-or-uuid* は、指定したストレージプールの XML 設定ファイルを開いて編集します。

この方法は、適用する前にエラーをチェックするため、XML 設定ファイルの編集に使用すべき唯一の方法です。

14.11.6. ストレージプールの変換

pool-name *uuid* コマンドは、指定された UUID をプール名に変換します。

pool-uuid *pool* コマンドは、指定されたプールの UUID を返します。

14.12. ストレージボリュームコマンド

本セクションでは、ストレージボリュームの作成、削除、および管理を行うすべてのコマンドを説明します。ストレージプール名または UUID が必要になるため、ストレージプールを作成したら、これを行うのが最適です。ストレージプールの詳細は、12章*ストレージプール* を参照してください。ストレージボリュームに関する情報は、13章 *ボリューム* を参照してください。

14.12.1. ストレージボリュームの作成

vol-create-from *pool-or-uuid file* --inputpool *pool-or-uuid vol-name-or-key-or-path* コマンドは、コンテンツのテンプレートとして別のストレージボリュームを使用して、ストレージボリュームを作成します。このコマンドは、ボリュームを作成するストレージプールの名前または UUID である *pool-or-uuid* を必要とします。

file 引数は、ボリューム定義を含む XML ファイルとパスを指定します。**--inputpool *pool-or-uuid*** オプションは、ソースボリュームが含まれるストレージプールの名前または uuid を指定します。*vol-name-or-key-or-path* 引数は、ソースボリュームの名前またはキーまたはパスを指定します。一部の例については、「ボリュームの作成」 を参照してください。

vol-create-as コマンドは、一連の引数からボリュームを作成します。*pool-or-uuid* 引数は、ボリュームを作成するストレージプールの名前または UUID を含みます。

```
vol-create-as pool-or-uuid name capacity --allocation <size> --format <string> --backing-vol <vol-name-or-key-or-path> --backing-vol-format <string>
```

name は、新しいボリュームの名前です。*capacity* は、作成されるボリュームのサイズであり、スケーリングされた整数であり、接尾辞がない場合はデフォルトでバイトになります。**--allocation <size>** は、ボリュームに割り当てられる初期サイズであり、デフォルトでバイトにスケーリングされた整数としても使用されます。**--format <string>** は、ファイルベースのストレージプールで使用され、コンマで区切られた許容可能な形式の文字列であるボリュームファイル形式を指定します。使用可能な形式には、**raw**、**bochs**、**qcow**、**qcow2**、**vmdk**、 **--backing-vol *vol-name-or-key-or-path*** は、既存のボリュームのスナップショットを作成する場合に使用されるソースバッキングボリュームです。**--backing-vol-format *string*** は、スナップショットバッキングボリュームのフォーマットであり、コンマで区切られたフォーマットのストリングです。許可される値は、**raw**、**bochs**、**qcow**、**qcow2**、、**vmdk**、および **host_device** です。ただし、これはファイルベースのストレージプールのみを対象としています。

14.12.1.1. XML ファイルからのストレージボリュームの作成

vol-create *pool-or-uuid file* は、保存された XML ファイルからストレージボリュームを作成します。このコマンドには、ボリュームが作成されるストレージプールの名前または UUID である *pool-or-uuid* も必要です。*file* 引数には、ボリューム定義の XML ファイルを含むパスが含まれています。XML ファイルを作成する簡単な方法は、**vol-dumpxml** コマンドを使用して既存のボリュームの定義を取得し、それを変更してから保存してから、**vol-create** を実行することです。

```
virsh vol-dumpxml --pool storagepool1 appvolume1 > newvolume.xml
virsh edit newvolume.xml
virsh vol-create differentstoragepool newvolume.xml
```

使用可能なその他のオプションは以下のとおりです。

- **--inactive** オプションは、非アクティブなゲスト仮想マシン (つまり、定義されているが現在アクティブではないゲスト仮想マシン) を一覧表示します。
- **--all** オプションは、すべてのゲスト仮想マシンを一覧表示します。

14.12.1.2. ストレージボリュームのクローンの作成

vol-clone --pool *pool-or-uuid vol-name-or-key-or-path name* コマンドは、既存のストレージボリュームのクローンを作成します。**vol-create-from** も使用できますが、ストレージボリュームのクローンを作成する方法としてはお勧めしません。**--pool *pool-or-uuid*** オプションは、ボリュームを作成するストレージプールの名前または UUID です。*vol-name-or-key-or-path* 引数は、ソースボリュームの名前またはキーまたはパスです。*name* 引数を使用すると、新しいボリュームの名前を参照します。

14.12.2. ストレージボリュームの削除

vol-delete --pool *pool-or-uuid vol-name-or-key-or-path* コマンドは、指定されたボリュームを削除します。このコマンドには、特定の **--pool *pool-or-uuid*** が必要です。これは、ボリュームが存在するストレージプールの名前または UUID です。***vol-name-or-key-or-path*** オプションは、削除するボリュームの名前またはキーまたはパスを指定します。

vol-wipe --pool *pool-or-uuid* --algorithm *algorithm vol-name-or-key-or-path* コマンドは、ボリュームをワイプして、以前にボリュームにあったデータが将来の読み取りにアクセスできないようにします。このコマンドには、**--pool *pool-or-uuid*** が必要です。これは、ボリュームが存在するストレージプールの名前または UUID です。*vol-name-or-key-or-path* には、ワイプするボリュームの名前またはキーまたはパスが含まれています。デフォルト (ストレージボリュームのすべてのセクターが値 0 で書き込まれる) の代わりに、異なるワイピングアルゴリズムを選択できることに注意してください。ワイピングアルゴリズムを指定するには、サポートされている次の *アルゴリズム* タイプのいずれかで **--algorithm** オプションを使用します。

- **zero** - 1-pass all zeroes
- **nnsa** - 4-pass NNSA Policy Letter NAP-14.1-C (XVI-8) for sanitizing removable and non-removable hard disks: random x2, 0x00, verify.
- **dod** - 4-pass DoD 5220.22-M section 8-306 procedure for sanitizing removable and non-removable rigid disks: random, 0x00, 0xff, verify.
- **bsi** - ドイツ情報技術セキュリティーセンター (http://www.bsi.bund.de) が推奨する 9-pass メソッド: 0xff、0xfe、0xfd、0xfb、0xf7、0xef、0xdf、0xbf、0x7f。
- **gutmann** - The canonical 35-pass sequence described in Gutmann's paper.

- **schneier** - "Applied Cryptography"(1996) の Bruce Schneier で記述される 7-pass メソッド: 0x00、0xff、random x5
- **pfitzner7** - Roy Pfitzner's 7-random-pass method: random x7
- **pfitzner33** - Roy Pfitzner's 33-random-pass method: random x33.
- **random** - 1-pass pattern: random

注記

ホストにインストールされているスクラブバイナリーのバージョンによって、使用可能なアルゴリズムが制限されます。

14.12.3. ストレージボリューム情報の XML ファイルへのダンプ

vol-dumpxml --pool *pool-or-uuid vol-name-or-key-or-path* コマンドは、ボリューム情報を指定されたファイルへの XML ダンプとして取得します。

このコマンドには、**--pool *pool-or-uuid*** が必要です。これは、ボリュームが存在するストレージプールの名前または UUID です。*vol-name-or-key-or-path* は、結果の XML ファイルを配置するボリュームの名前またはキーまたはパスです。

14.12.4. ボリューム情報の一覧表示

vol-info --pool *pool-or-uuid vol-name-or-key-or-path* コマンドは、指定されたストレージボリューム **--pool** に関する基本情報を一覧表示します。ここで、*pool-or-uuid* はボリュームが存在するストレージプールの名前または UUID です。*vol-name-or-key-or-path* は、情報を返すボリュームの名前またはキーまたはパスです。

vol-list --pool *pool-or-uuid* --details は、指定されたストレージプール内のすべてのボリュームを一覧表示します。このコマンドには、ストレージプールの名前または UUID である **--pool *pool-or-uuid*** が必要です。**--details** オプションは、virsh に、ボリュームタイプおよびキャパシティー関連の情報 (利用可能な場合) を追加で表示するように指示します。

14.12.5. ストレージボリュームの情報の取得

vol-pool --uuid *vol-key-or-path* コマンドは、指定されたボリュームのプール名または UUID を返します。デフォルトでは、プール名が返されます。**--uuid** オプションを指定すると、代わりにプール UUID が返されます。このコマンドには、要求された情報を返すボリュームのキーまたはパスである *vol-key-or-path* が必要です。

vol-path --pool *pool-or-uuid vol-name-or-key* コマンドは、指定したボリュームのパスを返します。このコマンドには、**--pool *pool-or-uuid*** が必要です。これは、ボリュームが存在するストレージプールの名前または UUID です。また、パスが要求されているボリュームの名前または鍵である *vol-name-or-key* も必要です。

vol-name *vol-key-or-path* コマンドは、指定したボリュームの名前を返します。*vol-key-or-path* は、名前を返すボリュームのキーまたはパスです。

vol-key --pool *pool-or-uuid vol-name-or-path* コマンドは、指定したボリュームのボリュームキーを返します。 **--pool *pool-or-uuid*** はストレージプールの名前または UUID で、 *vol-name-or-path* はボリュームキーを返すボリュームの名前またはパスになります。

14.12.6. ストレージボリュームのアップロードとダウンロード

このセクションでは、ストレージボリュームとの間で情報をアップロードおよびダウンロードする方法について説明します。

14.12.6.1. ストレージボリュームへのコンテンツのアップロード

vol-upload --pool *pool-or-uuid* --offset bytes --length bytes *vol-name-or-key-or-path local-file* コマンドは、指定された *ローカルファイル* の内容をストレージボリュームにアップロードします。このコマンドには、ボリュームが存在するストレージプールの名前または UUID である **--pool *pool-or-uuid*** が必要です。また、ワイプするボリュームの名前またはキーまたはパスである *vol-name-or-key-or-path* も必要です。**--offset** オプションは、データの書き込みを開始するストレージボリューム内の位置です。**--length *length*** は、アップロードされるデータ量の上限を示します。*local-file* が指定した **--length** よりも大きい場合は、エラーが発生します。

14.12.6.2. ストレージボリュームからコンテンツをダウンロードする

```
# vol-download --pool pool-or-uuid --offset bytes --length bytes vol-name-or-key-or-path local-file
```

このコマンドは、ローカルファイルの内容をストレージボリュームからダウンロードします。これは、ボリュームのあるストレージプールの名前または UUID である **-pool *pool-or-uuid* を**必要とします。また、ワイプするボリュームの名前またはキーまたはパスである *vol-name-or-key-or-path* も必要です。オプション **--offset** を使用すると、データの読み取りを開始するストレージボリューム内の位置が決まります。**--length *length*** は、ダウンロードされるデータ量の上限を示します。

14.12.7. ストレージボリュームのサイズ変更

```
# vol-resize --pool pool-or-uuid vol-name-or-path pool-or-uuid capacity --allocate --delta --shrink
```

このコマンドコマンドは、指定されたボリュームの容量をバイト単位でサイズ変更します。このコマンドには、ボリュームが存在するストレージプールの名前または UUID である **--pool *pool-or-uuid*** が必要です。このコマンドでは、*vol-name-or-key-or-path* が、サイズを変更するボリュームの名前またはキーまたはパスであることも必要です。

--allocate オプションが指定されていない限り、新しい容量によって *sparse file* が作成される可能性があります。通常、容量は新しいサイズになりますが、**--delta** が存在する場合は、既存のサイズに追加されます。**--shrink** オプションが存在しない限り、ボリュームを縮小しようとすると失敗します。

--shrink オプションが指定されていない限り、容量を負にすることはできず、負の符号は必要ありません。*capacity* はスケーリングされた整数であり、接尾辞がない場合はデフォルトでバイトになります。このコマンドは、アクティブなゲストが使用していないストレージボリュームに対してのみ安全であることに注意してください。ライブのサイズ変更については、「blockresize を使用してドメインパスのサイズを変更する」 を参照してください。

14.13. ゲストごとの仮想マシン情報の表示

このセクションでは、各ゲストの仮想マシン情報の表示に関する情報を提供します。

14.13.1. ゲスト仮想マシンの表示

ゲスト仮想マシンのリストとその現在の状態を **virsh** で表示するには:

```
# virsh list
```

使用可能なその他のオプションは以下のとおりです。

- **--inactive** オプションは、非アクティブなゲスト仮想マシン (つまり、定義されているが現在アクティブではないゲスト仮想マシン) を一覧表示します。

- **--all** オプションは、すべてのゲスト仮想マシンを一覧表示します。以下に例を示します。

```
# virsh list --all
 Id Name                 State
----------------------------------
  0 Domain-0             running
  1 Domain202            paused
  2 Domain010            inactive
  3 Domain9600           crashed
```

このコマンドを使用して表示できる 7 つの状態があります。

  - Running - **running** 状態は、CPU 上で現在アクティブなゲスト仮想マシンを指します。

  - Idle - **idle** 状態は、ドメインがアイドル状態であり、実行されていないか、実行できない可能性があることを示します。これは、ドメインが IO (従来の待機状態) を待機しているか、他に何もすることがなかったためにスリープ状態になっていることが原因である可能性があります。

  - Paused - **paused** 状態は、一時停止されているドメインを一覧表示します。これは、管理者が **virt-manager** または **virsh suspend** で **paused** ボタンを使用すると発生します。ゲスト仮想マシンが一時停止すると、メモリーとその他のリソースを消費しますが、ハイパーバイザーからのスケジューリングと CPU リソースには対応していません。

  - Shutdown - **shutdown** 状態は、シャットダウン中のゲスト仮想マシン用です。ゲスト仮想マシンにシャットダウンシグナルが送信され、操作を正常に停止するプロセスにあるはずです。これは、すべてのゲスト仮想マシンのオペレーティングシステムでは機能しない可能性があり、一部のオペレーティングシステムはこれらのシグナルに応答しない場合があります。

  - Shut off - **shut off** 状態は、ドメインが実行されていないことを示します。これは、ドメインが完全にシャットダウンした場合、またはドメインが開始されていない場合に発生する可能性があります。

  - Crashed - **crashed** 状態は、ドメインがクラッシュしたことを示し、ゲスト仮想マシンがクラッシュ時に再起動しないように設定されている場合にのみ発生する可能性があります。

  - Dying - **dying**状態のドメインは停止中です。これは、ドメインが完全にシャットダウンまたはクラッシュしていない状態です。

- **--managed-save** このオプションだけではドメインをフィルターリングしませんが、管理された保存状態が有効になっているドメインが一覧表示されます。ドメインを実際に個別にリストするには、**--inactive** オプションも使用する必要があります。

- **--name** は指定されていますドメイン名はリストに出力されます。**--uuid** が指定されている場合、代わりにドメインの UUID が出力されます。オプション **--table** を使用すると、テーブルスタイルの出力を使用する必要があることを指定します。3 つのコマンドはすべて相互に排他的です

- **--title** このコマンドは **--table** 出力とともに使用する必要があります。**--title** を指定すると、短いドメインの説明 (title) を含む追加の列がテーブルに作成されます。

- **--persistent** には、リストに永続ドメインが含まれます。**--transient** オプションを使用します。

- **--with-managed-save** は、管理された保存で設定されたドメインをリストします。それを指定せずにコマンドを一覧表示するには、コマンド **--without-managed-save** を使用します
- **--state-running** は、実行中のドメインを除外し、**--state-paused** は一時停止されたドメインを除外し、**--state-shutoff** はオフになっているドメインを除外し、**--state-other** はすべての状態をフォールバックとして一覧表示します。
- **--autostart** このオプションを使用すると、自動開始ドメインが一覧表示されます。この機能が無効になっているドメインを一覧表示するには、オプション **--no-autostart** を使用します。
- **--with-snapshot** は、スナップショットイメージを一覧表示できるドメインを一覧表示します。スナップショットのないドメインをフィルターリングするには、オプション **--without-snapshot** を使用します

```
$ virsh list --title --name

 Id     Name                           State     Title
 0      Domain-0                         running   Mailserver1
 2      rhelvm                         paused
```

virsh vcpuinfo の出力例は、以下を参照してください。「仮想 CPU 情報の表示」

14.13.2. 仮想 CPU 情報の表示

virsh を使用してゲスト仮想マシンから仮想 CPU を表示するには、次のコマンドを実行します。

```
# virsh vcpuinfo {domain-id, domain-name or domain-uuid}
```

virsh vcpuinfo の出力例を以下に示します。

```
# virsh vcpuinfo rhel6
VCPU:          0
CPU:           2
State:         running
CPU time:      7152.4s
CPU Affinity:  yyyy

VCPU:          1
CPU:           2
State:         running
CPU time:      10889.1s
CPU Affinity:  yyyy
```

14.13.3. 仮想 CPU アフィニティーの設定

物理 CPU を使用して仮想 CPU のアフィニティーを設定するには、以下を参照します 例14.3「vCPU のホストの物理マシンの CPU へのピニング」。

例14.3 vCPU のホストの物理マシンの CPU へのピニング

virsh vcpupin は、仮想 CPU を物理 CPU に割り当てます。

```
# virsh vcpupin rhel6
VCPU: CPU Affinity
```

```
----------------------------------
  0: 0-3
  1: 0-3
```

vcpupin は、次のオプションを選択できます。

- **--vcpu** には vcpu 番号が必要です。
- **[--cpulist] >string<** は、設定するホスト物理マシンの CPU 番号を一覧表示するか、オプションのクエリーを省略します
- **--config** は次回のブートに影響します。
- **--live** は実行中のドメインに影響します
- **--current** は現在のドメインに影響します

14.13.4. ドメインの仮想 CPU 数に関する情報の表示

virsh vcpucount には、 *ドメイン*名またはドメイン ID が必要です。以下に例を示します。

```
# virsh vcpucount rhel6
maximum      config         2
maximum      live           2
current      config         2
current      live           2
```

vcpucount は、次のオプションを選択できます。

- **--maximum** は、使用可能な vCPU の最大数を表示します
- **--active** は、現在アクティブな vCPU の数を表示します
- **--live** は、実行中のドメインからの値を表示します
- **--config** は、ゲスト仮想マシンの次回の起動時に設定される値を表示します
- **--current** は、現在のドメイン状態に応じた値を表示します
- **--guest** は、返されるカウントをゲストの観点から表示します

14.13.5. 仮想 CPU アフィニティーの設定

物理 CPU を使用して仮想 CPU のアフィニティーを設定するには、以下を実行します。

```
# virsh vcpupin domain-id vcpu cpulist
```

domain-id パラメーターは、ゲスト仮想マシンの ID 番号または名前です。

vcpu パラメーターは、ゲスト仮想マシンに割り当てられている仮想 CPU の数を示します。 **vcpu** パラメーターを指定する必要があります。

cpulist パラメーターは、物理 CPU ID 番号の一覧で、コンマで区切ります。 **cpulist** パラメーターは、VCPU を実行できる物理 CPU を指定します。

--config などの追加パラメーターは次回の起動に影響しますが、**--live** は実行中のドメインに影響し、**--current** は現在のドメインに影響します。

14.13.6. 仮想 CPU 数の設定

ゲスト仮想マシンに割り当てられている CPU の数を変更するには、**virsh setvcpus** コマンドを使用します。

```
# virsh setvcpus {domain-name, domain-id or domain-uuid} count [[--config] [--live] | [--current] [--guest]
```

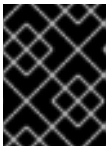
virsh setvcpus コマンドには、次のパラメーターを設定できます。

- ***{domain-name, domain-id or domain-uuid}*** - 仮想マシンを指定します。
- ***count*** - 設定する仮想 CPU の数を指定します。

注記

count 値は、作成時にゲスト仮想マシンに割り当てられた CPU の数を超えることはできません。また、ホストまたはハイパーバイザーによって制限される場合もあります。Xen の場合、ドメインが準仮想化されている場合にのみ、実行中のドメインの仮想 CPU を調整できます。

- ***--live*** - デフォルトのオプション。何も指定されていない場合に使用されます。設定の変更は、実行中のゲスト仮想マシンで有効になります。これは、vCPU の数が増加した場合は **ホットプラグ** と呼ばれ、減少した場合は **ホットアンプラグ** と呼ばれます。



重要

vCPU ホットアンプラグ機能はテクノロジープレビューです。したがって、これはサポートされておらず、高い値のデプロイメントでの使用は推奨されません。

- ***--config*** - 設定の変更は、ゲストの次回の再起動時に有効になります。ハイパーバイザーで対応している場合は、**--config** オプションと **--live** オプションの両方を指定できます。
- ***--current*** - 設定の変更は、ゲスト仮想マシンの現在の状態に影響します。実行中のゲストで使用する場合、次のように機能します ***--live***、シャットダウンされたゲストで使用される場合、次のように機能します ***--config***。
- ***--maximum*** - ゲストの次回の再起動時にホットプラグできる最大 vCPU 制限を設定します。そのため、**--config** オプションでのみ使用する必要があり、**--live** では使用しないでください。
- ***--guest*** - ホットプラグまたはホットアンプラグの代わりに、QEMU ゲストエージェントは、vCPU を有効または無効にすることにより、実行中のゲストの vCPU カウントを直接変更します。このオプションは、gueet の現在の仮想 CPU 数よりも多くの ***count*** の値とすることはできません。また、ゲストの再起動時に ***--guest*** でセットされる設定がリセットされます。

例14.4 vCPU のホットプラグとホットアンプラグ

vCPU をホットプラグするには、単一の vCPU を使用するゲストで次のコマンドを実行します。

```
virsh setvcpus guestVM1 2 --live
```

これにより、guestVM1 の vCPU の数が 2 つに増えます。この変更は、***--live*** オプションにあるように、guestVM1 の実行中に実行されます。

同じ実行中のゲストから 1 つの vCPU をホットプラグ解除するには、次のコマンドを実行します。

```
virsh setvcpus guestVM1 1 --live
```

ただし、現在、vCPU ホットアンプラグを使用すると、vCPU カウントをさらに変更すると問題が発生する可能性があることに注意してください。

14.13.7. メモリー割り当ての設定

virsh でゲスト仮想マシンのメモリー割り当てを変更するには、次のコマンドを実行します。

```
# virsh setmem {domain-id or domain-name} count
```

```
# virsh setmem vr-rhel6u1-x86_64-kvm --kilobytes 1025000
```

countはキロバイト単位で指定する必要があります。新しいカウント値は、ゲスト仮想マシンを作成したときに指定した量を超えることはできません。64MB 未満の値は、ほとんどのゲスト仮想マシンのオペレーティングシステムでは使用できない可能性があります。最大メモリー値を上げても、アクティブなゲスト仮想マシンには影響しません。新しい値が利用可能なメモリーよりも小さい場合は、縮小するため、ゲスト仮想マシンがクラッシュする可能性があります。

このコマンドには、以下のオプションがあります。

- [--domain] <string> - ドメイン名、ID、または uuid
- [--size] <number> スケーリングされた整数としての新しいメモリーサイズ (デフォルトは KiB)

  有効なメモリーユニットは以下のとおりです。

  - バイトの場合は ***b*** または ***bytes*** になります。
  - キロバイトの場合は、***KB*** になります (10 ^{3} または 1,000 バイトのブロック)
  - キビバイトの場合は ***k*** または ***KiB*** (2 ^{10} または 1024 バイトのブロック)
  - ***MB*** メガバイトの場合 (10 ^{6} または 1,000,000 バイトのブロック)
  - メビバイトの場合は ***M*** または ***MiB*** (2^{20} または 1,048,576 バイトのブロック)
  - ***GB*** (ギガバイト) (10^{9} または 1,000,000,000 バイトのブロック)
  - ギビバイトの場合は ***G*** または ***GiB*** (^{30} または 1,073,741,824 バイトのブロック)
  - エクサバイトの場合は ***TB*** (10^{12} または 1,000,000,000,000 バイト のブロック)
  - テビバイトの場合は ***T*** または ***TiB*** (2 ^{40} または 1,099,511,627,776 のブロック)

  すべての値は、libvirt により、最も近い kibibyte に丸められます。また、ハイパーバイザーが対応する粒度に丸めることもできます。一部のハイパーバイザーは、4000KiB(4000 x 2 ^{10} ま

たは 4,096,000 バイト) などの最小値も適用します。この値の単位は、オプションの属性 ***memory unit*** により決定されます。デフォルトは、測定単位としての KiB (キビバイト) になります。この値は、2^{10} または 1024 バイトのブロックで乗算されます。

- --config - 次回の起動に影響します
- --live は実行中のドメインのメモリーを制御します
- --current は、現在のドメインのメモリーを制御します

14.13.8. ドメインのメモリー割り当ての変更

virsh setmaxmem *domain size* --config --live --current を使用すると、以下のように、ゲスト仮想マシンの最大メモリー割り当て量を設定できます。

```
virsh setmaxmem rhel6 1024 --current
```

最大メモリーサイズに指定できるサイズは、拡張された整数です。拡張子に対応するものを除き、デフォルトではキビバイト単位で表されます。次のオプションは、このコマンドと使用できます。

- **--config** - 次回の起動に影響します
- **--live** - 実行中のドメインのメモリーを制御します。これにより、すべてのハイパーバイザーで、最大メモリー制限のライブ変更が許可されないため、ハイパーバイザーがこの動作に対応できるようになります。
- **--current** - 現在のドメインのメモリーを制御します

14.13.9. ゲスト仮想マシンブロックのデバイス情報の表示

virsh domblkstat を使用して、実行中のゲスト仮想マシンのブロックデバイス統計情報を表示します。

```
# virsh domblkstat GuestName block-device
```

14.13.10. ゲスト仮想マシンのネットワークデバイス情報の表示

virsh domifstat を使用して、実行中のゲスト仮想マシンのネットワークインターフェイス統計情報を表示します。

```
# virsh domifstat GuestName interface-device
```

14.14. 仮想ネットワークの管理

本セクションでは、**virsh** コマンドを使用した仮想ネットワークの管理を説明します。仮想ネットワークの一覧を表示するには、

```
# virsh net-list
```

このコマンドは、以下のような出力を生成します。

```
# virsh net-list
Name                 State      Autostart
```

```
----------------------------------------
default              active     yes
vnet1              active     yes
vnet2              active     yes
```

特定の仮想ネットワークのネットワーク情報を表示するには、次のコマンドを実行します。

```
# virsh net-dumpxml NetworkName
```

指定した仮想ネットワークに関する情報を XML 形式で表示します。

```
# virsh net-dumpxml vnet1
<network>
  <name>vnet1</name>
  <uuid>98361b46-1581-acb7-1643-85a412626e70</uuid>
  <forward dev='eth0'/>
  <bridge name='vnet0' stp='on' forwardDelay='0' />
  <ip address='192.168.100.1' netmask='255.255.255.0'>
    <dhcp>
      <range start='192.168.100.128' end='192.168.100.254' />
    </dhcp>
  </ip>
</network>
```

仮想ネットワークの管理で使用される**virsh** コマンドは、以下のとおりです。

- **virsh net-autostart network-name *network-name*** - *network-name* として指定したネットワークを自動的に起動します。
- **virsh net-create *XMLfile*** - 既存の XML ファイルを使用して新しいネットワークを生成して開始します。
- **virsh net-define *XMLfile*** - 既存の XML ファイルを起動せずに新しいネットワークデバイスを生成します。
- **virsh net-destroy *network-name*** - *network-name* として指定されたネットワークを破棄します。
- **virsh net-name *networkUUID*** - 指定された *networkUUID* をネットワーク名に変換します。
- **virsh net-uuid *network-name*** - 指定された *ネットワーク名* をネットワーク UUID に変換します。
- **virsh net-start *nameOfInactiveNetwork*** - 非アクティブなネットワークを開始します。
- **virsh net-undefine *nameOfInactiveNetwork*** - 非アクティブなネットワークの定義を削除します。

14.15. VIRSH を使用したゲスト仮想マシンの移行

virsh を使用した移行に関する情報は、Live KVM Migration with virsh というタイトルのセクションにあります。「virsh を使用した KVM のライブ移行」 を参照してください。

14.15.1. インターフェイスコマンド

以下のコマンドはホストインターフェイスを操作するため、ゲスト仮想マシンからは実行しないでください。これらのコマンドは、ホストの物理マシンにある端末から実行する必要があります。

警告

本セクションのコマンドは、マシンで NetworkManager サービスが無効になっており、代わりに **network** サービスを使用している場合に限りサポートされます。

多くの場合、このようなホストインターフェイスは、ドメインの **<interface>** 要素 (システムが作成したブリッジインターフェイスなど) 内で名前で使用できますが、ホストインターフェイスが特定のゲスト設定 XML と完全に関連付けられる必要はありません。ホストインターフェイスのコマンドの多くは、ドメインに使用されるコマンドと似ており、インターフェイスに名前を付ける方法は、名前または MAC アドレスのいずれかです。ただし、**iface** オプションに MAC アドレスを使用する場合、そのアドレスが一意である場合にのみ機能します (多くの場合、インターフェイスとブリッジが同じ MAC アドレスを共有しており、その場合、その MAC アドレスを使用すると曖昧さによりエラーが発生し、代わりに名前を使用する必要があります)。

14.15.1.1. XML ファイルを使用したホストの物理マシンインターフェイスの定義および起動

virsh iface-define *file* コマンドは、XML ファイルからホストインターフェイスを定義します。このコマンドはインターフェイスのみを定義し、起動はしません。

```
virsh iface-define iface.xml
```

定義済みのインターフェイスを起動する場合は、**iface-start *interface*** を実行します。*interface* は、インターフェイスの名前になります。

14.15.1.2. ホストインターフェイスの XML 設定ファイルの編集

コマンド **iface-edit *interface*** は、ホストインターフェイスの XML 設定ファイルを編集します。これは、XML 設定ファイルを変更する際に推奨される **唯一** の方法です。(これらのファイルについての詳細は、20章 *ドメイン XML の操作* を参照してください。)

14.15.1.3. アクティブなホストインターフェイスの一覧表示

iface-list --inactive --all は、アクティブなホストインターフェイスのリストを表示します。**--all** が指定されている場合、このリストには、定義されているが非アクティブなインターフェイスも含まれます。**--inactive** を指定すると、非アクティブのインターフェイスのみが一覧表示されます。

14.15.1.4. MAC アドレスのインターフェイス名への変換

iface-name *interface* コマンドは、MAC アドレスがホストのインターフェイス間で一意である場合に、ホストインターフェイスの MAC をインターフェイス名に変換します。このコマンドには、インターフェイスの MAC アドレスである *interface* が必要です。

iface-mac *interface* コマンドは、ホストのインターフェイス名を MAC アドレスに変換します。この場合、*interface* がインターフェイス名になります。

14.15.1.5. 特定のホスト物理マシンインターフェイスを停止する

virsh iface-destroy *interface* コマンドは、指定したホストインターフェイスを破壊 (停止) します。これは、ホストで **if-down** を実行するのと同じです。このコマンドは、そのインターフェイスのアクティブな使用を無効にし、すぐに有効にします。

インターフェイスの定義を解除する場合は、インターフェイス名を指定して **iface-undefine *interface*** コマンドを実行します。

14.15.1.6. ホスト設定ファイルの表示

virsh iface-dumpxml *interface* --inactive は、ホストインターフェイスの情報を、stdout への XML ダンプとして表示します。**--inactive** オプションを指定すると、出力には、次回起動したときに使用されるインターフェイスの永続的な状態が反映されます。

14.15.1.7. ブリッジデバイスの作成

iface-bridge は、bridge という名前のブリッジデバイスを作成し、既存のネットワークデバイス interface を新しいブリッジに割り当てます。この新しいブリッジはすぐに機能し、STP が有効になり、遅延は 0 になります。

```
# virsh iface-bridge interface bridge --no-stp delay --no-start
```

これらの設定は、-no-stp、-no-start、および遅延の整数秒数で変更できることに注意してください。インターフェイスのすべての IP アドレス設定は、新しいブリッジデバイスに移動します。ブリッジの破棄に関する情報は、「ブリッジデバイスの破損」 を参照してください。

14.15.1.8. ブリッジデバイスの破損

iface-unbridge *bridge* --no-start コマンドは、指定したブリッジデバイス *bridge* を削除し、その基盤となるインターフェイスを通常の使用に戻し、すべての IP アドレスの設定をブリッジデバイスから基盤となるデバイスに移動します。**--no-start** オプションを使用しない限り、基本となるインターフェイスが再起動しますが、通常は再起動しないことを念頭に置いてください。ブリッジの作成に使用するコマンドは、「ブリッジデバイスの作成」 を参照してください。

14.15.1.9. インターフェイススナップショットの操作

iface-begin コマンドは、現在のホストインターフェイス設定のスナップショットを作成します。このスナップショットは、後でコミット (**iface-commit** を使用) または復元 (**iface-rollback**) できます。スナップショットがすでに存在する場合は、以前のスナップショットがコミットまたは復元されるまで、このコマンドは失敗します。スナップショットの作成時と、その最終的なコミットまたはロールバックの間に、libvirt API 外でホストインターフェイスに外部変更が加えられると、定義されていない動作が発生します。

iface-commit を実行して、前回の **iface-begin** 以降に加えた変更をすべて動作として宣言し、ロールバックポイントを削除します。**iface-begin** を介してインターフェイススナップショットがまだ開始されていない場合、このコマンドは失敗します。

iface-rollback を使用して、ホストインターフェイスの設定を、**iface-begin** コマンドの最終実行時を記録した状態に戻します。**iface-begin** コマンドが以前に実行されていなかった場合、**iface-rollback** は失敗します。ホスト物理マシンの再起動は、暗黙のロールバックポイントとしても機能することに注意してください。

14.15.2. スナップショットの管理

次のセクションでは、ドメインスナップショットを操作するために実行できるアクションについて説明

します。*スナップショット* は、指定された時点でのドメインのディスク、メモリー、およびデバイスの状態を取得し、将来の使用のために保存します。スナップショットには、OS イメージのクリーンなコピーを保存することから、潜在的に破壊的な操作になる前にドメインの状態を保存することまで、多くの用途があります。スナップショットは、一意の名前で識別されます。スナップショットのプロパティーを表すのに使用される XML 形式のドキュメントは、libvirt の Web サイト を参照してください。

14.15.2.1. スナップショットの作成

virsh snapshot-create コマンドは、ドメイン XML ファイルで指定されたプロパティー (<name> 要素と <description> 要素、および <disks> など) を使用してドメインのスナップショットを作成します。

スナップショットを作成するには、以下のコマンドを実行します。

```
# snapshot-create <domain> <xmlfile> [--redefine] [--current] [--no-metadata] [--reuse-external]
```

ドメイン名、ID、または UID は、ドメイン要件として使用できます。XML 要件は、文字列に <name>、<description>、および <disks> 要素が含まれている必要があることです。

注記

ライブスナップショットは、Red Hat Enterprise Linux ではサポートされていません。ライブスナップショットで使用するために **virsh snapshot-create** コマンドで使用できる追加のオプションがあります。これは、libvirt には表示されますが、Red Hat Enterprise Linux 6 ではサポートされていません。

Red Hat Enterprise Linux で利用可能なオプションは次のとおりです。

- **--redefine** は、**snapshot-dumpxml** によって生成されたすべての XML 要素が有効であるかどうかを指定します。これは、スナップショット階層をあるマシンから別のマシンに移行したり、一時的なドメインがなくなって後で同じ名前と UUID で再作成された場合の階層を再作成したり、スナップショットメタデータにわずかな変更を加えたりするために使用できます (スナップショットに埋め込まれたドメイン XML のホスト固有の側面など)。このオプションが指定されている場合、**xmlfile** 引数は必須であり、**--current** オプションも指定されていない限り、ドメインの現在のスナップショットは変更されません。

- **--no-metadata** はスナップショットを作成しますが、メタデータは即座に破棄されます (つまり、libvirt は、スナップショットを現在のものとして扱いません。また、メタデータに関する libvirt を教えるために **--redefine** を使用しない限り、スナップショットに戻ることはできません)。

- **--reuse-external**を使用する場合、このオプションは、使用する既存の外部 XML スナップショットの場所を指定します。既存の外部スナップショットがまだ存在しない場合、既存のファイルの内容が失われないようにするために、コマンドはスナップショットの作成に失敗します。

14.15.2.2. 現在のドメインのスナップショットを作成する

virsh snapshot-create-as domain コマンドは、ドメイン XML ファイルで指定されたプロパティー (<name> 要素や <description> 要素など) を使用して、ドメインのスナップショットを作成します。これらの値が XML 文字列に含まれていない場合、libvirt 値を選択します。スナップショットを作成するには、以下のコマンドを実行します。

```
# virsh snapshot-create-as domain {[--print-xml] | [--no-metadata] [--reuse-external]} [name]
[description] [--diskspec] diskspec]
```

残りのオプションは次のとおりです。

- **--print-xml** は、実際にスナップショットを作成するのではなく、出力として **snapshot-create** に適切な XML を作成します。
- **--diskspec** オプションを使用して、**--disk-only** および外部チェックポイントによる外部ファイルの作成方法を制御できます。このオプションは、ドメイン XML 内の <disk> 要素の数に従って、複数回指定できます。各 <diskspec> は、disk **[,snapshot=type][,driver=type][,file=name]**の形式です。ディスクまたは**file=name**にリテラルのコンマを含める場合は、2 番目のコンマでエスケープします。domain、name、および description の 3 つすべてが含まれている場合を除き、ディスクスペックの前にリテラル **--diskspec** を指定する必要があります。たとえば、ディスクスペックが **vda,snapshot=external,file=/path/to,,new** の場合は、以下の XML になります。

```
<disk name='vda' snapshot='external'>
  <source file='/path/to,new'/>
</disk>
```

- **--reuse-external** は、既存のファイルを宛先として再利用する外部スナップショットを作成します (このファイルが上書きされることを意味します)。この宛先が存在しない場合、既存のファイルの内容が失われないように、スナップショット要求は拒否されます。
- **--no-metadata** はスナップショットデータを作成しますが、メタデータはすぐに破棄されます (つまり、libvirt はスナップショットを最新として処理せず、また、snapshot-create が、その後、libvirt にメタデータを再度教えるために使用されない限り、スナップショットに戻ることはできません)。このオプションは **--print-xml** と互換性がありません。

14.15.2.3. 現在のドメインのスナップショットを作成する

このコマンドは、現在使用されているスナップショットをクエリーするために使用されます。使用するには、以下を実行します

```
# virsh snapshot-current domain {[--name] | [--security-info] | [snapshotname]}
```

snapshotname が使用されていない場合、ドメインの現在のスナップショット (存在する場合) のスナップショット XML が出力として表示されます。**--name** を指定すると、完全な XML ではなく、現在のスナップショット名のみが出力として送信されます。**--security-info** を指定すると、セキュリティーの機密情報が XML に含まれます。**snapshotname** を使用して、libvirt は、既存の名前付きスナップショットをドメインに戻さずに、現在のスナップショットになるようにリクエストを生成します。

14.15.2.4. snapshot-edit-domain

このコマンドは、現在使用中のスナップショットを編集するために使用されます。使用するには、以下を実行します

```
#virsh snapshot-edit domain [snapshotname] [--current] {[--rename] [--clone]}
```

snapshotname と **--current** の両方を指定すると、編集したスナップショットが現在のスナップショットになります。**snapshotname** を省略する場合は、現在のスナップショットを編集するには、**--current** を指定する必要があります。

これは、以下のコマンドシーケンスと同等ですが、いくつかのエラーチェックも含まれます。

```
# virsh snapshot-dumpxml dom name > snapshot.xml
# vi snapshot.xml [note - this can be any editor]
# virsh snapshot-create dom snapshot.xml --redefine [--current]
```

--rename を指定すると、編集されたファイルは別のファイル名に保存されます。**--clone** を指定してスナップショット名を変更すると、スナップショットメタデータのクローンが作成されます。いずれも指定されていない場合、編集によりスナップショット名は変更されません。スナップショット名の変更には十分な注意が必要です。1 つの qcow2 ファイル内の内部スナップショットなど、一部のスナップショットの内容には、作成元のスナップショットファイル名からしかアクセスできないためです。

14.15.2.5. snapshot-info-domain

snapshot-info-domain は、スナップショットに関する情報を表示します。使用するには、以下を実行します

```
# snapshot-info domain {snapshot | --current}
```

指定した**snapshot**、または**--current** で作成中のスナップショットの基本情報を出力します。

14.15.2.6. snapshot-list-domain

指定されたドメインで使用可能なすべてのスナップショットを一覧表示します。デフォルトでは、スナップショット名、作成時間、およびドメイン状態の列が表示されます。使用するには、以下を実行します

```
#virsh snapshot-list domain [{--parent | --roots | --tree}] [{[--from] snapshot | --current} [--
descendants]] [--metadata] [--no-metadata] [--leaves] [--no-leaves] [--inactive] [--active] [--internal] [--
external]
```

残りのオプションオプションは次のとおりです。

- **--parent** は、各スナップショットの親の名前を示す列を、出力テーブルに追加します。このオプションは、**--roots** や **--tree** と併用できません。
- **--roots** は、一覧をフィルターリングして、親がないスナップショットのみを表示します。このオプションは、**--parent** または **--tree** では使用できません。
- **--tree** は、スナップショット名のみを一覧表示し、出力をツリー形式で表示します。これらの 3 つのオプションは相互に排他的です。このオプションは、**--roots** や **--parent** と併用できません。
- **--from** は、指定したスナップショットの子であるスナップショットの一覧をフィルターリングします。または、**--current** が指定されている場合は、一覧を現在のスナップショットから開始するようにします。分離時または **--parent** と併用する場合、**--descendants** が存在しない限り、一覧は直接の子に制限されます。**--tree** と一緒に使用すると、**--descendants** の使用が暗黙的になります。このオプションは、**--roots** とは互換性がありません。**--from** または **--current** の開始点は、**--tree** オプションが指定されていない限り、一覧に含まれていないことに注意してください。
- **--leaves** を指定すると、この一覧は、子がないスナップショットのみを対象にフィルター処理されます。同様に、**--no-leaves** を指定すると、一覧は子を持つスナップショットのみにフィルターリングされます。(両方のオプションを省略するとフィルターリングはされませんが、両方のオプションを指定すると、サーバーがオプションを認識するかどうかに応じて同じリストまたはエラーが生成されることに注意してください) フィルターリングオプションは **--tree** と互換性がありません。

- **--metadata** が指定されている場合、リストは libvirt メタデータを含むスナップショットのみにフィルターされ、永続ドメインの定義解除を防止したり、一時ドメインの破棄時に失われたりします。同様に、**--no-metadata** が指定されている場合には、一覧が libvirt メタデータを必要としないスナップショットだけにフィルターリングされます。

- **--metadata** が指定されている場合、リストはドメインがシャットオフされたときに取得されたスナップショットにフィルターされます。**--active** を指定すると、ドメインの実行中に取得したスナップショットに一覧がフィルターリングされ、スナップショットには、その稼働状態に戻すためのメモリーの状態が含まれます。**--disk-only** が指定されている場合、リストは、ドメインの実行中に取得されたスナップショットにフィルターされますが、スナップショットにはディスクの状態のみが含まれます。

- **--internal** を指定すると、一覧は、既存ディスクイメージの内部ストレージを使用するスナップショットにフィルターリングされます。--external を指定すると、リストは、ディスクイメージまたはメモリー状態に外部ファイルを使用するスナップショットにフィルターリングされます。

14.15.2.7. snapshot-dumpxml ドメインスナップショット

virsh snapshot-dumpxml domain snapshot は、snapshot という名前のドメインのスナップショットのスナップショット XML を出力します。使用するには、以下を実行します

```
# virsh snapshot-dumpxml domain snapshot [--security-info]
```

--security-info オプションには、セキュリティーの機密情報も含まれます。**snapshot-current** を使用すると、現行スナップショットの XML に簡単にアクセスできます。

14.15.2.8. snapshot-parent ドメイン

指定したスナップショットについて、親スナップショットが存在する場合はその名前を出力し、存在しない場合は**--current** で現在のスナップショットの名前を出力します。使用するには、以下を実行します

```
#virsh snapshot-parent domain {snapshot | --current}
```

14.15.2.9. snapshot-revert ドメイン

指定したドメインを、**snapshot** で指定したスナップショットか、**--current** で指定した現在のスナップショットに戻します。

警告

これは破壊的なアクションであることに注意してください。最後にスナップショットを取得してからドメイン内に加えられた変更はすべて失われます。また、**snapshot-revert** の完了後のドメインの状態は、元のスナップショット取得時のドメインの状態であることに注意してください。

スナップショットを元に戻すには、次のコマンドを実行します。

```
# snapshot-revert domain {snapshot | --current} [{--running | --paused}] [--force]
```

通常、スナップショットに戻すと、ドメインはスナップショット作成時の状態のままになります。ただし、ゲスト仮想マシンの状態がないディスクスナップショットでは、ドメインは非アクティブの状態のままになります。**--running** または **--paused** のいずれかを渡すと、状態の変更 (非アクティブなドメインの起動や、実行中のドメインの一時停止など) が行われます。一時ドメインは非アクティブの状態にできないため、一時ドメインのディスクスナップショットに戻す場合には、このオプションのいずれかを使用する必要があります。

snapshot revertに追加の危険が伴う場合が 2 つあり、その場合は**--force** を使用して続行する必要があります。1 つ目は、設定を戻すための完全なドメイン情報がないスナップショットの場合です。libvirt は、現在の設定がスナップショット時の使用状況と一致することを証明できないため、**--force** を提供することで、スナップショットが現在の設定と互換性があることを libvirt に保証します (一致しないとドメインの実行に失敗する可能性が高くなります)。もう 1 つは、実行中のドメインから、既存のハイパーバイザーを再利用するのではなく、新しいハイパーバイザーを作成する必要があるアクティブなドメインに戻す場合です。これは、既存の VNC 接続または Spice 接続を壊すなどの欠点を意味するからです。これは、証明可能な互換性のない設定を使用するアクティブなスナップショットと、**--start** フラグまたは **--pause** オプションを組み合わせた非アクティブなスナップショットで発生します。

14.15.2.10. snapshot-delete ドメイン

snapshot-delete domain は、指定されたドメインのスナップショットを削除します。これを実行するには、以下を実行します。

```
# virsh snapshot-delete domain {snapshot | --current} [--metadata] [{--children | --children-only}]
```

このコマンドは、**snapshot** という名前のドメインに対するスナップショット、または **--current** を使用した現行スナップショットを削除します。このスナップショットに子スナップショットが含まれる場合は、このスナップショットの変更は子スナップショットにマージされます。**--children** を使用すると、このスナップショットと、その子スナップショットがすべて削除されます。**--children-only** を使用すると、このスナップショットの子はすべて削除されますが、このスナップショットはそのままになります。これらの 2 つのオプションは相互に排他的です。

この**--metadata**を使用すると、libvirt が管理するスナップショットのメタデータが削除されます。外部ツールによるアクセス用に、スナップショットのコンテンツはそのまま残されます。それ以外の場合は、スナップショットを削除すると、その時点のデータコンテンツも削除されます。

14.16. ゲスト仮想マシンの CPU モデル設定

このセクションでは、ゲスト仮想マシンの CPU モデル設定に関する情報を提供します。

14.16.1. はじめに

すべてのハイパーバイザーには、ゲスト仮想マシンがデフォルトで CPU に表示するものに関する独自のポリシーがあります。一部のハイパーバイザーは、ゲスト仮想マシンで使用できる CPU ホスト物理マシンの機能を決定します。一方、QEMU/KVM は、ゲスト仮想マシンを、**qemu32** または **qemu64** という名前の一般的なモデルで表示します。このようなハイパーバイザーは、より高度なフィルターリングを実行し、すべての物理 CPU を少数のグループに分類し、ゲスト仮想マシンに提示される各グループに対してベースライン CPU モデルを 1 つ用意します。このような動作により、ホストの物理マシン間でゲスト仮想マシンを安全に移行できます。ただし、ゲスト仮想マシンすべてに、同じグループに分類される物理 CPU がある場合に限ります。libvirt は通常、ポリシー自体を適用せず、より高いレイヤーで必要なポリシーを定義するメカニズムを提供します。ホストの物理マシン間でゲスト仮想マシンの移行が正常に実行されるようにするには、CPU モデル情報を取得し、適切なゲスト仮想マシンの

CPU モデルを定義する方法を理解することが重要になります。ハイパーバイザーは、認識している機能のみをエミュレートでき、ハイパーバイザーがリリースされてから作成された機能はエミュレートできない場合がある点に注意してください。

14.16.2. ホスト物理マシンの CPU モデルの学習

virsh capabilities コマンドは、ハイパーバイザー接続とホスト物理マシンの機能を説明する XML ドキュメントを表示します。表示されている XML スキーマが拡張され、ホスト物理マシンの CPU モデルに関する情報が提供されるようになりました。CPU モデルを説明する際の大きな課題の 1 つは、すべてのアーキテクチャーで、その機能を公開するアプローチが異なることです。x86 では、最新の CPU の機能は CPUID 命令を介して公開されます。基本的に、これは 32 ビット整数のセットになり、各ビットには特定の意味が与えられます。幸い、AMD と Intel は、これらのビットの共通のセマンティクスに同意しています。他のハイパーバイザーは、CPUID マスクの概念をゲスト仮想マシン設定形式で直接公開します。ただし、QEMU/KVM は x86 アーキテクチャーだけでなく、CPUID が標準的な設定形式として適切ではないことは明らかです。QEMU は、CPU モデル名文字列と一連の名前付きオプションを組み合わせたスキームを使用することになりました。x86 では、CPU モデルはベースライン CPUID マスクにマップされ、オプションを使用してマスクのビットのオンとオフを切り替えることができます。libvirt はこのリードに従うことを決定し、モデル名とオプションの組み合わせを使用します。

既知の CPU モデルをすべて一覧表示するデータベースは実用的ではないので、libvirt のベースライン CPU モデル名の一覧は少ないです。CPUID ビットの最大数を実際のホストマシン CPU と共有し、残りのビットを名前付き機能として一覧表示するものを選択します。libvirt では、ベースライン CPU に含まれる機能が表示されないことに注意してください。一見、これは欠陥のように思われますが、本セクションで説明するように、この情報を実際に把握する必要はありません。

14.16.3. ホスト物理マシンのプールに対応するための互換性のある CPU モデルの決定

1 台のホスト物理マシンに搭載されている CPU 機能を確認できるようになりました。次の手順は、ゲスト仮想マシンに公開するのに最も適した CPU 機能を判断することです。ゲスト仮想マシンを別のホスト物理マシンに移行する必要がないことが分かっている場合は、ホスト物理マシンの CPU モデルを変更せずにそのまま渡すことができます。仮想化データセンターには、すべてのサーバーで 100% 同一の CPU が保証されるように設定されている場合があります。ここでも、ホスト物理マシンの CPU モデルは、変更せずにそのまま渡すことができます。ただし、より一般的なケースは、ホストの物理マシン間で CPU にバリエーションがある場合です。この混合 CPU 環境では、最も低い公共分母の CPU を決定する必要があります。これは完全に単純なものではないため、libvirt はこのタスクの API を提供します。libvirt が、ホストの物理マシンの CPU モデルをそれぞれ説明する XML ドキュメントの一覧を提供している場合、libvirt は、これらを CPUID マスクに内部的に変換し、その共通部分を計算して、CPUID マスクの結果を XML CPU 説明に戻します。

以下は、**virsh capabilities** の実行時に、libvirt が基本的なワークステーションの機能として報告する内容の例になります。

図14.3 ホストの物理マシンの CPU モデル情報をプルする

```
<capabilities>
  <host>
    <cpu>
      <arch>i686</arch>
      <model>pentium3</model>
      <topology sockets='1' cores='2' threads='1'/>
      <feature name='lahf_lm'/>
      <feature name='lm'/>
      <feature name='xtpr'/>
      <feature name='cx16'/>
      <feature name='ssse3'/>
      <feature name='tm2'/>
      <feature name='est'/>
      <feature name='vmx'/>
      <feature name='ds_cpl'/>
      <feature name='monitor'/>
      <feature name='pni'/>
      <feature name='pbe'/>
      <feature name='tm'/>
      <feature name='ht'/>
      <feature name='ss'/>
      <feature name='sse2'/>
      <feature name='acpi'/>
      <feature name='ds'/>
      <feature name='clflush'/>
      <feature name='apic'/>
    </cpu>
  </host>
</capabilities>
```

次に、同じ **virsh capabilities** コマンドを使用してランダムサーバーと比較します。

図14.4 ランダムなサーバーから CPU 説明を生成する

```
<capabilities>
  <host>
    <cpu>
      <arch>x86_64</arch>
      <model>phenom</model>
      <topology sockets='2' cores='4' threads='1'/>
      <feature name='osvw'/>
      <feature name='3dnowprefetch'/>
      <feature name='misalignsse'/>
      <feature name='sse4a'/>
      <feature name='abm'/>
      <feature name='cr8legacy'/>
      <feature name='extapic'/>
      <feature name='cmp_legacy'/>
      <feature name='lahf_lm'/>
      <feature name='rdtscp'/>
      <feature name='pdpe1gb'/>
      <feature name='popcnt'/>
      <feature name='cx16'/>
      <feature name='ht'/>
      <feature name='vme'/>
    </cpu>
    ...snip...
```

この CPU 説明が、以前のワークステーションの CPU 説明と互換性があるかどうかを確認するには、**virsh cpu-compare** コマンドを使用します。

縮小した内容は、**virsh-caps-workstation-cpu-only.xml** という名前のファイルに保存されており、このファイルで **virsh cpu-compare** コマンドを実行できます。

```
# virsh cpu-compare virsh-caps-workstation-cpu-only.xml
Host physical machine CPU is a superset of CPU described in virsh-caps-workstation-cpu-only.xml
```

この出力から分かるように、libvirt は、CPU との厳密な互換性がないことを正しく報告しています。これは、クライアント CPU にないサーバー CPU の機能が複数あるためです。クライアントとサーバー間で移行を行うには、XML ファイルを開いて、一部の機能をコメントアウトする必要があります。削除する機能を特定するには、両方のマシンの CPU 情報が含まれる**both-cpus.xml**で **virsh cpu-baseline** コマンドを実行します。 **# virsh cpu-baseline both-cpus.xml** を実行すると、以下が行われます。

図14.5 複合 CPU ベースライン

```
<cpu match='exact'>
  <model>pentium3</model>
  <feature policy='require' name='lahf_lm'/>
  <feature policy='require' name='lm'/>
  <feature policy='require' name='cx16'/>
  <feature policy='require' name='monitor'/>
  <feature policy='require' name='pni'/>
  <feature policy='require' name='ht'/>
  <feature policy='require' name='sse2'/>
  <feature policy='require' name='clflush'/>
  <feature policy='require' name='apic'/>
</cpu>
```

この複合ファイルでは、共通する要素が示されます。共通していないものはすべてコメントアウトする必要があります。

14.17. ゲスト仮想マシンの CPU モデルの設定

単純なデフォルトの場合、ゲスト仮想マシンの CPU 設定は、ホストの物理マシンの capabilities XML が公開するものと同じ基本的な XML 表現を受け入れます。つまり、**cpu-baseline** virsh コマンドーの XML を、<domain> 要素の最上位にあるゲスト仮想マシン XML に直接コピーできるようになりました。 上記の XML スニペットでは、ゲスト仮想マシン XML で CPU を記述する際に利用できる追加の属性がいくつかあります。これらはほとんど無視してもかまいませんが、ここではその実行内容を簡単に説明します。トップレベルの <cpu> 要素には、**match** という名前の属性があります。この属性には、以下の値を指定できます。

- match='minimum' - ホストの物理マシン CPU には、ゲスト仮想マシン XML で説明されている CPU 機能が少なくとも必要です。ホストの物理マシンに、ゲスト仮想マシンの設定以外の機能がある場合は、ゲスト仮想マシンにも公開されます。

- match='exact' - ホストの物理マシンの CPU には、ゲスト仮想マシンの XML で説明されている CPU 機能が少なくとも必要です。ホストの物理マシンに、ゲスト仮想マシンの設定以外の機能がある場合は、ゲスト仮想マシンからマスクアウトされます。

- match='strict' - ホストの物理マシン CPU には、ゲスト仮想マシン XML で説明されている CPU 機能とまったく同じものが必要です。

次の機能拡張では、<feature>要素はそれぞれ、以下の可能な値を持つ追加の 'policy' 属性を持つことができます。

- policy='force' - ホストの物理マシンに機能がない場合でも、ゲスト仮想マシンにその機能を公開します。通常、これはソフトウェアエミュレーションの場合にのみ役立ちます。

- policy='require' - 機能をゲスト仮想マシンに公開し、ホストの物理マシンにその機能がない場合には失敗します。これは妥当なデフォルトです。

- policy='optional' - 機能に対応している場合にゲスト仮想マシンに公開します。

- policy='disable' - ホストの物理マシンにこの機能がある場合は、ゲスト仮想マシンに表示しないようにします。

- policy='forbid' - ホストの物理マシンにこの機能がある場合は、ゲスト仮想マシンの起動に失敗して拒否されます。

'forbid' ポリシーは、機能が正しくないアプリケーションが CPUID マスクになくても機能を使用しようとし、その機能を持つホストの物理マシンで誤ってゲスト仮想マシンを実行しないようにするニッチのシナリオ向けです。'optional' ポリシーには、移行に関する特別な動作があります。ゲスト仮想マシンが最初に起動するときにこのパラメーターはオプションですが、ゲスト仮想マシンがライブマイグレーションされるときには、機能が移行中に消えないため、このポリシーは必須になります。

14.18. ゲスト仮想マシンのリソースの管理

virsh では、ゲスト仮想マシンベースでリソースをグループ化し、割り当てることができます。これは、*cgroups* を作成し、ゲスト仮想マシンの代わりにこれを管理する libvirt デーモンによって管理されます。システム管理者ができることは、指定したゲスト仮想マシンに対して調整可能パラメーターを照会するか設定することだけです。以下の調整可能パラメーターを使用できます。

- **memory** - メモリーコントローラーにより、RAM の制限やスワップの使用量を設定し、グループ内の全プロセスの累積使用量を照会できます。
- **cpuset** - CPU セットコントローラーは、グループ内のプロセスを CPU セットにバインドし、CPU 間の移行を制御します。
- **cpuacct** - CPU アカウンティングコントローラーは、プロセスのグループに対する CPU 使用率の情報を提供します。
- **cpu** - CPU スケジューラーコントローラーは、グループ内のプロセスの優先順位を制御します。これは、**nice** レベルの権限を付与するのと似ています。
- **devices** - デバイスコントローラーは、文字デバイスおよびブロックデバイスのアクセス制御リストを付与します。
- **freezer** - フリーザーコントローラーは、グループ内のプロセスを一時停止して再開します。これは、グループ全体の **SIGSTOP** と似ています。
- **net_cls** - ネットワーククラスコントローラーは、プロセスを **tc** ネットワーククラスに関連付けることで、ネットワークの使用率を管理します。

グループ階層を作成する際、cgroup はマウントポイントとディレクトリーの設定を完全に管理者の裁量に任せ、いくつかのマウントポイントを **/etc/fstab** に追加するよりも複雑になります。ディレクトリー階層を設定し、その中にプロセスを配置する方法を決定する必要があります。これは、次の virsh コマンドで実行できます。

- **schedinfo** - 「スケジュールパラメーターの設定」 で説明しています。
- **blkiotune** - 「ブロック I/O パラメーターの表示または設定」 で説明しています。
- **domiftune** - 「ネットワークインターフェイスの帯域幅パラメーターの設定」 で説明しています。
- **memtune** - 「メモリーの調整の設定」 で説明しています。

14.19. スケジュールパラメーターの設定

schedinfo を使用すると、スケジューラーパラメーターをゲスト仮想マシンに渡すことができます。以下のコマンド形式を使用する必要があります。

```
#virsh schedinfo domain --set --weight --cap --current --config --live
```

各パラメーターの説明を以下に示します。

- **domain** - これはゲスト仮想マシンドメインです
- **--set** - ここに置かれるストリングは、呼び出されるコントローラーまたはアクションです。必要な場合には、追加のパラメーターまたは値も追加する必要があります。
- **--current** - **--set** と一緒に使用すると、指定した**set**文字列を現在のスケジューラー情報として使用します。一緒に使用しない場合は、現在のスケジューラー情報が表示されます。
- **--config** - - **--set** と併用すると、次回のシステムの再起動時に指定した**set** 文字列を使用します。一緒に使用しない場合は、設定ファイルに保存されているスケジューラー情報が表示されます。
- **--live** - **--set** と一緒に使用すると、現在実行しているゲスト仮想マシンで指定した**set** 文字列を使用します。一緒に使用しない場合は、実行中の仮想マシンが現在使用している設定が表示されます。

スケジューラーは、***cpu_shares***、***vcpu_period***、および ***vcpu_quota*** のいずれかのパラメーターで設定できます。

例14.5 schedinfo show

この例は、シェルゲスト仮想マシンのスケジュール情報を示しています。

```
# virsh schedinfo shell
Scheduler      : posix
cpu_shares     : 1024
vcpu_period    : 100000
vcpu_quota     : -1
```

例14.6 schedinfo set

この例では、cpu_shares が 2046 に変更されています。これは現在の状態に影響しますが、設定ファイルには影響しません。

```
# virsh schedinfo --set cpu_shares=2046 shell
Scheduler      : posix
cpu_shares     : 2046
vcpu_period    : 100000
vcpu_quota     : -1
```

14.20. ブロック I/O パラメーターの表示または設定

blkiotune は、指定されたゲスト仮想マシンの I/O パラメーターを設定または表示します。以下の形式を使用する必要があります。

```
# virsh blkiotune domain [--weight weight] [--device-weights device-weights] [[--config] [--live] | [--current]]
```

このコマンドの詳細は、『Virtualization Tuning and Optimization Guide』 を参照してください。

14.21. メモリーの調整の設定

virsh memtune virtual_machine --parameter size は、『Virtualization Tuning and Optimization Guide』で説明しています。

14.22. 仮想ネットワークコマンド

次のコマンドは、仮想ネットワークを操作します。libvirt には、ドメインが使用でき、実際のネットワークデバイスにリンクできる仮想ネットワークを定義する機能があります。この機能の詳細は、libvirt Web サイト のドキュメントを参照してください。仮想ネットワーク用のコマンドの多くは、ドメインに使用されるコマンドと似ていますが、仮想ネットワークに名前を付ける方法は、名前または UUID のいずれかになります。

14.22.1. 仮想ネットワークの自動起動

このコマンドは、ゲスト仮想マシンの起動時に自動的に開始されるように仮想ネットワークを設定します。このコマンドを実行するには:

```
# virsh net-autostart network [--disable]
```

このコマンドは、**--disable** オプションを受け入れます。これにより、自動開始コマンドが無効になります。

14.22.2. XML ファイルからの仮想ネットワークの作成

このコマンドは、XML ファイルから仮想ネットワークを作成します。libvirt が使用する XML ネットワーク形式の説明は、libvirt の Web サイト を参照してください。このコマンド *file* は、XML ファイルへのパスになります。XML ファイルから仮想ネットワークを作成するには、次のコマンドを実行します。

```
# virsh net-create file
```

14.22.3. XML ファイルからの仮想ネットワークの定義

このコマンドは、XML ファイルから仮想ネットワークを定義します。ネットワークは定義されたばかりですが、インスタンス化されていません。仮想ネットワークを定義するには、次のコマンドを実行します。

```
# net-define file
```

14.22.4. 仮想ネットワークの停止

このコマンドは、名前または UUID で指定された特定の仮想ネットワークを破棄 (停止) します。これはすぐに有効になります。指定されたネットワークを停止するには、*network* が必要です。

```
# net-destroy network
```

14.22.5. ダンプファイルの作成

このコマンドは、仮想ネットワーク情報を XML ダンプとして指定された仮想ネットワークの stdout に出力します。**--inactive** を指定すると、物理機能は、関連付けられた仮想機能に展開されません。ダンプファイルを作成するには、次のコマンドを実行します。

```
# virsh net-dumpxml network [--inactive]
```

14.22.6. 仮想ネットワークの XML 設定ファイルの編集

以下のコマンドは、ネットワークの XML 設定ファイルを編集します。

```
# virsh net-edit network
```

XML ファイルの編集に使用するエディターは、$VISUAL 環境変数または $EDITOR 環境変数で提供できます。デフォルトは **vi** です。

14.22.7. 仮想ネットワークに関する情報の取得

このコマンドは、*ネットワークオブジェクト*に関する基本情報を返します。ネットワーク情報を取得するには、次のコマンドを実行します。

```
# virsh net-info network
```

14.22.8. 仮想ネットワークに関する情報の一覧表示

--all が指定されている場合は、アクティブなネットワーク一覧を返します。**--inactive**が指定されている場合は、非アクティブなネットワークのみが一覧表示されます。返されたネットワークを **--persistent** でフィルターリングして永続的なネットワークを、**--transient** でフィルターリングして一時的なネットワークを、**--autostart** でフィルターリングして自動起動が有効になっているネットワークを、そして **--no-autostart** でフィルターリングして自動起動が無効になっているネットワークを一覧表示することも推奨されます。

注記: 古いサーバーと通信する場合、このコマンドは一連の API コールと固有の競合を使用することを強制されます。ここでは、リストの収集中にプールの状態が変更した場合に、プールが一覧に表示されなかったり、複数回表示されたりすることがあります。新しいサーバーにはこの問題がありません。

仮想ネットワークの一覧を表示するには、次のコマンドを実行します。

```
# net-list [--inactive | --all] [--persistent] [<--transient>] [--autostart] [<--no-autostart>]
```

14.22.9. ネットワーク UUID のネットワーク名への変換

このコマンドは、ネットワーク UUID をネットワーク名に変換します。これを実行するには、以下を実行します。

```
# virsh net-name network-UUID
```

14.22.10. (定義済みの) 非アクティブなネットワークの起動

このコマンドは、(以前に定義された) 非アクティブなネットワークを開始します。これを実行するには、以下を実行します。

```
# virsh net-start network
```

14.22.11. 非アクティブなネットワークの設定の定義解除

このコマンドは、非アクティブなネットワークの設定の定義を解除します。これを実行するには、以下を実行します。

```
# net-undefine network
```

14.22.12. ネットワーク名のネットワーク UUID への変換

このコマンドは、ネットワーク名をネットワーク UUID に変換します。これを実行するには、以下を実行します。

```
# virsh net-uuid network-name
```

14.22.13. 既存のネットワーク定義ファイルの更新

このコマンドは、既存のネットワーク定義の特定のセクションを更新し、ネットワークを破棄して再起動することなく、すぐに有効になります。このコマンドは、add-first、add-last、add (add-last の同義語)、delete、または modify のいずれかです。セクションは、bridge、domain、ip、ip-dhcp-host、ip-dhcp-range、forward、forward-interface、forward-pf、portgroup、dns-host、dns-txt、または dns-srv。各セクションは、変更される要素につながる xml 要素階層の連結によって名前が付けられます。たとえば、ip-dhcp-host は、ネットワークの **<ip>** 要素内に **<dhcp>** 要素に含まれる **<host>** 要素を変更します。xml は、変更するタイプの完全な xml 要素のテキストです (たとえば、<host mac="00:11:22:33:44:55' ip='192.0.2.1'/>)、または完全な xml 要素が含まれるファイルの名前です。あいまいさの除去は、提供されているテキストの最初の文字を調べて行います。最初の文字が "<" の場合は XML テキストで、最初の文字が ">" でない場合は、使用される xml テキストが含まれるファイル名になります。--parent-index オプションは、要求された要素が、複数の親要素のうちどの要素にあるのかを指定するために使用します (0-ベース)。たとえば、dhcp **<host>** 要素は、ネットワーク内の複数の **<ip>** 要素のいずれかに置くことができます。parent-index が指定されていない場合は最も適切な **<ip>** 要素が選択されます (通常は、**<dhcp>** 要素がすでに存在するものだけが対象となります)。ただし、***--parent***-index が指定されていると、**<ip>** の特定のインスタンスが修正されます。***--live*** が指定されている場合は、実行中のネットワークに影響を及ぼします。***--config*** を指定した場合は、次に永続的なネットワークを起動する際に影響を及ぼします。***-- current*** が指定されている場合、現在のネットワーク状態に影響します。***--live*** および ***--config*** オプションの両方を指定できますが、--current は排他的です。オプションを指定しない場合は、***--current*** を指定することと同じです。

設定ファイルを更新するには、次のコマンドを実行します。

```
# virsh net-update network command section xml [--parent-index index] [[--live] [--config] | [--current]]
```

第15章 VIRTUAL MACHINE MANAGER を使用したゲストの管理 (VIRT-MANAGER)

本セクションでは、Virtual Machine Manager (**virt-manager**) のウィンドウ、ダイアログボックス、およびさまざまな GUI コントロールについて説明します。

virt-manager は、ホストシステムおよびリモートホストシステム上のハイパーバイザーとゲストのグラフィカルビューを提供します。**virt-manager** は、次のような仮想化管理タスクを実行できます。

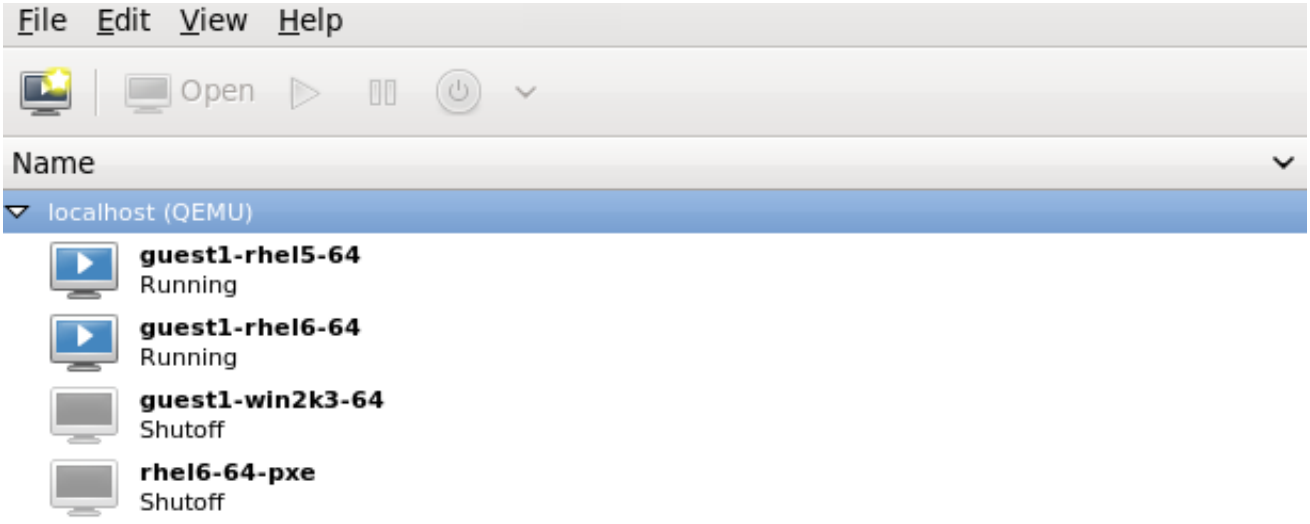
- ゲストの定義と作成
- メモリーの割り当て
- 仮想 CPU の割り当て
- 運用パフォーマンスの監視
- ゲストの保存、復元、一時停止、再開、およびシャットダウンと起動
- テキストコンソールやグラフィカルコンソールへのリンク
- ライブマイグレーションおよびオフラインマイグレーション。

15.1. VIRT-MANAGER の起動

virt-manager セッションを開始するには、Applications メニューを開き、System Tools メニューの Virtual Machine Manager (**virt-manager**) を選択します。

virt-manager のメインウィンドウが表示されます。

図15.1 virt-manager の起動



または、以下のコマンドで示すように、ssh を使用して、**virt-manager** をリモートで起動できます。

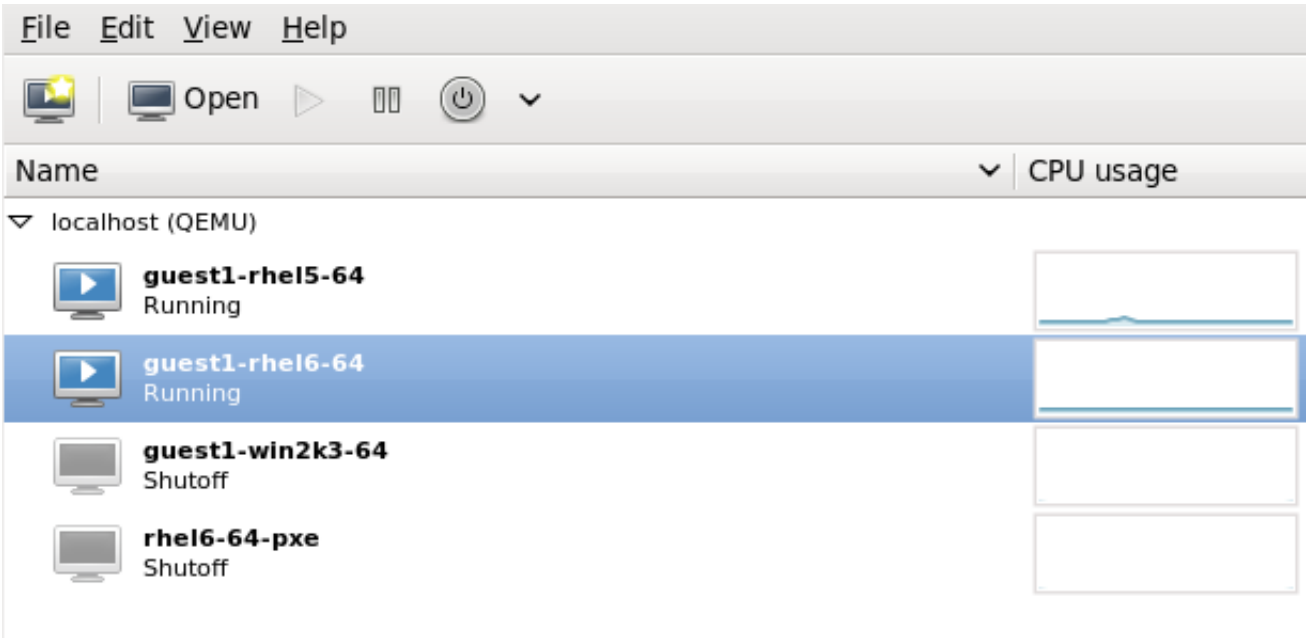
```
ssh -X host's address
[remotehost]# virt-manager
```

ssh を使用した仮想マシンおよびホストの管理については、「SSH を使用したリモート管理」 を参照してください。

15.2. VIRTUAL MACHINE MANAGER のメインウィンドウ

このメインウィンドウには、実行中のゲストと、ゲストが使用するリソースがすべて表示されます。ゲストの名前をダブルクリックして、ゲストを選択します。

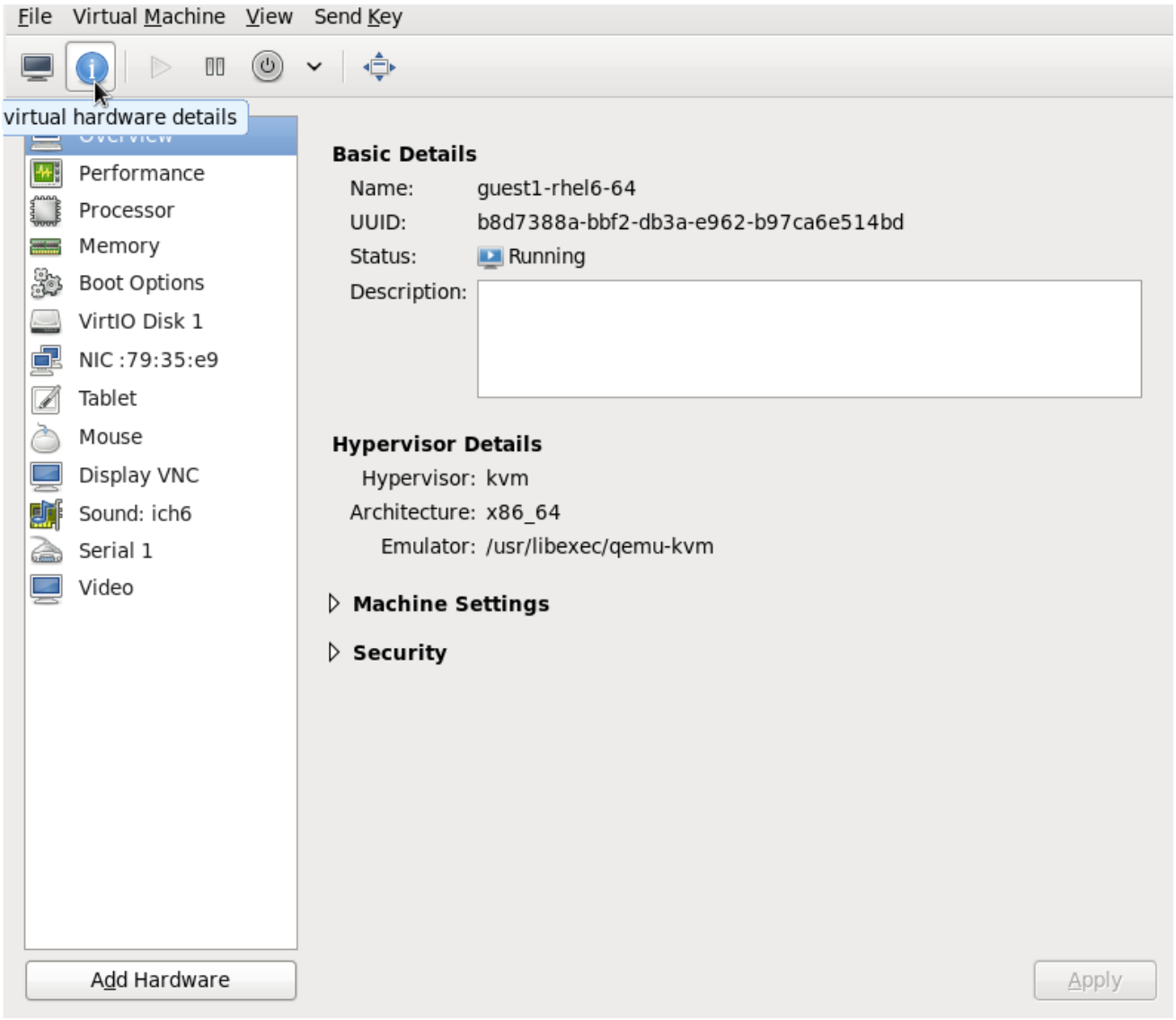
図15.2 Virtual Machine Manager のメインウィンドウ



15.3. 仮想ハードウェアの詳細ウィンドウ

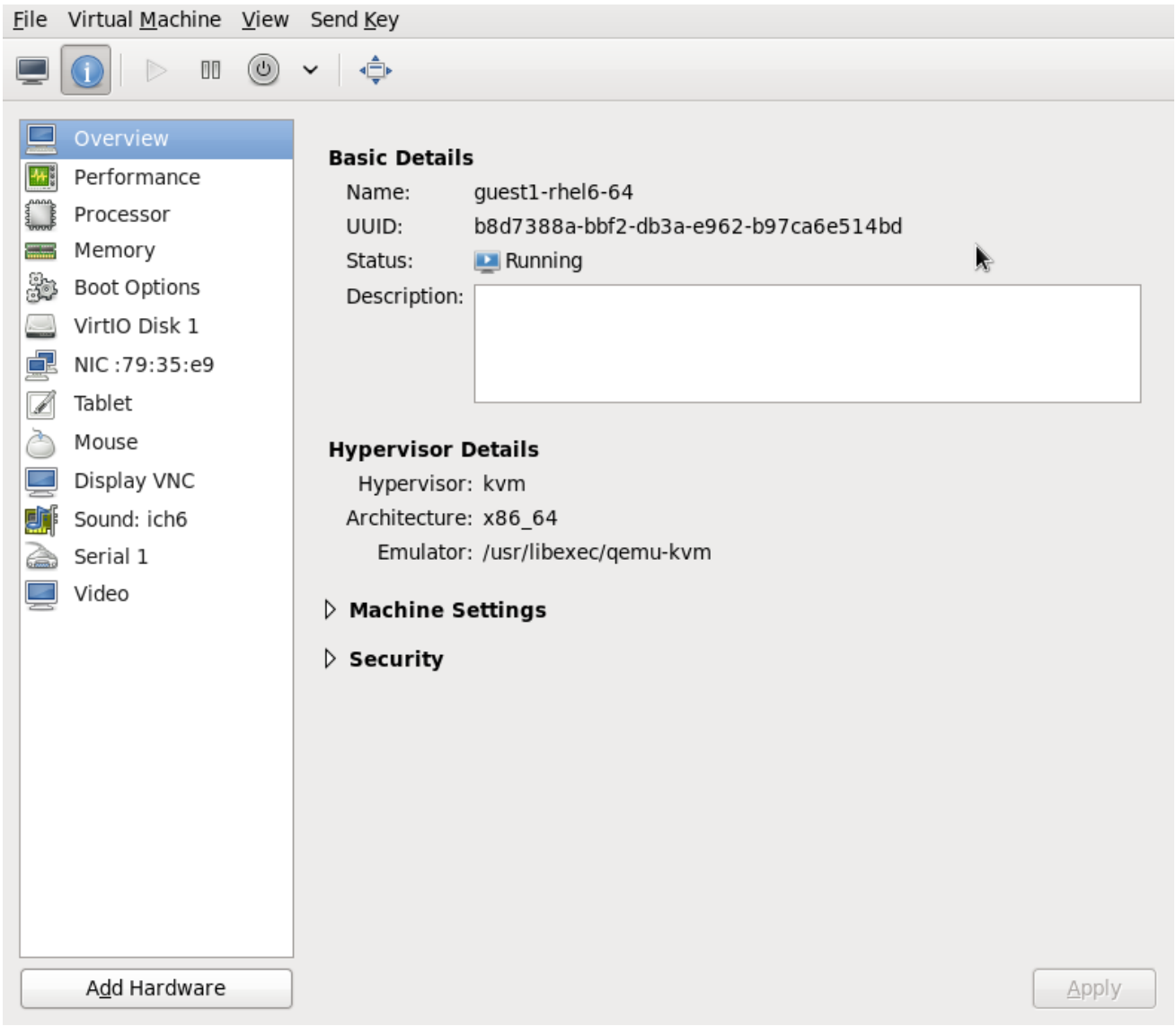
仮想ハードウェアの詳細ウィンドウには、ゲストに設定された仮想ハードウェアに関する情報が表示されます。仮想ハードウェアリソースは、このウィンドウで追加、削除、および変更できます。仮想ハードウェアの詳細ウィンドウにアクセスするには、ツールバーのアイコンをクリックします。

図15.3 仮想ハードウェアの詳細アイコン



アイコンをクリックすると、仮想ハードウェアの詳細ウィンドウが表示されます。

図15.4 仮想ハードウェアの詳細ウィンドウ



15.3.1. ゲスト仮想マシンへの USB デバイスの接続

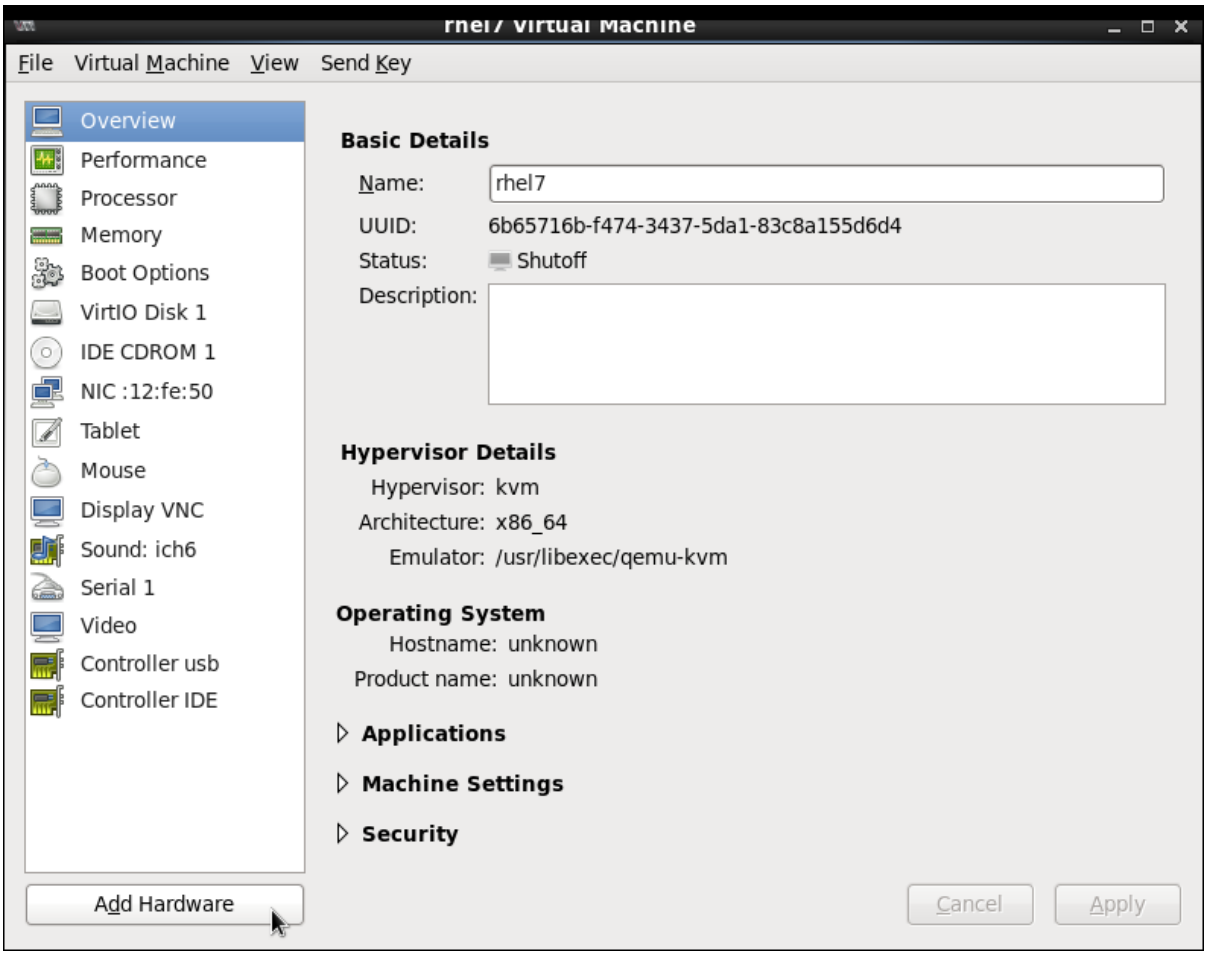
注記

USB デバイスをゲスト仮想マシンに接続するには、まずホストの物理マシンに接続し、デバイスが機能していることを確認します。ゲストを実行している場合は、続行する前にシャットダウンする必要があります。

手順15.1 Virt-Manager を使用した USB デバイスの接続

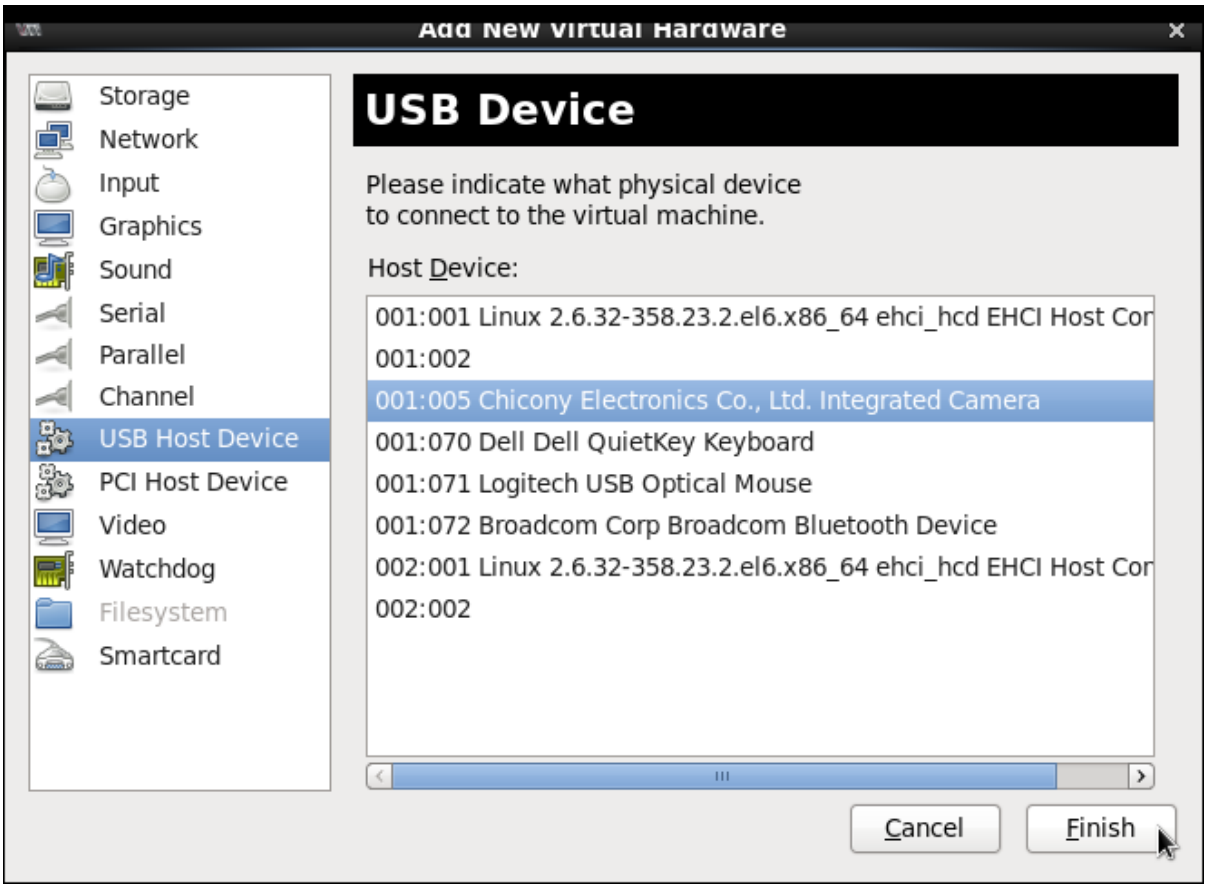
1. ゲスト仮想マシンの Virtual Machine Details 画面を開きます。
2. **Add Hardware** をクリックします。

図15.5 ハードウェアボタンの追加



3. Add New Virtual Hardware ポップアップで、USB Host Device を選択し、一覧から接続するデバイスを選択して、**Finish** をクリックします。

図15.6 USB デバイスの追加

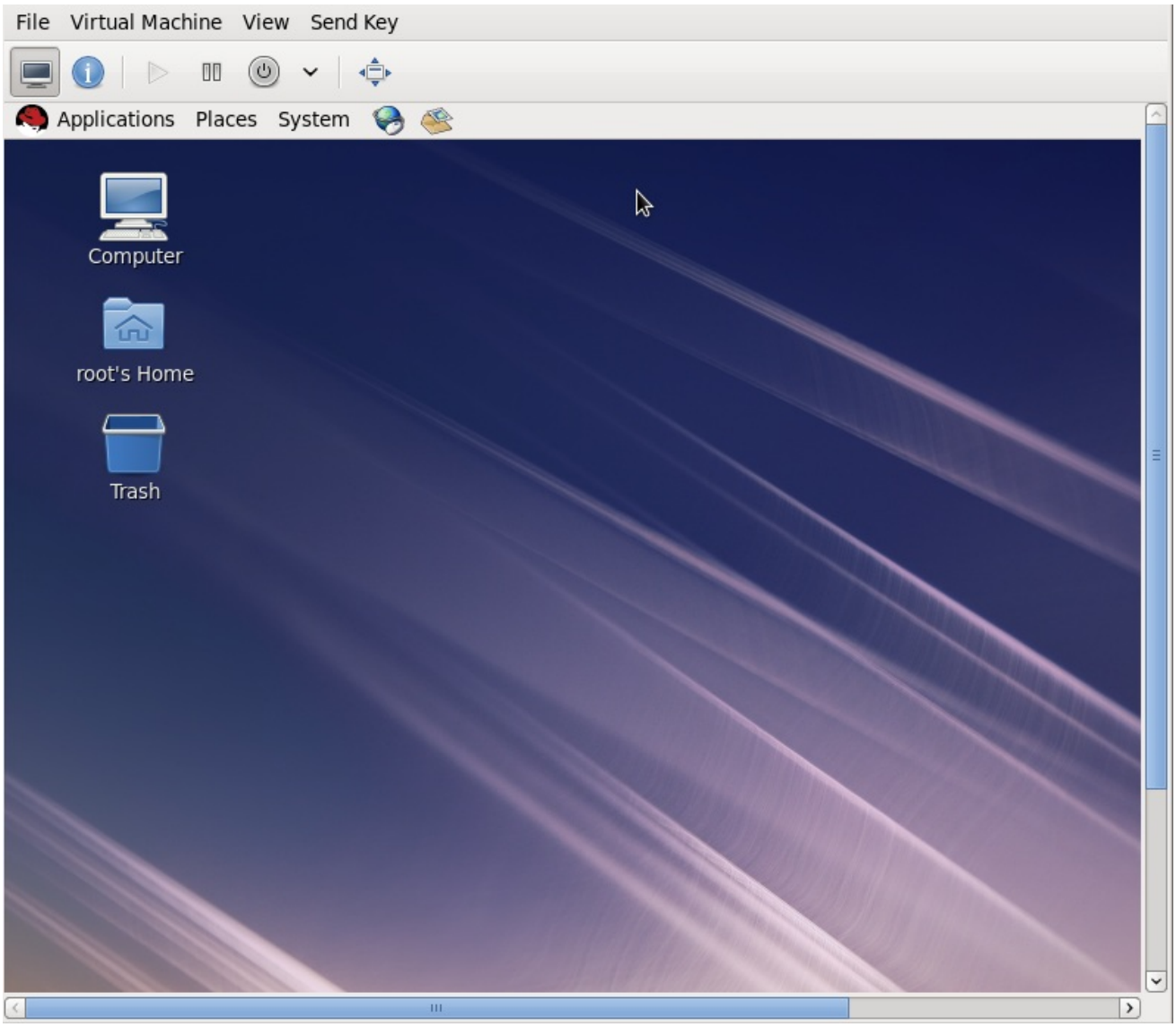


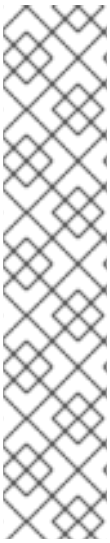
4. ゲスト仮想マシンで USB デバイスを使用するには、ゲスト仮想マシンを起動します。

15.4. 仮想マシンのグラフィカルコンソール

このウィンドウには、ゲストのグラフィカルコンソールが表示されます。ゲストは、複数のプロトコルを使用してグラフィカルフレームバッファーをエクスポートできます。**virt-manager** は、VNC および SPICE をサポートします。仮想マシンが認証を要求するように設定されている場合、仮想マシンのグラフィカルコンソールは、ディスプレイが表示される前にパスワードの入力を求めます。

図15.7 グラフィカルコンソールウィンドウ





注記

VNC は多くのセキュリティー専門家によって安全ではないと見なされますが、Red Hat Enterprise Linux の仮想化で VNC のセキュアな使用を可能にするためにいくつかの変更が加えられました。ゲストマシンは、ローカルホストのループバックアドレス (**127.0.0.1**) のみをリッスンします。これにより、ホストでシェル特権を持つユーザーのみが、VNC を介して virt-manager および仮想マシンにアクセスできるようになります。virt-manager は、他のパブリックネットワークインターフェイスをリッスンするように設定されており、別の方法を設定することもできますが、推奨されません。

リモート管理は、トラフィックを暗号化する SSH 経由のトンネルにより実行できます。VNC は、SSH 経由でのトンネルを使用せずにリモートからアクセスするように設定できますが、セキュリティー上の理由から推奨されません。ゲストをリモートで管理するには、*5章 ゲストのリモート管理* の手順に従います。TLS は、ゲストおよびホストシステムを管理するためのエンタープライズレベルのセキュリティーを提供できます。

ローカルデスクトップは、ゲストマシンに送信しないようにキーの組み合わせ (Ctrl+Alt+F1 キーなど) を傍受できます。**Send key** メニューオプションを使用すると、これらのシーケンスを送信できます。ゲストマシンウィンドウで **Send key** メニューをクリックし、送信するキーシーケンスを選択します。また、このメニューから画面出力をキャプチャーすることもできます。

SPICE は、Red Hat Enterprise Linux で利用可能な VNC の代替手段です。

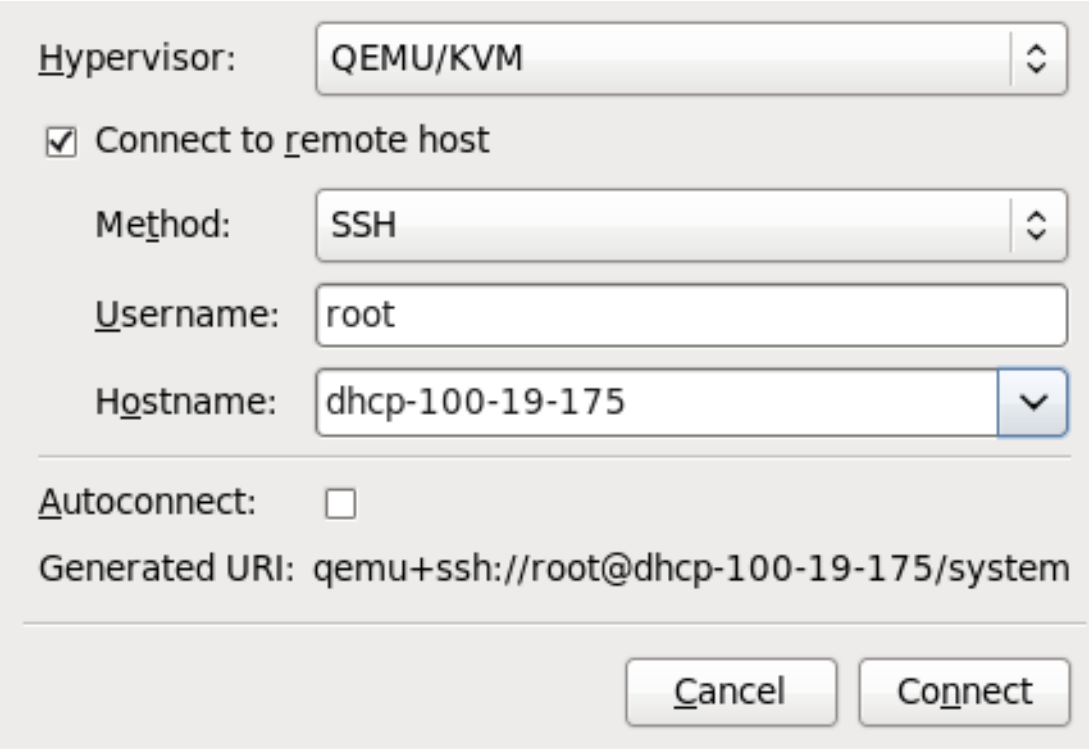
15.5. リモート接続の追加

この手順では、**virt-manager** を使用してリモートシステムへの接続を設定する方法を説明します。

1. 新規接続を作成するには、**File** メニューを開き、**Add Connection...** メニューアイテムを選択します。

2. **Add Connection** ウィザードが表示されます。ハイパーバイザーを選択します。Red Hat Enterprise Linux 6 の場合は、システムで **QEMU/KVM** を選択します。ローカルシステムの Local、またはリモート接続オプションのいずれかを選択し、**Connect** をクリックします。この例では、SSH 経由でリモートトンネルを使用します。これは、デフォルトのインストールで機能します。リモート接続の設定の詳細は、*5章 ゲストのリモート管理* を参照してください。

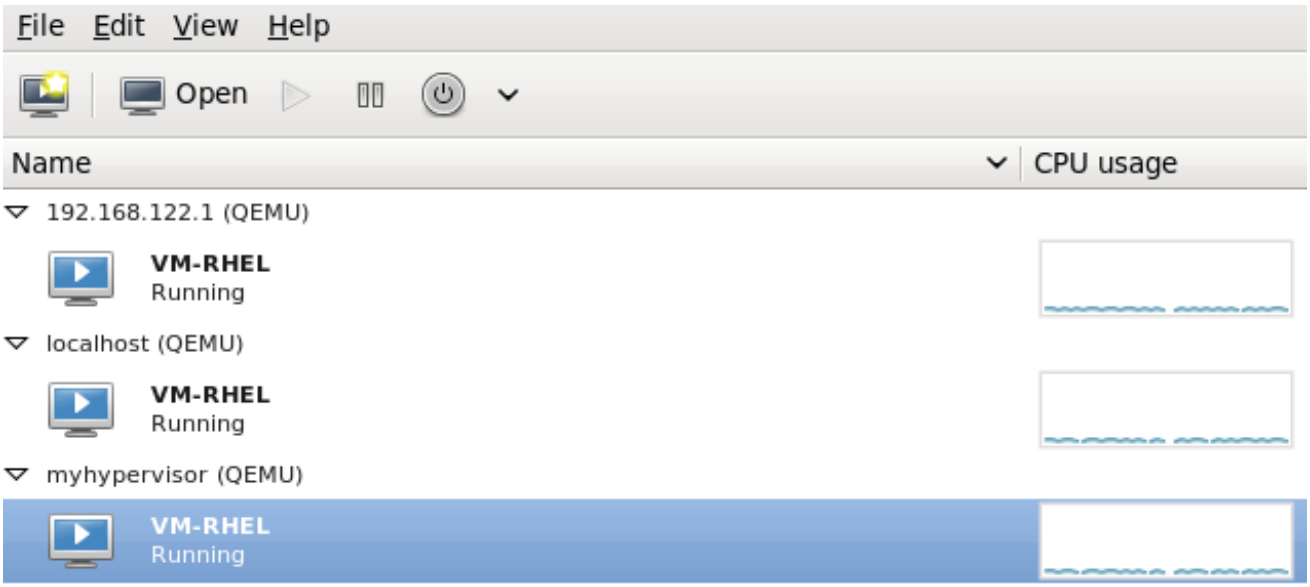
   図15.8 接続の追加



3. プロンプトが表示されたら、選択したホストの root パスワードを入力します。

これで、リモートホストコンピューターが接続され、メイン**virt-manager** ウィンドウに表示されるようになります。

図15.9 メインの virt-manager ウィンドウのリモートホスト



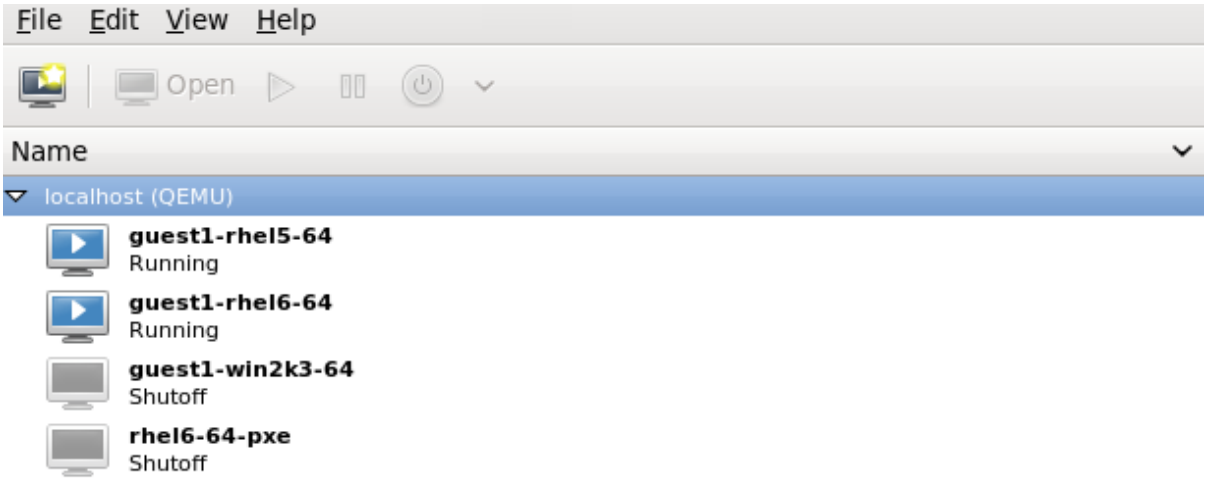
15.6. ゲストの詳細の表示

Virtual Machine Monitor を使用すると、システム上の仮想マシンのアクティビティー情報を表示できます。

仮想システムの詳細を表示するには、次のコマンドを実行します。

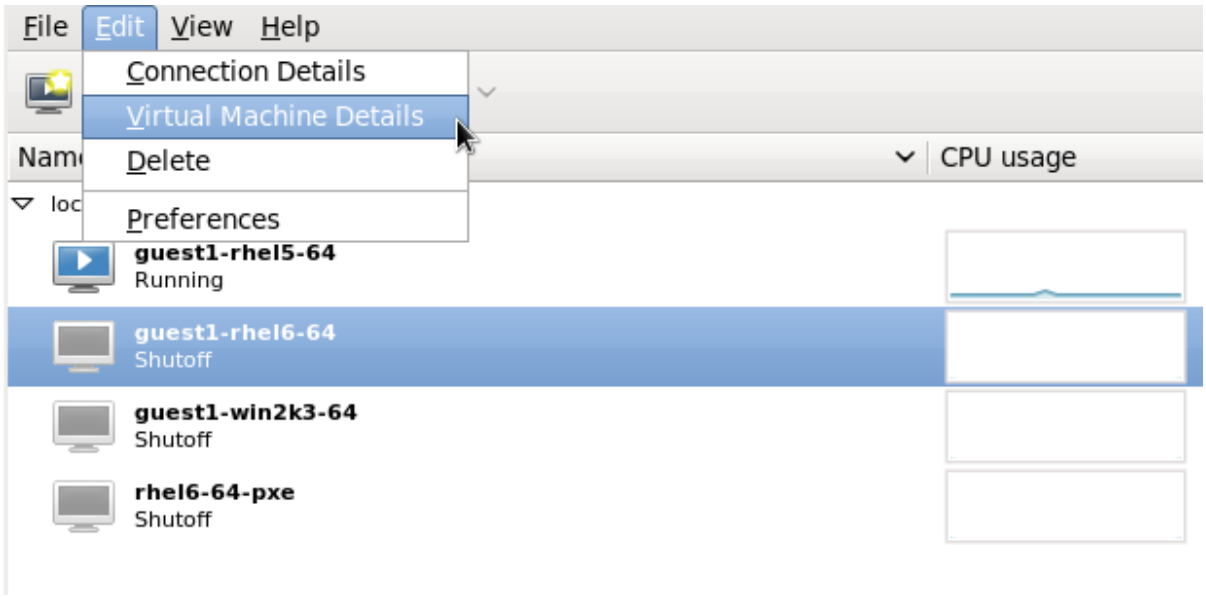
1. Virtual Machine Manager のメインウィンドウで、表示する仮想マシンを強調表示します。

   図15.10 表示する仮想マシンの選択

2. 仮想マシンマネージャーの**Edit** メニューから、**Virtual Machine Details** を選択します。

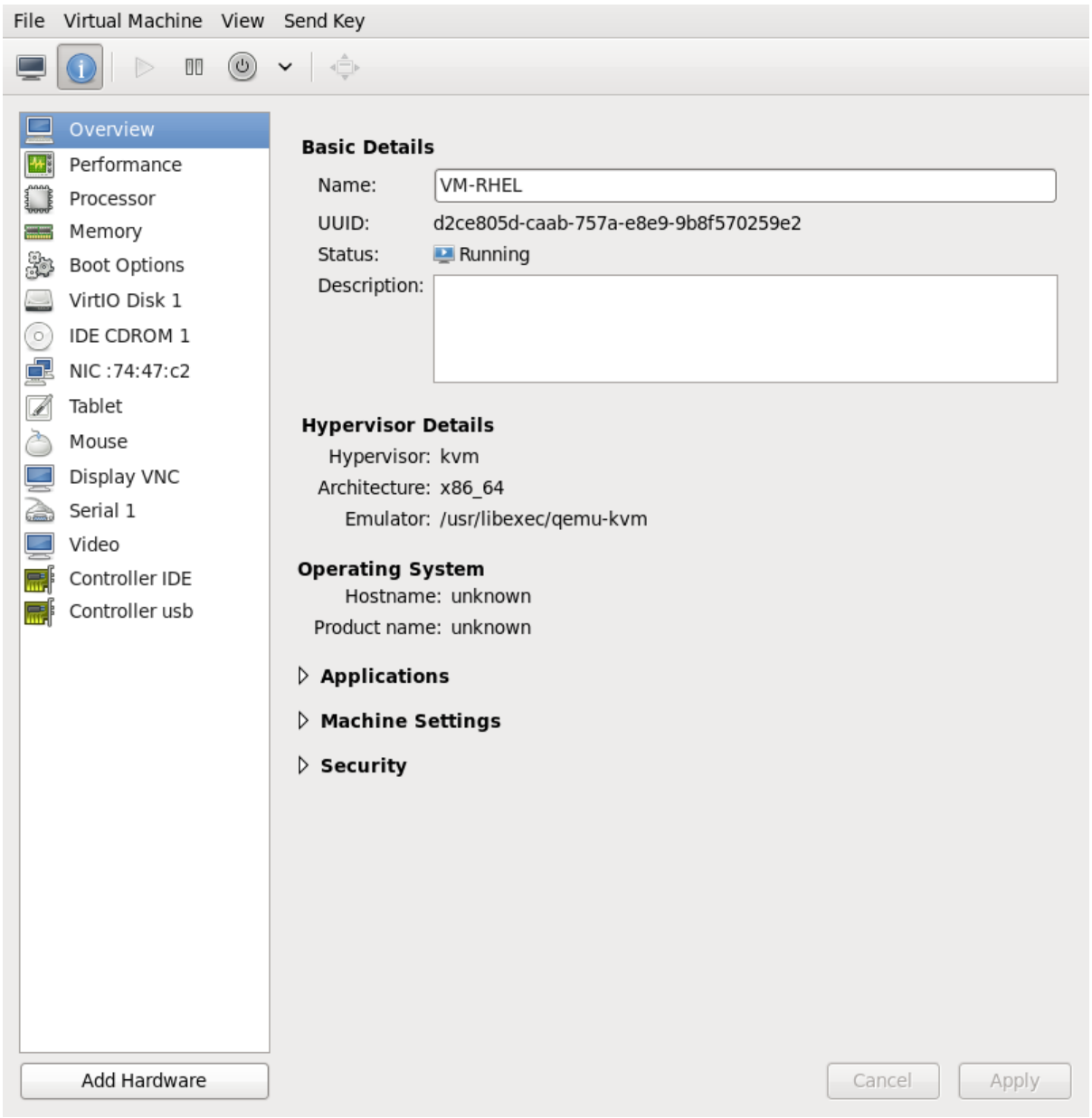
図15.11 仮想マシンの詳細の表示



仮想マシンの詳細ウィンドウが開くと、コンソールが表示される場合があります。これが発生した場合は、**View** をクリックし、**Details** を選択します。デフォルトでは、Overview ウィンドウが最初に開きます。このウィンドウに戻るには、左側のナビゲーションペインから **Overview** を選択します。

Overview ビューには、ゲストの設定詳細の概要が表示されます。

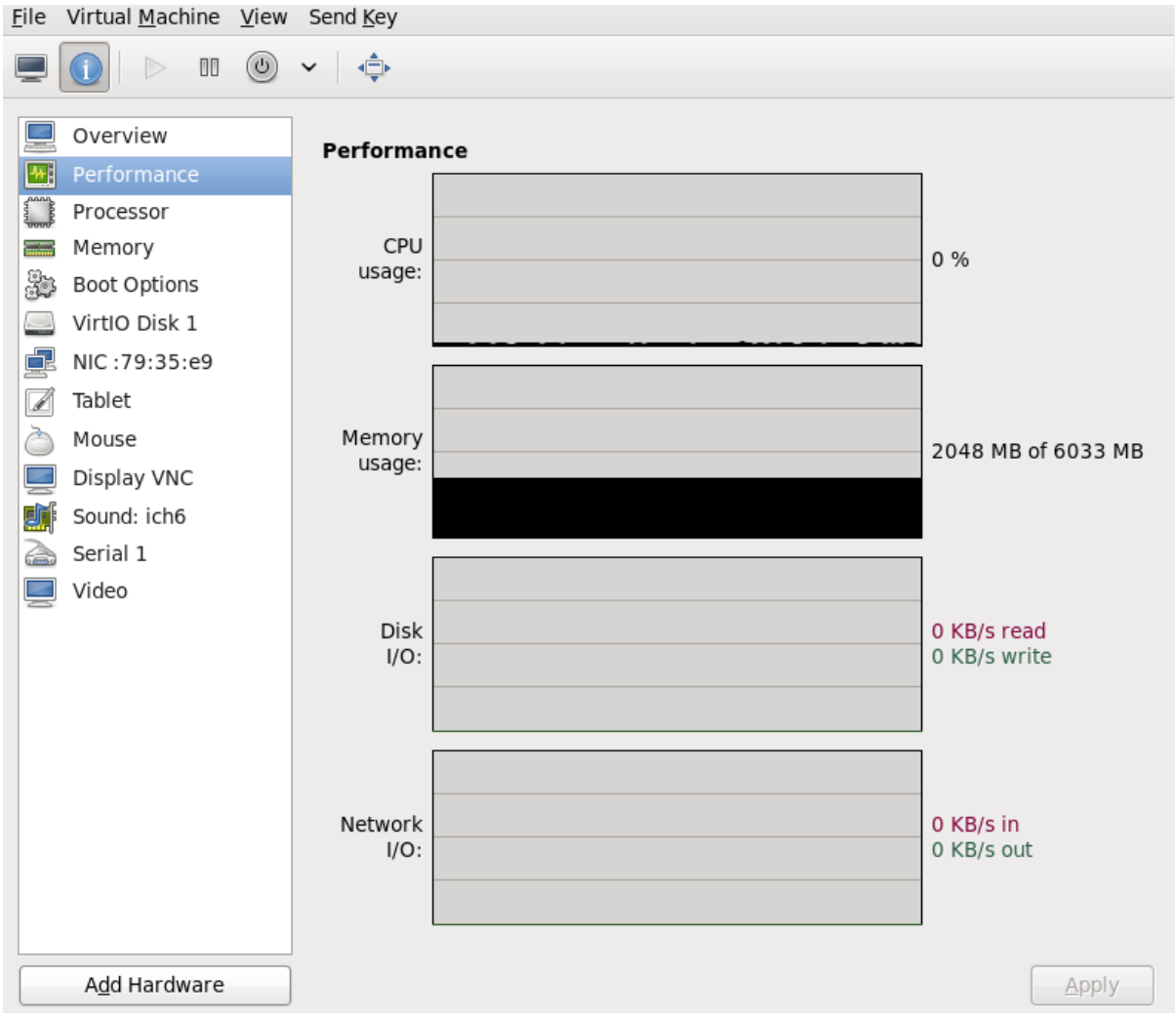
図15.12 ゲストの詳細の概要の表示



3. 左側のナビゲーションペインから **Performance** を選択します。

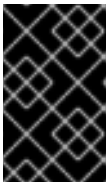
   Performance ビューには、CPU とメモリーの使用量など、ゲストのパフォーマンスの概要が表示されます。

図15.13 ゲストのパフォーマンスの詳細の表示



4. 左側のナビゲーションペインから **Processor** を選択します。**Processor** ビューでは、現在のプロセッサー割り当てを表示したり、変更したりできます。

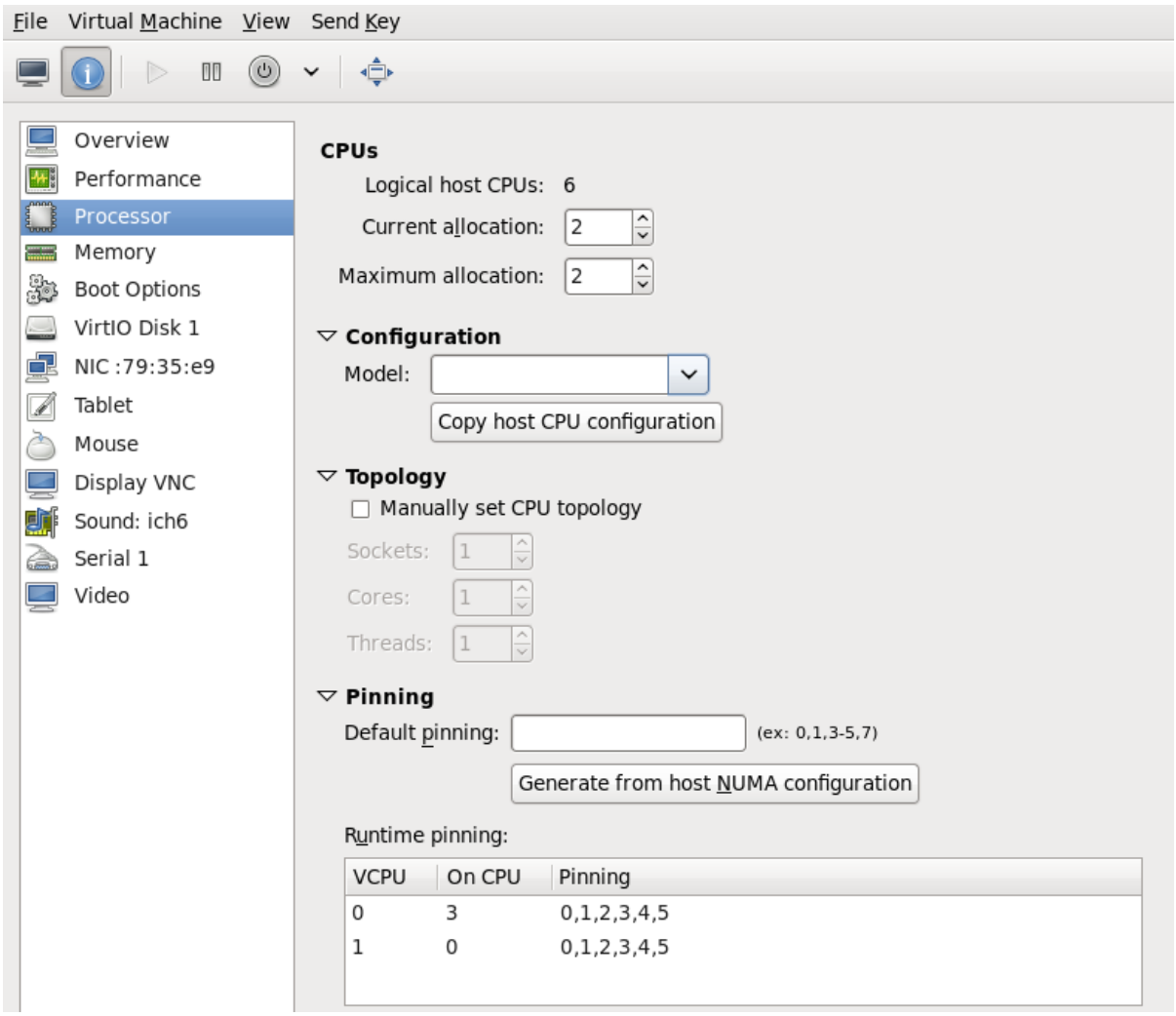
   仮想マシンの実行中に仮想 CPU(vCPU) の数を変更することもできます。これは、*ホットプラグ*および *ホットアンプラグ*と呼ばれます。



重要

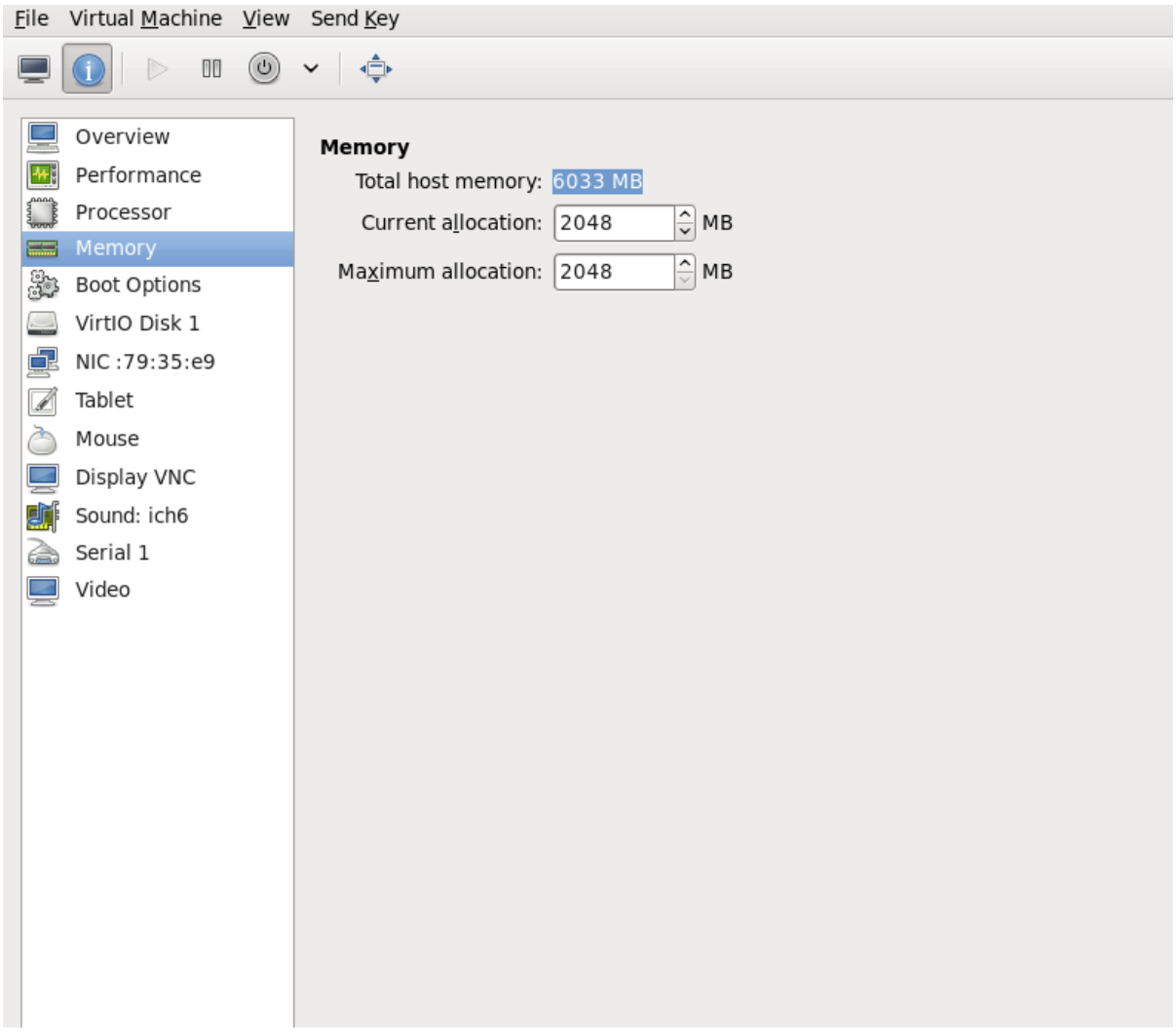
ホットアンプラグ機能は、テクノロジープレビューとしてのみ使用できます。したがって、これはサポートされておらず、高い値のデプロイメントでの使用は推奨されません。

図15.14 プロセッサー割り当てパネル



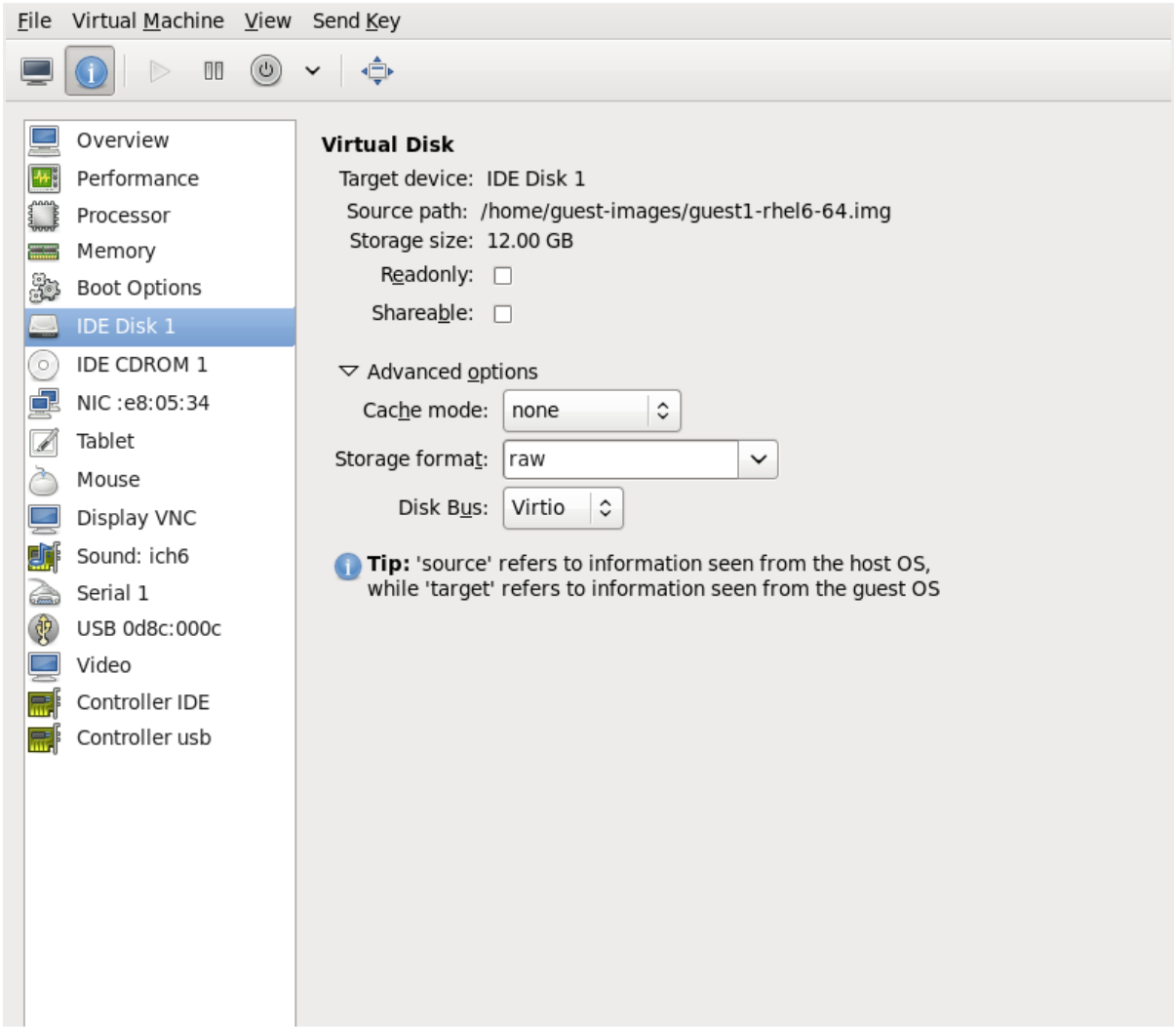
5. 左側のナビゲーションペインから **Memory** を選択します。**Memory** ビューでは、現在のメモリー割り当てを表示または変更できます。

図15.15 メモリー割り当ての表示



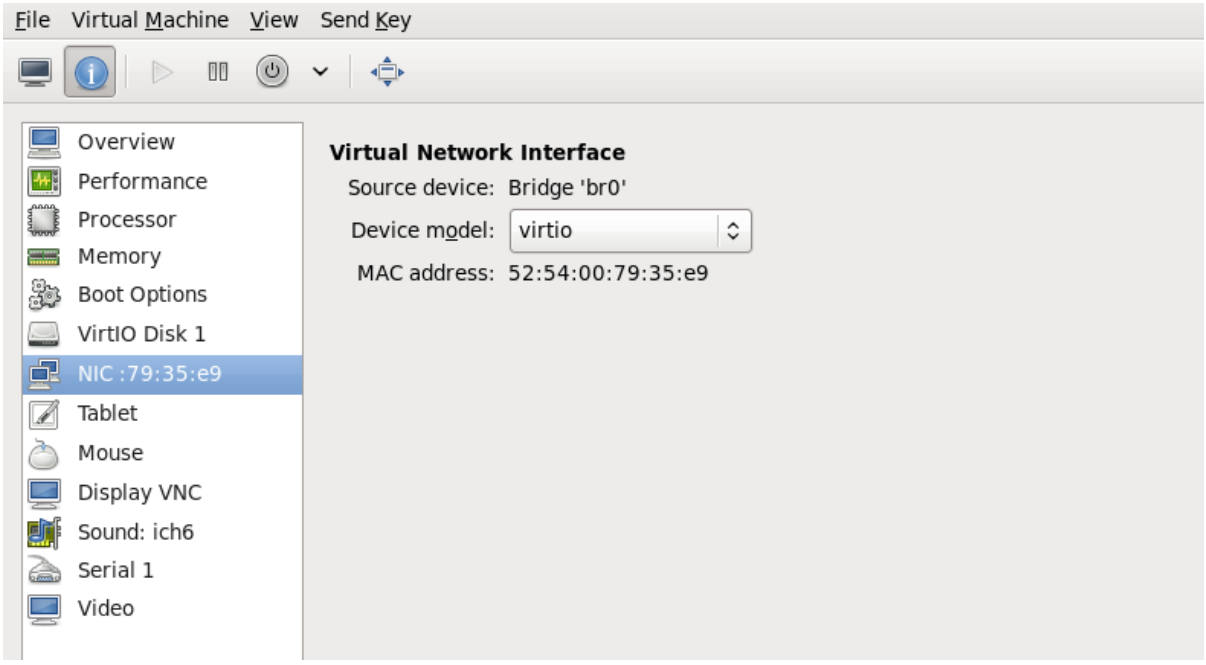
6. 仮想マシンに接続されている各仮想ディスクがナビゲーションペインに表示されます。仮想ディスクをクリックして、変更または削除します。

図15.16 ディスク設定の表示



7. 仮想マシンに接続されている各仮想ネットワークインターフェイスがナビゲーションペインに表示されます。仮想ネットワークインターフェイスをクリックして、変更または削除します。

図15.17 ネットワーク設定の表示



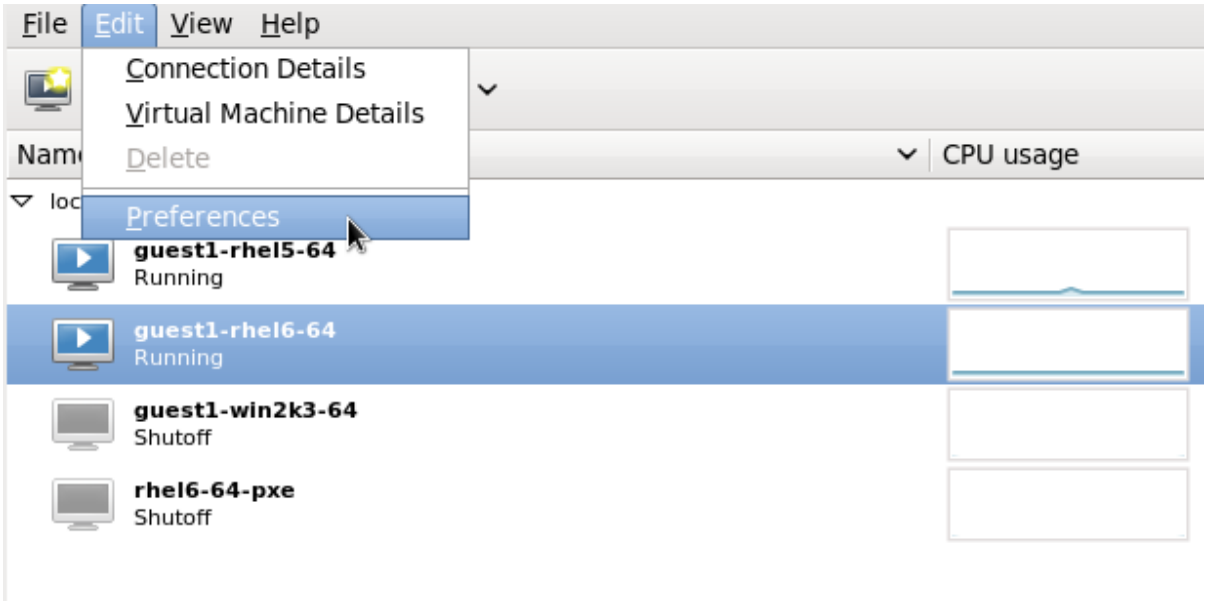
15.7. パフォーマンスのモニターリング

パフォーマンスモニターリングの設定は、**virt-manager** の設定ウィンドウで変更できます。

パフォーマンスモニターリングを設定するには、以下を実行します。

1. **Edit** メニューから、**Preferences** を選択します。

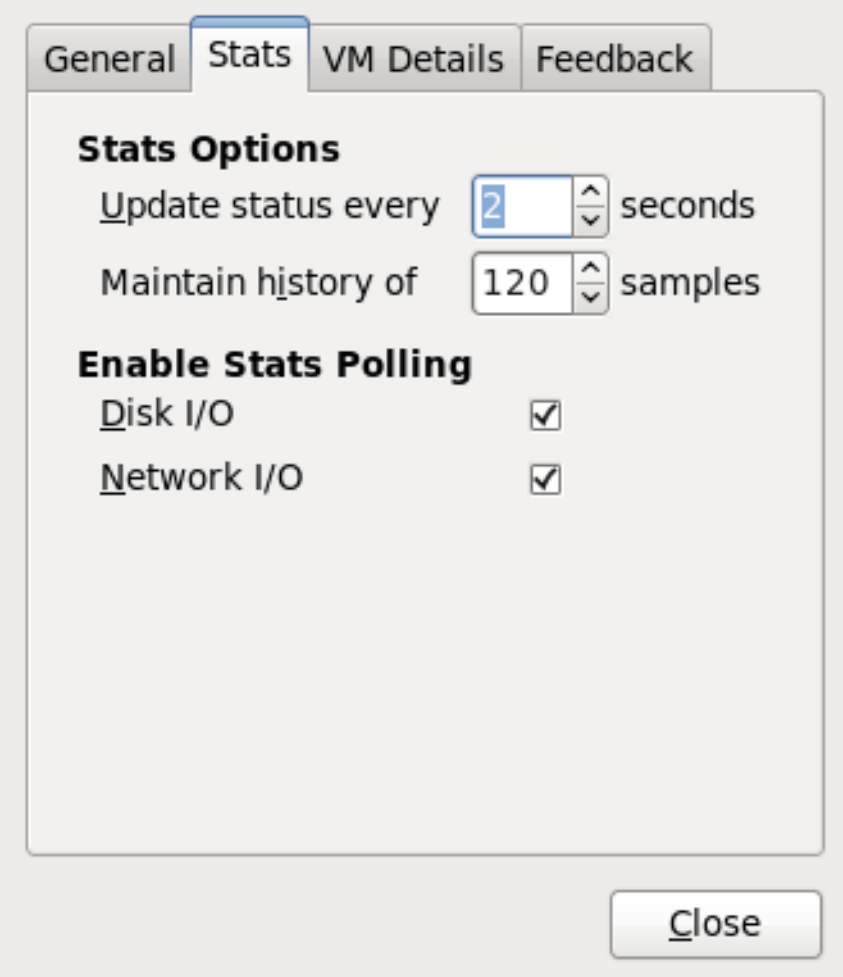
   図15.18 ゲスト設定の変更

   Preferences ウィンドウが表示されます。

2. **Stats** タブから、時間を秒単位または統計ポーリングオプションで指定します。

図15.19 パフォーマンスモニターリングの設定

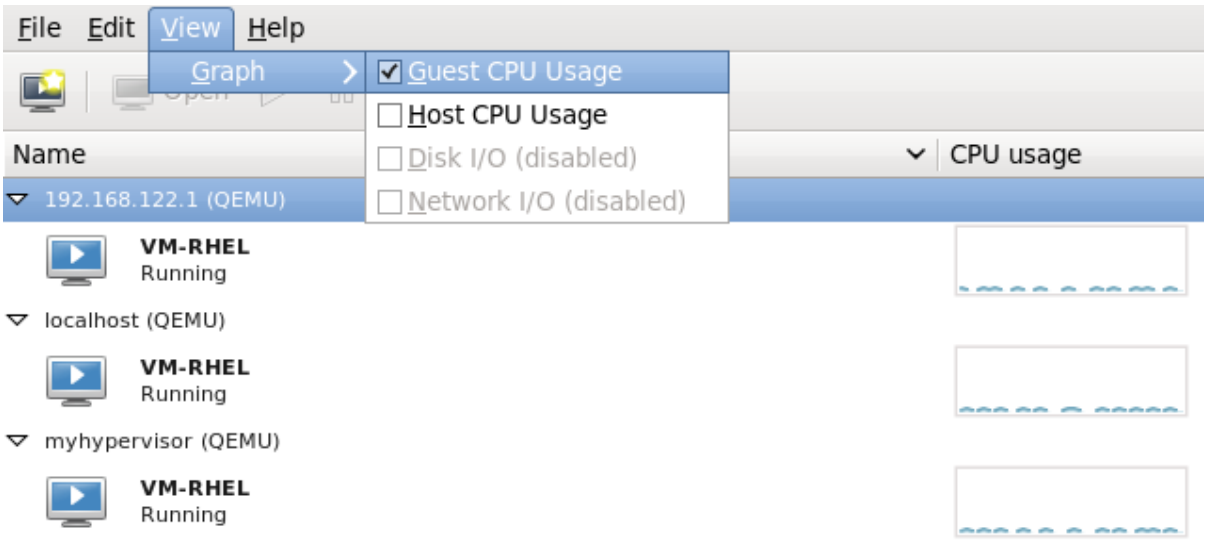


15.8. ゲストの CPU 使用率の表示

システム上のすべてのゲストの CPU 使用率を表示するには、以下を実行します。

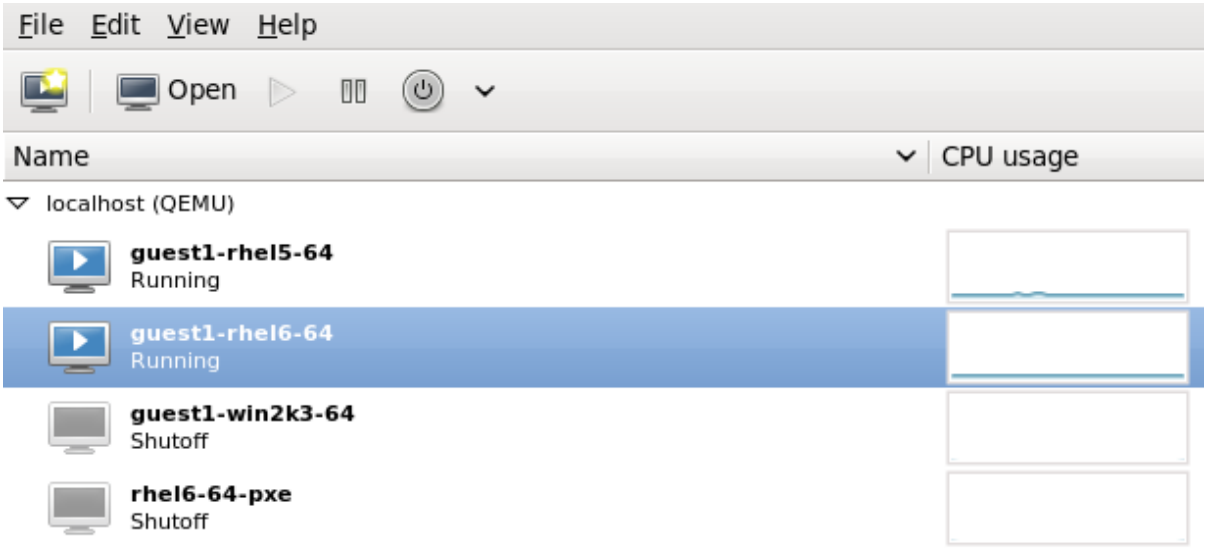
1. **View** メニューから **Graph** を選択し、**Guest CPU Usage** チェックボックスをオンにします。

図15.20 ゲスト CPU 使用率統計情報グラフの有効化



2. Virtual Machine Manager は、システム上のすべての仮想マシンの CPU 使用率グラフを表示します。

図15.21 ゲスト CPU 使用率グラフ

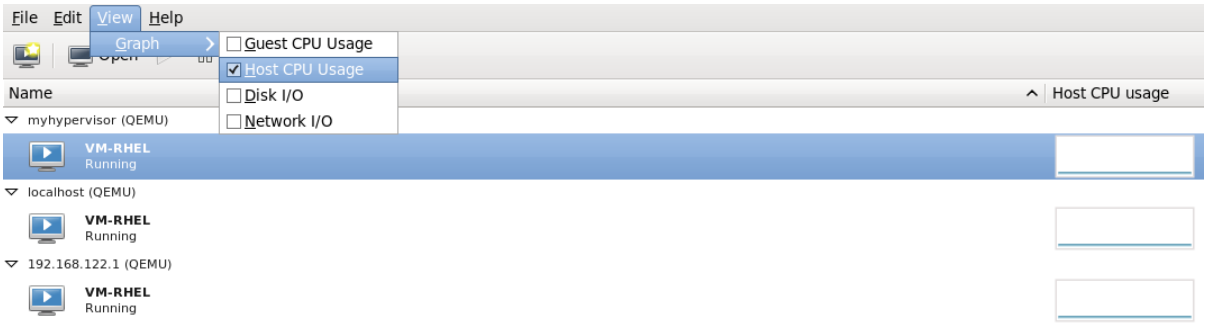


15.9. ホストの CPU 使用率の表示

システム上のすべてのホストの CPU 使用率を表示するには、以下を実行します。

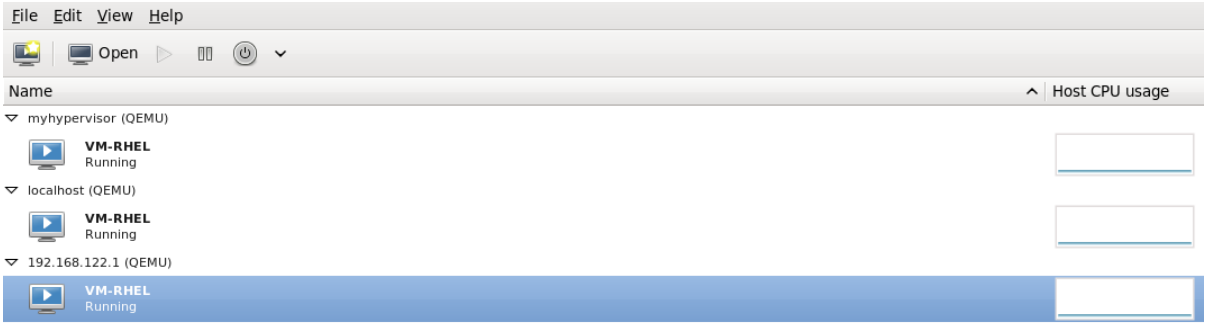
1. **View** メニューから **Graph** を選択し、**Host CPU Usage** チェックボックスをオンにします。

図15.22 ホスト CPU 使用率統計情報グラフの有効化



2. Virtual Machine Manager は、システム上のホストの CPU 使用率グラフを表示します。

図15.23 ホストの CPU 使用率グラフ

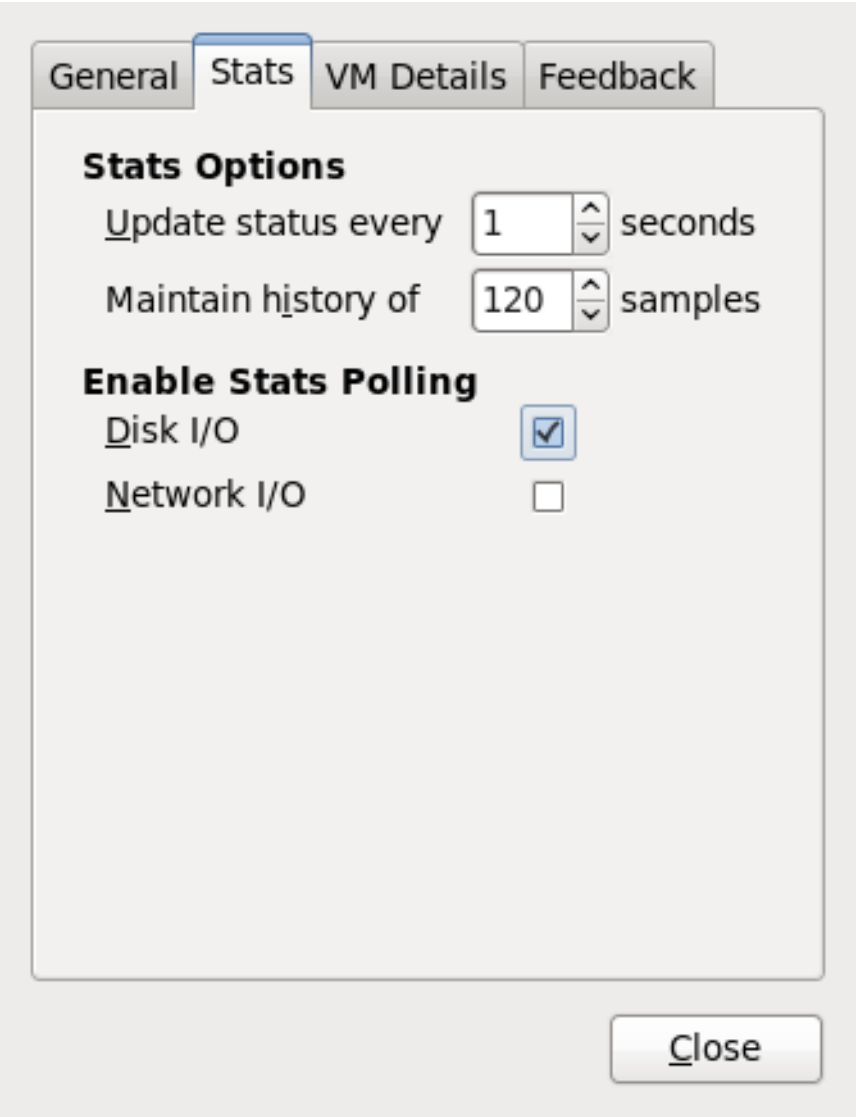


15.10. ディスク I/O の表示

システム上のすべての仮想マシンのディスク I/O を表示するには、以下を実行します。

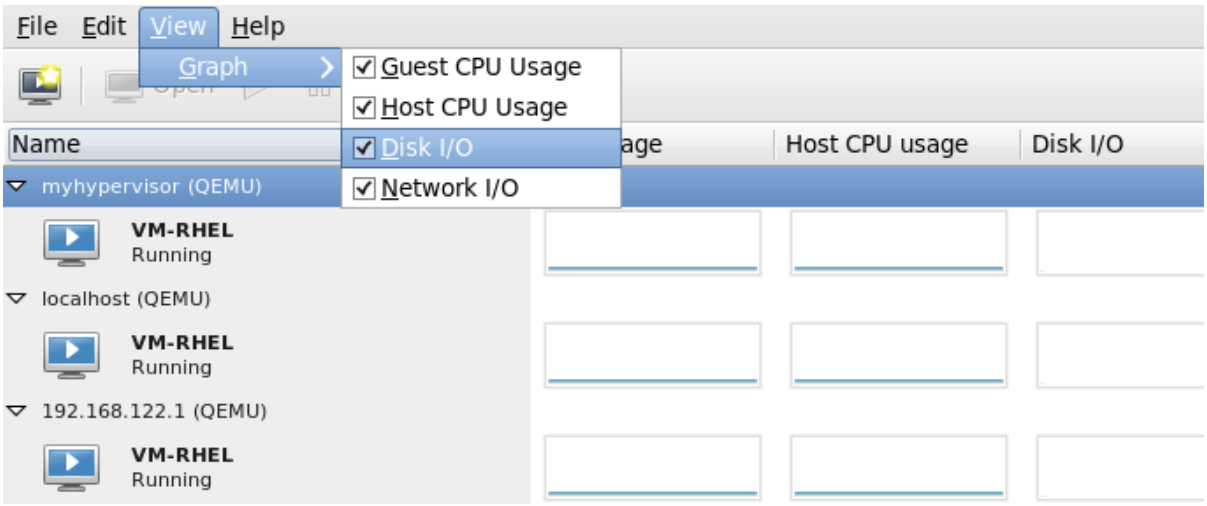
1. ディスク I/O 統計情報の収集が有効になっていることを確認してください。これを行うには、**Edit** メニューから **Preferences** を選択し、**Stats** タブをクリックします。
2. **Disk I/O** チェックボックスをオンにします。

   図15.24 ディスク I/O の有効化



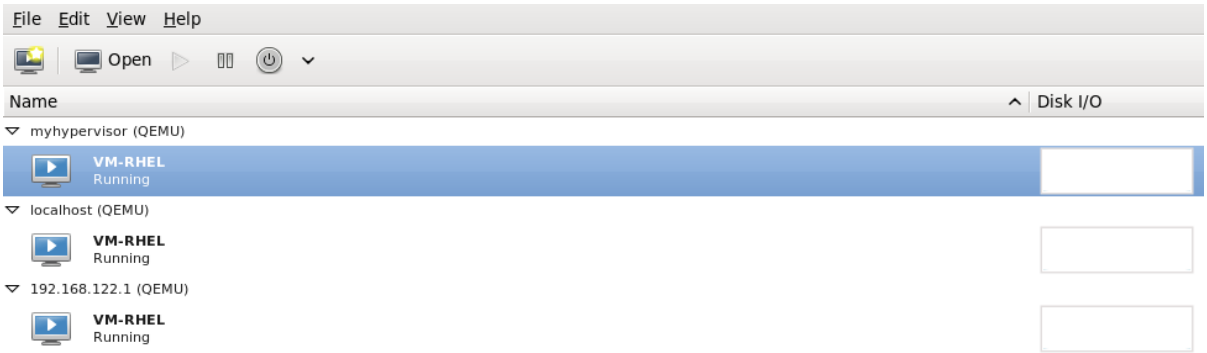
3. ディスク I.O 表示を有効にするには、**View** メニューから **Graph** を選択し、**Disk I/O** チェックボックスをオンにします。選択します。

図15.25 ディスク I/O の選択



4. Virtual Machine Manager は、システム上のすべての仮想マシンのディスク I/O のグラフを表示します。

図15.26 ディスク I/O の表示



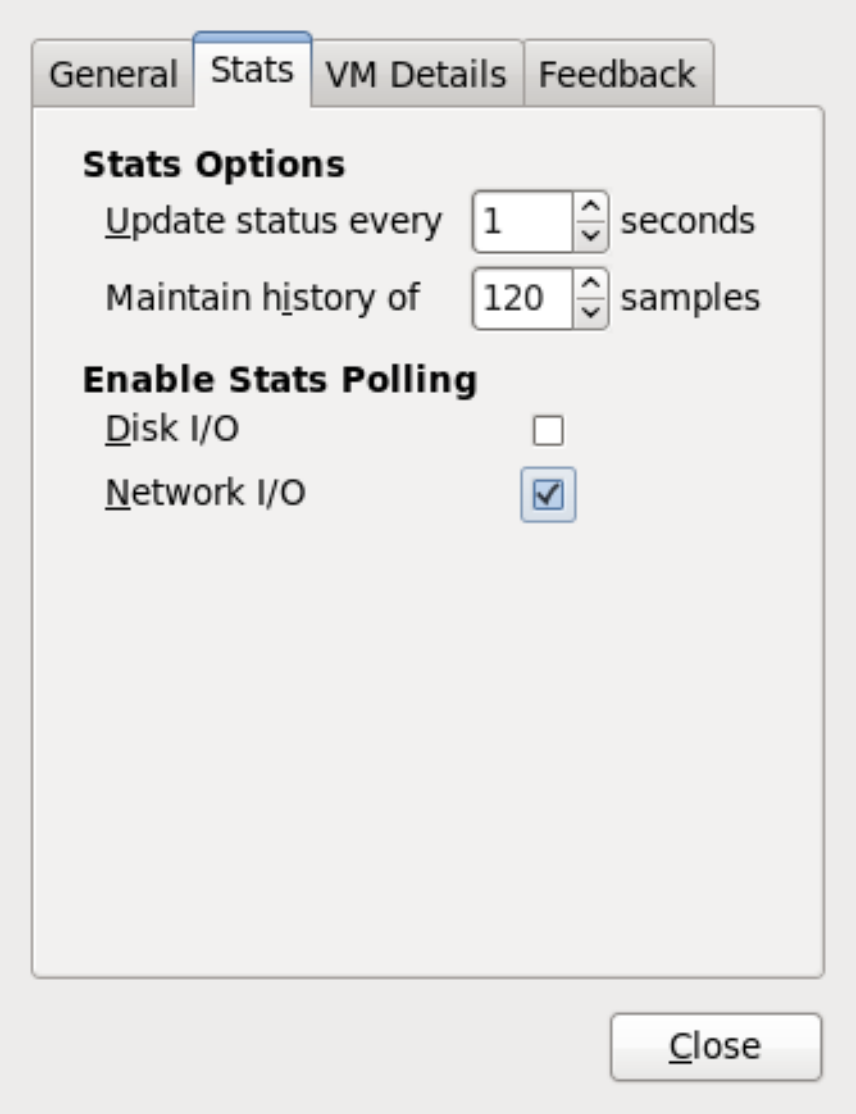
15.11. ネットワーク I/O の表示

システム上のすべての仮想マシンのネットワーク I/O を表示するには、以下を実行します。

1. ネットワーク I/O 統計情報の収集が有効になっていることを確認してください。これを行うには、**Edit** メニューから **Preferences** を選択し、**Stats** タブをクリックします。

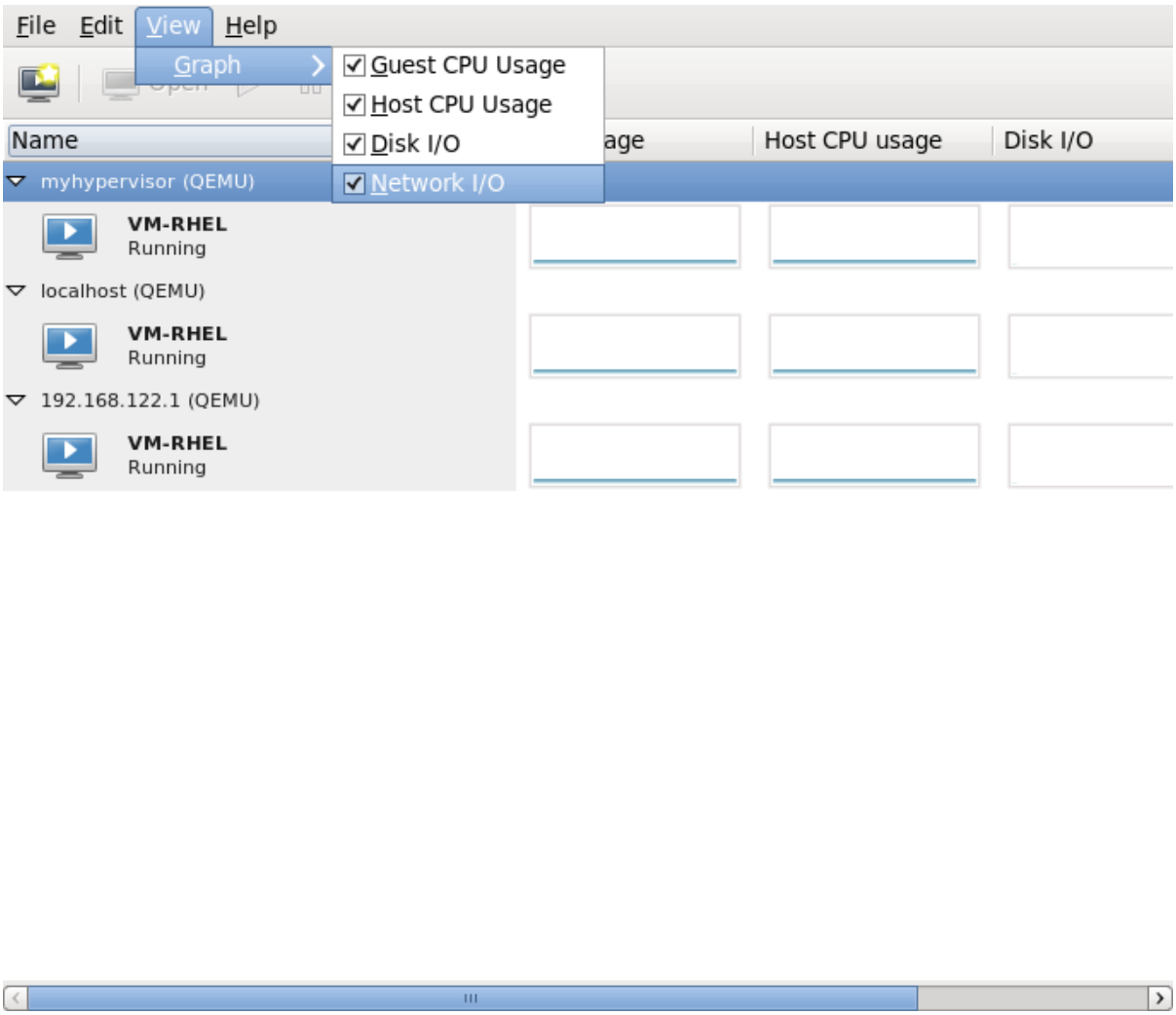
2. **Network I/O** チェックボックスをオンにします。

図15.27 ネットワーク I/O の有効化



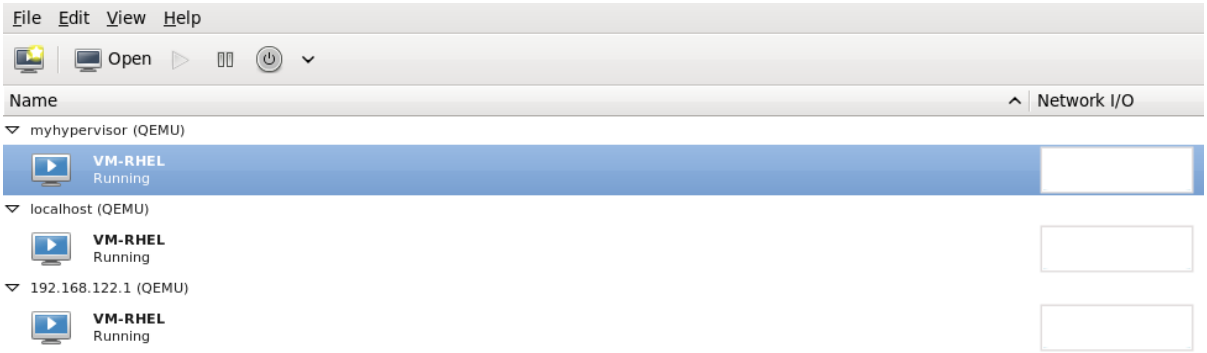
3. ネットワーク I/O 統計情報を表示するには、**View** メニューから **Graph** を選択し、**Network I/O** チェックボックスをオンにします。

図15.28 ネットワーク I/O の選択



4. Virtual Machine Manager は、システム上のすべての仮想マシンのネットワーク I/O グラフを表示します。

図15.29 ネットワーク I/O の表示



第16章 オフラインツールを使用したゲスト仮想マシンのディスクアクセス

16.1. はじめに

Red Hat Enterprise Linux 6 には、ホストの物理マシンディスクまたはその他のディスクイメージにアクセス、編集、および作成するためのツールが付属しています。これらのツールには、次のようないくつかの用途があります。

- ホストの物理マシンディスクにあるファイルの表示またはダウンロード。
- ファイルを編集するか、ホストの物理マシンディスクにアップロードします。
- ホストの物理マシン設定の読み取りまたは書き込み。
- Windows ホスト物理マシンでの Windows レジストリーの読み取りまたは書き込み。
- ファイル、ディレクトリー、ファイルシステム、パーティション、論理ボリューム、およびその他のオプションを含む新しいディスクイメージの準備
- 起動に失敗した、または起動設定の変更が必要なホスト物理マシンのレスキューと修復。
- ホスト物理マシンのディスク使用量をモニターします。
- 組織のセキュリティー規格など、ホストの物理マシンのコンプライアンスを監査します。
- テンプレートのクローンを作成して変更することにより、ホスト物理マシンをデプロイします。
- CD および DVD ISO とフロッピーディスクイメージの読み取り。

警告

これらのツールを使用して、実行中の仮想マシンに接続されているホスト物理マシンまたはディスクイメージに書き込んだり、書き込みモードでそのようなディスクイメージを開いたりし **ない**でください。実行すると、ゲスト仮想マシンのディスクが破損します。ツールはこれを阻止しようとしますが、すべてのケースを捕らえるわけではありません。ゲスト仮想マシンが実行されている疑いがある場合は、ツールを使用しないか、少なくとも **常に**読み取り専用モードでツールを使用することを強くお勧めします。

注記

Red Hat Enterprise Linux 6 の仮想化コマンドの中には、リモートの libvirt 接続を指定できるものがあります。以下に例を示します。

```
virt-df -c qemu://remote/system -d Guest
```

ただし、Red Hat Enterprise Linux 6 の libguestfs はリモートゲストにアクセスできず、このようなリモート URL を使用するコマンドは期待どおりに機能しません。これは、次の Red Hat Enterprise Linux 6 コマンドに影響します。

- guestfish
- guestmount
- virt-alignment-scan
- virt-cat
- virt-copy-in
- virt-copy-out
- virt-df
- virt-edit
- virt-filesystems
- virt-inspector
- virt-inspector2
- virt-list-filesystems
- virt-list-partitions
- virt-ls
- virt-rescue
- virt-sysprep
- virt-tar
- virt-tar-in
- virt-tar-out
- virt-win-reg

16.2. 用語

本セクションでは、本章全体で使用される用語を説明します。

- **libguestfs (Guest file system library)**- ディスクイメージを開いたり、ファイルを読み書きしたりするために使用される基本的な機能を提供する C ライブラリーです。この API に直接 C プログラムを書くことができますが、それはかなり低レベルです。

- **guestfish (Guest file system interactive shell)**は、コマンドラインまたはシェルスクリプトから使用できる対話式のシェルです。これは、libguestfs API のすべての機能を公開します。

- さまざまな virt ツールが libguestfs の上に構築され、コマンドラインから特定の 1 つのタスクを実行する方法を提供します。ツールには、**virt-df**、**virt-rescue**、**virt-resize**、および **virt-edit** が含まれます。

- **hivex** と **Augeas** は、それぞれ Windows レジストリーと Linux 設定ファイルを編集するためのライブラリーです。これらは libguestfs とは別のものですが、libguestfs の価値の多くは、これらのツールの組み合わせに由来します。

- **guestmount** は、libguestfs と FUSE との間のインターフェイスです。これは、主に、ホストの物理マシンのディスクイメージからファイルシステムをマウントするために使用されます。この機能は必須ではありませんが、便利です。

16.3. インストールシステム

libguestfs、guestfish、libguestfs ツール、guestmount、および Windows ゲスト仮想マシンのサポートをインストールするには、Red Hat Enterprise Linux V2WIN チャネルにサブスクライブし、Red Hat Web サイト にアクセスして、次のコマンドを実行します。

```
# yum install libguestfs guestfish libguestfs-tools libguestfs-winsupport
```

言語バインディングを含むすべての libguestfs 関連パッケージをインストールするには、以下のコマンドを実行します。

```
# yum install '*guestf*'
```

16.4. GUESTFISH シェル

guestfish は、コマンドラインまたはシェルスクリプトから、ゲスト仮想マシンファイルシステムにアクセスするために使用できるインタラクティブなシェルです。libguestfs API の機能はすべて、シェルから利用できます。

仮想マシンのディスクイメージの表示または編集を開始するには、次のコマンドを実行して、目的のディスクイメージへのパスを置き換えます。

```
guestfish --ro -a /path/to/disk/image
```

--ro は、ディスクイメージが読み取り専用で開かれていることを意味します。このモードは常に安全ですが、書き込みアクセスは許可しません。ゲスト仮想マシンが実行されていないことが**確実**な場合、またはディスクイメージがライブゲスト仮想マシンに接続されていない場合にのみ、このオプションを省略してください。**libguestfs** を使用してライブゲスト仮想マシンを編集することはできません。これを試みると、ディスクが破損し、元に戻せなくなります。

/path/to/disk/image は、ディスクへのパスです。これは、ファイル、ホスト物理マシン論理ボリューム (/dev/VG/LV など)、ホスト物理マシンデバイス (/dev/cdrom)、または SAN LUN(/dev/sdf3) の場合があります。

注記

libguestfs および guestfish には root 権限は必要ありません。アクセスされているディスクイメージが root から読み取りまたは書き込み、あるいは両方を必要とする場合にのみ、これらを root として実行する必要があります。

guestfish を対話的に起動すると、以下のプロンプトが表示されます。

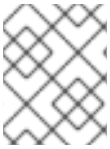
```
 guestfish --ro -a /path/to/disk/image

Welcome to guestfish, the libguestfs filesystem interactive shell for editing virtual machine
filesystems.

 Type: 'help' for help on commands
       'man' to read the manual
       'quit' to quit the shell

><fs>
```

プロンプトで **実行** と入力して、ライブラリーを開始し、ディスクイメージを割り当てます。これは、最初に実行するときに最大 30 秒かかることがあります。その後の起動は、非常に速く完了します。



注記

libguestfs は、KVM (利用可能な場合) などのハードウェア仮想化アクセラレーションを使用して、このプロセスを高速化します。

run コマンドを入力すると、以下のセクションで示すように、他のコマンドを使用できます。

16.4.1. guestfish を使用したファイルシステムの表示

このセクションでは、guestfish でファイルを表示する方法について説明します。

16.4.1.1. 手動による一覧表示と表示

list-filesystems コマンドは、libguestfs が検出したファイルシステムの一覧を表示します。この出力は、Red Hat Enterprise Linux 4 ディスクイメージを示しています。

```
><fs> run
><fs> list-filesystems
/dev/vda1: ext3
/dev/VolGroup00/LogVol00: ext3
/dev/VolGroup00/LogVol01: swap
```

この出力は、Windows ディスクイメージを示しています。

```
><fs> run
><fs> list-filesystems
/dev/vda1: ntfs
/dev/vda2: ntfs
```

その他の便利なコマンドは、**list-devices**、**list-partitions**、**lvs**、**pvs**、**vfs-type**、および **file** です。**help command** と入力すると、以下の出力のように、コマンドの詳細とヘルプを表示できます。

```
><fs> help vfs-type
 NAME
   vfs-type - get the Linux VFS type corresponding to a mounted device

 SYNOPSIS
    vfs-type device

 DESCRIPTION
   This command gets the file system type corresponding to the file system on
   "device".

   For most file systems, the result is the name of the Linux VFS module
   which would be used to mount this file system if you mounted it without
   specifying the file system type. For example a string such as "ext3" or
   "ntfs".
```

ファイルシステムの実際の内容を表示するには、最初にマウントする必要があります。この例では、前の出力 (/**dev**/**vda2**) に示されている Windows パーティションの 1 つを使用します。この場合、これは **C:**ドライブに対応することがわかっています。

```
><fs> mount-ro /dev/vda2 /
><fs> ll /
total 1834753
 drwxrwxrwx  1 root root       4096 Nov  1 11:40 .
 drwxr-xr-x 21 root root       4096 Nov 16 21:45 ..
 lrwxrwxrwx  2 root root         60 Jul 14  2009 Documents and Settings
 drwxrwxrwx  1 root root       4096 Nov 15 18:00 Program Files
 drwxrwxrwx  1 root root       4096 Sep 19 10:34 Users
 drwxrwxrwx  1 root root      16384 Sep 19 10:34 Windows
```

ls、**ll**、**cat**、その**more**、**download**、**tar-out** などの guestfish コマンドを使用して、ファイルおよびディレクトリーを表示およびダウンロードできます。

注記

このシェルには、現在の作業ディレクトリーの概念はありません。通常のシェルとは異なり、**cd** コマンドを使用してディレクトリーを変更することはできません。すべてのパスは、先頭のスラッシュ (**/**) 文字で始まる完全修飾する必要があります。**Tab** 鍵を使用して、パスを完成させます。

guestfish シェルを終了するには、**exit** と入力するか、**Ctrl+d** と入力します。

16.4.1.2. guestfish 検査の使用

ファイルシステムを手動で一覧表示してマウントする代わりに、guestfish 自体がイメージを検証して、ゲスト仮想マシンのようにファイルシステムをマウントできます。これを行うには、コマンドラインに **-i** を追加します。

```
guestfish --ro -a /path/to/disk/image -i

Welcome to guestfish, the libguestfs filesystem interactive shell for
 editing virtual machine filesystems.

 Type: 'help' for help on commands
```

```
      'man' to read the manual
      'quit' to quit the shell

Operating system: Red Hat Enterprise Linux AS release 4 (Nahant Update 8)
/dev/VolGroup00/LogVol00 mounted on /
/dev/vda1 mounted on /boot

><fs> ll /
total 210
drwxr-xr-x. 24 root root  4096 Oct 28 09:09 .
drwxr-xr-x  21 root root  4096 Nov 17 15:10 ..
drwxr-xr-x.  2 root root  4096 Oct 27 22:37 bin
drwxr-xr-x.  4 root root  1024 Oct 27 21:52 boot
drwxr-xr-x.  4 root root  4096 Oct 27 21:21 dev
drwxr-xr-x. 86 root root 12288 Oct 28 09:09 etc
[etc]
```

インスペクションおよびマウントを実行するために、guestfish は libguestfs バックエンドを起動する必要があるため、**-i** オプションを使用する場合は **run** コマンドを使用する必要はありません。**-i** オプションは、多くの一般的な Linux および Windows ゲスト仮想マシンで機能します。

16.4.1.3. 名前によるゲスト仮想マシンのアクセス

ゲスト仮想マシンは、libvirt として知られる名前 (つまり **virsh list --all** にあるように) を指定するとコマンドラインからアクセスできます。**-d** オプションを使用して、ゲスト仮想マシンに名前 (**-i** オプションの有無にかかわらず) でアクセスします。

```
guestfish --ro -d GuestName -i
```

16.4.2. guestfish を使用したファイルの変更

ファイルを変更したり、ディレクトリーを作成したり、ゲスト仮想マシンにその他の変更を加えたりする場合は、最初に本セクションの冒頭にある ゲスト仮想マシンをシャットダウンする必要がある に関する警告に留意してください。guestfish で実行中のディスクを編集または変更すると、ディスクが **破損します**。ここでは、**/boot/grub/grub.conf** ファイルーを編集する例を説明します。ゲスト仮想マシンがシャットダウンしていることを確認したら、以下のようなコマンドを使用して書き込みアクセスを取得するために **--ro** オプションを省略できます。

```
guestfish -d RHEL3 -i

Welcome to guestfish, the libguestfs filesystem interactive shell for
 editing virtual machine filesystems.

 Type: 'help' for help on commands
       'man' to read the manual
       'quit' to quit the shell

 Operating system: Red Hat Enterprise Linux AS release 3 (Taroon Update 9)
 /dev/vda2 mounted on /
 /dev/vda1 mounted on /boot

><fs> edit /boot/grub/grub.conf
```

ファイルを編集するコマンドには、**edit**、**vi**、および **emacs** が含まれます。**write**、**mkdir**、**upload**、**tar-in** など、ファイルとディレクトリーを作成する多くのコマンドも存在します。

16.4.3. guestfish でのその他のアクション

また、**mkfs**、**part-add**、**lvresize**、**lvcreate**、**vgcreate**、**pvcreate** などのコマンドを使用して、ファイルシステムのフォーマットを設定し、パーティションを作成し、LVM 論理ボリュームなどのサイズを変更できます。

16.4.4. guestfish によるシェルスクリプト設定

guestfish を対話的に使用することに慣れたら、必要に応じてシェルスクリプトの記述が役に立つ場合があります。以下は、新しい MOTD (message of the day) をゲストに追加する単純なシェルスクリプトです。

```
#!/bin/bash -
 set -e
 guestname="$1"

 guestfish -d "$guestname" -i <<'EOF'
   write /etc/motd "Welcome to Acme Incorporated."
   chmod 0644 /etc/motd
 EOF
```

16.4.5. Augeas スクリプトと libguestfs スクリプト

libguestfs と Augeas を組み合わせると、Linux ゲスト仮想マシンの設定を操作するスクリプトを作成する場合に役立ちます。たとえば、次のスクリプトは、Augeas を使用してゲスト仮想マシンのキーボード設定を解析し、レイアウトを出力します。この例は、Red Hat Enterprise Linux を実行しているゲスト仮想マシンでのみ機能することに注意してください。

```
#!/bin/bash -
 set -e
 guestname="$1"

 guestfish -d "$1" -i --ro <<'EOF'
   aug-init / 0
   aug-get /files/etc/sysconfig/keyboard/LAYOUT
 EOF
```

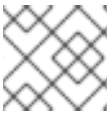
Augeas は、設定ファイルの変更にも使用できます。上記のスクリプトを修正して、キーボードレイアウトを**変更**できます。

```
#!/bin/bash -
 set -e
 guestname="$1"

 guestfish -d "$1" -i <<'EOF'
   aug-init / 0
   aug-set /files/etc/sysconfig/keyboard/LAYOUT '"gb"'
   aug-save
 EOF
```

2 つのスクリプト間で、以下の 3 つの変更が行われたことに注意してください。

1. 2 番目の例では **--ro** オプションが削除され、ゲスト仮想マシンに書き込む機能が利用できるようになりました。
2. **aug-get** コマンドが、フェッチではなく値を変更するように **aug-set** に変更されました。新しい値は**"gb"** (引用符を含む) になります。
3. ここでは **aug-save** コマンドを使用しているため、Augeas は変更をディスクに書き込みます。



注記

Augeas の詳細は、Web サイト http://augeas.net を参照してください。

guestfish には、本書で説明する以上の機能があります。たとえば、最初からディスクイメージを作成する場合は、次のコマンドを実行します。

```
guestfish -N fs
```

または、ディスクイメージからディレクトリー全体をコピーする場合は、次のコマンドを実行します。

```
><fs> copy-out /home /tmp/home
```

詳細は、man ページの guestfish(1) を参照してください。

16.5. その他のコマンド

本セクションでは、guestfish を使用してゲスト仮想マシンのディスクイメージを表示および編集するのと同等の、簡単なツールを説明します。

- **virt-cat** は、guestfish **download** コマンドに似ています。ゲスト仮想マシンに 1 つのファイルをダウンロードして表示します。以下に例を示します。

  ```
  # virt-cat RHEL3 /etc/ntp.conf | grep ^server
  server      127.127.1.0       # local clock
  ```

- **virt-edit** は guestfish **edit** コマンドと似ています。これを使用すると、ゲスト仮想マシン内の 1 つのファイルを対話形式で編集できます。たとえば、Linux ベースのゲスト仮想マシンで、起動しない**grub.conf** ファイルを修正する必要がある場合があります。

  ```
  # virt-edit LinuxGuest /boot/grub/grub.conf
  ```

  virt-edit には別のモードがあり、このモードを使用すると、1 つのファイルにシンプルな非対話的な変更を加えることができます。これには、**-e** オプションが使用されます。たとえば、次のコマンドは、Linux ゲスト仮想マシンの root パスワードをパスワードなしに変更します。

  ```
  # virt-edit LinuxGuest /etc/passwd -e 's/^root:.*?:/root::/'
  ```

- **virt-ls** は guestfish の **ls** コマンド、**ll** コマンド、および **find** コマンドに似ています。これは、ディレクトリーを (再帰的に) 一覧表示するために使用されます。たとえば、以下のコマンドは、Linux ゲスト仮想マシンの /home 配下にファイルとディレクトリーを再帰的に一覧表示します。

```
# virt-ls -R LinuxGuest /home/ | less
```

16.6. VIRT-RESCUE: レスキューシェル

このセクションでは、レスキューシェルの使用に関する情報を提供します。

16.6.1. はじめに

本セクションでは、仮想マシンのレスキュー CD と似ていると思われる **virt-rescue** を説明します。ゲスト仮想マシンをレスキューシェルで起動して、メンテナーンスを実行してエラーを修正し、ゲスト仮想マシンを修復できるようにします。

virt-rescue と guestfish には、重複する部分があります。virt-rescue は、通常の Linux ファイルシステムツールを使用して対話型のアドホックな変更を行うために使用されます。これは、障害が発生したゲスト仮想マシンのレスキューにとりわけ適しています。virt-rescue はスクリプトを使用できません。

一方、guestfish は対話的に使用することもできますが、正式なコマンドセット (libguestfs API) を使用して、スクリプトを使用し、構造化した変更を行う場合にとりわけ役立ちます。

16.6.2. virt-rescue の実行

ゲスト仮想マシンで **virt-rescue** を使用する前に、ゲスト仮想マシンが実行していないことを確認してください。そうでないと、ディスクが破損します。ゲスト仮想マシンが稼働していないことを確認したら、次のコマンドを実行します。

```
virt-rescue GuestName
```

(GuestName は、libvirt が認識しているゲスト名です。) または、以下のコマンドを実行します。

```
virt-rescue /path/to/disk/image
```

(パスは、ゲスト仮想マシンのディスクを含むファイル、論理ボリューム、LUN などを指定できます)。

virt-rescue がレスキュー仮想マシンを起動すると、最初に出力がスクロールして過去のものが表示されます。最後に、以下が表示されます。

```
Welcome to virt-rescue, the libguestfs rescue shell.

Note: The contents of / are the rescue appliance.
You have to mount the guest virtual machine's partitions under /sysroot
before you can examine them.

bash: cannot set terminal process group (-1): Inappropriate ioctl for device
bash: no job control in this shell
><rescue>
```

シェルプロンプトは通常の bash シェルで、通常の Red Hat Enterprise Linux コマンドを減らしたものが利用できます。たとえば、次のように入力します。

```
><rescue> fdisk -l /dev/vda
```

上記のコマンドは、ディスクパーティションの一覧を表示します。ファイルシステムをマウントするには、**/sysroot** にマウントすることが推奨されます。これは、ユーザーが好きなものをマウントするため

のレスキューマシンの空のディレクトリーです。/ のファイルは、レスキュー仮想マシン自体のファイルであることに注意してください。

```
><rescue> mount /dev/vda1 /sysroot/
EXT4-fs (vda1): mounted filesystem with ordered data mode. Opts: (null)
><rescue> ls -l /sysroot/grub/
 total 324
 -rw-r--r--. 1 root root     63 Sep 16 18:14 device.map
 -rw-r--r--. 1 root root  13200 Sep 16 18:14 e2fs_stage1_5
 -rw-r--r--. 1 root root  12512 Sep 16 18:14 fat_stage1_5
 -rw-r--r--. 1 root root  11744 Sep 16 18:14 ffs_stage1_5
 -rw-------. 1 root root   1503 Oct 15 11:19 grub.conf
 [...]
```

ゲスト仮想マシンのレスキューが終了したら、**exit** または **Ctrl+d** を入力してシェルを終了します。

virt-rescue には多くのコマンドラインオプションがあります。最もよく使用されるオプションは以下のとおりです。

- **--ro**: ゲスト仮想マシンで読み取り専用モードで操作します。変更は保存されません。これを使用して、ゲスト仮想マシンを実験できます。シェルを終了すると、変更はすべて破棄されます。
- **--network**: レスキューシェルからのネットワークアクセスを有効にします。たとえば、RPM またはその他のファイルをゲスト仮想マシンにダウンロードする必要がある場合に使用します。

16.7. VIRT-DF: ディスク使用量のモニターリング

このセクションでは、**virt-df** を使用したディスク使用量の監視に関する情報を提供します。

16.7.1. はじめに

本セクションでは、ディスクイメージまたはゲスト仮想マシンからファイルシステムの使用状況を表示する **virt-df** を説明します。Linux の **df** コマンドと似ていますが、これは仮想マシン用です。

16.7.2. virt-df の実行

ディスクイメージにあるすべてのファイルシステムのファイルシステム使用状況を表示するには、次のコマンドを実行します。

```
# virt-df /dev/vg_guests/RHEL6
 Filesystem                1K-blocks       Used  Available  Use%
 RHEL6:/dev/sda1              101086      10233      85634   11%
 RHEL6:/dev/VolGroup00/LogVol00 7127864    2272744    4493036   32%
```

(**/dev/vg_guests/RHEL6** は Red Hat Enterprise Linux 6 ゲスト仮想マシンディスクイメージです。この場合のパスは、このディスクイメージが置かれているホストの物理マシンの論理ボリュームです。)

virt-df を単独で使用して、すべてのゲスト仮想マシン (すなわち libvirt に認識されているもの) の情報を一覧表示することも可能です。**virt-df** は、**-h** (人間が判読できる) や **-i** (ブロックの代わりに inode を表示) など、通常の**df**と同じオプションの一部を認識します。

virt-df は、Windows ゲスト仮想マシンでも機能します。

```
# virt-df -h
 Filesystem                      Size       Used  Available  Use%
 F14x64:/dev/sda1                484.2M      66.3M     392.9M   14%
 F14x64:/dev/vg_f14x64/lv_root    7.4G       3.0G       4.4G   41%
 RHEL6brewx64:/dev/sda1          484.2M      52.6M     406.6M   11%
 RHEL6brewx64:/dev/vg_rhel6brewx64/lv_root
                                 13.3G       3.4G       9.2G   26%
 Win7x32:/dev/sda1               100.0M      24.1M      75.9M   25%
 Win7x32:/dev/sda2                19.9G       7.4G      12.5G   38%
```

注記

virt-df は読み取り専用アクセスのみが必要であるため、ライブゲスト仮想マシンでは安全に使用できます。ただし、ゲスト仮想マシン内で実行している **df** コマンドと同じ数字になることを想定することはできません。これは、ディスク上のものが、ライブゲスト仮想マシンの状態とわずかに同期が外れるためです。それでも、分析と監視の目的には十分な近似値となります。

virt-df は、統計を監視ツールやデータベースなどに統合できるように設計されています。これにより、システム管理者は、ディスク使用状況の傾向に関するレポートを生成し、ゲスト仮想マシンでディスク領域が不足するとアラートを生成できます。これを行うには、**--csv** を使用して、マシンが判読可能なコンマ区切り値 (CSV) 出力を生成する必要があります。CSV 出力は、ほとんどのデータベース、スプレッドシートソフトウェア、およびその他のさまざまなツールとプログラミング言語で読み取り可能です。raw の CSV は以下のようになります。

```
# virt-df --csv WindowsGuest
 Virtual Machine,Filesystem,1K-blocks,Used,Available,Use%
 Win7x32,/dev/sda1,102396,24712,77684,24.1%
 Win7x32,/dev/sda2,20866940,7786652,13080288,37.3%
```

この出力を処理してトレンドとアラートを生成する方法に関するリソースとアイデアについては、次の URL を参照してください。 http://libguestfs.org/virt-df.1.html。

16.8. VIRT-RESIZE: ゲスト仮想マシンのオフラインでのサイズ変更

本セクションでは、オフラインのゲスト仮想マシンのサイズを変更する方法を説明します。

16.8.1. はじめに

本セクションでは、ゲスト仮想マシンを拡張または縮小するツールである **virt-resize** を説明します。これは、オフラインのゲスト仮想マシン (シャットダウン) でのみ機能します。これは、ゲストの仮想マシンイメージをコピーして、元のディスクイメージをそのままにして動作します。元のイメージをバックアップとして使用できるため、これは理想的ですが、2 倍のディスク容量が必要になるというトレードオフがあります。

16.8.2. ディスクイメージの拡張

このセクションでは、ディスクイメージを拡張する簡単な例を示します。

1. サイズを変更するディスクイメージを探します。コマンド **virsh dumpxml GuestName** は、libvirt ゲスト仮想マシンに使用できます。

2. ゲスト仮想マシンを拡張する方法を決定します。次の出力に示すように、ゲスト仮想マシンディスクで **virt-df -h** および **virt-list-partitions -lh** を実行します。

```
# virt-df -h /dev/vg_guests/RHEL6
Filesystem                      Size       Used  Available  Use%
RHEL6:/dev/sda1                   98.7M      10.0M      83.6M   11%
RHEL6:/dev/VolGroup00/LogVol00  6.8G       2.2G       4.3G   32%

# virt-list-partitions -lh /dev/vg_guests/RHEL6
/dev/sda1 ext3 101.9M
/dev/sda2 pv 7.9G
```

この例では、次の方法を示します。

- 最初の (ブート) パーティションのサイズを、約 100MB から 500MB に増やします。
- ディスクの合計サイズを 8GB から 16GB に増やします。
- 2 番目のパーティションを展開して、残りの領域を埋めます。
- **/dev/VolGroup00/LogVol00** を展開し、2 番目のパーティションの新しい領域を埋めます。

1. ゲスト仮想マシンがシャットダウンしていることを確認します。
2. 元のディスクの名前をバックアップとして変更します。これを行う方法は、元のディスクのホスト物理マシンのストレージ環境によって異なります。ファイルとして保存されている場合は、**mv** コマンドを実行します。(この例で示しているように) 論理ボリュームの場合は、**lvrename** を使用します。

```
# lvrename /dev/vg_guests/RHEL6 /dev/vg_guests/RHEL6.backup
```

3. 新しいディスクを作成します。この例の要件は、ディスクの合計サイズを最大 16GB まで拡張することです。論理ボリュームが使用されるため、以下のコマンドが使用されます。

```
# lvcreate -L 16G -n RHEL6 /dev/vg_guests
Logical volume "RHEL6" created
```

4. 手順 2 の要件は、以下のコマンドで表されます。

```
# virt-resize \
      /dev/vg_guests/RHEL6.backup /dev/vg_guests/RHEL6 \
      --resize /dev/sda1=500M \
      --expand /dev/sda2 \
      --LV-expand /dev/VolGroup00/LogVol00
```

最初の 2 つの引数は、入力ディスクと出力ディスクです。**--resize /dev/sda1=500M** は、最初のパーティションのサイズを最大 500MB に変更します。**--expand /dev/sda2** は、残りのすべてのスペースを埋めるために 2 番目のパーティションを展開します。**--LV-expand /dev/VolGroup00/LogVol00** は、ゲスト仮想マシンの論理ボリュームを拡張して、2 番目のパーティションの余分なスペースを埋めます。

virt-resize は、出力で実行している処理を説明します。

```
Summary of changes:
```

```
/dev/sda1: partition will be resized from 101.9M to 500.0M
/dev/sda1: content will be expanded using the 'resize2fs' method
/dev/sda2: partition will be resized from 7.9G to 15.5G
/dev/sda2: content will be expanded using the 'pvresize' method
/dev/VolGroup00/LogVol00: LV will be expanded to maximum size
/dev/VolGroup00/LogVol00: content will be expanded using the 'resize2fs' method
Copying /dev/sda1 ...
[##################################################]
Copying /dev/sda2 ...
[##################################################]
Expanding /dev/sda1 using the 'resize2fs' method
Expanding /dev/sda2 using the 'pvresize' method
Expanding /dev/VolGroup00/LogVol00 using the 'resize2fs' method
```

5. 仮想マシンを起動してみてください。正常に機能している場合 (および徹底的なテストを行った後) は、バックアップディスクを削除できます。失敗した場合は、仮想マシンをシャットダウンして、新しいディスクを削除し、バックアップディスクの名前を元の名前に戻します。

6. **virt-df** または **virt-list-partitions** を使用して、新しいサイズを表示します。

```
# virt-df -h /dev/vg_pin/RHEL6
  Filesystem                    Size       Used  Available  Use%
  RHEL6:/dev/sda1               484.4M      10.8M     448.6M    3%
  RHEL6:/dev/VolGroup00/LogVol00 14.3G       2.2G      11.4G   16%
```

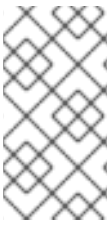
ゲスト仮想マシンのサイズ変更は正確な科学ではありません。**virt-resize** が失敗した場合は、virt-resize(1) の man ページに、確認して試すことができるヒントが複数あります。一部の古い Red Hat Enterprise Linux ゲスト仮想マシンに場合は、GRUB に関するヒントに特に注意を払う必要がある場合があります。

16.9. VIRT-INSPECTOR: ゲスト仮想マシンの検査

このセクションでは、**virt-inspector** を使用したゲスト仮想マシンの検査に関する情報を提供します。

16.9.1. はじめに

virt-inspector は、ディスクイメージを調べて、そこに含まれるオペレーティングシステムを確認するツールです。



注記

Red Hat Enterprise Linux 6.2 では、このプログラムには 2 つのバリエーションが提供されています。**virt-inspector** は Red Hat Enterprise Linux 6.0 にあるオリジナルプログラムであり、アップストリームで非推奨となりました。**virt-inspector2** は、新しいアップストリーム **virt-inspector** プログラムと同じです。

16.9.2. インストールシステム

virt-inspector とドキュメントをインストールするには、次のコマンドを入力します。

```
# yum install libguestfs-tools libguestfs-devel
```

Windows ゲスト仮想マシンを処理するには、libguestfs-winsupport もインストールする必要がありま

す。詳細は、「インストールシステム」 を参照してください。サンプルの XML 出力や、出力の Relax-NG スキーマを含むドキュメントが、**/usr/share/doc/libguestfs-devel-*/** にインストールされます。"*" は、libguestfs のバージョン番号に置き換えられます。

16.9.3. virt-inspector の実行

以下の例に示すように、**virt-inspector** は任意のディスクイメージまたは libvirt ゲスト仮想マシンに対して実行できます。

```
virt-inspector --xml disk.img > report.xml
```

または、以下のようになります。

```
virt-inspector --xml GuestName > report.xml
```

結果は XML レポート (**report.xml**) になります。XML ファイルの主なコンポーネントは、通常、以下のように単一の <operatingsystem> 要素が含まれるトップレベルの <operatingsystems> 要素です。

```
 <operatingsystems>
  <operatingsystem>

    <!-- the type of operating system and Linux distribution -->
    <name>linux</name>
    <distro>rhel</distro>
    <!-- the name, version and architecture -->
    <product_name>Red Hat Enterprise Linux Server release 6.4 </product_name>
    <major_version>6</major_version>
    <minor_version>4</minor_version>
    <package_format>rpm</package_format>
    <package_management>yum</package_management>
    <root>/dev/VolGroup/lv_root</root>
    <!-- how the filesystems would be mounted when live -->
    <mountpoints>
      <mountpoint dev="/dev/VolGroup/lv_root">/</mountpoint>
      <mountpoint dev="/dev/sda1">/boot</mountpoint>
      <mountpoint dev="/dev/VolGroup/lv_swap">swap</mountpoint>
    </mountpoints>

   < !-- filesystems-->
     <filesystem dev="/dev/VolGroup/lv_root">
       <label></label>
       <uuid>b24d9161-5613-4ab8-8649-f27a8a8068d3</uuid>
       <type>ext4</type>
       <content>linux-root</content>
       <spec>/dev/mapper/VolGroup-lv_root</spec>
     </filesystem>
     <filesystem dev="/dev/VolGroup/lv_swap">
       <type>swap</type>
       <spec>/dev/mapper/VolGroup-lv_swap</spec>
     </filesystem>
    <!-- packages installed -->
    <applications>
      <application>
        <name>firefox</name>
        <version>3.5.5</version>
```

```
        <release>1.fc12</release>
      </application>
    </applications>

  </operatingsystem>
</operatingsystems>
```

これらのレポートの処理は、W3C 標準の XPath クエリーを使用して行うのが最適です。Red Hat Enterprise Linux 6 には、単純なインスタンスに使用できるコマンドラインプログラム (**xpath**) が付属しています。ただし、長期的かつ高度な使用法については、XPath ライブラリーをお気に入りのプログラミング言語と一緒に使用することを検討する必要があります。

たとえば、以下の XPath クエリーを使用して、すべてのファイルシステムデバイスを一覧表示できます。

```
virt-inspector --xml GuestName | xpath //filesystem/@dev
 Found 3 nodes:
 -- NODE --
 dev="/dev/sda1"
 -- NODE --
 dev="/dev/vg_f12x64/lv_root"
 -- NODE --
 dev="/dev/vg_f12x64/lv_swap"
```

または、次のコマンドを実行してインストールしたすべてのアプリケーションの名前を一覧表示します。

```
virt-inspector --xml GuestName | xpath //application/name
[...long list...]
```

16.10. VIRT-WIN-REG: WINDOWS レジストリーの読み取りと編集

16.10.1. はじめに

virt-win-reg は、Windows ゲスト仮想マシンのレジストリーを操作するツールです。レジストリーキーの読み取りに使用できます。これを使用してレジストリーに変更を加えることもできますが、ライブ/実行中のゲスト仮想マシンに対してこれを実行しようとし **ない**でください。ディスクが破損する可能性があります。

16.10.2. インストールシステム

virt-win-reg を使用するには、次を実行する必要があります。

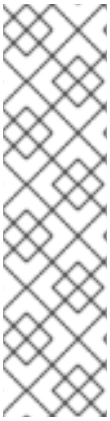
```
# yum install /usr/bin/virt-win-reg
```

16.10.3. virt-win-reg を使用する

レジストリーキーを読み取るには、ゲスト仮想マシン (またはそのディスクイメージ) の名前とレジストリーキーの名前を指定します。目的のキーの名前を囲むには、一重引用符を使用する必要があります。

```
# virt-win-reg WindowsGuest \
    'HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall' \
    | less
```

出力は、Windows の **.REG**ファイルで使用される標準のテキストベースの形式です。



注記

この形式では文字列の移植可能なエンコード方法が適切に定義されていないため、文字列には 16 進引用符が使用されます。これは、あるマシンから別のマシンに **.REG**ファイルを転送するときに忠実度を確保する唯一の方法です。

この単純な Perl スクリプトを介して **virt-win-reg** の出力をパイプ処理することにより、16 進引用符で囲まれた文字列を印刷可能にすることができます。

```
perl -MEncode -pe's?hex\((\d+)\):(\S+)?
$t=$1;$_=$2;s,\,,,g;"str($t):\"".decode(utf16le=>pack("H*",$_))."\""?eg'
```

変更をオフラインゲスト仮想マシンの Windows レジストリーにマージするには、最初に **.REG** ファイルを準備する必要があります。この作業に関するドキュメントは、ここで入手できます。**.REG** ファイルを準備したら、次のように入力します。

```
# virt-win-reg --merge WindowsGuest input.reg
```

これにより、ゲスト仮想マシンのレジストリーが更新されます。

16.11. プログラミング言語からの API の使用

libguestfs API は、Red Hat Enterprise Linux 6.2 の次の言語から直接使用できます:C、C ++、Perl、Python、Java、Ruby、および OCaml。

- C バインディングおよび C++ バインディングをインストールするには、以下のコマンドを実行します。

  ```
  # yum install libguestfs-devel
  ```

- Perl バインディングをインストールするには、以下のコマンドを実行します。

  ```
  # yum install 'perl(Sys::Guestfs)'
  ```

- Python バインディングをインストールするには、以下のコマンドを実行します。

  ```
  # yum install python-libguestfs
  ```

- Java バインディングをインストールするには、以下のコマンドを実行します。

  ```
  # yum install libguestfs-java libguestfs-java-devel libguestfs-javadoc
  ```

- Ruby バインディングをインストールするには、以下のコマンドを実行します。

  ```
  # yum install ruby-libguestfs
  ```

- OCaml バインディングをインストールするには、以下のコマンドを実行します。

```
# yum install ocaml-libguestfs ocaml-libguestfs-devel
```

各言語のバインディングは基本的に同じですが、構文が少し変更されています。C ステートメント:

```
guestfs_launch (g);
```

Perl では、以下のように表示されます。

```
$g->launch ()
```

または、OCaml で以下のようになります。

```
g#launch ()
```

本セクションでは、C の API のみを説明します。

C バインディングおよび C++ バインディングでは、エラーを手動で確認する必要があります。他のバインディングでは、エラーは例外に変換されます。以下の例で示している追加のエラーチェックは他の言語には必要ありませんが、逆に例外をキャッチするコードを追加することもできます。libguestfs API のアーキテクチャーに関する肝心な点については、以下の一覧を参照してください。

- libguestfs API は同期的です。各呼び出しは、完了するまでブロックされます。非同期で呼び出しを行う場合は、スレッドを作成する必要があります。
- libguestfs の API はスレッドセーフではありません。各ハンドルは、1 つのスレッドからのみ使用する必要があります。または、スレッド間でハンドルを共有する場合は、独自のミューテックスを実装して、2 つのスレッドが同時に 1 つのハンドルでコマンドを実行できないようにします。
- 同じディスクイメージで複数のハンドルを開くことはできません。すべてのハンドルが読み取り専用である場合は許容されますが、推奨されません。
- そのディスクイメージ (たとえば、ライブ仮想マシン) を使用するものがない場合は、書き込み用にディスクイメージを追加しないでください。これを行うと、ディスクが破損します。
- 現在使用中のディスクイメージ (ライブ VM など) で読み取り専用ハンドルを開くことができます。ただし、特にディスクイメージを読み取っているときに大量に書き込まれている場合は、結果が予測できないか、一貫性がない可能性があります。

16.11.1. C プログラムを介した API との相互作用

C プログラムは、<guestfs.h> ヘッダーファイルをインクルードし、ハンドルを作成することから始める必要があります。

```
#include <stdio.h>
#include <stdlib.h>
#include <guestfs.h>

int
main (int argc, char *argv[])
{
  guestfs_h *g;
```

```
  g = guestfs_create ();
  if (g == NULL) {
    perror ("failed to create libguestfs handle");
    exit (EXIT_FAILURE);
  }

  /* ... */

  guestfs_close (g);

  exit (EXIT_SUCCESS);
}
```

このプログラムをファイル (**test.c**) に保存します。このプログラムをコンパイルし、以下の 2 つのコマンドを実行します。

```
gcc -Wall test.c -o test -lguestfs
./test
```

この段階では、出力はありません。このセクションの残りの部分では、このプログラムを拡張して新しいディスクイメージを作成し、パーティションを作成し、ext4 ファイルシステムでフォーマットして、ファイルシステムにいくつかのファイルを作成する方法を示す例を紹介します。ディスクイメージは**disk.img** と呼ばれ、現在のディレクトリーに作成されます。

このプログラムの概要は、以下のとおりです。

- ハンドルを作成します。
- ハンドルにディスクを追加します。
- libguestfs バックエンドを起動します。
- パーティション、ファイルシステム、およびファイルを作成します。
- ハンドルを閉じて、終了します。

変更したプログラムを以下に示します。

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <guestfs.h>

int
main (int argc, char *argv[])
{
  guestfs_h *g;
  size_t i;

  g = guestfs_create ();
  if (g == NULL) {
    perror ("failed to create libguestfs handle");
    exit (EXIT_FAILURE);
```

```
}

/* Create a raw-format sparse disk image, 512 MB in size. */
int fd = open ("disk.img", O_CREAT|O_WRONLY|O_TRUNC|O_NOCTTY, 0666);
if (fd == -1) {
  perror ("disk.img");
  exit (EXIT_FAILURE);
}
if (ftruncate (fd, 512 * 1024 * 1024) == -1) {
  perror ("disk.img: truncate");
  exit (EXIT_FAILURE);
}
if (close (fd) == -1) {
  perror ("disk.img: close");
  exit (EXIT_FAILURE);
}

/* Set the trace flag so that we can see each libguestfs call. */
guestfs_set_trace (g, 1);

/* Set the autosync flag so that the disk will be synchronized
 * automatically when the libguestfs handle is closed.
 */
guestfs_set_autosync (g, 1);

/* Add the disk image to libguestfs. */
if (guestfs_add_drive_opts (g, "disk.img",
      GUESTFS_ADD_DRIVE_OPTS_FORMAT, "raw", /* raw format */
      GUESTFS_ADD_DRIVE_OPTS_READONLY, 0,   /* for write */
      -1 /* this marks end of optional arguments */ )
    == -1)
  exit (EXIT_FAILURE);

/* Run the libguestfs back-end. */
if (guestfs_launch (g) == -1)
  exit (EXIT_FAILURE);

/* Get the list of devices.  Because we only added one drive
 * above, we expect that this list should contain a single
 * element.
 */
char **devices = guestfs_list_devices (g);
if (devices == NULL)
  exit (EXIT_FAILURE);
if (devices[0] == NULL || devices[1] != NULL) {
  fprintf (stderr,
           "error: expected a single device from list-devices\n");
  exit (EXIT_FAILURE);
}

/* Partition the disk as one single MBR partition. */
if (guestfs_part_disk (g, devices[0], "mbr") == -1)
  exit (EXIT_FAILURE);

/* Get the list of partitions.  We expect a single element, which
 * is the partition we have just created.
```

```
     */
    char **partitions = guestfs_list_partitions (g);
    if (partitions == NULL)
      exit (EXIT_FAILURE);
    if (partitions[0] == NULL || partitions[1] != NULL) {
      fprintf (stderr,
               "error: expected a single partition from list-partitions\n");
      exit (EXIT_FAILURE);
    }

    /* Create an ext4 filesystem on the partition. */
    if (guestfs_mkfs (g, "ext4", partitions[0]) == -1)
      exit (EXIT_FAILURE);

    /* Now mount the filesystem so that we can add files. */
    if (guestfs_mount_options (g, "", partitions[0], "/") == -1)
      exit (EXIT_FAILURE);

    /* Create some files and directories. */
    if (guestfs_touch (g, "/empty") == -1)
      exit (EXIT_FAILURE);

    const char *message = "Hello, world\n";
    if (guestfs_write (g, "/hello", message, strlen (message)) == -1)
      exit (EXIT_FAILURE);

    if (guestfs_mkdir (g, "/foo") == -1)
      exit (EXIT_FAILURE);

    /* This uploads the local file /etc/resolv.conf into the disk image. */
    if (guestfs_upload (g, "/etc/resolv.conf", "/foo/resolv.conf") == -1)
      exit (EXIT_FAILURE);

    /* Because 'autosync' was set (above) we can just close the handle
     * and the disk contents will be synchronized.  You can also do
     * this manually by calling guestfs_umount_all and guestfs_sync.
     */
    guestfs_close (g);

    /* Free up the lists. */
    for (i = 0; devices[i] != NULL; ++i)
      free (devices[i]);
    free (devices);
    for (i = 0; partitions[i] != NULL; ++i)
      free (partitions[i]);
    free (partitions);

    exit (EXIT_SUCCESS);
}
```

このプログラムをコンパイルして、以下の 2 つのコマンドを実行します。

```
gcc -Wall test.c -o test -lguestfs
./test
```

プログラムが正常に完了すると、**disk.img** というディスクイメージが残ります。これは guestfish で調べることができます。

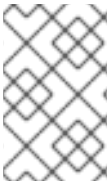
```
guestfish --ro -a disk.img -m /dev/sda1
><fs> ll /
><fs> cat /foo/resolv.conf
```

デフォルトでは (C バインディングおよび C++ バインディングのみ)、libguestfs は stderr にエラーを出力します。この動作は、エラーハンドラーを設定することで変更できます。man ページの guestfs(3) では、これを詳細に説明しています。

16.12. VIRT-SYSPREP: 仮想マシン設定のリセット

virt-sysprep コマンドラインツールを使用すると、ゲスト仮想マシンをリセットまたは設定解除して、クローンを作成できます。このプロセスには、SSH ホストキー、永続的なネットワーク MAC 設定、およびユーザーアカウントの削除が含まれます。**virt-sysprep** は、SSH キー、ユーザー、ロゴなどを追加して、仮想マシンをカスタマイズすることもできます。各手順は、必要に応じて有効または無効にできます。

sysprep という用語は、Microsoft Windows システムで使用されるシステム準備ツール (sysprep.exe) に由来します。それにもかかわらず、このツールは現在 Windows ゲストでは機能しません。



注記

libguestfs と guestfish ルート権限は必要ありません。アクセスされているディスクイメージが root アクセスから読み取りまたは書き込み、あるいは両方を必要とする場合にのみ、これらを root として実行する必要があります。

virt-sysprep ツールは、libguestfs-tools-c パッケージの一部で、以下のコマンドでインストールされます。

```
$ yum install libguestfs-tools-c
```

または、次のコマンドを使用して、**virt-sysprep** ツールのみをインストールすることもできます。

```
$ yum install /usr/bin/virt-sysprep
```

重要

virt-sysprep は、ゲストまたはディスクイメージを所定の位置に変更します。**virt-sysprep** を使用するには、ゲスト仮想マシンがオフラインである必要があるため、シャットダウンしてからコマンドを実行してください。ゲスト仮想マシンの既存のコンテンツを保持するには、最初にディスクのスナップショット、コピー、またはクローンを作成する必要があります。ディスクのコピーとクローン作成の詳細については、libguestfs.org を参照してください。

次のコマンドは、**virt-sysprep** で使用できます。

表16.1 virt-sysprepコマンド

コマンド	説明	例
--help	特定のコマンドまたはパッケージ全体に関する簡単なヘルプエントリーを表示します。詳細なヘルプは、man ページの virt-sysprep を参照してください。	$ **virt-sysprep --help**
-a [*file*] or --add [*file*]	指定された*file*を追加します。これは、ゲスト仮想マシンからのディスクイメージである必要があります。ディスクイメージの形式は自動検出されます。これを上書きして、特定の形式に強制する場合は、**--format** オプションを使用します。	$ **virt-sysprep --add /dev/vms/disk.img**
-c [*URI*] or --connect [*URI*]	libvirt を使用している場合は、指定の URI に接続します。省略した場合、KVM ハイパーバイザーを介して接続します。ゲストブロックデバイスを直接指定 (**virt-sysprep -a**) すると、libvirt はまったく使用されません。	$ **virt-sysprep -c qemu:///system**
-d [*guest*] or --domain [*guest*]	指定したゲスト仮想マシンのすべてのディスクを追加します。ドメイン名の代わりにドメイン UUID を使用できます。	$ **virt-sysprep --domain *90df2f3f-8857-5ba9-2714-7d95907b1c9e***
-n または --dry-run または --dryrun	ゲスト仮想マシンで、読み取り専用の "dry run" sysprep 操作を実行します。これにより sysprep 操作が実行されますが、末尾のディスクに対する変更はすべて破棄されます。	$ **virt-sysprep -n**
--enable [*operations*]	指定した*operations*を有効にします。可能な操作の一覧を表示するには、--list コマンドを使用します。	$ **virt-sysprep --enable ssh-hotkeys,udev-persistent-net**

コマンド	説明	例
--format [**raw**|**qcow2**|**auto**]	-a オプションのデフォルトは、ディスクイメージの形式を自動検出することです。これを使用すると、コマンドラインに続く -a オプションのディスク形式が強制されます。--format auto を使用すると、後続の -a オプションの自動検出に戻ります (上述の -a コマンドを参照)。	$ **virt-sysprep --format raw -a *disk.img*** は、disk.img に対して raw 形式 (自動検出なし) を強制しますが、**virt-sysprep --format raw -a *disk.img* --format auto -a *another.img*** *は、**disk.img** に対して raw 形式 (自動検出なし) を強制し、**another.img** では自動検出に戻ります。信頼できない raw 形式のゲストディスクイメージがある場合は、このオプションを使用してディスク形式を指定する必要があります。これにより、悪意のあるゲストで発生する可能性のあるセキュリティーの問題を回避できます。*
--list-operations	virt-sysprep プログラムでサポートされている操作を一覧表示します。これらは、1 行に 1 つずつ表示され、1 つ以上の空白で区切られたフィールドがあります。出力の最初のフィールドは操作名であり、**--enable** フラグに指定できます。2 番目のフィールドは、操作がデフォルトで有効になっている場合は * 文字で、そうでない場合は空白になります。同じ行のその他のフィールドには、操作の説明が記載されています。	$ **virt-sysprep --list-operations**
--mount-options	ゲスト仮想マシンの各マウントポイントにマウントオプションを設定します。mountpoint:options のペアのセミコロンで区切られたリストを使用します。この一覧をシェルから保護するために、この一覧の周囲に引用符を付ける必要がある場合があります。	$ **virt-sysprep --mount-options "/:notime"** は、**notime** 操作で root ディレクトリーをマウントします。

コマンド	説明	例
--selinux-relabel および --no-selinux-relabel	virt-sysprep は、ゲストの最初の起動時に SELinux の再ラベル付けを常にスケジュールするわけではありません。場合によっては、再ラベル付けが実行されます (たとえば、virt-sysprep がファイルを変更した場合)。ただし、すべての操作でファイルのみを削除する場合 (たとえば、**--enable delete --delete /some/file** を使用する場合)、再ラベル付けはスケジュールされません。**--selinux-relabel** オプションを使用すると、常に SELinux の再ラベル付けが強制されますが、**--no-selinux-relabel** が設定されている場合、再ラベル付けはスケジュールされません。**--selinux-relabel** を使用して、ファイルに正しい SELinux ラベルが付いていることを確認することをお勧めします。	$ **virt-sysprep --selinux-relabel**
-q or --quiet	ログメッセージを出力しないようにします。	$ **virt-sysprep -q**
-v or --verbose	デバッグ目的で詳細なメッセージを有効にします。	$ **virt-sysprep -v**
-V または --version	virt-sysprep のバージョン番号を表示し、終了します。	$ **virt-sysprep -V**
--root-password	root パスワードを設定します。新しいパスワードを明示的に指定する場合や、選択したファイルの最初の行の文字列を使用する場合に使用できます。これにより、安全性が高まります。	$ **virt-sysprep --root-password password:***123456* **-a** *guest.img* または $ **virt-sysprep --root-password file:***SOURCE_FILE_PATH* **-a** *guest.img*

詳細については、libguestfs のドキュメント を参照してください。

16.13. トラブルシューティング

libguestfs が機能しているかどうかを確認するテストツールを利用できます。libguestfs (root アクセスは必要ありません) をインストールした後、以下のコマンドを実行して、通常の操作をテストします。

```
$ libguestfs-test-tool
```

このツールは、libguestfs の動作をテストするために、大量のテキストを出力します。テストが成功すると、出力の末尾に以下のテキストが表示されます。

```
===== TEST FINISHED OK =====
```

16.14. その他のドキュメントの入手先

libguestfs とツールのドキュメントの主な情報源は、Unix の man ページです。API は guestfs(3) に文書化されています。guestfish は guestfish(1) に記載されています。virt ツールは、独自の man ページ (例: virt-df(1)) に記載されています。

第17章 ゲスト仮想マシン管理用のグラフィカルユーザーインターフェイスツール

virt-manager のほかに、Red Hat Enterprise Linux 6 には、ゲスト仮想マシンのコンソールにアクセスできるようにする以下のツールが備わっています。

17.1. VIRT-VIEWER

virt-viewer は、ゲスト仮想マシンのグラフィカルコンソールを表示するための最小限のコマンドラインユーティリティーです。コンソールには、VNC または SPICE プロトコルを使用してアクセスします。ゲストは、名前、ID、または UUID で参照できます。ゲストが実行していない場合は、ビューアーが起動するまで待機してからコンソールに接続するように設定できます。ビューアーはリモートホストに接続してコンソール情報を取得し、同じネットワークトランスポートを使用してリモートコンソールにも接続できます。

virt-manager と比較すると、**virt-viewer** では機能セットが小さくなりますが、リソースへの要求は低くなります。また、**virt-manager** とは異なり、ほとんどの場合、**virt-viewer** は、libvirt への読み取りと書き込みのパーミッションを必要としません。したがって、ゲストに接続して表示できる非特権ユーザーよって使用されることが可能ですが、設定することはできません。

virt-viewer ユーティリティーをインストールするには、以下を実行します。

```
# sudo yum install virt-viewer
```

構文
virt-viewer コマンドラインの基本的な構文は次のようになります。

```
# virt-viewer [OPTIONS] {guest-name|id|uuid}
```

virt-viewer コマンドラインの基本的な構文は次のようになります。

ゲスト仮想マシンへの接続
オプションを付けずに使用すると、**virt-viewer** は、ローカルシステムのデフォルトハイパーバイザーで接続できるゲストの一覧を表示します。

デフォルトのハイパーバイザーを使用するゲスト仮想マシンに接続するには:

```
# virt-viewer guest-name-or-UUID
```

KVM-QEMU ハイパーバイザーを使用するゲスト仮想マシンに接続するには、次のコマンドを実行します。

```
# virt-viewer --connect qemu:///system guest-name-or-UUID
```

TLS を使用してリモートコンソールに接続するには、次のコマンドを実行します。

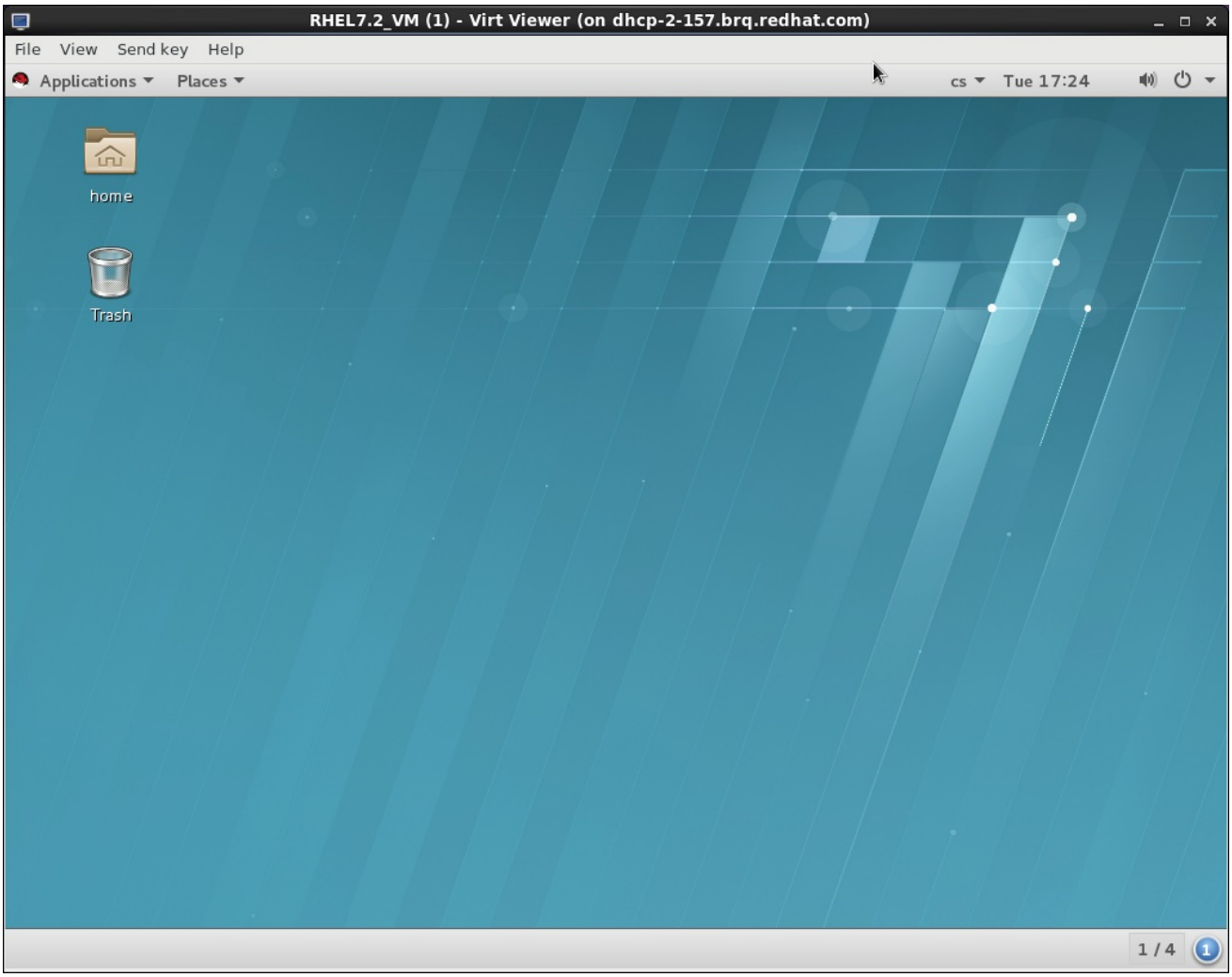
```
# virt-viewer --connect xen://example.org/ guest-name-or-UUID
```

SSH を使用してリモートホストのコンソールに接続するには、ゲスト設定を調べてから、コンソールにトンネリングされていない直接接続を確立します。

```
# virt-viewer --direct --connect xen+ssh://root@example.org/ guest-name-or-UUID
```

Interface
デフォルトでは、**virt-viewer** インターフェイスは、ゲストと対話するための基本的なツールのみを提供します。

図17.1 サンプルの virt-viewer インターフェイス



ホットキーの設定
virt-viewerセッション用にカスタマイズしたキーボードショートカット (ホットキーとも呼ばれます) を作成する場合は、**--hotkeys** オプションを使用します。

```
# virt-viewer --hotkeys=action1=key-combination1[,action2=key-combination2] guest-name-or-UUID
```

ホットキーに割り当てることができるアクションは、次のとおりです。

- toggle-fullscreen
- release-cursor
- smartcard-insert
- smartcard-remove

キーと名前の組み合わせのホットキーでは、大文字と小文字が区別されません。ホットキーの設定は、今後の **virt-viewer** セッションには引き継がれないことに注意してください。

例17.1 virt-viewer ホットキーの設定

KVM-QEMU ゲストに接続して testguest と呼ばれる 全画面モードに変更するホットキーを追加するには、次のコマンドを実行します。

```
# virt-viewer --hotkeys=toggle-fullscreen=shift+f11 qemu:///system testguest
```

キオスクモード
キオスクモードでは、**virt-viewer** により、ユーザーは接続したデスクトップとのみ対話でき、ゲストがシャットダウンしない限り、ゲスト設定またはホストシステムと対話するオプションは提供されません。これは、管理者がユーザーのアクションの範囲を指定したゲストに制限したい場合などに役立ちます。

キオスクモードを使用するには、**-k** または **--kiosk** オプションでゲストに接続します。

例17.2 キオスクモードでの virt-viewer の使用

マシンのシャットダウン後に終了する KVM-QEMU 仮想マシンにキオスクモードで接続する場合は、次のコマンドを使用します。

```
# virt-viewer --connect qemu:///system guest-name-or-UUID --kiosk --kiosk-quit on-disconnect
```

ただし、キオスクモードのみでは、ゲストのシャットダウン後に、ユーザーがホストシステムやゲスト設定と対話しないようにすることはできません。これには、ホストのウィンドウマネージャーを無効にするなど、さらなるセキュリティー対策が必要になります。

17.2. REMOTE-VIEWER

remote-viewer は、SPICE および VNC に対応するシンプルなリモートデスクトップディスプレイクライアントです。これは、機能のほとんどと制限を virt-viewer と共有します。

ただし、virt-viewer とは異なり、**remote-viewer** では、リモートゲストディスプレイに接続するのに libvirt が必要ありません。このため、**remote-viewer** は、たとえば、libvirt と対話したり SSH 接続を使用したりするためのパーミッションを提供しないリモートホスト上の仮想マシンに接続するために使用できます。

remote-viewer ユーティリティーをインストールするには、以下を実行します。

```
# sudo yum install virt-viewer
```

構文
remote-viewer コマンドラインの基本的な構文は次のようになります。

```
# remote-viewer [OPTIONS] {guest-name|id|uuid}
```

remote-viewer で使用できるオプションの完全リストを表示するには、**man remote-viewer** を使用します。

ゲスト仮想マシンへの接続
オプションを付けずに使用すると、**remote-viewer** は、ローカルシステムのデフォルトの URI で接続できるゲストの一覧を表示します。

remote-viewer を使用して特定のゲストに接続するには、VNC/SPICE URI を使用します。URI の取得方法は、「グラフィック表示への接続の URI の表示」 を参照してください。

例17.3 SPICE を使用したゲストディスプレイへの接続

以下を使用して、SPICE 通信にポート 5900 を使用する "testguest" と呼ばれるマシンの SPICE サーバーに接続します。

```
# remote-viewer spice://testguest:5900
```

例17.4 VNC を使用したゲストディスプレイへの接続

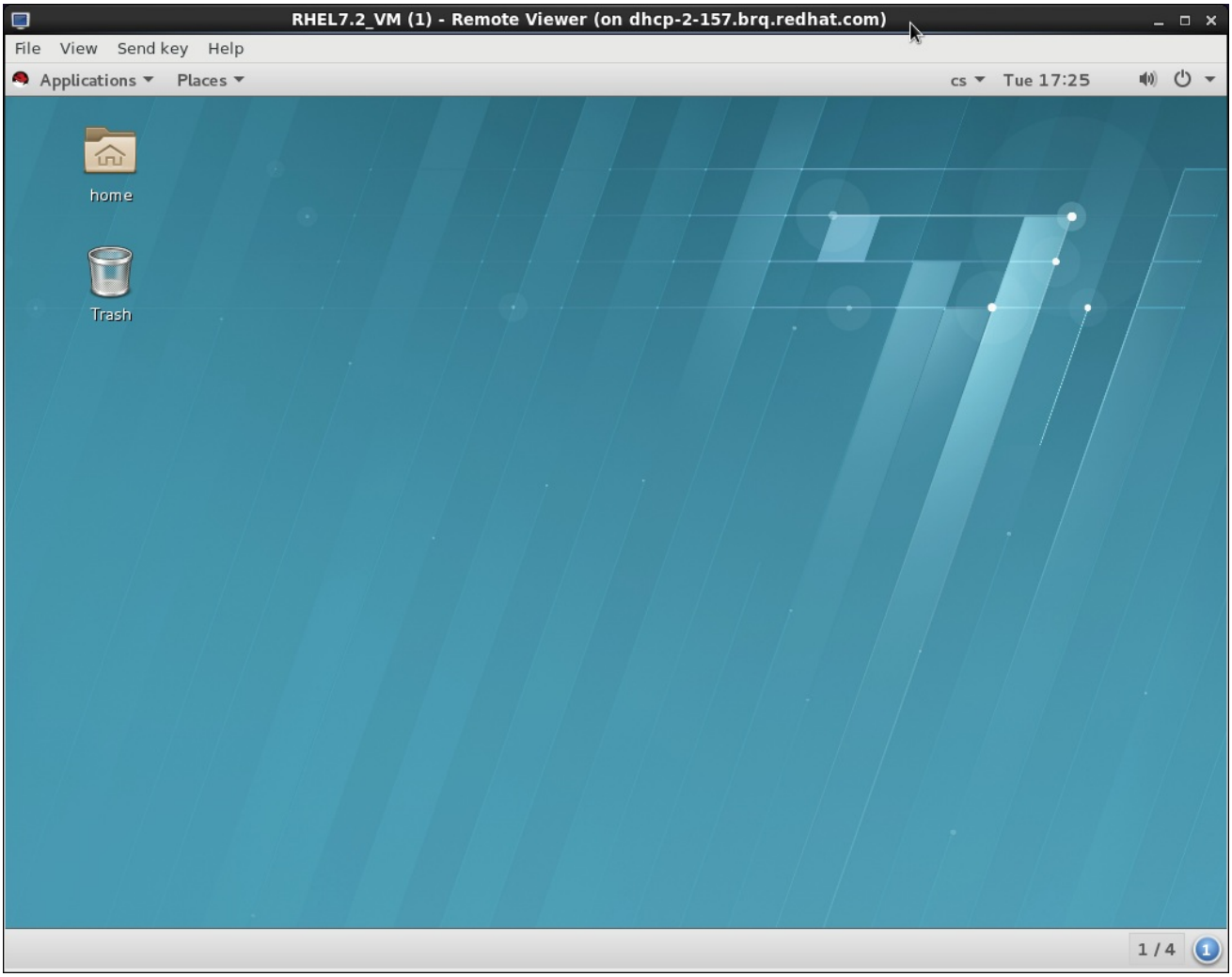
以下を使用して、VNC 通信にポート 5900 を使用する **testguest2** と呼ばれるマシンの VNC サーバーに接続します。

```
# remote-viewer vnc://testguest2:5900
```

Interface

デフォルトでは、**remote-viewer** インターフェイスは、ゲストと対話するための基本的なツールのみを提供します。

図17.2 サンプルの remote-viewer インターフェイス



第18章 仮想ネットワーク

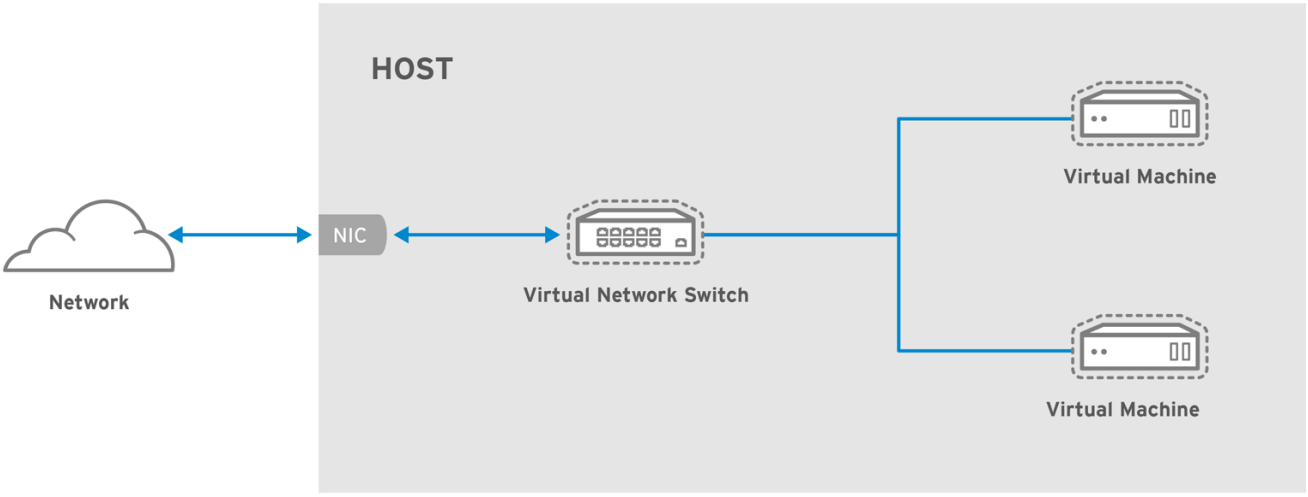
本章では、libvirt で仮想ネットワークを作成、起動、停止、削除、および変更するために必要な概念を説明します。

詳細は、libvirt のリファレンスの章を参照してください。

18.1. 仮想ネットワークスイッチ

Libvirt 仮想ネットワークは、**仮想ネットワークスイッチ** という概念を使用します。仮想ネットワークスイッチは、ホストの物理マシンサーバーで動作し、仮想マシン (ゲスト) が接続するソフトウェア構造です。ゲストのネットワークトラフィックは、このスイッチを介して転送されます。

図18.1 2 つのゲストを持つ仮想ネットワークスイッチ



Linux ホストの物理マシンサーバーは、ネットワークインターフェイスとして仮想ネットワークスイッチを表します。libvirtd デーモン (**libvirtd**) を最初にインストールして起動すると、仮想ネットワークスイッチを表すデフォルトのネットワークインターフェイスが **virbr0** になります。

この**virbr0**インターフェイスは、他のインターフェイスと同様に、**ip** コマンドで表示できます。

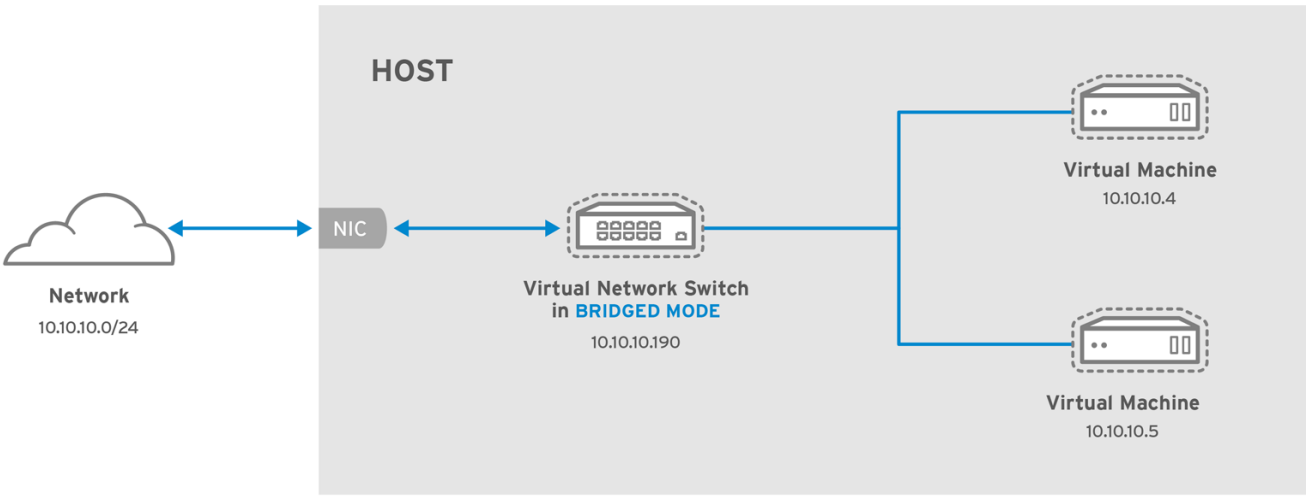
```
$ ip addr show virbr0
3: virbr0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
    link/ether 1b:c4:94:cf:fd:17 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
```

18.2. ブリッジモード

ブリッジモード を使用する場合は、ゲスト仮想マシンがすべて、ホストの物理マシンと同じサブネット内に表示されます。同じ物理ネットワーク上のその他のすべての物理マシンは、仮想マシンを認識しており、仮想マシンにアクセスできます。ブリッジングは、OSI ネットワークモデルのレイヤー 2 で動作します。

ハイパーバイザーで複数の物理インターフェイスを使用する場合は、ボンドで複数のインターフェイスを結合します。ボンディングがブリッジに追加され、ゲスト仮想マシンもブリッジに追加されます。ただし、ボンディングドライバーには動作モードが複数あり、このモードのごく一部は、仮想ゲストマシンが使用されているブリッジで機能します。

図18.2 ブリッジモードの仮想ネットワークスイッチ



> **警告**
>
> ゲスト仮想マシンで使用する必要があるボンディングモードは、モード 1、モード 2、およびモード 4 のみです。いかなる状況でも、モード 0、3、5、または 6 を使用しないでください。また、arp-monitoring は機能しないため、mii-monitoring を使用してボンディングモードを監視する必要があることにも注意してください。

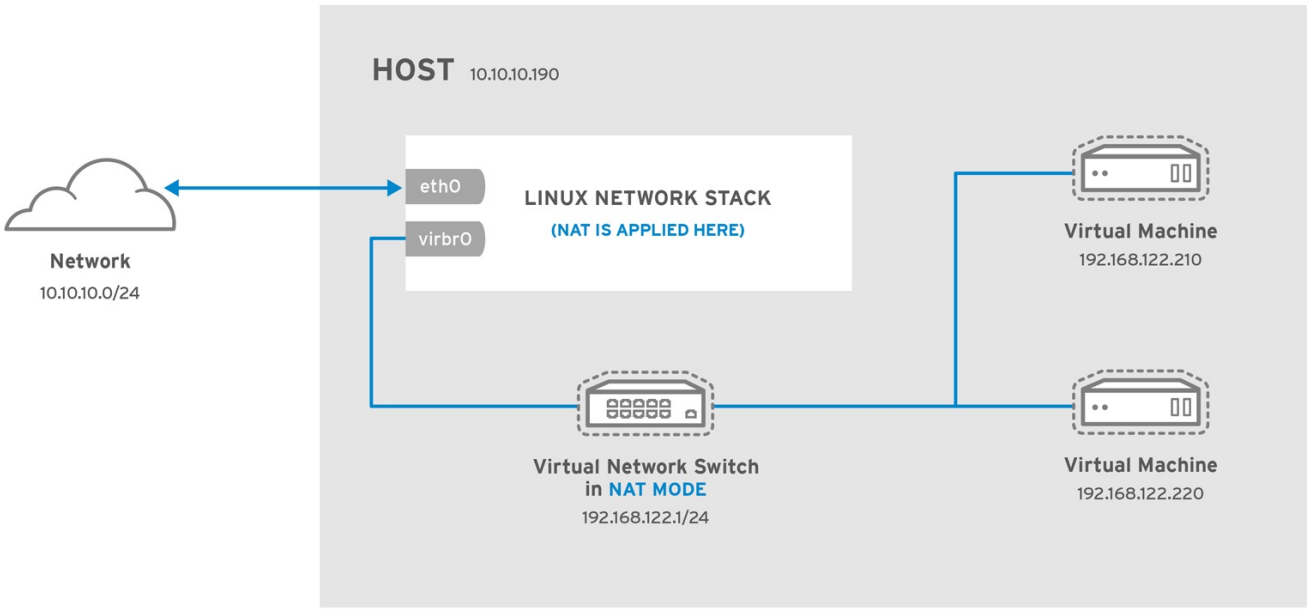
ボンディングモードの詳細は、ナレッジベースのボンディングモード、または Red Hat Enterprise Linux 6 デプロイメントガイドを参照してください。

bridge_opts パラメーターの詳細な説明は、『Red Hat Virtualization 管理ガイド』を参照してください。

18.3. ネットワークアドレス変換モード

デフォルトでは、仮想ネットワークスイッチは NAT モードで動作します。SNAT (Source-NAT) や DNAT (Destination-NAT)) ではなく、IP マスカレードを使用します。IP マスカレードを使用すると、接続したゲストが、ホストの物理マシンの IP アドレスを使用して外部ネットワークと通信できます。次の図に示すように、デフォルトでは、仮想ネットワークスイッチが NAT モードで動作している場合、ホスト物理マシンの外部に配置されたコンピューターは内部のゲストと通信できません。

図18.3 2 つのゲストで NAT を使用する仮想ネットワークスイッチ



警告

仮想ネットワークスイッチは、iptables ルールで設定された NAT を使用します。スイッチの実行中にこのルールを編集することは推奨されていません。誤ったルールがあると、スイッチが通信できなくなる可能性があるためです。

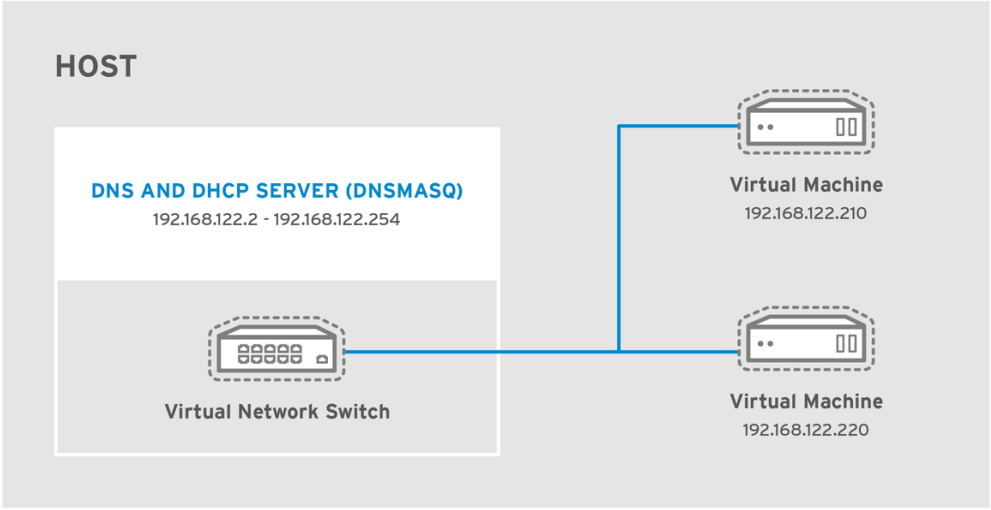
スイッチが実行していない場合は、正引きモードの NAT のパブリック IP 範囲を設定して、ポートマスカレードの範囲を作成するには、以下のコマンドを実行します。

```
# iptables -j SNAT --to-source [start]-[end]
```

18.3.1. DNS および DHCP

IP 情報は DHCP を介してゲストに割り当てることができます。このため、仮想ネットワークスイッチには、アドレスのプールを割り当てることができます。Libvirt は、これに **dnsmasq** プログラムを使用します。dnsmasq のインスタンスは、それを必要とする各仮想ネットワークの libvirt により自動的に設定され、起動します。

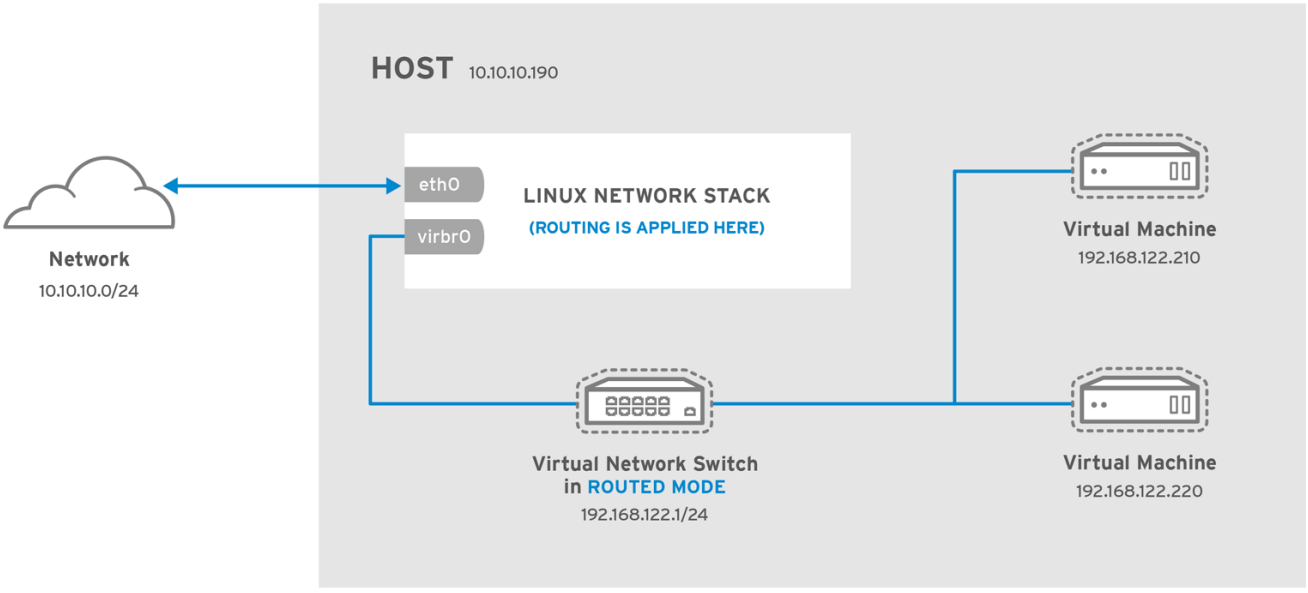
図18.4 dnsmasq を実行している仮想ネットワークスイッチ



18.4. ルーティングモード

ルーティングモード を使用する場合は、仮想スイッチを、ホストマシンに接続された物理 LAN に接続し、NAT を使用せずにトラフィックをやり取りします。仮想スイッチは、すべてのトラフィックを調べ、ネットワークパケットに含まれる情報を使用して、ルーティングの決定を行うことができます。このモードを使用すると、仮想マシンはすべて自身のサブネットにあり、仮想スイッチを介してルーティングされます。物理ネットワーク上の他のホスト物理マシンが手動の物理ルーター設定なしで仮想マシンを認識しておらず、仮想マシンにアクセスできないため、この状況は必ずしも理想的ではありません。ルーティングモードは、OSI ネットワークモデルのレイヤー 3 で動作します。

図18.5 ルーティングモードの仮想ネットワークスイッチ

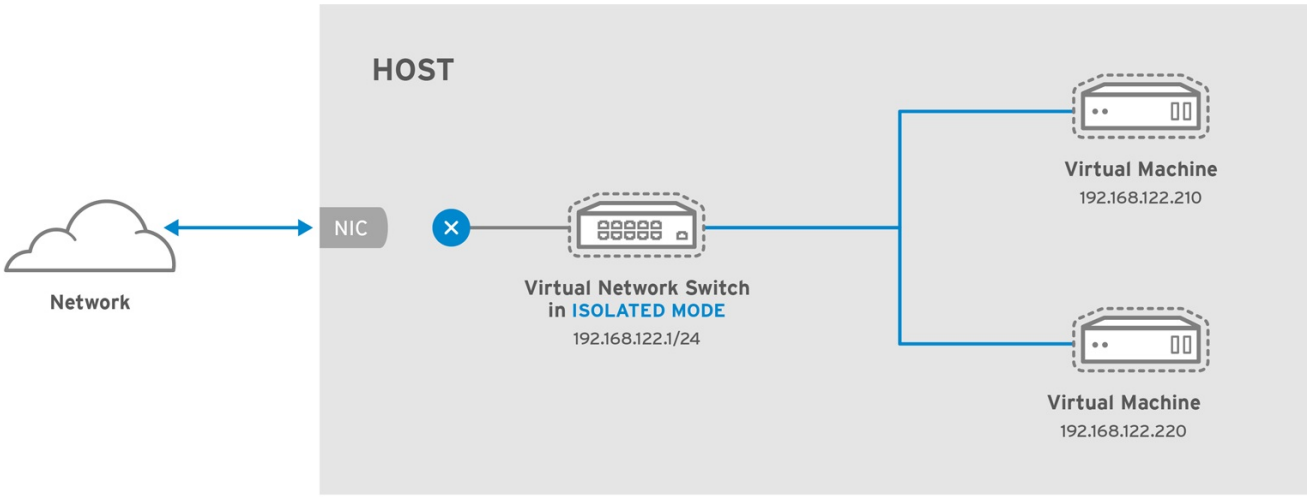


18.5. 分離モード

分離モード を使用する場合は、仮想スイッチに接続したゲストが、ホストの物理マシンと相互通信できます。ただし、ゲストのトラフィックはホストの物理マシンの外部を通過しないため、ホストの物理マシンの外部からのトラフィックを受信できません。DHCP などの基本的な機能には、このモードの

dnsmasq を使用する必要があります。ただし、このネットワークが物理ネットワークから分離されていても、DNS 名は解決されません。そのため、DNS 名が解決しても、ICMP echo 要求 (ping) コマンドが失敗すると、問題が発生する可能性があります。

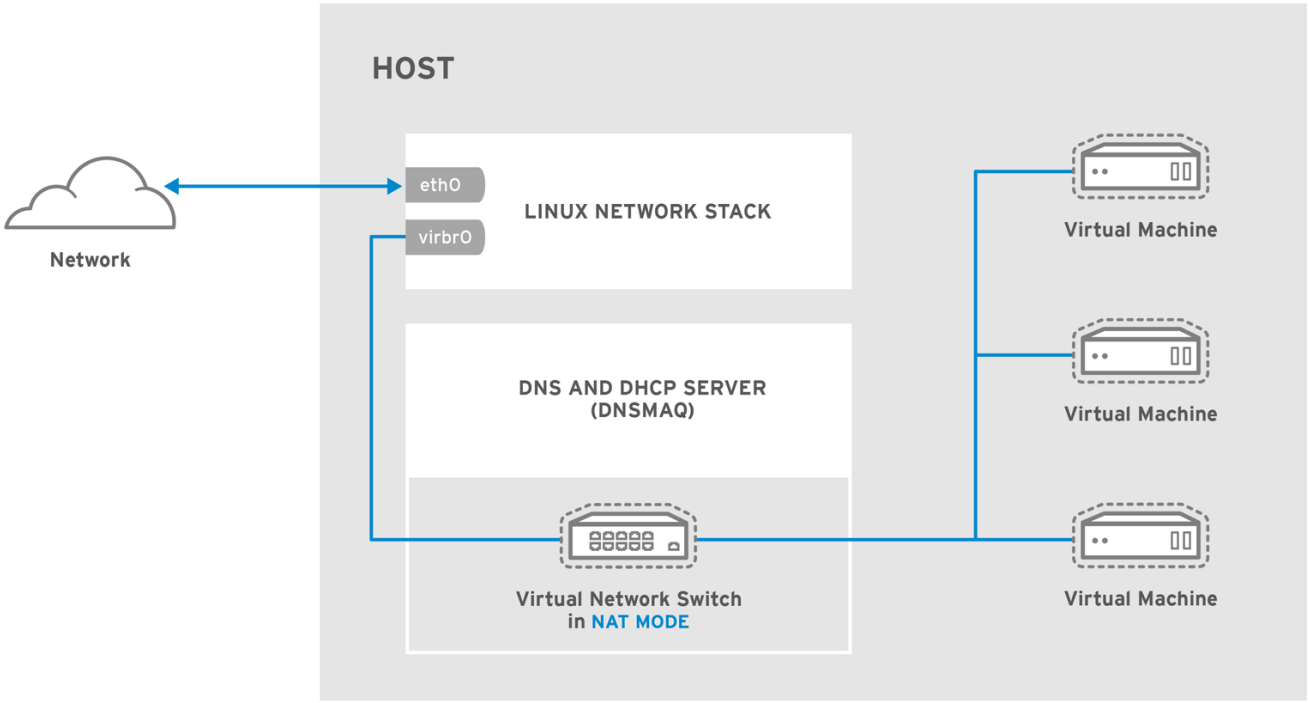
図18.6 分離モードの仮想ネットワークスイッチ



18.6. デフォルト設定

libvirtd デーモン (**libvirtd**) を最初にインストールすると、NAT モードの最初の仮想ネットワークスイッチ設定が含まれます。この設定は、インストールしたゲストが、ホストの物理マシンを介して外部ネットワークと通信できるように使用されます。以下のイメージは、**libvirtd** におけるこの既定の設定を示しています。

図18.7 デフォルトの libvirt ネットワーク設定



注記

仮想ネットワークは、特定の物理インターフェイスに制限できます。これは、複数のインタフェースを持つ物理システムで役立つ場合があります (たとえば、**eth0**、**eth1**、および **eth2**)。これは、ルーティングモードおよび NAT モードでのみ役に立ち、新しい仮想ネットワークを作成する場合は、**dev=<interface>** オプション、または **virt-manager** で定義できます。

18.7. 一般的なシナリオの例

本セクションでは、さまざまな仮想ネットワークモードと、いくつかのシナリオ例を示します。

18.7.1. ブリッジモード

ブリッジモードは、OSI モデルのレイヤー 2 で動作します。これを使用すると、ゲスト仮想マシンはすべて、ホストの物理マシンと同じサブネットに表示されます。ブリッジモードにおける最も一般的なユースケースには、たとえば以下のようなものがあります。

- 既存ネットワークにゲスト仮想マシンをホスト物理マシンとともにデプロイすると、仮想マシンと物理マシンの違いがエンドユーザーに透過的になります。
- 既存の物理ネットワーク設定を変更せずに仮想マシンをデプロイします。
- 既存の物理ネットワークに簡単にアクセスできる必要があるゲスト仮想マシンをデプロイします。既存のブロードキャストドメイン (DHCP など) のサービスにアクセスする必要がある物理ネットワークにゲスト仮想マシンを配置します。
- VLAN が使用されている既存のネットワークにゲスト仮想マシンを接続する。

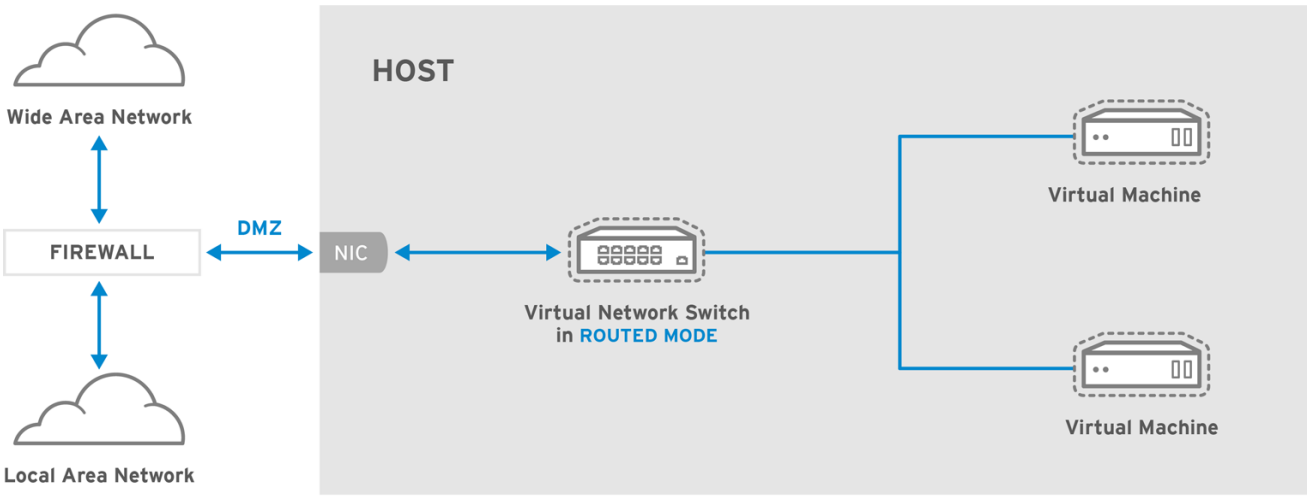
18.7.2. ルーティングモード

このセクションでは、ルーテッドモードに関する情報を提供します。

DMZ

セキュリティー上の理由から、1 つ以上のノードが制御されたサブネットワークに配置されているネットワークを検討してください。このような特別なサブネットワークのデプロイメントは一般的な慣例であり、サブネットワークは DMZ と呼ばれています。このレイアウトの詳細は、以下の図を参照してください。

図18.8 サンプルの DMZ 設定

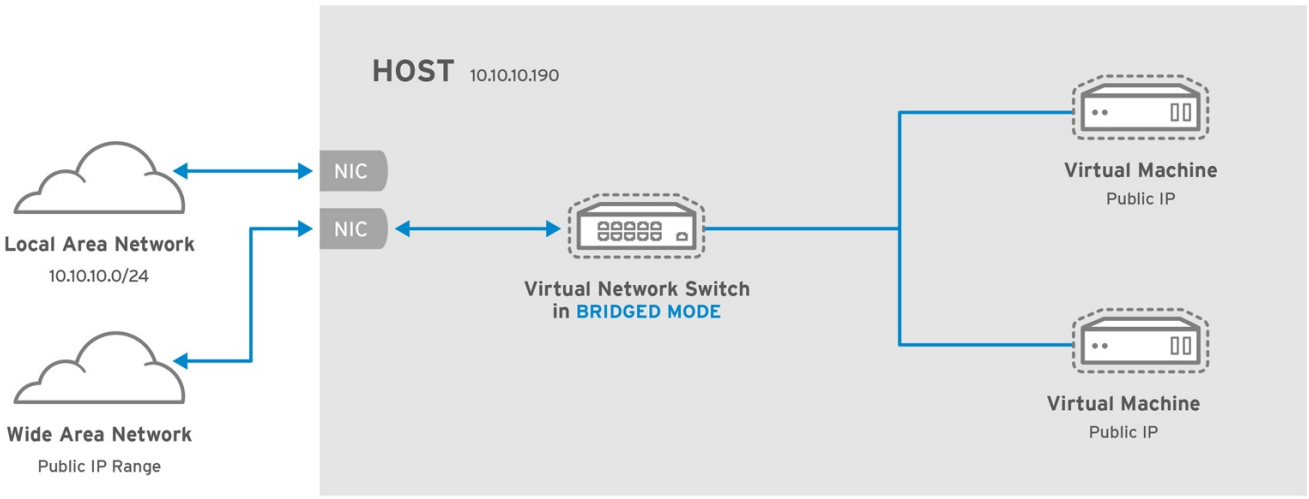


通常、DMZ のホスト物理マシンは、WAN (外部) ホスト物理マシンおよび LAN (内部) ホスト物理マシンにサービスを提供します。これは複数の場所からアクセスできる必要があり、この場所はセキュリティーおよび信頼レベルに基づいて異なる方法で制御および操作されるため、ルーティングモードはこの環境に最適な設定になります。

仮想サーバーのホスト

それぞれが 2 つの物理ネットワーク接続のある複数のホスト物理マシンを持つ仮想サーバーホスティング会社について考えてみます。管理とアカウンティングにはいずれかのインターフェイスが使用されており、もう 1 つは仮想マシンによる接続用です。各ゲストには独自のパブリック IP アドレスがありますが、ホストの物理マシンはプライベート IP アドレスを使用するため、ゲストの管理は内部管理者しか行えません。このシナリオを理解するには、以下の図を参照してください。

図18.9 仮想サーバーホスティングのサンプル設定



18.7.3. NAT モード

NAT (Network Address Translation) モードがデフォルトモードです。直接ネットワークの可視性が必要ない場合にテストに使用できます。

18.7.4. 分離モード

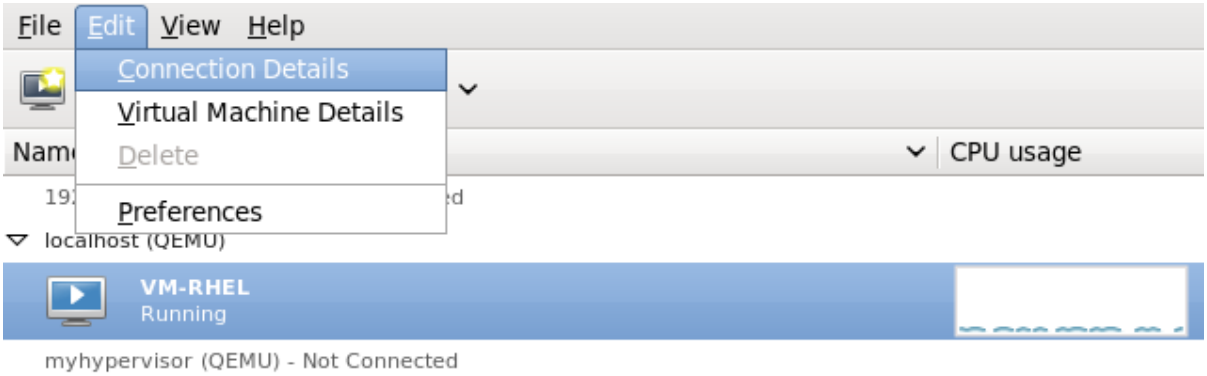
分離モードでは、仮想マシンが相互にのみ通信できます。物理ネットワークと対話することはできません。

18.8. 仮想ネットワークの管理

システムに仮想ネットワークを設定するには、次のコマンドを実行します。

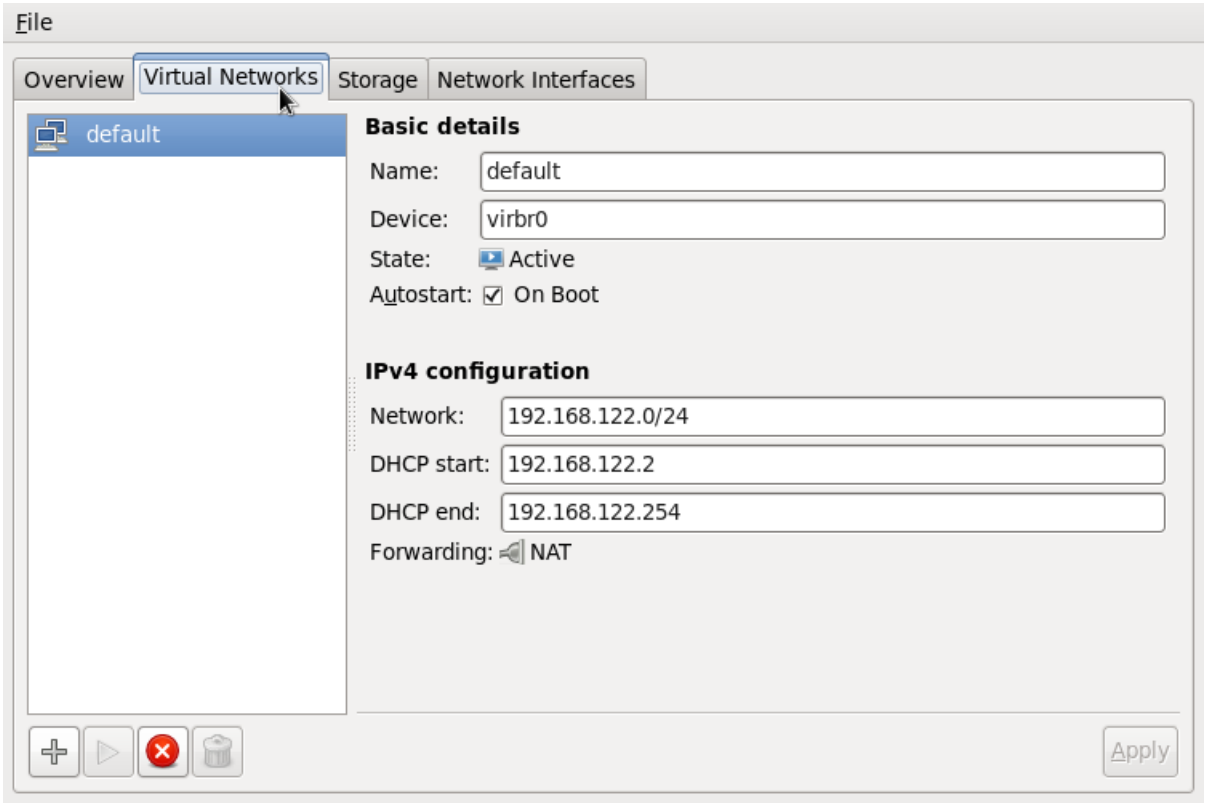
1. **Edit** メニューから、**Connection Details** を選択します。

図18.10 ホスト物理マシンの詳細の選択



2. これにより、Connection Details メニューーが開きます。**Virtual Networks** タブをクリックします。

図18.11 仮想ネットワークの設定



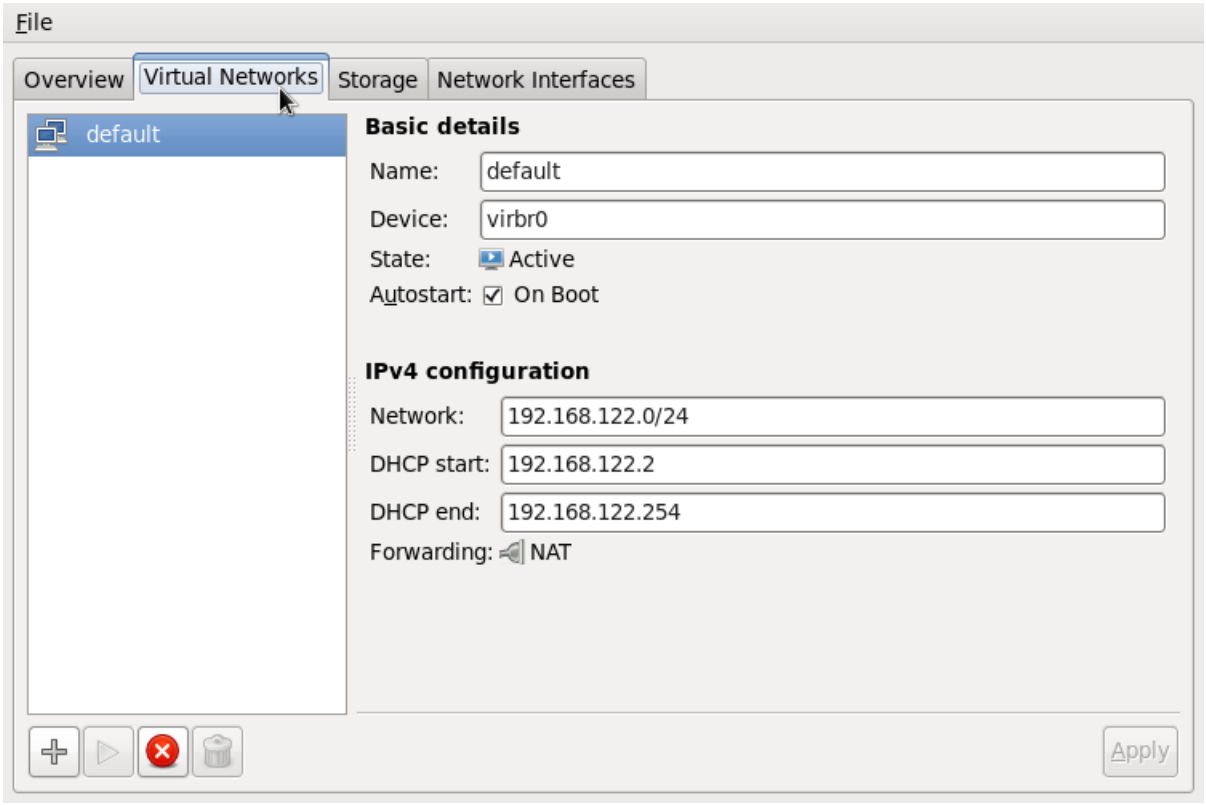
3. 使用可能なすべての仮想ネットワークは、メニューの左側のボックスに一覧表示されます。このボックスから仮想ネットワークを選択し、必要に応じて編集することで、仮想ネットワークの設定を編集できます。

18.9. 仮想ネットワークの作成

システムに仮想ネットワークを作成するには:

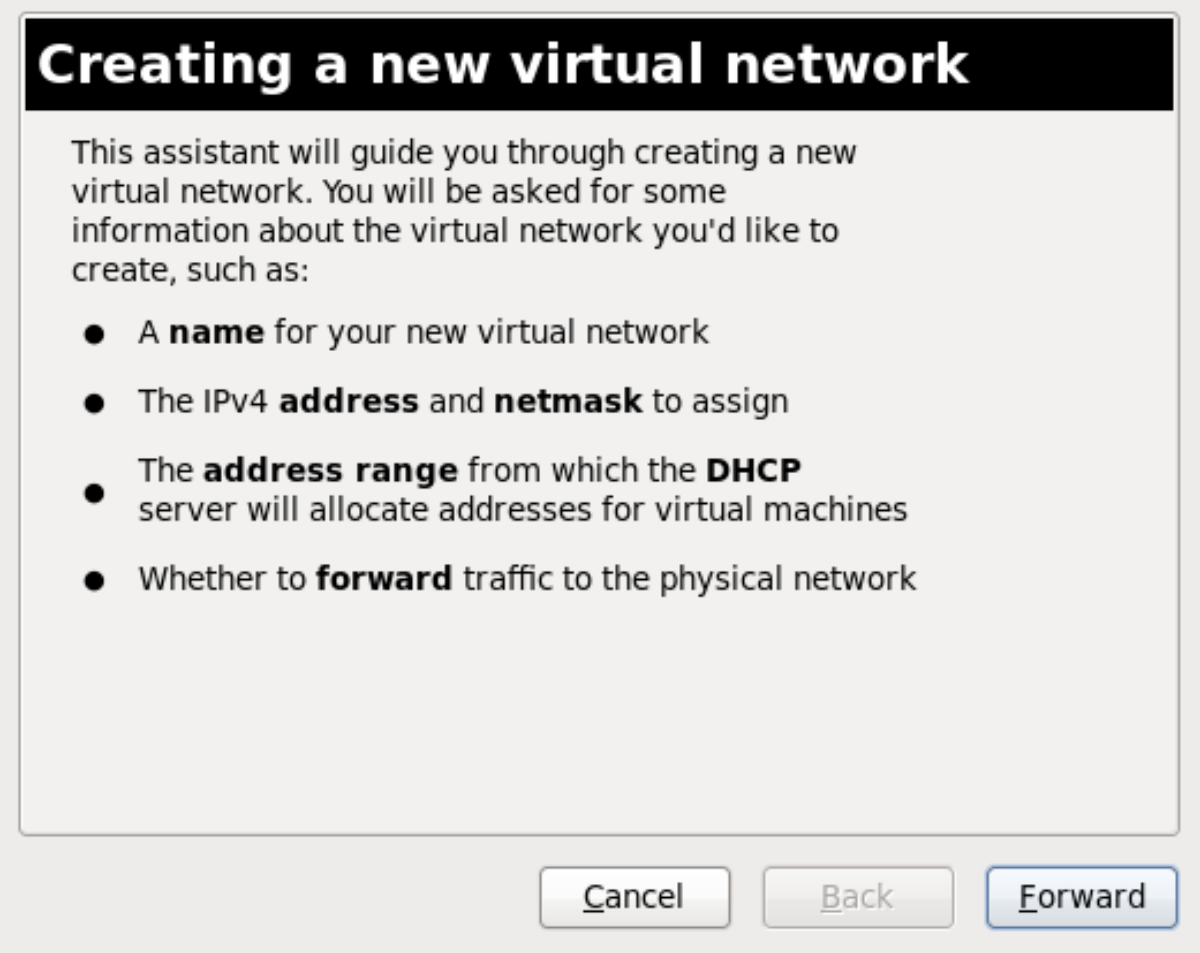
1. **Connection Details** メニューから **Virtual Networks** タブを開きます。プラス記号 (+) のアイコンで識別される **Add Network** ボタンをクリックします。詳細は、「仮想ネットワークの管理」を参照してください。

図18.12 仮想ネットワークの設定



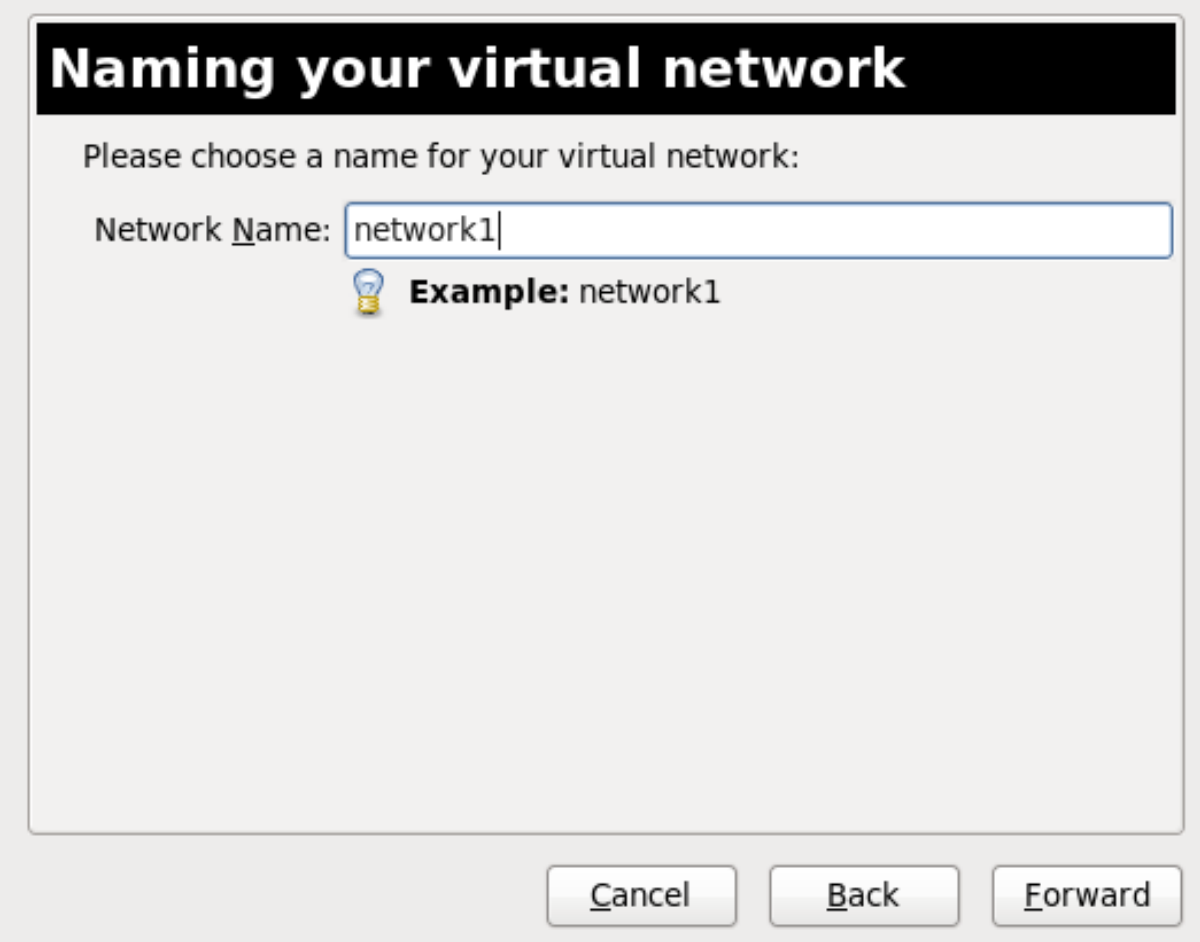
これにより、Create a new virtual networkウィンドウが開きます。**進む** をクリックして続けます。

図18.13 新しい仮想ネットワークの作成



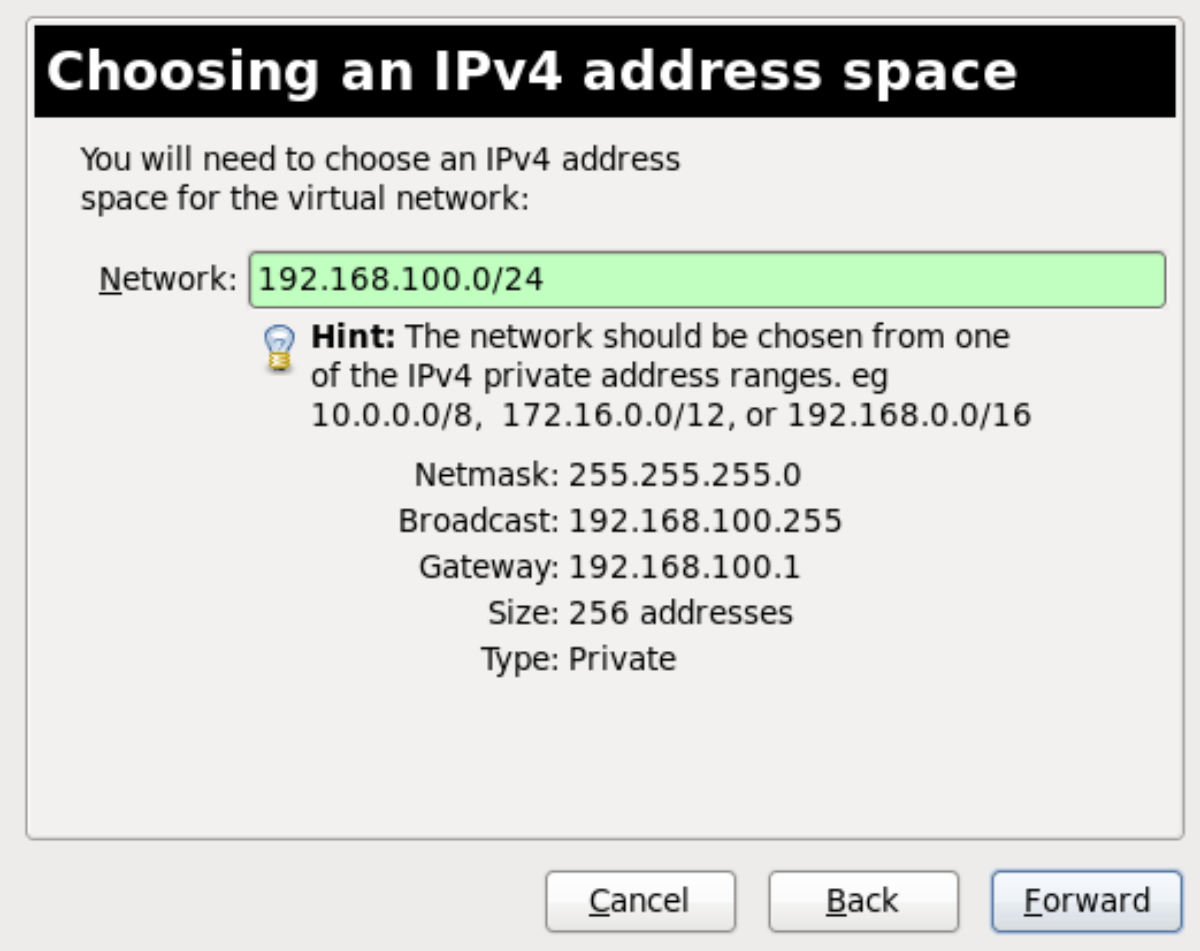
2. 仮想ネットワークに適切な名前を入力し、**Forward** をクリックします。

図18.14 仮想ネットワークに名前を付ける



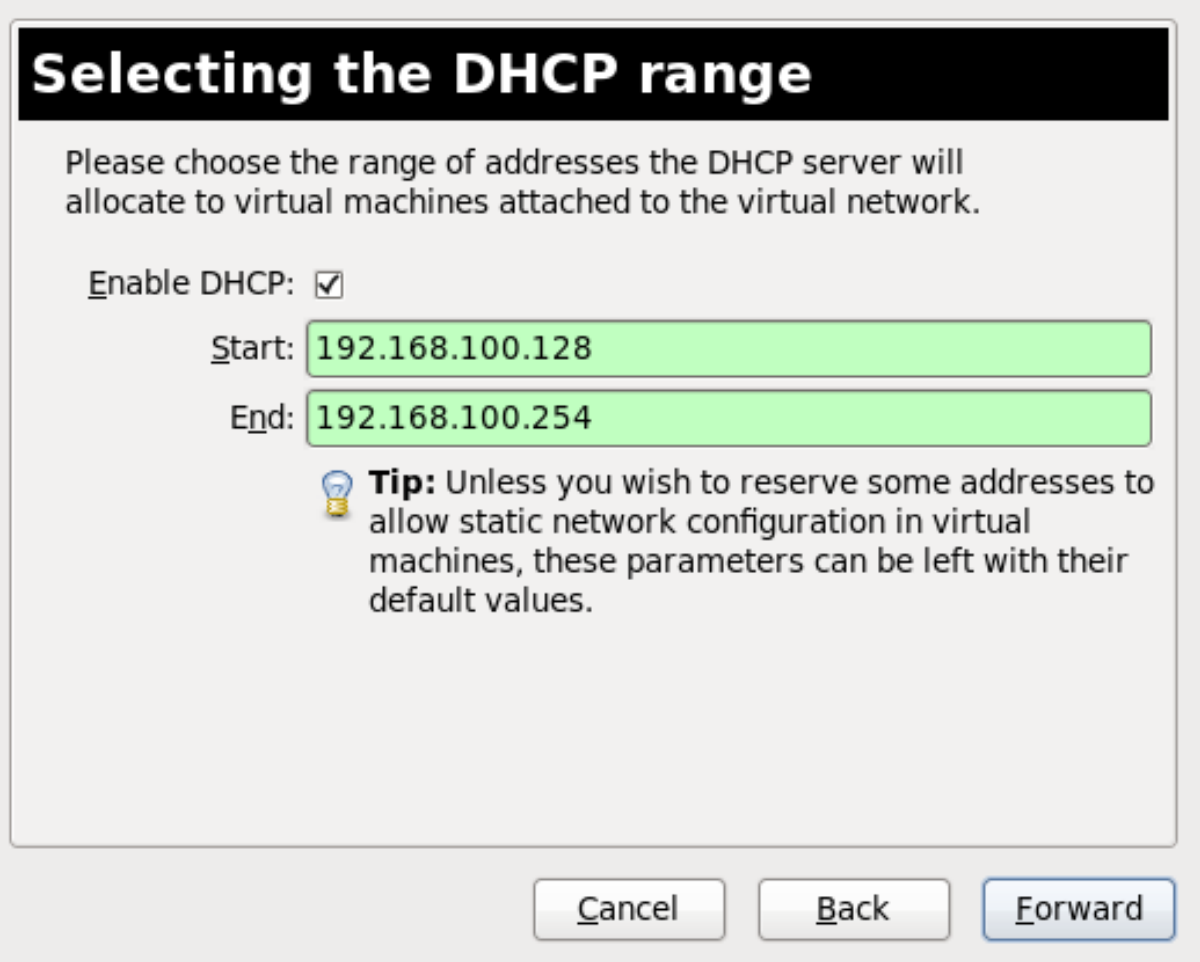
3. 仮想ネットワークの IPv4 アドレス空間を入力し、**Forward** をクリックします。

図18.15 IPv4 アドレス空間の選択



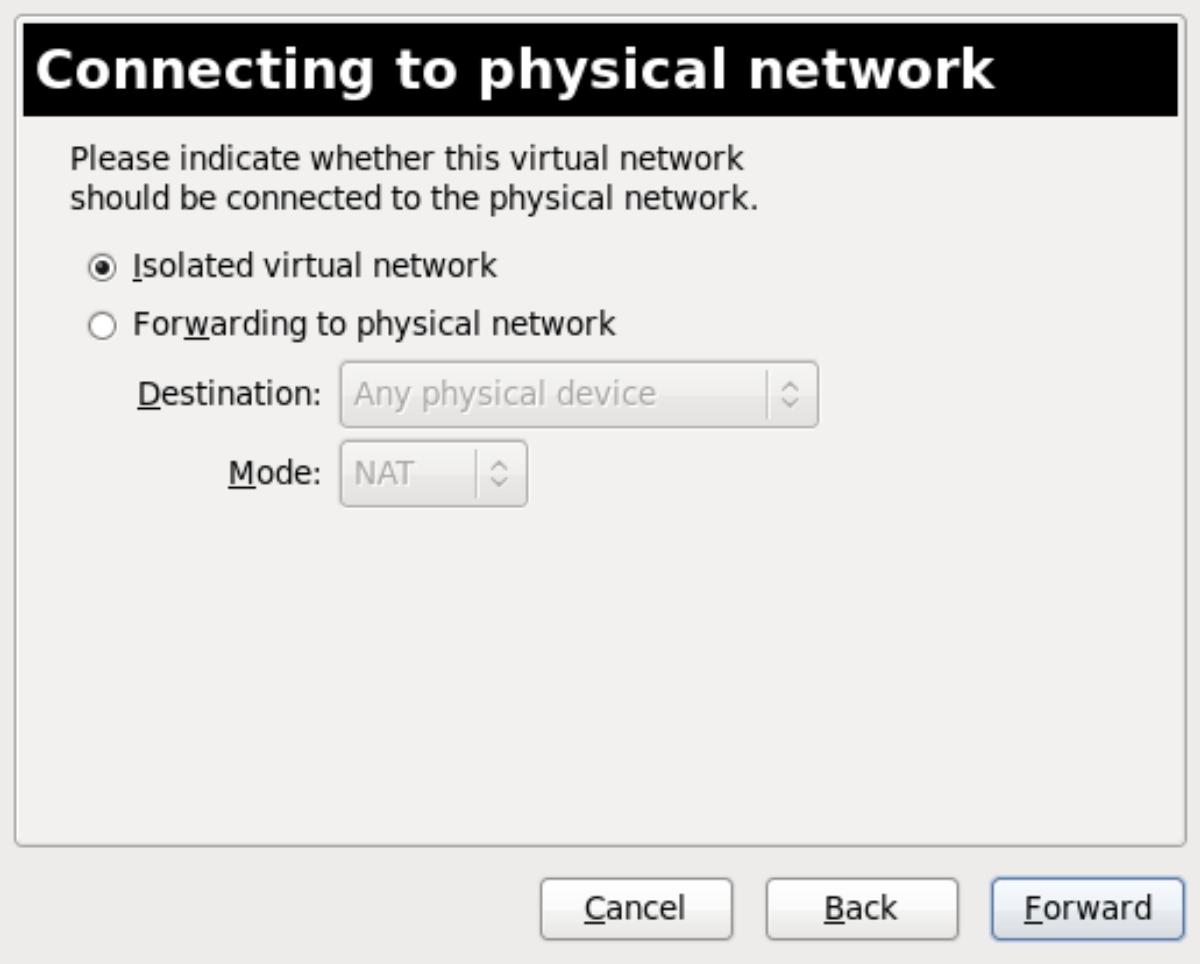
4. IP アドレスの **Start** 範囲および **End** 範囲を指定して、仮想ネットワークに DHCP 範囲を定義します。**進む** をクリックして続けます。

図18.16 DHCP 範囲の選択



5. 仮想ネットワークを物理ネットワークに接続する方法を選択します。

図18.17 物理ネットワークへの接続

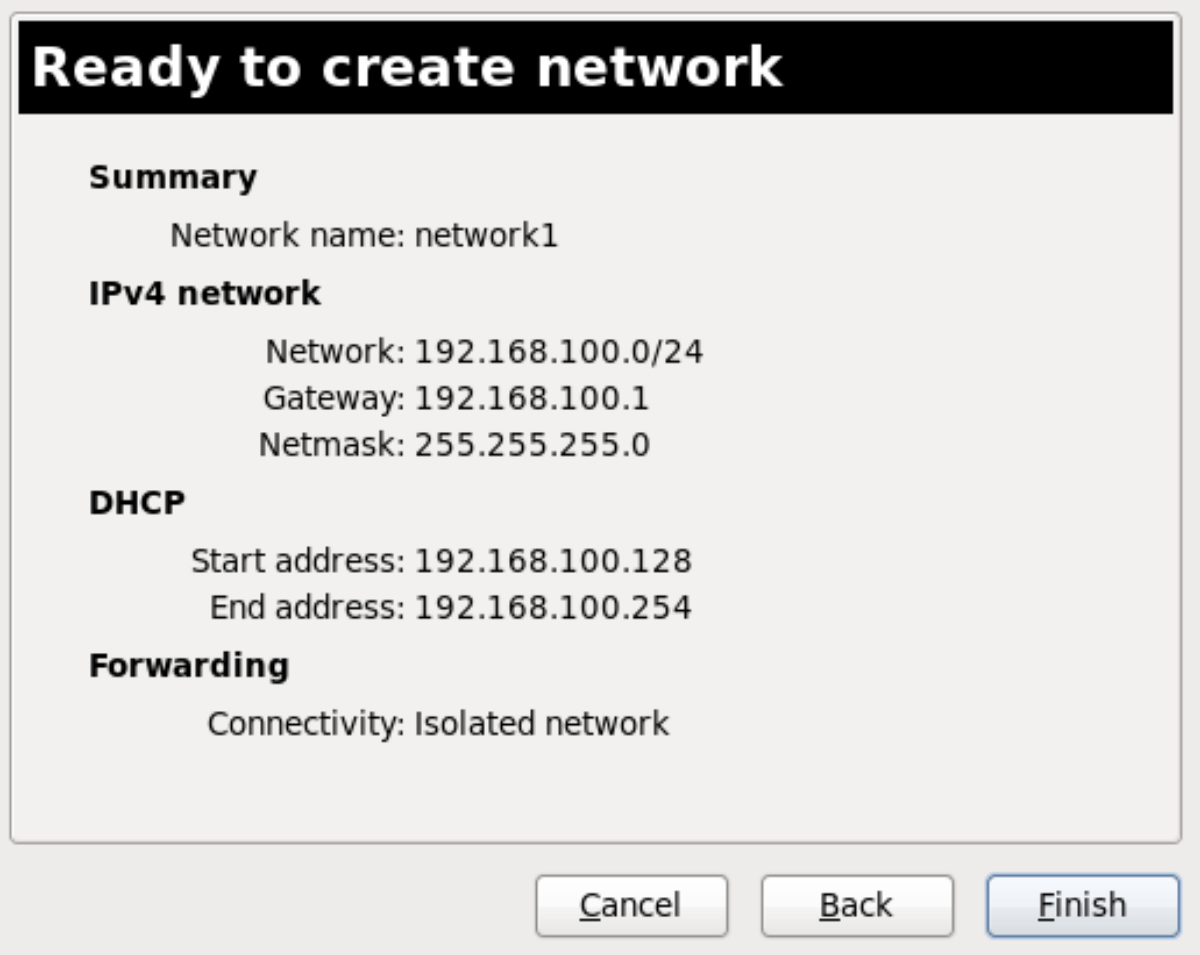


Forwarding to physical networkを選択した場合は、**Destination**を **Any physical device** にするか、特定の物理デバイスにするかを選択します。また、**Mode**を **NAT** にするか、**Routed** にするかを選択します。

進む をクリックして続けます。

6. これで、ネットワークを作成する準備が整いました。ネットワークの設定を確認し、**Finish** をクリックします。

図18.18 ネットワークを作成する準備ができました



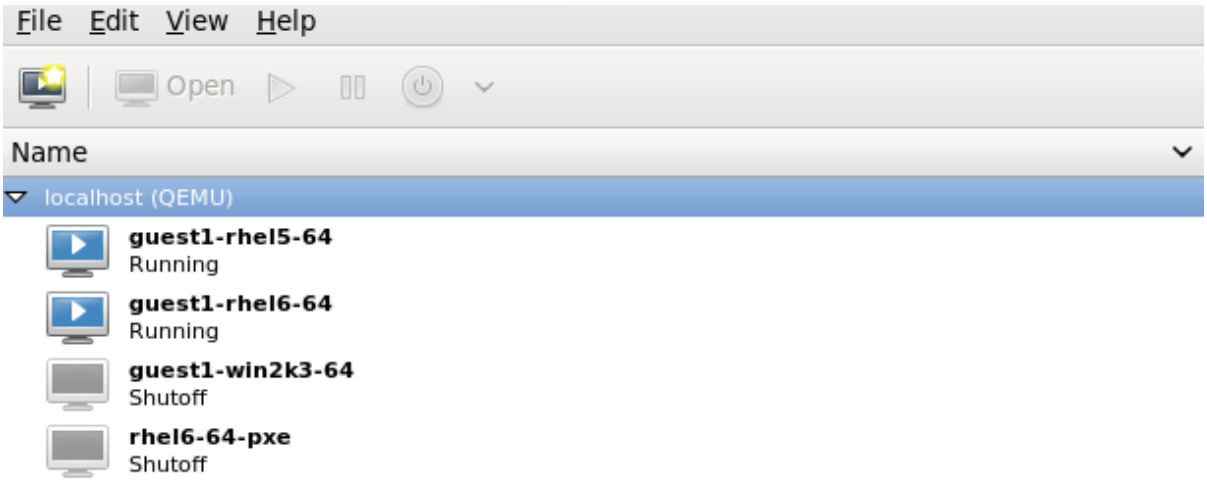
7. 新しい仮想ネットワークが、 **Connection Details** ウィンドウの **Virtual Networks** タブで利用できるようになります。

18.10. 仮想ネットワークのゲストへの割り当て

仮想ネットワークをゲストに接続するには、次のコマンドを実行します。

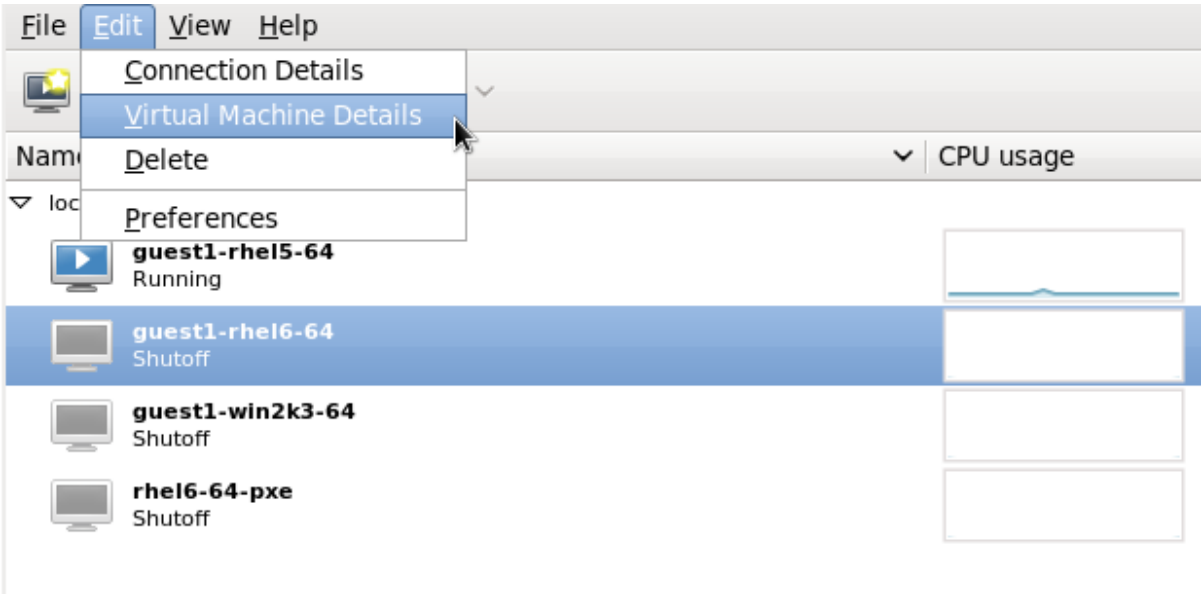
1. **Virtual Machine Manager** ウィンドウで、ネットワークが割り当てられるゲストを強調表示します。

図18.19 表示する仮想マシンの選択



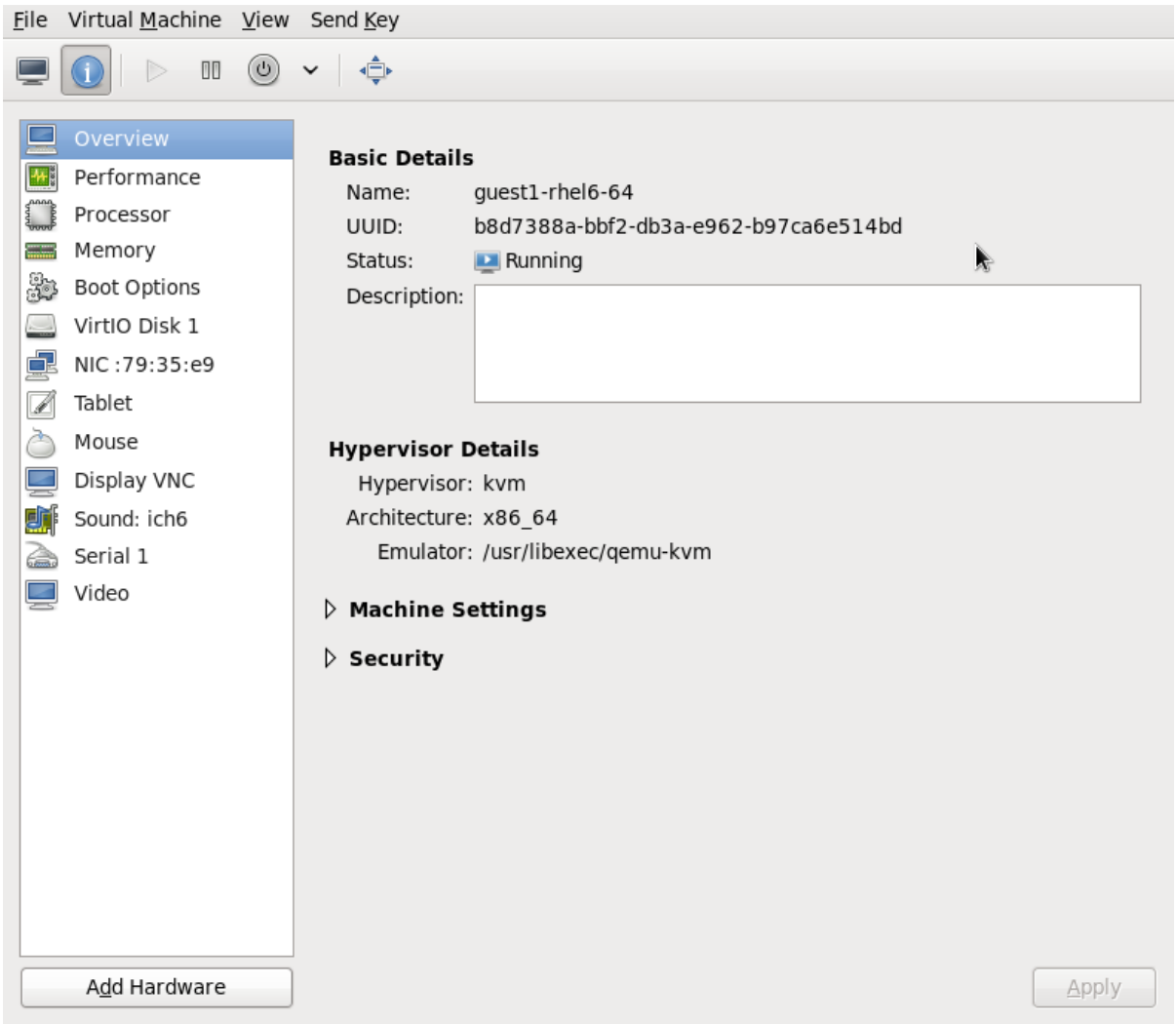
2. 仮想マシンマネージャーの**Edit** メニューから、**Virtual Machine Details** を選択します。

図18.20 仮想マシンの詳細の表示



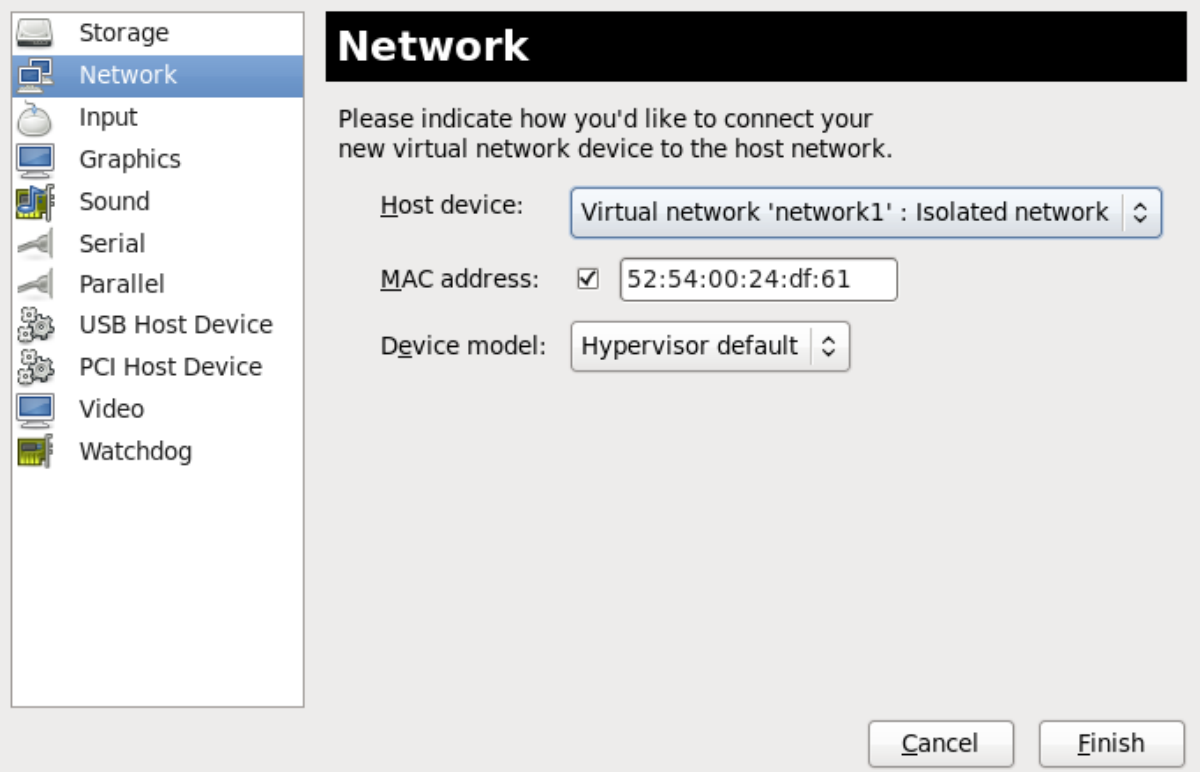
3. Virtual Machine Details ウィンドウの **Add Hardware** ボタンをクリックします。

図18.21 仮想マシンの詳細ウィンドウ



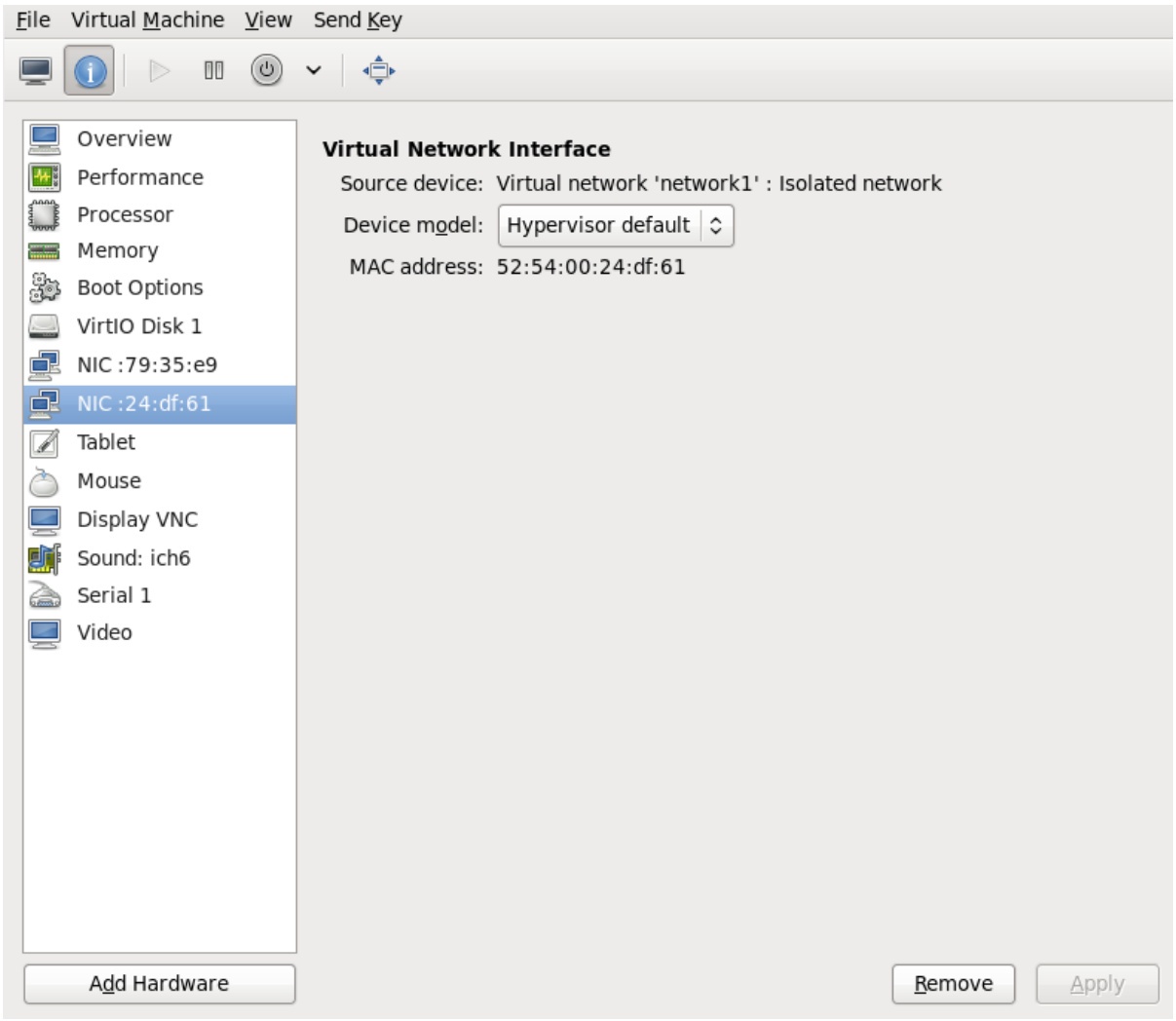
4. **Add new virtual hardware** ウィンドウで、左側のペインから **Network** を選択し、**Host device** メニューからネットワーク名 (この例では **network1**) を選択して、**Finish** をクリックします。

図18.22 Add new virtual hardware ウィンドウからネットワークを選択します



5. 新しいネットワークが、起動時にゲストに表示される仮想ネットワークインターフェイスとして表示されるようになりました。

図18.23 ゲストハードウェア一覧に新しいネットワークが表示されました。



18.11. 物理インターフェイスへの仮想 NIC の直接接続

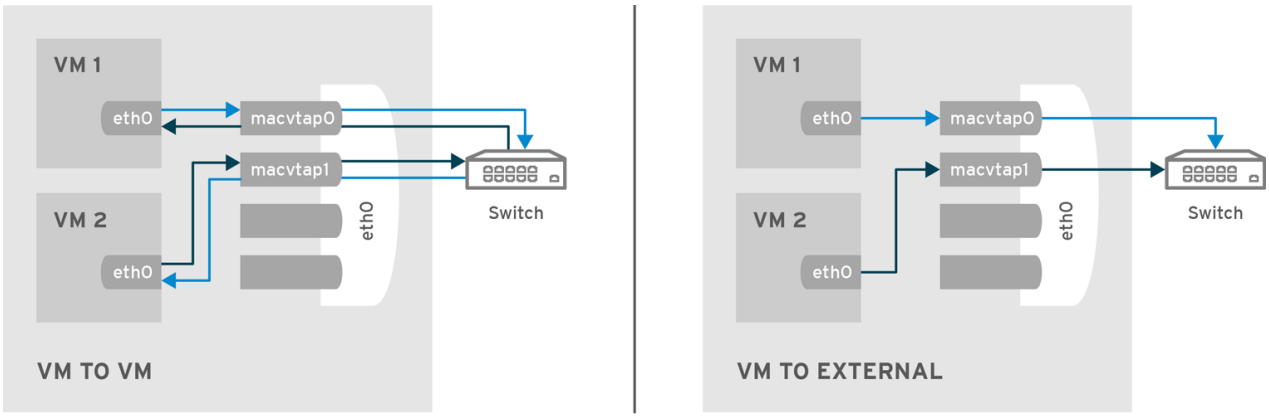
デフォルトの NAT 接続の代わりに、*macvtap* ドライバーを使用して、ゲストの NIC を、ホストマシンの指定された物理インターフェイスに直接接続できます。これを デバイスの割り当て (パススルーとも呼ばれます) と混同しないようにしてください。Macvtap 接続には次のモードがあり、それぞれに異なる利点とユースケースがあります。

物理インターフェイスの配信モード

VEPA

仮想イーサネットポートアグリゲーター (VEPA) モードでは、ゲストからのすべてのパケットが外部スイッチに送信されます。これにより、ユーザーはスイッチを介してゲストトラフィックを強制的に送信できます。VEPA モードが適切に機能するには、外部スイッチも **ヘアピンモード** に対応する必要があります。これにより、宛先がソースゲストと同じホストマシンのゲストであるパケットが、外部スイッチによりホストに返されます。

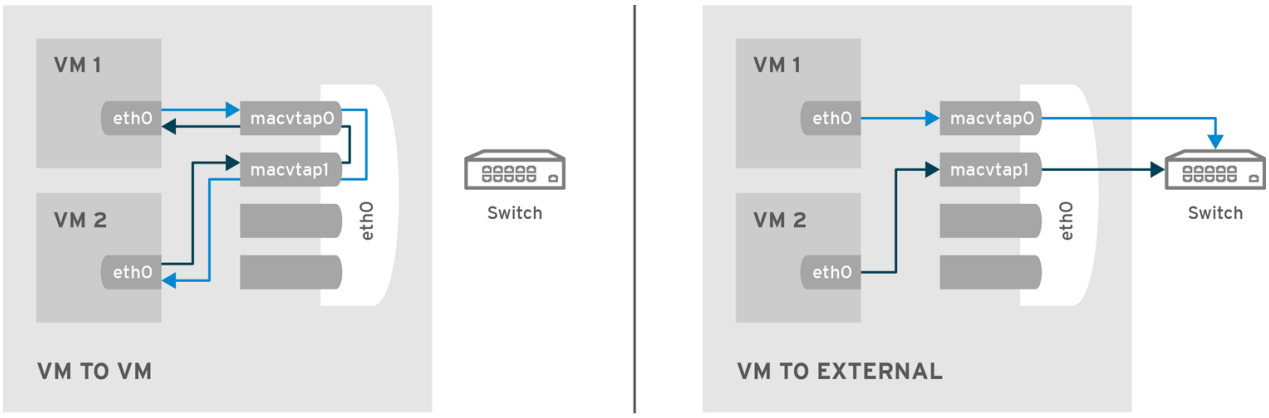
図18.24 VEPA モード



bridge

宛先がソースゲストと同じホストマシン上にあるパケットは、ターゲットの macvtap デバイスに直接配信されます。直接配信を成功させるには、ソースデバイスとターゲットデバイスの両方がブリッジモードになっている必要があります。いずれかのデバイスが VEPA モードの場合は、ヘアピン対応の外部スイッチが必要です。

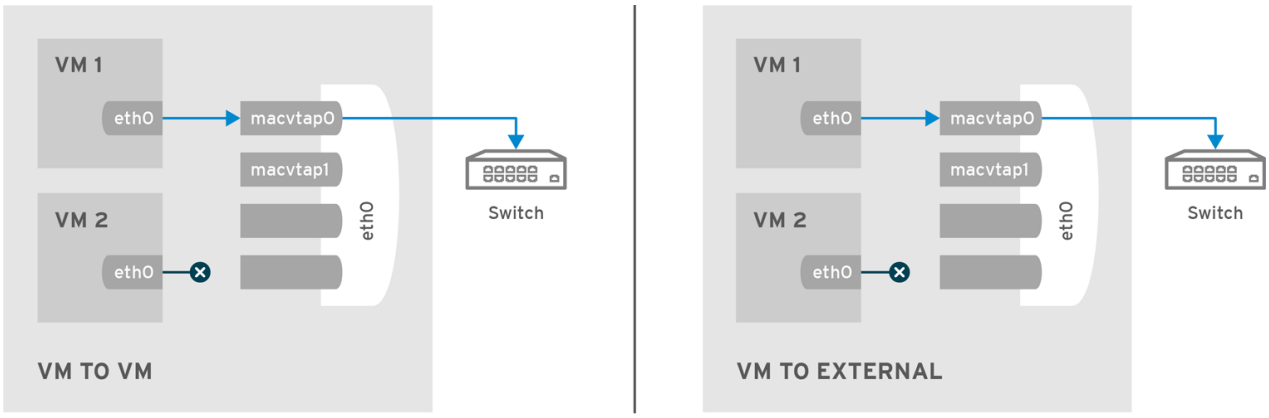
図18.25 ブリッジモード



プライベート

すべてのパケットは外部スイッチに送信されます。また、外部ルーターまたはゲートウェイを介して送信され、これらがホストに送り返す場合に、同じホストマシン上のターゲットゲストにのみ配信されます。プライベートモードを使用すると、1 台のホスト上にある個別の仮想マシン間での通信を阻止することができます。移行元デバイスまたは移行先デバイスのいずれかがプライベートモードの場合は、以下の手順が行われます。

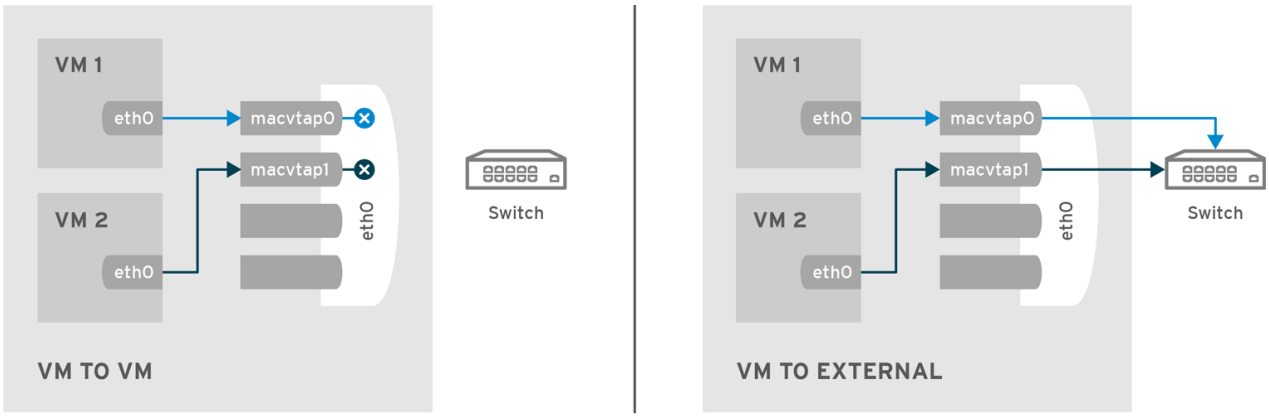
図18.26 プライベートモード



パススルー

この機能は、移行機能を失うことなく、物理インターフェイスデバイスまたはSR-IOV 仮想機能 (VF) をゲストに直接接続します。すべてのパケットは、指定されたネットワークデバイスに直接送信されます。パススルーモードではネットワークデバイスをゲスト間で共有できないため、ネットワークデバイスは 1 つのゲストにしか渡せないことに注意してください。

図18.27 passthrough モード



4 つのモードはそれぞれ、ドメイン XML ファイルを変更することによって設定されます。このファイルを開いたら、以下のようにモード設定を変更します。

```
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0' mode='vepa'/>
  </interface>
</devices>
```

直接接続のゲスト仮想マシンのネットワークアクセスは、ホスト物理マシンの物理インターフェイスが接続されているハードウェアスイッチによって管理できます。

スイッチが IEEE 802.1Qbg 規格に準拠している場合、インターフェイスには以下のような追加パラメーターを設定できます。virtualport 要素のパラメーターについては、IEEE 802.1Qbg 標準で詳細が説明されています。この値はネットワーク固有のもので、ネットワーク管理者が指定する必要があります。802.1Qbg の用語では、Virtual Station Interface (VSI) は仮想マシンの仮想インターフェイスを表します。

また、IEEE 802.1Qbg では VLAN ID にゼロ以外の値が必要です。また、スイッチが IEEE 802.1Qbh 標準に準拠している場合、値はネットワーク固有であり、ネットワーク管理者が提供する必要があります。

Virtual Station Interface のタイプ

managerid

VSI Manager ID は、VSI タイプおよびインスタンス定義を含むデータベースを識別します。これは整数値で、値 0 は予約されています。

typeid

VSI タイプ ID は、ネットワークアクセスを特徴付ける VSI タイプを識別します。VSI の種類は通常、ネットワーク管理者が管理します。これは整数値です。

typeidversion

VSI タイプバージョンでは、複数のバージョンの VSI タイプが許可されます。これは整数値です。

instanceid

VSI インスタンス ID 識別子は、VSI インスタンス (仮想マシンの仮想インターフェイス) が作成されると生成されます。これは、グローバルに一意の識別子です。

profileid

プロファイル ID には、このインターフェイスに適用されるポートプロファイルの名前が含まれます。この名前は、ポートプロファイルデータベースにより、ポートプロファイルからネットワークパラメーターに解決され、これらのネットワークパラメーターはこのインターフェイスに適用されます。

4 つのタイプはそれぞれ、ドメイン xml ファイルを変更することによって設定されます。このファイルを開いたら、以下のようにモード設定を変更します。

```
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0.2' mode='vepa'/>
    <virtualport type="802.1Qbg">
      <parameters managerid="11" typeid="1193047" typeidversion="2" instanceid="09b11c53-8b5c-
4eeb-8f00-d84eaa0aaa4f"/>
    </virtualport>
  </interface>
</devices>
```

プロファイル ID が表示されます。

```
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0' mode='private'/>
    <virtualport type='802.1Qbh'>
      <parameters profileid='finance'/>
    </virtualport>
```

```
  </interface>
</devices>
...
```

18.12. ネットワークフィルターの適用

本セクションは、libvirt のネットワークフィルターと、その目的、概念、および XML 形式の概要を説明します。

18.12.1. はじめに

ネットワークフィルターリングの目的は、仮想システムの管理者が、仮想マシンでネットワークトラフィックフィルターリングルールを設定および適用し、仮想マシンが送受信できるネットワークトラフィックのパラメーターを管理できるようにすることです。ネットワークトラフィックフィルターリングルールは、仮想マシンの起動時にホストの物理マシンに適用されます。フィルターリングルールは仮想マシン内からは回避できないため、仮想マシンユーザーの視点からは必須となります。

ゲスト仮想マシンの視点からは、ネットワークフィルターリングシステムにより、各仮想マシンのネットワークトラフィックフィルターリングルールをインターフェイスごとに個別に設定できます。このルールは、仮想マシンの起動時にホストの物理マシンに適用され、仮想マシンの実行中に変更できます。後者は、ネットワークフィルターの XML 記述を変更することで実現できます。

複数の仮想マシンが、同じ汎用ネットワークフィルターを使用できます。このフィルターを変更すると、このフィルターを参照する実行中の仮想マシンのネットワークトラフィックフィルターリングルールがすべて更新されます。起動時に、実行していないマシンが更新されます。

前述のように、ネットワークトラフィックフィルターリングルールを適用する場合は、特定のタイプのネットワーク設定に設定された個別のネットワークインターフェイスを使用できます。対応しているネットワークタイプは、次のとおりです。

- network
- ethernet -- bridging モードで使用する必要があります。
- bridge

例18.1 ネットワークフィルターリングの例

インターフェイス XML は、トップレベルフィルターを参照するために使用されます。以下の例では、インターフェイスの説明は、フィルターの clean-traffic を参照します。

```
<devices>
 <interface type='bridge'>
   <mac address='00:16:3e:5d:c7:9e'/>
   <filterref filter='clean-traffic'/>
 </interface>
</devices>
```

ネットワークフィルターは XML で記述されており、他のフィルターへの参照、トラフィックフィルターリングのルール、または両方の組み合わせを含むことができます。上記の参照フィルターの clean-traffic は、他のフィルターへの参照のみが含まれ、実際のフィルタールールがないフィルターです。他のフィルターへの参照も使用できるため、フィルターツリーを構築できます。clean-traffic フィルターは、**# virsh nwfilter-dumpxml clean-traffic** コマンドを使用して表示できます。

前述のように、1つのネットワークフィルターを、複数の仮想マシンで参照できます。インターフェ

イスは通常、各トラフィックフィルターリングルールに関連付けられた個々のパラメーターを持つため、フィルターの XML で説明されているルールは、変数を使用して一般化できます。この場合、フィルター XML では変数名が使用され、フィルターが参照される場所に名前と値が提供されます。

例18.2 説明の拡張

以下の例では、インターフェイスの説明が、パラメーター IP と、ドット付き IP アドレスを値として拡張されています。

```
<devices>
  <interface type='bridge'>
    <mac address='00:16:3e:5d:c7:9e'/>
    <filterref filter='clean-traffic'>
      <parameter name='IP' value='10.0.0.1'/>
    </filterref>
  </interface>
</devices>
```

この例では、clean-traffic ネットワークトラフィックフィルターは IP アドレスパラメーター 10.0.0.1 で表されます。ルールでは、このインターフェイスからのすべてのトラフィックが常にソース IP アドレスとして 10.0.0.1 を使用することを規定しており、これはこの特定のフィルターの目的の 1 つになります。

18.12.2. チェーンのフィルター

フィルターリングルールはフィルターチェーンで編成されています。このようなチェーンは、パケットフィルターリングルールを持つツリー構造を、個々のチェーン (分岐) のエントリーとして持つと考えることができます。

パケットは root チェーンでフィルター評価を開始し、他のチェーンで評価を継続し、これらのチェーンから root チェーンに戻るか、トラバースされたチェーンの 1 つでフィルタールールによりドロップまたは許可されます。

libvirt のネットワークフィルターリングシステムは、ユーザーがトラフィックフィルターリングのアクティブ化を選択する仮想マシンのネットワークインターフェイスごとに、個々の root チェーンを自動的に作成します。ユーザーは、root チェーンで直接インスタンス化されるフィルターリングルールを作成するか、プロトコル固有のルールを効率的に評価するためにプロトコル固有のフィルターリングチェーンを作成できます。

以下のチェーンが存在します。

- root
- mac
- stp (スパニングツリープロトコル)
- vlan
- arp と rarp
- ipv4

- ipv6

mac、stp、vlan、arp、rarp、ipv4、または ipv6 のプロトコルを評価する複数のチェーンは、プロトコル名をチェーン名の接頭辞としてのみ使用して作成できます。

例18.3 ARP トラフィックフィルターリング

この例では、名前が arp-xyz または arp-test のチェーンを指定し、そのチェーンで ARP プロトコルパケットが評価されるようにします。

以下のフィルター XML は、arp チェーンでの ARP トラフィックのフィルターリング例を示しています。

```
<filter name='no-arp-spoofing' chain='arp' priority='-500'>
  <uuid>f88f1932-debf-4aa1-9fbe-f10d3aa4bc95</uuid>
  <rule action='drop' direction='out' priority='300'>
    <mac match='no' srcmacaddr='$MAC'/>
  </rule>
  <rule action='drop' direction='out' priority='350'>
    <arp match='no' arpsrcmacaddr='$MAC'/>
  </rule>
  <rule action='drop' direction='out' priority='400'>
    <arp match='no' arpsrcipaddr='$IP'/>
  </rule>
  <rule action='drop' direction='in' priority='450'>
    <arp opcode='Reply'/>
    <arp match='no' arpdstmacaddr='$MAC'/>
  </rule>
  <rule action='drop' direction='in' priority='500'>
    <arp match='no' arpdstipaddr='$IP'/>
  </rule>
  <rule action='accept' direction='inout' priority='600'>
    <arp opcode='Request'/>
  </rule>
  <rule action='accept' direction='inout' priority='650'>
    <arp opcode='Reply'/>
  </rule>
  <rule action='drop' direction='inout' priority='1000'/>
</filter>
```

root チェーンなどではなく、ARP 固有のルールを arp チェーンに配置すると、ARP 以外のパケットプロトコルは ARP プロトコル固有のルールで評価される必要がなくなります。これにより、トラフィックフィルターリングの効率が向上します。ただし、その他のルールは評価されないため、指定したプロトコルのフィルターリングルールのみをチェーンに組み込むことに注意する必要があります。たとえば、IPv4 プロトコルパケットは ARP チェーンを通過しないため、IPv4 ルールは ARP チェーンで評価されません。

18.12.3. チェーンの優先度のフィルターリング

前述のように、フィルタールールを作成すると、すべてのチェーンが root チェーンに接続されます。これらのチェーンへのアクセス順序は、チェーンの優先度の影響を受けます。優先度を割り当てることができるチェーンと、そのデフォルトの優先度を以下の表に示します。

表18.1 チェーンのデフォルトの優先度の値のフィルターリング

チェーン (接頭辞)	デフォルトの優先度
stp	-810
mac	-800
vlan	-750
ipv4	-700
ipv6	-600
arp	-500
rarp	-400

注記

優先度が低いチェーンは、優先度が高いチェーンの前にアクセスされます。

表18.1「チェーンのデフォルトの優先度の値のフィルターリング」 に記載されているチェーンは、[-1000 から 1000] の範囲の値をフィルターノードの priority (XML) 属性に書き込むことで、カスタム優先度を割り当てることもできます。「チェーンのフィルター」filter は、arp チェーンのデフォルト優先度 -500 を表示します。以下に例を示します。

18.12.4. フィルターにおける変数の使用

ネットワークトラフィックフィルターサブシステム (MAC と IP) で使用するために予約されている変数には、次の 2 つがあります。

MAC は、ネットワークインターフェイスの MAC アドレスに指定されます。この変数を参照するフィルターリングルールは、自動的にインターフェイスの MAC アドレスに置き換えられます。これは、MAC パラメーターを明示的に指定する必要なく機能します。上記の IP パラメーターに似た MAC パラメーターを指定できる場合でも、libvirt は、インターフェイスが使用する MAC アドレスを認識しているため、推奨できません。

パラメーター***IP*** は、仮想マシン内のオペレーティングシステムが、指定のインターフェイスで使用する IP アドレスを表します。パラメーターが明示的に指定されずに参照されている場合に、インターフェイスで使用されている IP アドレス (つまり IP パラメーターの値) を libvirt デーモンが判断しようとする限り、IP パラメーターは特別なものとなります。現在の IP アドレス検出の制限は、この機能の使用方法と、その使用時に想定される制限に関する「制限事項」を参照してください。「チェーンのフィルター」 に示す XML ファイルには、フィルター ***no-arp-spoofing*** が含まれています。これは、ネットワークフィルター XML を使用して MAC 変数および IP 変数を参照する例です。

参照変数の前には、常に ***$*** という文字が付くことに注意してください。変数の値の形式は、XML で識別されるフィルター属性で期待されるタイプである必要があります。上記の例では、***IP*** パラメーターは、標準形式の正しい IP アドレスを保持する必要があります。正しい構造を指定しないと、フィルター変数が値に置き換わりません。また、ホットプラグの使用時に、仮想マシンの起動が妨げられたり、インターフェイスの接続が妨げられたりします。各 XML 属性で期待されるタイプの一部を、サンプル例18.4「サンプルの変数の種類」に示します。

例18.4 サンプルの変数の種類

変数には要素の一覧が含まれる場合があります (変数 IP には、特定のインターフェイスで有効な 複数 IP アドレスを指定できるなど)。IP 変数に複数の要素を指定する表記は以下のようになります。

```
<devices>
  <interface type='bridge'>
    <mac address='00:16:3e:5d:c7:9e'/>
    <filterref filter='clean-traffic'>
      <parameter name='IP' value='10.0.0.1'/>
      <parameter name='IP' value='10.0.0.2'/>
      <parameter name='IP' value='10.0.0.3'/>
    </filterref>
  </interface>
</devices>
```

この XML ファイルは、インターフェイスごとに複数の IP アドレスを有効にするフィルターを作成します。各 IP アドレスは、個別のフィルターリングルールになります。そのため、上記の XML および以下のルールを使用して、3 つの個別のフィルターリングルール (各 IP アドレスに 1 つ) が作成されます。

```
<rule action='accept' direction='in' priority='500'>
  <tcp srpipaddr='$IP'/>
</rule>
```

要素の一覧を含む変数の個々の要素にアクセスできるように、以下のようなフィルターリングルールは、変数*DSTPORTS* の 2 番目の要素にアクセスします。

```
<rule action='accept' direction='in' priority='500'>
  <udp dstportstart='$DSTPORTS[1]'/>
</rule>
```

例18.5 さまざまな変数の使用

$VARIABLE[@<iterator id="x">] 表記を使用して、異なるリストから可能なルールの組み合わせをすべて表示するフィルターリングルールを作成できます。以下の規則では、仮想マシンが、*DSTPORTS* で指定された一連のポートで、*SRCIPADDRESSES* で指定された送信元 IP アドレスのセットからトラフィックを受信できるようにします。この規則では、2 つの独立したイテレーターを使用して、変数 *DSTPORTS* の要素と *SRCIPADDRESSES* の要素の組み合わせをすべて生成します。

```
<rule action='accept' direction='in' priority='500'>
  <ip srcipaddr='$SRCIPADDRESSES[@1]' dstportstart='$DSTPORTS[@2]'/>
</rule>
```

以下のように、*SRCIPADDRESSES* および *DSTPORTS* に具体的な値を割り当てます。

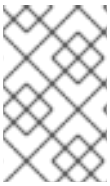
```
SRCIPADDRESSES = [ 10.0.0.1, 11.1.2.3 ]
DSTPORTS = [ 80, 8080 ]
```

$SRCIPADDRESSES[@1] および **$DSTPORTS[@2]** を使用して変数に値を割り当てると、次に示すように、アドレスおよびポートの組み合わせがすべて作成されます。

- 10.0.0.1, 80
- 10.0.0.1, 8080
- 11.1.2.3, 80
- 11.1.2.3, 8080

表記法 **$SRCIPADDRESSES[@1]** と **$DSTPORTS[@1]** を使用するなど、1 つのイテレーターを使用して同じ変数にアクセスすると、両方のリストにパラレルアクセスが行われ、結果が以下のようになります。

- 10.0.0.1, 80
- 11.1.2.3, 8080



注記

$VARIABLE は、**$VARIABLE[@0]** の省略形です。以前の表記法は、このセクションの上部にある最初の段落に示されているように、**iterator id="0"** が追加されたイテレーターのロールを常に想定しています。

18.12.5. 自動 IP アドレス検出と DHCP スヌーピング

本セクションでは、自動 IP アドレス検出および DHCP スヌーピングに関する情報を提供します。

18.12.5.1. はじめに

仮想マシンのインターフェイスで使用される IP アドレスの検出は、変数 IP が参照されているにもかかわらず、その変数に値が割り当てられていない場合に、自動的にアクティブになります。変数 ***CTRL_IP_LEARNING*** を使用して、使用する IP アドレスの学習方法を指定できます。有効な値は、*any*、*dhcp*、または *none* です。

any 値は、仮想マシンが使用しているアドレスを判断するために、libvirt にパケットを使用するように指示します。これは、変数 ***TRL_IP_LEARNING*** が設定されていない場合のデフォルト設定です。この方法では、インターフェイスごとに 1 つの IP アドレスのみが検出されます。ゲスト仮想マシンの IP アドレスが検出されると、IP アドレススプーフィングがフィルターのいずれかで阻止された場合などに、そのアドレスへの IP ネットワークトラフィックがロックされます。この場合、仮想マシンのユーザーはゲスト仮想マシン内のインターフェイスで IP アドレスを変更できません。このインターフェイスは、IP アドレスのスプーフィングと見なされます。ゲスト仮想マシンが別のホスト物理マシンに移行されるか、サスペンド操作後に再開されると、ゲスト仮想マシンによって送信される最初のパケットによって、ゲスト仮想マシンが特定のインターフェイスで使用できる IP アドレスが再度決定されます。

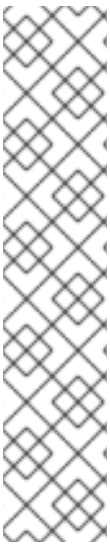
dhcp の値は、有効なリースを持つ DHCP サーバーが割り当てられたアドレスのみを有効にするように libvirt に指示します。この方法は、インターフェイスごとに複数の IP アドレスの検出と使用をサポートします。一時停止の操作後にゲスト仮想マシンが再開すると、有効な IP アドレスリースがそのフィルターに適用されます。これを行わないと、ゲスト仮想マシンは DHCP を使用して新しい IP アドレスを取得することが期待されます。ゲスト仮想マシンを別の物理ホストの物理マシンに移行する場合は、ゲスト仮想マシンが DHCP プロトコルを再実行する必要があります。

CTRL_IP_LEARNING を *none* に設定すると、libvirt は、明示的な値を割り当てずに IP アドレスの学習および参照を行いません。これはエラーになります。

18.12.5.2. DHCP スヌーピング

CTRL_IP_LEARNING=*dhcp* (DHCP スヌーピング) では、追加のスプーフィング対策セキュリティーが提供されます。これは、信頼できる DHCP サーバーのみが IP アドレスを割り当てることができるフィルターと併用できます。これを有効にするには、変数 ***DHCPSERVER*** を、有効な DHCP サーバーの IP アドレスに設定し、この変数を使用して着信 DHCP 応答をフィルターリングするフィルターを提供します。

DHCP スヌーピングが有効で、DHCP リースが期限切れになると、ゲスト仮想マシンは、DHCP サーバーから新しい有効なリースを取得するまで IP アドレスを使用できなくなります。ゲスト仮想マシンを移行する場合は、(仮想マシンインターフェイスをダウンして再度起動するなどして) IP アドレスを使用するために、新しい有効な DHCP リースを取得する必要があります。



注記

自動 DHCP 検出は、ゲスト仮想マシンがインフラストラクチャーの DHCP サーバーと交換する DHCP トラフィックをリッスンします。libvirt に対するサービス拒否攻撃を回避するために、これらのパケットの評価の速度が制限されます。つまり、インターフェイスで 1 秒あたりに過剰な数の DHCP パケットを送信するゲスト仮想マシンでは、これらのパケットがすべて評価されないため、フィルターが調整されない可能性があります。通常の DHCP クライアントの動作は、1 秒あたりの DHCP パケット数が少ないことが想定されます。さらに、インフラストラクチャー内のすべてのゲスト仮想マシンに適切なフィルターを設定して、DHCP パケットを送信できないようにすることが重要です。したがって、ゲスト仮想マシンがポート 67 からポート 68 への UDP および TCP のトラフィックを送信しないようにするか、DHCPSERVER 変数をすべてのゲスト仮想マシンで使用して、DHCP サーバーのメッセージが信頼された DHCP サーバーからの送信のみを許可するように制限する必要があります。同時に、サブネット内のすべてのゲスト仮想マシンで スプーフィング対策を有効にする必要があります。

例18.6 DHCP スヌーピングに対する IP のアクティブ化

以下の XML は、DHCP スヌーピング方法を使用した IP アドレス学習のアクティベーションの例を示しています。

```
<interface type='bridge'>
  <source bridge='virbr0'/>
  <filterref filter='clean-traffic'>
    <parameter name='CTRL_IP_LEARNING' value='dhcp'/>
  </filterref>
</interface>
```

18.12.6. 予約されている変数

表18.2「予約変数」 は、予約済みと見なされ、libvirt が使用する変数を示しています。

表18.2 予約変数

変数名	定義
MAC	インターフェイスの MAC アドレス
IP	インターフェイスで使用されている IP アドレスの一覧

変数名	定義
IPV6	現在実装されていません。インターフェイスで使用中の IPV6 アドレスの一覧を表示します。
DHCPSERVER	信頼できる DHCP サーバーの IP アドレスの一覧
DHCPSERVERV6	現在実装されていない: 信頼できる DHCP サーバーの IPv6 アドレスのリスト
CTRL_IP_LEARNING	IP アドレス検出モードの選択

18.12.7. 要素と属性の概要

すべてのネットワークフィルターに必要な root 要素には、2 つの属性を持つ **<filter>** という名前が付けられています。**name** 属性は、指定されたフィルターの一意の名前を提供します。**chain** 属性は任意ですが、基本となるホスト物理マシンのファイアウォールサブシステムを使用すると、効率的に処理するために、特定のフィルターをより適切に編成できます。現在、システムがサポートしているのは、**root**、**ipv4**、**ipv6**、**arp**、および **rarp** のチェーンのみです。

18.12.8. その他のフィルターの参照

任意のフィルターは、他のフィルターへの参照を保持できます。各フィルターは、フィルターツリーで複数回参照できますが、フィルター間の参照ではループが発生しません。

例18.7 クリーンなトラフィックフィルターの例

以下は、その他のいくつかのフィルターを参照するクリーントラフィックネットワークフィルターの XML を示しています。

```
<filter name='clean-traffic'>
  <uuid>6ef53069-ba34-94a0-d33d-17751b9b8cb1</uuid>
  <filterref filter='no-mac-spoofing'/>
  <filterref filter='no-ip-spoofing'/>
  <filterref filter='allow-incoming-ipv4'/>
  <filterref filter='no-arp-spoofing'/>
  <filterref filter='no-other-l2-traffic'/>
  <filterref filter='qemu-announce-self'/>
</filter>
```

別のフィルターを参照するには、フィルターノード内に XML ノード filterref を提供する必要があります。このノードには、参照するフィルターの名前が含まれる属性フィルターが必要です。

新しいネットワークフィルターはいつでも定義でき、libvirt には知られていないネットワークフィルターへの参照を含むことができます。ただし、仮想マシンを起動するか、フィルターを参照しているネットワークインターフェイスをホットプラグすると、フィルターツリーの全ネットワークフィルターが利用可能になります。そうしないと、仮想マシンが起動しなくなり、ネットワークインターフェイスが接続できなくなります。

18.12.9. フィルタールール

次の XML は、発信 IP パケットの IP アドレス (変数 IP の値によって提供される) が予期されたものではない場合に、トラフィックをドロップするルールを実装するネットワークトラフィックフィルターの簡単な例を示しています。これにより、VM による IP アドレスのスプーフィングを防ぎます。

例18.8 ネットワークトラフィックフィルターリングの例

```
<filter name='no-ip-spoofing' chain='ipv4'>
  <uuid>fce8ae33-e69e-83bf-262e-30786c1f8072</uuid>
  <rule action='drop' direction='out' priority='500'>
    <ip match='no' srcipaddr='$IP'/>
  </rule>
</filter>
```

トラフィックフィルターリングルールは、ルールノードで開始します。このノードには、次の属性のうち最大 3 つを含むことができます。

- action は必須で、次の値を指定できます。
  - drop (ルールに一致すると、分析は行われずにパケットが静かに破棄されます)
  - reject (ルールを一致させると、分析は行われずに ICMP 拒否メッセージが生成されます)
  - accept (ルールを一致させるとパケットを受け付けますが、これ以上分析は行いません)
  - return (ルールを一致させるとこのフィルターが渡されますが、制御を呼び出しフィルターに戻して詳細な分析を行います)
  - continue (ルールの一致は次のルールに進み、詳細な分析が行われます)
- direction は必須で、次の値を指定できます。
  - 着信トラフィックには in
  - 送信トラフィックには out
  - 着信および送信トラフィックには inout
- priority は任意です。ルールの優先度は、他のルールに関連してルールがインスタンス化される順序を制御します。値が低いルールは、値が大きいルールの前にインスタンス化されます。有効な値の範囲は、-1000 から 1000 です。この属性を指定しないと、優先度 500 がデフォルトで割り当てられます。root チェーンのフィルターリングルールは、優先順位に従って、root チェーンに接続されたフィルターでソートルールが分類されることに注意してください。これにより、フィルターリングルールとフィルターチェーンへのアクセスをインターリーブできます。詳細は、「チェーンの優先度のフィルターリング」 を参照してください。
- statematch はオプションです。設定可能な値は 0 または false で、基本的な接続状態の一致を無効にします。デフォルトの設定は true または 1 です。

詳しくは、「高度なフィルター設定のトピック」を参照する。

上記の例の例18.7「クリーンなトラフィックフィルターの例」は、*type ip* のトラフィックがチェーン*ipv4*に関連付けられ、ルールが **priority=*500*** であることを示しています。たとえば、別のフィルターが参照されており、そのトラフィックの *type ip* がチェーン*ipv4* にも関連付けられている場合は、そのフィルターのルールが、示されているルールの **priority=*500*** を基準にして順序付けされます。

ルールには、トラフィックをフィルターリングするためのルールを 1 つだけ含めることができます。上記の例は、タイプ ip のトラフィックがフィルターリングされることを示しています。

18.12.10. サポートされているプロトコル

以下のセクションでは、ネットワークフィルターサブシステムが対応しているプロトコルを一覧表示し、詳細を示します。このタイプのトラフィックルールは、ルールノードでネストされたノードとして提供されます。ルールがフィルターリングするトラフィックタイプに応じて、属性は異なります。上記の例は、ip トラフィックフィルターリングノード内で有効な単一の属性 ***srcipaddr*** を示しています。以下のセクションでは、どの属性が有効で、どのタイプのデータが期待されるかを示します。使用可能なデータ型は以下のとおりです。

- UINT8 : 8 ビット整数。範囲 0〜255
- UINT16 : 16 ビット整数。範囲 0〜65535
- MAC_ADDR: 00:11:22:33:44:55 などのドット付き 10 進形式の MAC アドレス
- MAC_MASK: FF:FF:FF:FC:00:00 などの MAC アドレス形式の MAC アドレスマスク
- IP_ADDR: 10.1.2.3 などのドット付き 10 進形式の IP アドレス
- P_MASK: ドット付き 10 進数形式 (255.255.248.0) または CIDR マスク (0-32) の IP アドレスマスク
- IPV6_ADDR: FFFF::1 などの数値形式の IPv6 アドレス
- IPV6_MASK: 数値形式の IPv6 マスク (FFFF:FFFF:FC00::) または CIDR マスク (0-128)
- STRING: 文字列
- BOOLEAN - 'true'、'yes'、'1'、または 'false'、'no'、'0'
- IPSETFLAGS: ipset の送信元フラグおよび宛先フラグで、パケットヘッダーの送信元部分または宛先部分から機能を選択する最大 6 つの 'src' 要素または 'dst' 要素 (例: src、src、dst) で記述されます。ここで提供する 'selectors' の数は、参照される ipset のタイプによって異なります。

IP_MASK または ***IPV6_MASK*** タイプのものを除くすべての属性は、値が *no* の match 属性を使用して除外できます。複数の否定属性をまとめてグループ化できます。以下の XML フラグメントは、抽象属性を使用したこのような例を示しています。

```
[...]
  <rule action='drop' direction='in'>
    <protocol match='no' attribute1='value1' attribute2='value2'/>
    <protocol attribute3='value3'/>
  </rule>
[...]
```

ルールの動作は、ルールを評価するとともに、指定のプロトコル属性の境界内で論理的に調べます。したがって、1 つの属性の値が、ルールで指定された値と一致しない場合は、評価プロセス中にルール全体がスキップされます。したがって、上記の例では、受信トラフィックは次の場合にのみドロップされます。プロトコルプロパティーは ***attribute1 value1***と一致せず、プロトコルプロパティー***attribute2***は ***value2*** と一致せず、プロトコルプロパティー ***attribute3*** は ***value3*** と一致します。

18.12.10.1. MAC(イーサネット)

プロトコル ID: mac

このタイプのルールは、root チェーンに置く必要があります。

表18.3 MAC プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
dstmacaddr	MAC_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク
protocolid	UINT16 (0x600-0xffff), STRING	レイヤー 3 プロトコル ID。有効な文字列は、[arp, rarp, ipv4, ipv6] などです。
comment	STRING	最大 256 文字のテキスト文字列

このフィルターは以下のように記述できます。

```
[...]
<mac match='no' srcmacaddr='$MAC'/>
[...]
```

18.12.10.2. VLAN (802.1Q)

プロトコル ID: vlan

このタイプのルールは、root チェーンまたは vlan チェーンに置く必要があります。

表18.4 VLAN プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
dstmacaddr	MAC_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク

属性名	データタイプ	定義
vlan-id	UINT16 (0x0-0xfff, 0 - 4095)	VLAN ID
encap-protocol	UINT16 (0x03c-0xfff), String	カプセル化レイヤー 3 プロトコル ID で、有効な文字列は arp、ipv4、ipv6 です。
comment	STRING	最大 256 文字のテキスト文字列

18.12.10.3. STP (スパニングツリープロトコル)

プロトコル ID: stp

このタイプのルールは、root チェーンまたは stp チェーンのいずれかに置く必要があります。

表18.5 STP プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
type	UINT8	BPDU (Bridge Protocol Data Unit) のタイプ
flags	UINT8	BPDU flagdstmacmask
root-priority	UINT16	root 優先度の範囲の開始
root-priority-hi	UINT16 (0x0-0xfff, 0 - 4095)	root 優先度の範囲の終了
root-address	MAC _ADDRESS	root の MAC アドレス
root-address-mask	MAC _MASK	root の MAC アドレスマスク
roor-cost	UINT32	root パスコスト (範囲開始)
root-cost-hi	UINT32	root パスコスト範囲の終了
sender-priority-hi	UINT16	送信者優先度の範囲の終了
sender-address	MAC_ADDRESS	BPDU 送信側 MAC アドレス
sender-address-mask	MAC_MASK	BPDU 送信側 MAC アドレスマスク

属性名	データタイプ	定義
port	UINT16	ポート識別子 (範囲開始)
port_hi	UINT16	ポート識別子の範囲の終了
msg-age	UINT16	メッセージエージタイマー (範囲開始)
msg-age-hi	UINT16	メッセージエージタイマーの範囲の終了
max-age-hi	UINT16	最大エージタイマーの範囲の終了
hello-time	UINT16	Hello time タイマー (範囲開始)
hello-time-hi	UINT16	Hello time タイマーの範囲の終了
forward-delay	UINT16	転送遅延 (範囲開始)
forward-delay-hi	UINT16	転送遅延の範囲の終了
comment	STRING	最大 256 文字のテキスト文字列

18.12.10.4. ARP/RARP

プロトコル ID: arp または rarp

このタイプのルールは、root チェーンまたは arp/rarp チェーンのいずれかに置く必要があります。

表18.6 ARP プロトコルタイプおよび RARP プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
dstmacaddr	MAC_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク
hwtype	UINT16	ハードウェアの種類
protocoltype	UINT16	プロトコルタイプ

属性名	データタイプ	定義
opcode	UINT16, STRING	Opcode の有効な文字列は、Request、Reply、Request_Reverse、Reply_Reverse、DRARP_Request、DRARP_Reply、DRARP_Error、InARP_Request、ARP_NAK です。
arpsrcmacaddr	MAC_ADDR	ARP/RARP パケットのソース MAC アドレス
arpdstmacaddr	MAC _ADDR	ARP/RARP パケットの宛先 MAC アドレス
arpsrcipaddr	IP_ADDR	ARP/RARP パケットのソース IP アドレス
arpdstipaddr	IP_ADDR	ARP/RARP パケットの宛先 IP アドレス
gratuitous	BOOLEAN	Gratuitous ARP パケットを確認するかどうかを示すブール値
comment	STRING	最大 256 文字のテキスト文字列

18.12.10.5. IPv4

プロトコル ID: ip

このタイプのルールは、root チェーンまたは ipv4 チェーンに存在する必要があります。

表18.7 IPv4 プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
dstmacaddr	MAC_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク
srcipaddr	IP_ADDR	ソース IP アドレス

属性名	データタイプ	定義
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
protocol	UINT8, STRING	レイヤー 4 プロトコル識別子。protocol に有効な文字列は、tcp、udp、udplite、esp、ah、icmp、igmp、sctp などです。
srcportstart	UINT16	有効なソースポートの範囲の開始。プロトコルが必要です。
srcportend	UINT16	有効なソースポートの範囲の終了。プロトコルが必要です。
dstportstart	UNIT16	有効な宛先ポートの範囲の開始。プロトコルが必要です。
dstportend	UNIT16	有効な宛先ポートの範囲の終了。プロトコルが必要です。
comment	STRING	最大 256 文字のテキスト文字列

18.12.10.6. IPv6

プロトコル ID: ipv6

このタイプのルールは、root チェーンまたは ipv6 チェーンに存在する必要があります。

表18.8 IPv6 プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信元の MAC アドレスに適用されるマスク
dstmacaddr	MAC_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク

属性名	データタイプ	定義
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
protocol	UINT8, STRING	レイヤー 4 プロトコル識別子。protocol に有効な文字列は、tcp、udp、udplite、esp、ah、icmpv6、sctp などです。
scrportstart	UNIT16	有効なソースポートの範囲の開始。プロトコルが必要です。
srcportend	UINT16	有効なソースポートの範囲の終了。プロトコルが必要です。
dstportstart	UNIT16	有効な宛先ポートの範囲の開始。プロトコルが必要です。
dstportend	UNIT16	有効な宛先ポートの範囲の終了。プロトコルが必要です。
comment	STRING	最大 256 文字のテキスト文字列

18.12.10.7. TCP/UDP/SCTP

プロトコル ID: tcp、udp、sctp

chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.9 TCP/UDP/SCTP プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。

属性名	データタイプ	定義
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
scripto	IP_ADDR	ソース IP アドレスの範囲の開始
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
scrportstart	UNIT16	有効なソースポートの範囲の開始。プロトコルが必要です。
srcportend	UINT16	有効なソースポートの範囲の終了。プロトコルが必要です。
dstportstart	UNIT16	有効な宛先ポートの範囲の開始。プロトコルが必要です。
dstportend	UNIT16	有効な宛先ポートの範囲の終了。プロトコルが必要です。
comment	STRING	最大 256 文字のテキスト文字列
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
flags	STRING	TCP のみ - mask/flags の形式で、mask と flags の各々が、SYN、ACK、URG、PSH、FIN、RST、NONE、ALL のコンマで区切りリストになります。
ipset	STRING	libvirt の外部で管理される IPSet の名前
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.10.8. ICMP

プロトコル ID: icmp

注記: chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.10 ICMP プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信側の MAC アドレスに適用されるマスク
dstmacaddr	MAD_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の開始
scripto	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
type	UNIT16	ICMP のタイプ
code	UNIT16	ICMP コード
comment	STRING	最大 256 文字のテキスト文字列
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
ipset	STRING	libvirt の外部で管理される IPSet の名前

属性名	データタイプ	定義
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.10.9. IGMP、ESP、AH、UDPLITE、'ALL'

プロトコル ID - igmp、esp、ah、udplite、all

chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.11 IGMP、ESP、AH、UDPLITE、'ALL'

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcmacmask	MAC_MASK	送信側の MAC アドレスに適用されるマスク
dstmacaddr	MAD_ADDR	宛先の MAC アドレス
dstmacmask	MAC_MASK	宛先の MAC アドレスに適用されるマスク
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の開始
scripto	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
comment	STRING	最大 256 文字のテキスト文字列

属性名	データタイプ	定義
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
ipset	STRING	libvirt の外部で管理される IPSet の名前
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.10.10. IPV6 経由の TCP/UDP/SCTP

プロトコル ID: tcp-ipv6、udp-ipv6、sctp-ipv6

chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.12 IPv6 のプロトコルタイプ経由の TCP、UDP、SCTP

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の開始
scripto	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
srcportstart	UINT16	有効なソースポートの範囲の開始
srcportend	UINT16	有効なソースポートの範囲の終了
dstportstart	UINT16	有効な宛先ポートの範囲の開始

属性名	データタイプ	定義
dstportend	UINT16	有効な宛先ポートの範囲の終了
comment	STRING	最大 256 文字のテキスト文字列
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
ipset	STRING	libvirt の外部で管理される IPSet の名前
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.10.11. ICMPv6

プロトコル ID: icmpv6

chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.13 ICMPv6 プロトコルタイプ

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の開始
scripto	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
type	UINT16	ICMPv6 のタイプ

属性名	データタイプ	定義
code	UINT16	ICMPv6 コード
comment	STRING	最大 256 文字のテキスト文字列
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
ipset	STRING	libvirt の外部で管理される IPSet の名前
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.10.12. IPv6 経由の IGMP, ESP, AH, UDPLITE, 'ALL'

プロトコル ID - igmp-ipv6、esp-ipv6、ah-ipv6、udplite-ipv6、all-ipv6

chain パラメーターはこのタイプのトラフィックでは無視されるため、省略するか、root に設定してください。

表18.14 IPv プロトコルタイプ経由の IGMP、ESP、AH、UDPLITE、'ALL'

属性名	データタイプ	定義
srcmacaddr	MAC_ADDR	送信側の MAC アドレス
srcipaddr	IP_ADDR	ソース IP アドレス
srcipmask	IP_MASK	マスクがソース IP アドレスに適用されました。
dstipaddr	IP_ADDR	宛先 IP アドレス
dstipmask	IP_MASK	宛先 IP アドレスに適用されるマスク
srcipfrom	IP_ADDR	ソース IP アドレスの範囲の開始
scripto	IP_ADDR	ソース IP アドレスの範囲の終了
dstipfrom	IP_ADDR	宛先 IP アドレスの範囲の開始
dstipto	IP_ADDR	宛先 IP アドレスの範囲の終了
comment	STRING	最大 256 文字のテキスト文字列

属性名	データタイプ	定義
state	STRING	NEW、ESTABLISHED、RELATED、INVALID、または NONE のコンマ区切りのリスト
ipset	STRING	libvirt の外部で管理される IPSet の名前
ipsetflags	IPSETFLAGS	IPSet のフラグ。ipset 属性が必要です。

18.12.11. 高度なフィルター設定のトピック

次のセクションでは、高度なフィルター設定のトピックについて説明します。

18.12.11.1. 接続追跡

ネットワークフィルターサブシステム (Linux の場合) は、IP テーブルの接続追跡サポートを利用します。これにより、ネットワークトラフィックの方向を強制 (状態の一致) したり、ゲスト仮想マシンへの同時接続数をカウントして制限したりできます。たとえば、ゲスト仮想マシンで TCP ポート 8080 がサーバーとして開いていると、クライアントはポート 8080 でゲスト仮想マシンに接続できます。接続の追跡と方向の強制により、ゲスト仮想マシンが (TCP クライアント) ポート 8080 からホスト物理マシンへの接続を開始して、リモートホスト物理マシンに戻らないようにします。さらに重要なのは、トラッキングにより、リモートの攻撃者がゲスト仮想マシンへの接続を確立できないようにすることです。たとえば、ゲスト仮想マシン内のユーザーが攻撃者サイトのポート 80 への接続を確立すると、攻撃者は TCP ポート 80 からゲスト仮想マシンへの接続を開始できなくなります。デフォルトでは、接続追跡を有効にした後にトラフィックの方向を強制する接続状態一致が有効になります。

> **例18.9 TCP ポートへの接続を無効にするなどの XML**
>
> 以下は、TCP ポート 12345 への着信接続でこの機能が無効になっている XML フラグメントの例を示しています。
>
> ```
> [...]
>  <rule direction='in' action='accept' statematch='false'>
>    <cp dstportstart='12345'/>
>  </rule>
> [...]
> ```
>
> これにより、TCP ポート 12345 への着信トラフィックが許可されるようになりましたが、仮想マシン内の (クライアント) TCP ポート 12345 からの開始も有効になります。これは望ましい場合とそうでない場合があります。

18.12.11.2. 接続数の制限

ゲスト仮想マシンが確立できる接続の数を制限するには、特定タイプのトラフィックの接続の制限を設定するルールを提供する必要があります。たとえば、仮想マシンが、一度に 1 つの IP アドレスに対してのみ ping を実行できるようにし、同時に 1 つの ssh 受信接続のみをアクティブにする必要がある場合など。

例18.10 接続に制限を設定する XML サンプルファイル

以下の XML フラグメントを使用して、接続を制限できます。

```
[...]
<rule action='drop' direction='in' priority='400'>
  <tcp connlimit-above='1'/>
</rule>
<rule action='accept' direction='in' priority='500'>
  <tcp dstportstart='22'/>
</rule>
<rule action='drop' direction='out' priority='400'>
  <icmp connlimit-above='1'/>
</rule>
<rule action='accept' direction='out' priority='500'>
  <icmp/>
</rule>
<rule action='accept' direction='out' priority='500'>
  <udp dstportstart='53'/>
</rule>
<rule action='drop' direction='inout' priority='1000'>
  <all/>
</rule>
[...]
```

注記

制限ルールは、トラフィックを許可するルールに先立ち、XML に記載する必要があります。例18.10「接続に制限を設定する XML サンプルファイル」 の XML ファイルに従い、ポート 22 に送信された DNS トラフィックがゲスト仮想マシンから送信されることを許可するための追加ルールが追加され、ssh デーモンによる DNS ルックアップの失敗に関連する理由で ssh セッションが確立されないことを回避します。このルールを適用しないと、接続しようとしているときに ssh クライアントが予想外にハングする可能性があります。トラフィックの追跡に関連するタイムアウトを処理する際には、注意が必要です。ゲスト仮想マシン内でユーザーが終了した ICMP ping は、ホスト物理マシンの接続追跡システムでタイムアウトが長くなる可能性があるため、別の ICMP ping を実行できません。

最善の解決策は、# **echo 3 > /proc/sys/net/netfilter/nf_conntrack_icmp_timeout** で、ホスト物理マシンの **sysfs** のタイムアウトを調整することです。このコマンドは、ICMP 接続トラッキングのタイムアウトを 3 秒に設定します。これは、ある ping が終了すると、別の ping が 3 秒後に開始できることになります。

なんらかの理由でゲスト仮想マシンが TCP 接続を適切に閉じていないと、特にホストの物理マシンで TCP タイムアウト値が大量に設定されている場合に、接続が長期間開かれたままになります。また、アイドル状態の接続では、パケットが交換されると再度アクティブにできる接続追跡システムのタイムアウトが発生する場合があります。

ただし、この制限を低く設定しすぎると、新しく開始した接続により、アイドル状態の接続が強制的に TCP バックオフになる場合があります。したがって、接続の制限は、新しい TCP 接続の変動によって、アイドル状態の接続に関連して奇妙なトラフィック動作が発生しないように、かなり高く設定する必要があります。

18.12.11.3. コマンドラインツール

virsh は、ネットワークフィルターのライフサイクルサポートにより拡張されました。ネットワークフィルターサブシステムに関連するすべてのコマンドは、***nwfilter*** という接頭辞で始まります。以下のコマンドを使用できます。

- **nwfilter-list** - UUID と、すべてのネットワークフィルターの名前の一覧を表示します。
- **nwfilter-define** - 新しいネットワークフィルターを定義するか、既存のフィルターを更新します (名前を指定する必要があります)。
- **nwfilter-undefine** - 指定したネットワークフィルターを削除します (名前を指定する必要があります)。現在使用しているネットワークフィルターを削除しないでください。
- **nwfilter-dumpxml** - 指定したネットワークフィルターを表示します (名前を指定する必要があります)。
- **nwfilter-edit** - 指定したネットワークフィルターを編集します (名前を指定する必要があります)。

18.12.11.4. 既存のネットワークフィルター

以下は、libvirt で自動的にインストールされるネットワークフィルターの例です。

表18.15 ICMPv6 プロトコルタイプ

Command Name	説明
no-arp-spoofing	ゲスト仮想マシンが ARP トラフィックをスプーフィングするのを防ぎます。このフィルターは、ARP 要求および応答メッセージのみを許可し、それらのパケットにゲスト仮想マシンの MAC アドレスと IP アドレスが含まれるようにします。
allow-dhcp	ゲスト仮想マシンが DHCP 経由で (任意の DHCP サーバーから) IP アドレスを要求できるようにします。
allow-dhcp-server	ゲスト仮想マシンが、指定した DHCP サーバーから IP アドレスを要求できるようにします。DHCP サーバーの、ドットで区切られた 10 進数の IP アドレスは、このフィルターを参照する必要があります。変数の名前は *DHCPSERVER* でなければなりません。
no-ip-spoofing	ゲスト仮想マシンが、パケット内の送信元 IP アドレスとは異なる IP パケットを送信しないようにします。
no-ip-multicast	ゲスト仮想マシンが IP マルチキャストパケットを送信しないようにします。

Command Name	説明
clean-traffic	MAC、IP、および ARP のスプーフィングを防ぎます。このフィルターは、他のいくつかのフィルターをビルディングブロックとして参照します。

これらのフィルターはビルディングブロックにすぎず、便利なネットワークトラフィックフィルターを提供するために、他のフィルターとの組み合わせが必要です。上記で最も使用されているのが *clean-traffic* フィルターです。たとえば、このフィルター自体を *no-ip-multicast* フィルターと組み合わせて、仮想マシンが IP マルチキャストトラフィックを送信しないようにし、パケットスプーフィングを防ぐことができます。

18.12.11.5. 独自のフィルターの作成

libvirt はネットワークフィルターの例をいくつか提供しているだけなので、独自のフィルターを作成することを検討してください。これを計画する際に、ネットワークフィルターサブシステムと、それが内部でどのように機能するかについて知っておく必要があります。フィルターをかけるプロトコルを必ず熟知して理解しておく必要があります。それにより、通過するトラフィック以上のトラフィックがなくなり、実際に許可するトラフィックが通過するようになります。

ネットワークフィルターサブシステムは、現在 Linux ホストの物理マシンでのみ利用でき、Qemu および KVM タイプの仮想マシンでのみ機能します。Linux では、ebtable、iptables、および ip6tables に対応し、その機能を利用します。「サポートされているプロトコル」 に記載されている一覧を考慮し、ebtable を使用して以下のプロトコルを実装できます。

- mac
- stp (スパニングツリープロトコル)
- vlan (802.1Q)
- arp, rarp
- ipv4
- ipv6

IPv4 で実行するプロトコルはすべて iptables を使用してサポートされ、IPv6 で実行するプロトコルは ip6tables を使用して実装されます。

Linux ホストの物理マシンを使用して、libvirt のネットワークフィルターリングサブシステムが作成したすべてのトラフィックフィルターリングルールは、最初に ebtables が実装したフィルターリングサポートを通過し、その後で iptables フィルターまたは ip6tables フィルターを通過します。フィルターツリーに、mac、stp、vlan arp、rarp、ipv4、または ipv6 などのプロトコルを持つルールがある場合は、一覧表示されている ebtable ルールと値が自動的に使用されます。

同じプロトコルに複数のチェーンを作成できます。チェーンの名前は、以前に列挙されたプロトコルのいずれかの接頭辞を持つ必要があります。ARP トラフィックを処理するための追加のチェーンを作成する場合は、たとえば、名前が arp-test のチェーンを指定できます。

たとえば、IP プロトコルフィルターを使用し、受け入れる UDP パケットのソース、宛先 IP、およびポートの属性を指定して、送信元および宛先のポートで UDP トラフィックでフィルターリングできます。これにより、ebtables を使用した UDP トラフィックの早期フィルターリングが可能になります。ただし、UDP パケットなどの IP または IPv6 パケットが ebtables レイヤーを通過し、iptables ルールまたは ip6tables ルールをインスタンス化するフィルターツリーに少なくとも 1 つのルールがある場

合、UDP パケットの通過を許可するルールもこれらのフィルターレイヤーに指定する必要があります。これは、適切な udp トラフィックフィルターノードまたは udp-ipv6 トラフィックフィルターノードを含むルールで実行できます。

例18.11 カスタムフィルターの作成

以下の要件の一覧を満たすためにフィルターが必要であるとします。

- 仮想マシンのインターフェイスによる MAC、IP、および ARP のスプーフィングの阻止
- 仮想マシンのインターフェイスの TCP ポート 22 および 80 のみを開く
- 仮想マシンがインターフェイスから ping トラフィックを送信できるようにしますが、インターフェイスで仮想マシンの ping を行わないようにします。
- 仮想マシンが DNS ルックアップ (ポート 53 への UDP) を実行できるようにします。

スプーフィングを防ぐための要件は、すでに存在する ***clean-traffic*** ネットワークフィルターにより満たされています。そのため、これを行う方法は、カスタムフィルターから参照することです。

TCP ポート 22 および 80 のトラフィックを有効にするには、このタイプのトラフィックを有効にするために 2 つのルールが追加されました。ゲスト仮想マシンが ping トラフィックを送信できるようにするため、ICMP トラフィックにルールが追加されました。簡単にするために、一般的な ICMP トラフィックはゲスト仮想マシンから開始することが許可され、ICMP エコー要求および応答メッセージには指定されません。その他のトラフィックはすべて、ゲスト仮想マシンに到達しないか、ゲスト仮想マシンにより開始されます。これを行うには、その他のトラフィックをすべて破棄するルールが追加されます。ゲスト仮想マシンが **test** と呼ばれ、フィルターを関連付けるインターフェイスが **eth0** と呼ばれていると、フィルターの名前は **test-eth0** となります。

この考慮事項の結果、ネットワークフィルターの XML が次のようになります。

```
<filter name='test-eth0'>
  <!- - This rule references the clean traffic filter to prevent MAC, IP and ARP spoofing. By not
providing an IP address parameter, libvirt will detect the IP address the guest virtual machine is
using. - ->
  <filterref filter='clean-traffic'/>

  <!- - This rule enables TCP ports 22 (ssh) and 80 (http) to be reachable - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='22'/>
  </rule>

  <rule action='accept' direction='in'>
    <tcp dstportstart='80'/>
  </rule>

  <!- - This rule enables general ICMP traffic to be initiated by the guest virtual machine including
ping traffic - ->
  <rule action='accept' direction='out'>
    <icmp/>
  </rule>>

  <!- - This rule enables outgoing DNS lookups using UDP - ->
  <rule action='accept' direction='out'>
    <udp dstportstart='53'/>
  </rule>
```

```
<!- - This rule drops all other traffic - ->
<rule action='drop' direction='inout'>
  <all/>
</rule>

</filter>
```

18.12.11.6. サンプルのカスタムフィルター

上記の XML のルールのいずれかに、送信元アドレスまたは宛先アドレスとしてゲスト仮想マシンの IP アドレスが含まれていますが、トラフィックのフィルターリングは正しく機能します。理由は、ルールの評価はインターフェイスごとに内部的に行われるのに対し、ルールは、送信元または宛先の IP アドレスではなく、パケットを送信または受信する (タップ) インターフェイスに基づいて追加で評価されるためです。

例18.12 ネットワークインターフェイスの説明のサンプル XML

テストゲスト仮想マシンのドメイン XML 内にある可能性のあるネットワークインターフェイスの説明の XML フラグメントは、以下のようになります。

```
[...]
 <interface type='bridge'>
   <source bridge='mybridge'/>
   <filterref filter='test-eth0'/>
 </interface>
[...]
```

ICMP トラフィックをより厳格に制御し、ゲスト仮想マシンから送信できるのは ICMP エコー要求のみで、ゲスト仮想マシンが受信できるのは ICMP エコー応答のみとなるようにするため、上記の ICMP ルールを、以下の 2 つのルールに置き換えることができます。

```
<!- - enable outgoing ICMP echo requests- ->
<rule action='accept' direction='out'>
  <icmp type='8'/>
</rule>
```

```
<!- - enable incoming ICMP echo replies- ->
<rule action='accept' direction='in'>
  <icmp type='0'/>
</rule>
```

例18.13 2 番目の例のカスタムフィルター

この例は、上記の例と同様のフィルターを作成する方法を示していますが、ゲスト仮想マシン内にある ftp サーバーを使用して要件のリストを拡張しています。このフィルターの要件は以下のとおりです。

- ゲスト仮想マシンのインターフェイスの MAC、IP、および ARP スプーフィングを防ぎます。

- ゲスト仮想マシンのインターフェイスで TCP ポート 22 および 80 のみを開きます。
- ゲスト仮想マシンがインターフェイスから ping トラフィックを送信できるようにしますが、インターフェイスでゲスト仮想マシンへの ping は許可しません。
- ゲスト仮想マシンが DNS ルックアップ (ポート 53 への UDP) を実行できるようにします。
- ftp サーバーを有効にし (アクティブモード)、ゲスト仮想マシン内で実行できるようにします。

FTP サーバーをゲスト仮想マシン内で実行できるようにするという追加の要件は、ポート 21 が FTP 制御トラフィックに到達できるようにするという要件と、ゲスト仮想マシンが、ゲスト仮想マシンの TCP ポート 20 からの発信 TCP 接続を FTP クライアント (FTP アクティブモード) に確立できるようにする要件にマッピングします。このフィルターを作成する方法はいくつかあります。この例では、2 つの解決策を説明します。

最初のソリューションは、TCP プロトコルの state 属性を使用します。この属性は、Linux ホストの物理マシンの接続追跡フレームワークにフックを提供します。ゲスト仮想マシンが開始する FTP データ接続 (FTP アクティブモード) の場合は、RELATED 状態を使用して、ゲスト仮想マシンが開始する FTP データ接続が既存の FTP 制御接続の結果である (または関連している) ことを検出できるようにします。これにより、ファイアウォールを介して、パケットを渡すことができます。ただし、RELATED 状態は、FTP データパスの送信 TCP 接続の最初のパケットに対してのみ有効です。その後、この状態は ESTABLISHED になり、着信方向および発信方向にも同様に適用されます。これはすべて、ゲスト仮想マシンの TCP ポート 20 から発信される FTP データトラフィックに関連します。これにより、以下のソリューションが得られます。

```
<filter name='test-eth0'>
  <!- - This filter (eth0) references the clean traffic filter to prevent MAC, IP, and ARP spoofing. By
 not providing an IP address parameter, libvirt will detect the IP address the guest virtual machine
 is using. - ->
  <filterref filter='clean-traffic'/>

  <!- - This rule enables TCP port 21 (FTP-control) to be reachable - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='21'/>
  </rule>

  <!- - This rule enables TCP port 20 for guest virtual machine-initiated FTP data connection
 related to an existing FTP control connection - ->
  <rule action='accept' direction='out'>
    <tcp srcportstart='20' state='RELATED,ESTABLISHED'/>
  </rule>

  <!- - This rule accepts all packets from a client on the FTP data connection - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='20' state='ESTABLISHED'/>
  </rule>

  <!- - This rule enables TCP port 22 (SSH) to be reachable - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='22'/>
  </rule>

  <!- -This rule enables TCP port 80 (HTTP) to be reachable - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='80'/>
```

```
  </rule>

  <!- - This rule enables general ICMP traffic to be initiated by the guest virtual machine, including
ping traffic - ->
  <rule action='accept' direction='out'>
    <icmp/>
  </rule>

  <!- - This rule enables outgoing DNS lookups using UDP - ->
  <rule action='accept' direction='out'>
    <udp dstportstart='53'/>
  </rule>

  <!- - This rule drops all other traffic - ->
  <rule action='drop' direction='inout'>
    <all/>
  </rule>

</filter>
```

RELATED 状態を使用してフィルターを試す前に、適切な接続追跡モジュールがホスト物理マシンのカーネルに読み込まれていることを確認する必要があります。カーネルのバージョンによっては、ゲスト仮想マシンと FTP 接続を確立する前に、次の 2 つのコマンドのいずれかを実行する必要があります。

- #**modprobe nf_conntrack_ftp** (利用可能な場合)
- #**modprobe ip_conntrack_ftp** (上記が利用可能でない場合)

FTP 以外のプロトコルを RELATED 状態と併用する場合は、対応するモジュールを読み込む必要があります。モジュールは、ftp、tftp、irc、sip、sctp、および amanda のプロトコルで利用できます。

2 番目のソリューションでは、以前のソリューションよりも多くの接続の状態フラグを使用します。このソリューションでは、トラフィックフローの最初のパケットが検出されると、接続の NEW 状態が有効であるという事実を利用します。その後、フローの最初のパケットが受け入れられると、フローは接続になるため、ESTABLISHED 状態になります。したがって、ESTABLISHED 接続のパケットをゲスト仮想マシンに到達できるようにしたり、ゲスト仮想マシンによって送信されるように、一般的なルールを記述することができます。これは、NEW 状態で識別される最も最初のパケットに特定のルールを書き込み、データが許容可能なポートを指定します。明示的に許可されていないポート向けのパケットはすべて破棄されるため、ESTABLISHED 状態に到達しません。そのポートから送信される後続のパケットもすべて破棄されます。

```
<filter name='test-eth0'>
  <!- - This filter references the clean traffic filter to prevent MAC, IP and ARP spoofing. By not
providing and IP address parameter, libvirt will detect the IP address the VM is using. - ->
  <filterref filter='clean-traffic'/>

  <!- - This rule allows the packets of all previously accepted connections to reach the guest virtual
machine - ->
  <rule action='accept' direction='in'>
    <all state='ESTABLISHED'/>
  </rule>

  <!- - This rule allows the packets of all previously accepted and related connections be sent from
the guest virtual machine - ->
```

```
  <rule action='accept' direction='out'>
    <all state='ESTABLISHED,RELATED'/>
  </rule>

  <!- - This rule enables traffic towards port 21 (FTP) and port 22 (SSH)- ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='21' dstportend='22' state='NEW'/>
  </rule>

  <!- - This rule enables traffic towards port 80 (HTTP) - ->
  <rule action='accept' direction='in'>
    <tcp dstportstart='80' state='NEW'/>
  </rule>

  <!- - This rule enables general ICMP traffic to be initiated by the guest virtual machine, including
ping traffic - ->
  <rule action='accept' direction='out'>
    <icmp state='NEW'/>
  </rule>

  <!- - This rule enables outgoing DNS lookups using UDP - ->
  <rule action='accept' direction='out'>
    <udp dstportstart='53' state='NEW'/>
  </rule>

  <!- - This rule drops all other traffic - ->
  <rule action='drop' direction='inout'>
    <all/>
  </rule>

</filter>
```

18.12.12. 制限事項

以下は、ネットワークフィルターサブシステムにおける現在知られている制限の一覧です。

- 仮想マシンの移行は、ゲスト仮想マシンのトップレベルフィルターが参照するフィルターツリー全体が、ターゲットホストの物理マシンで利用可能な場合にのみ対応します。たとえば、ネットワークフィルターの **clean-traffic** は、すべての libvirt インストールで利用可能である必要があり、このフィルターを参照するゲスト仮想マシンの移行を有効にする必要があります。バージョンの互換性が問題ではないことを確認するには、パッケージを定期的に更新して、最新バージョンの libvirt を使用していることを確認してください。

- インターフェイスに関連付けられたネットワークトラフィックフィルターを失わないように、バージョン 0.8.1 以降の libvirt インストール間で移行を行う必要があります。

- ゲスト仮想マシンが VLAN (802.1Q) パケットを送信しても、プロトコル ID の arp、rarp、ipv4、ipv6 のルールでフィルターをかけることはできません。これらは、プロトコル ID、MAC、および VLAN でのみフィルターにかけることができます。そのため、フィルターの clean-traffic の例 例18.1「ネットワークフィルターリングの例」 は予想通りに機能しません。

18.13. トンネルの作成

本セクションでは、さまざまなトンネルシナリオを実装する方法を説明します。

18.13.1. マルチキャストトンネルの作成

マルチキャストグループは、仮想ネットワークを表すように設定されています。ネットワークデバイスが同じマルチキャストグループにあるゲスト仮想マシンは、ホストの物理マシン全体であっても互いに通信できます。このモードは、特権を持たないユーザーでも使用できます。デフォルトの DNS または DHCP に対応せず、発信ネットワークアクセスもありません。外部へのネットワークアクセスを提供するには、ゲスト仮想マシンの 1 つに 2 つ目の NIC が必要です。この NIC は、最初の 4 つのネットワークタイプのいずれかに接続され、適切なルーティングを提供します。マルチキャストプロトコルは、ゲスト仮想マシンのユーザーモードと互換性があります。提供するソースアドレスは、マルチキャストアドレスブロックに使用されるアドレスからのものである必要があることに注意してください。

マルチキャストトンネルを作成するには、**<devices>** 要素に以下の XML の詳細を指定します。

図18.28 マルチキャストトンネル XML の例

```
...
<devices>
  <interface type='mcast'>
    <mac address='52:54:00:6d:90:01'>
    <source address='230.0.0.1' port='5558'/>
  </interface>
</devices>
...
```

18.13.2. TCP トンネルの作成

TCP のクライアント/サーバーアーキテクチャーは、仮想ネットワークを提供します。この設定では、1 つのゲスト仮想マシンがネットワークのサーバー側を提供し、その他のゲスト仮想マシンはすべてクライアントとして設定されます。すべてのネットワークトラフィックは、ゲスト仮想マシンサーバーを介してゲスト仮想マシンクライアント間でルーティングされます。このモードは、非特権ユーザーにも使用できます。このモードは、デフォルトの DNS または DHCP サポートを提供せず、送信ネットワークアクセスも提供しないことに注意してください。外部へのネットワークアクセスを提供するには、ゲスト仮想マシンの 1 つに 2 つ目の NIC が必要です。この NIC は、最初の 4 つのネットワークタイプのいずれかに接続され、適切なルーティングを提供します。

TCP トンネルを作成するには、以下の XML ディテールを **<devices>** 要素に配置します。

図18.29 TCP トンネルドメイン XMl の例

```
    ...
<devices>
  <interface type='server'>
    <mac address='52:54:00:22:c9:42'>
    <source address='192.168.0.1' port='5558'/>
  </interface>
  ...
  <interface type='client'>
    <mac address='52:54:00:8b:c9:51'>
    <source address='192.168.0.1' port='5558'/>
  </interface>
</devices>
...
```

18.14. VLAN タグの設定

仮想ローカルエリアネットワーク (vLAN) タグは、**virsh net-edit** コマンドを使用して追加します。このタグは、SR-IOV デバイスで PCI デバイスの割り当てと併用することもできます。詳細は、「SR-IOV デバイスを使用した PCI 割り当て (パススルー) の設定」 を参照してください。

図18.30 vVLAN タグの設定 (対応しているネットワークタイプのみ)

```
<network>
  <name>ovs-net</name>
  <forward mode='bridge'/>
  <bridge name='ovsbr0'/>
  <virtualport type='openvswitch'>
    <parameters interfaceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f'/>
  </virtualport>
  <vlan trunk='yes'>
    <tag id='42' nativeMode='untagged'/>
    <tag id='47'/>
  </vlan>
  <portgroup name='dontpanic'>
    <vlan>
      <tag id='42'/>
    </vlan>
  </portgroup>
</network>
```

ネットワークタイプが、ゲストに対して透過的な vlan タグ付けをサポートする場合にのみ、任意の **<vlan>** 要素は、このネットワークを使用してすべてのゲストのトラフィックに適用する 1 つ以上の vlan タグを指定できます (openvswitch および type='hostdev' SR-IOV ネットワークは、ゲストトラフィックの透過的な VLAN タグ付けをサポートします。標準の linux ブリッジや libvirt 自体の仮想ネットワークを含むその他の場合はすべて、これをサポートしません。802.1 Qbh (vn-link) スイッチおよび 802.1Qbg (VEPA) スイッチは、(libvirt の外部で) ゲストトラフィックを特定の vlan にタグ付けする独自の方法を提供します。) 想定どおりに、tag 属性は使用する vlan タグを指定します。ネットワークに複

数の**<vlan>**要素が定義されている場合は、指定されたすべてのタグを使用して VLAN トランクを実行することが想定されます。単一のタグを使用した VLAN トランキングが必要な場合は、オプションの属性 trunk='yes' を VLAN 要素に追加できます。

openvswitch を使用したネットワーク接続では、'native-tagged' モードおよび 'native-untagged' VLAN モードを設定できます。**<tag>** 要素でオプションの nativeMode 属性を使用します。nativeMode は、tagged または untagged に設定できます。要素の id 属性は、ネイティブ vlan を設定します。

<vlan> 要素は、**<portgroup>** 要素でも指定できます。また、ドメインの **<interface>** 要素でも指定できます。複数の場所で vlan タグが指定されている場合は、**<interface>** の設定が優先され、その後にインターフェイス設定で選択した**<portgroup>** の設定が優先されます。**<network>** の**<vlan>** は、**<portgroup>** または **<interface>** に指定がない場合にのみ選択されます。

18.15. 仮想ネットワークへの QOS の適用

Quality of Service (QoS) は、ネットワーク上のすべてのユーザーに最適なエクスペリエンスを保証するリソース制御システムを指し、遅延、ジッター、またはパケットロスがないことを確認します。QoS は、アプリケーション固有またはユーザー/グループ固有の場合があります。詳細は、「QoS (Quality of Service)」 を参照してください。

第19章 QEMU-KVM コマンド、フラグ、および引数

19.1. はじめに

注記

本章の主な目的は、Red Hat Enterprise Linux 6 でエミュレーターおよびハイパーバイザーとして使用される **qemu-kvm** ユーティリティーコマンド、フラグ、および引数のリストを提供することです。これは、機能することがわかっているが、自己責任で使用する必要があるオプションの包括的な要約です。Red Hat Enterprise Linux 6 は、基盤となる仮想化テクノロジーとして KVM を使用します。使用されるマシンエミュレーターとハイパーバイザーは、qemu-kvm と呼ばれる QEMU の修正バージョンです。このバージョンは、元の QEMU のすべての設定オプションをサポートしているわけではなく、いくつかの追加オプションも追加しています。

ここにリストされて **いない**オプションは実行しないでください。

ホワイトリスト形式

- **<name>** - 構文の説明で使用する場合、この文字列はユーザー定義の値に置き換える必要があります。
- **[a|b|c]** - 構文の説明で使用される場合、| で区切られた文字列の 1 つのみ使用されています。
- コメントが存在しない場合、オプションはすべての可能な値でサポートされます。

19.2. BASIC OPTIONS

このセクションでは、基本的なオプションについて説明します。

エミュレートされたマシン
-M <machine-type>

-machine <machine-type>**[,<property>[=<value>][,..]]**

プロセッサータイプ
-cpu <model>[,<FEATURE>][...]

-cpu ? コマンドを実行すると、追加のモデルが表示されます。

- **Opteron_G5** - AMD Opteron 63xx class CPU
- **Opteron_G4** - AMD Opteron 62xx class CPU
- **Opteron_G3** - AMD Opteron 23xx (AMD Opteron Gen 3)
- **Opteron_G2** - AMD Opteron 22xx (AMD Opteron Gen 2)
- **Opteron_G1** - AMD Opteron 240 (AMD Opteron Gen 1)
- **Westmere** - Westmere E56xx/L56xx/X56xx (Nehalem-C)
- **Haswell** - Intel Core Processor (Haswell)

- **SandyBridge** - Intel Xeon E312xx (Sandy Bridge)
- **Nehalem** - Intel Core i7 9xx (Nehalem Class Core i7)
- **Penryn** - Intel Core 2 Duo P9xxx (Penryn Class Core 2)
- **Conroe** - Intel Celeron_4x0 (Conroe/Merom Class Core 2)
- **cpu64-rhel5** - Red Hat Enterprise Linux 5 がサポートする QEMU 仮想 CPU バージョン
- **cpu64-rhel6** - Red Hat Enterprise Linux 6 がサポートする QEMU 仮想 CPU バージョン
- **default** - 特別なオプションは、上記のデフォルトのオプションを使用します。

プロセッサートポロジー
-smp <n>[,cores=<ncores>][,threads=<nthreads>][,sockets=<nsocks>][,maxcpus=<maxcpus>]

プロセッサートポロジーに関するハイパーバイザーとゲストオペレーティングシステムの制限が適用されます。

NUMA システム
-numa <nodes>[,mem=<size>][,cpus=<cpu[-cpu>]][,nodeid=<node>]

プロセッサートポロジーに関するハイパーバイザーとゲストオペレーティングシステムの制限が適用されます。

Memory Size
-m <megs>

サポートされる値は、ゲストの最小値と最大値、およびハイパーバイザーの制限によって制限されます。

キーボードのレイアウト
-k <language>

ゲスト名
-name <name>

ゲスト UUID
-uuid <uuid>

19.3. ディスクオプション

このセクションでは、ディスクオプションに関する情報を提供します。

ジェネリックドライブ
-drive <option>[,<option>[,<option>[,...]]]

次のオプションでサポートされています。

- **readonly**[on|off]
- **werror**[enospc|report|stop|ignore]
- **rerror**[report|stop|ignore]
- **id**=<id>

ドライブの ID には、if=none に対して次の制限があります。

- IDE ディスクには次の形式の <id> が必要です: drive-ide0-<BUS>-<UNIT>

  正しい形式の例:

  -drive if=none,id=drive-ide0-<BUS>-<UNIT>,... -device ide-drive,drive=drive-ide0-<BUS>-<UNIT>,bus=ide.<BUS>,unit=<UNIT>

- **file**=<file>

  <file> の値は、次のルールで解析されます。

  - フロッピーデバイスを <file> として渡すことはサポートされていません。
  - cd-rom デバイスを <file> として渡すことは、cdrom メディアタイプ (media=cdrom) でのみ、IDE ドライブ (if=ide または if=none + -device ide-drive) としてのみサポートされます。
  - <file> がブロックデバイスでも文字デバイスでもない場合は、: を含めることはできません。

- **if**=<interface>

  次のインターフェイスがサポートされています: none、ide、virtio、floppy。

- **index**=<index>
- **media**=<media>
- **cache**=<cache>

  サポートされている値: none、ライトバックまたはライトスルー。

- **copy-on-read**=[on|off]
- **snapshot**=[yes|no]
- **serial**=<serial>
- **aio**=<aio>
- 形式 =< **format**>

  このオプションは必須ではなく、省略できます。ただし、これはセキュリティーリスクを表すため、生のイメージにはお勧めしません。サポートされている形式は次のとおりです。

  - **qcow2**
  - **raw**

Boot Option
-boot [order=<drives>][,menu=[on|off]]

Snapshot Mode
-snapshot

19.4. 表示オプション

このセクションでは、表示オプションに関する情報を提供します。

グラフィックを無効にする
-nographic

VGA カードエミュレーション
-vga <type>

Supported types:

- **cirrus** - Cirrus Logic GD5446 ビデオカード。
- **std** - Bochs VBE 拡張機能を備えた標準 VGA カード。
- **qxl** - スパイス準仮想カード。
- **none** - VGA カードを無効にします。

VNC ディスプレイ
-vnc <display>[,<option>[,<option>[,...]]]

サポートされている表示値:

- [<host>]:<port>
- unix:<path>
- **share**[allow-exclusive|force-shared|ignore]
- none - 他のオプションを指定せずにサポートされます。

サポートされているオプションは次のとおりです。

- **to**=<port>
- **reverse**
- **password**
- **tls**
- **x509**=</path/to/certificate/dir> - **-tls**が指定されている場合にサポートされます。
- **x509verify**=</path/to/certificate/dir> - **tls**が指定されている場合にサポートされます。
- **sasl**
- **acl**

スパイスデスクトップ
-spice option[,option[,...]]

サポートされているオプションは次のとおりです。

- **port**=<number>

- **addr**=<addr>
- **ipv4**

  ipv6
- **password**=<secret>
- **disable-ticketing**
- **disable-copy-paste**
- **tls-port**=<number>
- **x509-dir**=</path/to/certificate/dir>
- **x509-key-file**=<file>

  x509-key-password=<file>

  x509-cert-file=<file>

  x509-cacert-file=<file>

  x509-dh-key-file=<file>
- **tls-cipher**=<list>
- **tls-channel**[main|display|cursor|inputs|record|playback]

  plaintext-channel[main|display|cursor|inputs|record|playback]
- **image-compression**=<compress>
- **jpeg-wan-compression**=<value>

  zlib-glz-wan-compression=<value>
- **streaming-video**=[off|all|filter]
- **agent-mouse**=[on|off]
- **playback-compression**=[on|off]
- **seamless-migratio**=[on|off]

19.5. ネットワークオプション

このセクションでは、ネットワークオプションに関する情報を提供します。

TAP ネットワーク
-netdev tap,id=<id>][,<options>...]

次のオプションがサポートされています (すべて name=value 形式を使用)。

- ifname

- fd
- script
- downscript
- sndbuf
- vnet_hdr
- vhost
- vhostfd
- vhostforce

19.6. デバイスオプション

このセクションでは、デバイスオプションに関する情報を提供します。

一般的なデバイス
-device <driver>[,<prop>[=<value>][,...]]

すべてのドライバーは、次のプロパティーをサポートしています

- id
- bus

次のドライバーがサポートされています (使用可能なプロパティーを使用)。

- **pci-assign**
  - host
  - bootindex
  - configfd
  - addr
  - rombar
  - romfile
  - multifunction

  デバイスに複数の機能がある場合は、それらすべてを同じゲストに割り当てる必要があります。
- **rtl8139**
  - mac
  - netdev
  - bootindex

    - addr
- **e1000**
    - mac
    - netdev
    - bootindex
    - addr
- **virtio-net-pci**
    - ioeventfd
    - vectors
    - 間接的
    - event_idx
    - csum
    - guest_csum
    - gso
    - guest_tso4
    - guest_tso6
    - guest_ecn
    - guest_ufo
    - host_tso4
    - host_tso6
    - host_ecn
    - host_ufo
    - mrg_rxbuf
    - status
    - ctrl_vq
    - ctrl_rx
    - ctrl_vlan
    - ctrl_rx_extra
    - mac

    - netdev
    - bootindex
    - x-txtimer
    - x-txburst
    - tx
    - addr
- **qxl**
    - ram_size
    - vram_size
    - revision
    - cmdlog
    - addr
- **ide-drive**
    - unit
    - のドライブ
    - physical_block_size
    - bootindex
    - ver
    - wwn
- **virtio-blk-pci**
    - class
    - のドライブ
    - logical_block_size
    - physical_block_size
    - min_io_size
    - opt_io_size
    - bootindex
    - ioeventfd
    - vectors

- indirect_desc
- event_idx
- scsi
- addr

- **virtio-scsi-pci** - 6.3 のテクノロジープレビュー。6.4 以降でサポートされています。

  Windows ゲストの場合、テクノロジープレビューであった Windows Server 2003 は 6.5 以降サポートされなくなりました。ただし、Windows Server 2008 と 2012、および Windows デスクトップ 7 と 8 は、6.5 以降完全にサポートされています。

  - vectors
  - indirect_desc
  - event_idx
  - num_queues
  - addr

- **isa-debugcon**
- **isa-serial**
  - index
  - iobase
  - irq
  - chardev
- **virtserialport**
  - nr
  - chardev
  - name
- **virtconsole**
  - nr
  - chardev
  - name
- **virtio-serial-pci**
  - vectors
  - class

    - indirect_desc
    - event_idx
    - max_ports
    - flow_control
    - addr
- **ES1370**
    - addr
- **AC97**
    - addr
- **intel-hda**
    - addr
- **hda-duplex**
    - CAD
- **hda-micro**
    - CAD
- **hda-output**
    - CAD
- **i6300esb**
    - addr
- **ib700** - プロパティーなし
- **sga** - プロパティーなし
- **virtio-balloon-pci**
    - indirect_desc
    - event_idx
    - addr
- **usb-tablet**
    - migrate
    - port
- **usb-kbd**

    - migrate
    - port
- **usb-mouse**
    - migrate
    - port
- **usb-ccid** - 6.2 以降でサポート
    - port
    - slot
- **usb-host** - 6.2 以降のテクノロジープレビュー
    - hostbus
    - hostaddr
    - hostport
    - vendorid
    - productid
    - isobufs
    - port
- **usb-hub** - 6.2 以降でサポート
    - port
- **usb-ehci** - 6.2 以降のテクノロジープレビュー
    - freq
    - maxframes
    - port
- **usb-storage** - 6.2 以降のテクノロジープレビュー
    - のドライブ
    - bootindex
    - serial
    - removable
    - port
- **usb-redir** - 6.3 のテクノロジープレビュー、6.4 以降サポート

- chardev
- filter

- **scsi-cd** - 6.3 のテクノロジープレビュー、6.4 以降サポート
  - のドライブ
  - logical_block_size
  - physical_block_size
  - min_io_size
  - opt_io_size
  - bootindex
  - ver
  - serial
  - scsi-id
  - lun
  - channel-scsi
  - wwn
- **scsi-hd** - 6.3 のテクノロジープレビュー、6.4 以降サポート
  - のドライブ
  - logical_block_size
  - physical_block_size
  - min_io_size
  - opt_io_size
  - bootindex
  - ver
  - serial
  - scsi-id
  - lun
  - channel-scsi
  - wwn
- **scsi-block** - 6.3 のテクノロジープレビュー、6.4 以降サポート

- のドライブ
- bootindex

- **scsi-disk** - 6.3 のテクノロジープレビュー
  - drive=drive
  - logical_block_size
  - physical_block_size
  - min_io_size
  - opt_io_size
  - bootindex
  - ver
  - serial
  - scsi-id
  - lun
  - channel-scsi
  - wwn
- **piix3-usb-uhci**
- **piix4-usb-uhci**
- **ccid-card-passthru**

グローバルデバイス設定
-global <device>.<property>=<value>

これらの追加デバイスで一般デバイスセクションのようにサポートされるデバイスとプロパティー:

- **isa-fdc**
  - driveA
  - driveB
  - bootindexA
  - bootindexB
- **qxl-vga**
  - ram_size
  - vram_size
  - revision

- cmdlog
- addr

キャラクターデバイス
-chardev back end,id=<id>[,<options>]

サポートされているバックエンドは次のとおりです。

- **null**,id=<id> - null device
- **socket**,id=<id>,port=<port>[,host=<host>][,to=<to>][,ipv4][,ipv6][,nodelay][,server][,nowait][,telnet] - tcp socket
- **socket**,id=<id>,path=<path>[,server][,nowait][,telnet] - unix socket
- **file**,id=<id>,path=<path> - trafit to file.
- **stdio**,id=<id> - standard i/o
- **spicevmc**,id=<id>,name=<name> - spice channel

USB を有効にする
-usb

19.7. LINUX/MULTIBOOT ブート

本セクションでは、Linux とマルチブートの起動に関する情報を提供します。

カーネルファイル
-kernel <bzImage>

注: マルチブートイメージはサポートされていません

Ram ディスク
-initrd <file>

コマンドラインパラメーター
-append <cmdline>

19.8. エキスパートオプション

このセクションでは、エキスパートオプションに関する情報を提供します。

KVM 仮想化
-enable-kvm

QEMU-KVM は KVM 仮想化のみをサポートし、使用可能な場合はデフォルトで使用されます。-enable-kvm が使用され、KVM が使用できない場合、qemu-kvm は失敗します。ただし、-enable-kvm が使用されておらず、KVM が使用できない場合、**qemu-kvm** はサポートされていない TCG モードで実行されます。

カーネルモードの PIT 再挿入を無効にする
-no-kvm-pit-reinjection

No Shutdown

-no-shutdown

再起動しない
-no-reboot

シリアルポート、モニター、QMP
-serial <dev>

-monitor <dev>

-qmp <dev>

サポートされているデバイスは次のとおりです。

- **stdio** - standard input/output
- **null** - null デバイス
- **file**:<filename> - ファイルに出力します。
- **tcp**:[<host>]:<port>[,server][,nowait][,nodelay] - TCP Net console.
- **unix**:<path>[,server][,nowait] - Unix ドメインソケット
- **mon**:<dev_string> - 上記のデバイス。モニターの多重化にも使用されます。
- **none** - disable、-serial に対してのみ有効です。
- **chardev**:<id> - chardev で作成された文字デバイス。

リダイレクトのモニター
-mon <chardev_id>[,mode=[readline|control]][,default=[on|off]]

手動 CPU スタート
-S

RTC
-rtc [base=utc|localtime|date][,clock=host|vm][,driftfix=none|slew]

Watchdog
-watchdog モデル

Watchdog の反応
-watchdog-action <action>

ゲストメモリーバッキング
-mem-prealloc -mem-path /dev/hugepages

SMBIOS エントリー
-SMBIOS type= 0[,vendor=<str>][,<version=str>][,date=<str>][,release=%d.%d]

-smbios type=1[,manufacturer=<str>][,product=<str>][,version=<str>][,serial=<str>][,uuid=<uuid>][,sku=<str>][,family=<str>]

19.9. ヘルプと情報オプション

本セクションでは、ヘルプと情報オプションに関する情報を提供します。

Help
-h

-help

バージョン
-version

Audio Help
-audio-help

19.10. その他のオプション

本セクションでは、その他のオプションに関する情報を提供します。

移行
-incoming

デフォルト設定なし
-nodefconfig

-nodefaults

-nodefaults なしでの実行はサポートされていません

デバイス設定ファイル
-readconfig <file>

-writeconfig <file>

ロードされた保存状態
-loadvm <file>

第20章 ドメイン XML の操作

本セクションでは、ドメインを表すために使用される XML 形式について説明します。ここで、*ドメイン*という用語は、すべてのゲスト仮想マシンに必要なルート **<ドメイン>**要素を指します。ドメイン XML には、2 つの属性があります。**type** は、ドメインの実行に使用されるハイパーバイザーを指定します。許可される値はドライバー固有ですが、**KVM** などが含まれます。**id** は、実行中のゲスト仮想マシンの一意の整数識別子です。アクティブでないマシンには、id 値が設定されていません。本章のセクションでは、ドメイン XML のコンポーネントについて説明します。ドメイン XML の操作が必要な場合は、本書のその他の章を参照してください。

注記

本章は libvirt アップストリームのドキュメント に基づいています。

20.1. 一般情報およびメタデータ

この情報は、ドメイン XML の以下の部分にあります。

図20.1 ドメイン XML メタデータ

```
<domain type='xen' id='3'>
  <name>fv0</name>
  <uuid>4dea22b31d52d8f32516782e98ab3fa0</uuid>
  <title>A short description - title - of the domain</title>
  <description>Some human readable description</description>
  <metadata>
    <app1:foo xmlns:app1="http://app1.org/app1/">..</app1:foo>
    <app2:bar xmlns:app2="http://app1.org/app2/">..</app2:bar>
  </metadata>
  ...
</domain>
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.1 一般的なメタデータ要素

要素	説明
<name>	仮想マシンの名前を割り当てます。この名前は英数字のみで設定され、1 台のホストの物理マシンの範囲内で固有である必要があります。永続的な設定ファイルを保存するためにファイル名を形成するためによく使用されます。
<uuid>	仮想マシンのグローバルに一意識別子を割り当てます。形式は RFC4122 に準拠している必要があります (例: **3e3fce45-4f53-4fa7-bb32-11f34168b82b**。新しいマシンの定義時/作成時に省略した場合は、ランダムな UUID が生成されます。sysinfo 仕様を使用して、UUID を提供することもできます。

要素	説明
<title>	**title** ドメインの簡単な説明のための領域を作成します。タイトルには改行を含めないでください。
<description>	タイトルとは異なり、このデータは libvirt によって使用されることはなく、ユーザーが表示したい情報を含めることができます。
<metadata>	アプリケーションで使用できるため、カスタムメタデータを XML ノード/ツリーの形式で保存できます。アプリケーションは、XML ノード/ツリーでカスタム名前空間を使用し、名前空間ごとにトップレベル要素を 1 つのみ使用する必要があります (アプリケーションが構造を必要とする場合は、その名前空間要素のサブ要素を持つ必要があります)。

20.2. オペレーティングシステムの起動

仮想マシンを起動するには、それぞれに長所と短所があるさまざまな方法があります。それぞれについて、次のサブセクションで説明します。BIOS ブートローダー、ホスト物理マシンブートローダー、および直接カーネルブート 。

20.2.1. BIOS ブートローダー

BIOS を介した起動は、完全仮想化をサポートするハイパーバイザーで利用できます。この場合、BIOS にはブート順序の優先順位 (フロッピー、ハードディスク、CD-ROM、ネットワーク) があり、ブートイメージを取得/検索する場所を決定します。ドメイン XML の OS セクションには、次の情報が含まれています。

図20.2 BIOS ブートローダードメイン XML

```
...
<os>
  <type>hvm</type>
  <loader>/usr/lib/xen/boot/hvmloader</loader>
  <boot dev='hd'/>
  <boot dev='cdrom'/>
  <bootmenu enable='yes'/>
  <smbios mode='sysinfo'/>
  <bios useserial='yes' rebootTimeout='0'/>
</os>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.2 BIOS ブートローダー要素

要素	説明
<type>	ゲスト仮想マシンで起動するオペレーティングシステムの種類を指定します。**hvm** は、OS がベアメタル上で実行するように設計されているため、完全仮想化が必要であることを示しています。**linux** は、Xen 3 ハイパーバイザーゲスト ABI をサポートする OS を指します。オプションの属性も 2 つあります。**arch** は仮想化に対する CPU アーキテクチャーを指定し、**machine** はマシンタイプを参照します。詳細については、『ドライバー機能』を参照してください。
<loader>	ドメイン作成プロセスを支援するために使用されるファームウェアの一部を指します。Xen の完全仮想化ドメインを使用する場合にのみ必要です。
<boot>	**fd**、**hd**、**cdrom**、または **network** のいずれかの値を取り、次に検討するブートデバイスを指定するために使用されます。ブート要素を複数回繰り返して、順番に試行するブートデバイスの優先順位リストを設定できます。同じタイプの複数のデバイスは、バスの順序を維持しながら、ターゲットに従ってソートされます。ドメインを定義した後、libvirt によって (virDomainGetXMLDesc を介して) 返される XML 設定は、ソートされた順序でデバイスをリストします。ソートされると、最初のデバイスが起動可能としてマークされます。詳細は、『BIOS ブートローダー』を参照してください。
<bootmenu>	ゲスト仮想マシンの起動時にインタラクティブな起動メニュープロンプトを有効にするかどうかを設定します。**enable** 属性は、**yes** または **no のいずれかになり** ます。指定しない場合は、ハイパーバイザーのデフォルトが使用されます。
<smbios>	ゲスト仮想マシンで SMBIOS 情報をどのように表示するかを指定します。**mode** 属性は、**emulate** (ハイパーバイザーがすべての値を生成できるようにする)、**host** (UUID を除くブロック 0 とブロック 1 のすべてをホスト物理マシンの SMBIOS 値からコピーします。virConnectGetSysinfo 呼び出しを使用して、どの値がコピーされているかを確認することができます)、または **sysinfo** (sysinfo 要素の値を使用) のいずれかとして指定する必要があります。指定しない場合は、ハイパーバイザーのデフォルト設定が使用されます。

要素	説明
<bios>	この要素には、**yes** または **no** の値を持つ属性 **useserial** があります。この属性は、ユーザーがシリアルポートで BIOS メッセージを表示できるようにする Serial Graphics Adapter を有効または無効にします。したがって、シリアルポートを定義する必要があります。別の属性、**rebootTimeout** があることに注意してください。これは、起動が失敗した場合にゲスト仮想マシンが再度起動を開始するかどうか、および開始する期間を制御します (BIOS による)。値はミリ秒単位で、最大は **65535** で、特別な値 **-1** は再起動を無効にします。

20.2.2. ホスト物理マシンブートローダー

準仮想化を採用しているハイパーバイザーは通常 BIOS をエミュレートしませんが、代わりにホストの物理マシンがオペレーティングシステムの起動を担当します。これは、ホスト物理マシンの疑似ブートローダーを使用して、ゲスト仮想マシンのカーネルを選択するためのインターフェイスを提供する場合があります。例として、Xen を使用した pygrub があります。

図20.3 ホスト物理マシンのブートローダードメイン XML

```
...
<bootloader>/usr/bin/pygrub</bootloader>
<bootloader_args>--append single</bootloader_args>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.3 BIOS ブートローダー要素

要素	説明
<bootloader>	ホスト物理マシン OS のブートローダー実行可能ファイルへの完全修飾パスを提供します。このブートローダーは、起動するカーネルを選択します。ブートローダーの必要な出力は、使用中のハイパーバイザーによって異なります。
<bootloader_args>	コマンドライン引数をブートローダーに渡すことができます (オプションのコマンド)

20.2.3. ダイレクトカーネルブート

新しいゲスト仮想マシン OS をインストールする場合、ホスト物理マシン OS に格納されているカーネルと initrd から直接起動して、コマンドライン引数をインストーラーに直接渡すことができると便利なことがよくあります。この機能は通常、準仮想化ゲスト仮想マシンと完全仮想化ゲスト仮想マシンの両方で使用できます。

図20.4 ダイレクトカーネルブート

```
...
<os>
  <type>hvm</type>
  <loader>/usr/lib/xen/boot/hvmloader</loader>
  <kernel>/root/f8-i386-vmlinuz</kernel>
  <initrd>/root/f8-i386-initrd</initrd>
  <cmdline>console=ttyS0 ks=http://example.com/f8-i386/os/</cmdline>
  <dtb>/root/ppc.dtb</dtb>
</os>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.4 ダイレクトカーネルブート要素

要素	説明
<type>	BIOS ブートのセクションで説明されているものと同じです。
<loader>	BIOS ブートのセクションで説明されているものと同じです。
<kernel>	ホスト物理マシン OS のカーネルイメージへの完全修飾パスを指定します
<initrd>	ホスト物理マシンの OS の (任意) ramdisk イメージへの完全修飾パスを指定します。
<cmdline>	システムの起動時にカーネル (またはインストーラー) に渡される引数を指定します。これは、代替のプライマリーコンソール (シリアルポートなど)、またはインストールメディアソース/キックスタートファイルを指定するためによく使用されます

20.3. SMBIOS システム情報

ハイパーバイザーの中には、ゲスト仮想マシンに表示されるシステム情報を制御できるものもあります (たとえば、SMBIOS フィールドはハイパーバイザーで入力し、ゲスト仮想マシンの **midecode** コマンドを使用して検証できます)。オプションの sysinfo 要素は、このようなカテゴリーの情報をすべてカバーします。

図20.5 SMBIOS システム情報

```
...
<os>
  <smbios mode='sysinfo'/>
  ...
</os>
<sysinfo type='smbios'>
  <bios>
    <entry name='vendor'>LENOVO</entry>
  </bios>
  <system>
    <entry name='manufacturer'>Fedora</entry>
    <entry name='vendor'>Virt-Manager</entry>
  </system>
</sysinfo>
...
```

<sysinfo> 要素には、サブ要素のレイアウトを決定する必須の属性**type**があり、以下のように定義されます。

- **smbios** - サブ要素は特定の SMBIOS 値を呼び出します。これは、**<os>** 要素の smbios サブ要素と併用するとゲスト仮想マシンに影響します。sysinfo の各サブ要素は、SMBIOS ブロックの名前を付け、その中の要素は、ブロック内のフィールドを記述するエントリー要素のリストになります。以下のブロックおよびエントリーが認識されます。
  - **bios** - SMBIOS の ブロック 0 で、エントリー名は **vendor**、**version**、**date**、および **release** から取得されます。
  - **<system>** - SMBIOS のブロック 1 で、エントリー名は **manufacturer**、**product**、**version**、**serial**、**uuid**、**sku**、および **family** から取得します。**uuid** エントリーが最上位の uuid 要素とともに指定されている場合は、その 2 つの値が一致している必要があります。

20.4. CPU ALLOCATION

図20.6 CPU の割り当て

```
<domain>
  ...
  <vcpu placement='static' cpuset="1-4,^3,6" current="1">2</vcpu>
  ...
</domain>
```

<cpu> 要素は、ゲスト仮想マシンのオペレーティングシステムに割り当てる仮想 CPU (vCPU) の最大数を定義します。この最大数は 1 から、ハイパーバイザーで対応している最大数までの間でなければなりません。この要素には、**cpuset** 属性を指定できます。属性は、ドメインプロセスおよび仮想 CPU をデフォルトで固定できる物理 CPU 番号のコンマ区切りのリストです。

ドメインプロセスおよび仮想 CPU のピン留めポリシーは、**cputune** 属性を使用して個別に指定できることに注意してください。**emulatorpin** 属性が **<cputune>** で指定されている場合、**<vcpu>** で指定された **cpuset**値は無視されます。

同様に、**vcpupin** に値を設定した仮想 CPU では、 **cpuset** 設定が無視されます。**vcpupin** が指定されていない仮想 CPU は、**cpuset** で指定された物理 CPU に固定されます。 **cpuset** 一覧の各要素は、単一の CPU 番号、CPU 番号の範囲、または前の範囲から除外される CPU 番号が後に続くキャレット (^) のいずれかになります。属性 **current** を使用して、有効にする仮想 CPU の最大数より少ない数を指定することができます。

オプションの属性 **placement** を使用して、ドメインプロセスの CPU 配置モードを指定できます。**placement** は、**static** または **auto** のいずれかに設定できます。**<vcpu placement='auto'>** を設定すると、システムは numad をクエリーし、**<numatune>** タグで指定された設定を使用し、**<vcpu>** の他の設定を無視します。**<vcpu placement='static'>** を設定すると、システムは **<numatune>** の設定ではなく、**<vcpu placement>** タグで指定された設定を使用します。

20.5. CPU チューニング

図20.7 CPU チューニング

```
<domain>
  ...
  <cputune>
    <vcpupin vcpu="0" cpuset="1-4,^2"/>
    <vcpupin vcpu="1" cpuset="0,1"/>
    <vcpupin vcpu="2" cpuset="2,3"/>
    <vcpupin vcpu="3" cpuset="0,4"/>
    <emulatorpin cpuset="1-3"/>
    <shares>2048</shares>
    <period>1000000</period>
    <quota>-1</quota>
    <emulator_period>1000000</emulator_period>
    <emulator_quota>-1</emulator_quota>
  </cputune>
  ...
</domain>
```

すべてオプションですが、ドメイン XML のこのセクションのコンポーネントは次のとおりです。

表20.5 CPU チューニング要素

要素	説明
<cputune>	ドメインの CPU 調整可能パラメーターに関する詳細を提供します。これは任意です。
<vcpupin>	ドメイン VCPU が固定されるホスト物理マシンの物理 CPU を指定します。これを省略し、要素 **<vcpu>**の属性 **cpuset**が指定されていない場合、vCPU はデフォルトですべての物理 CPU に固定されます。これには 2 つの必須属性が含まれ、**vcpu** 属性は **id** を指定し、**cpuset** 属性は要素 **<vcpu>** の属性 **cpuset** と同じです。

要素	説明
<emulatorpin>	エミュレーター vcpu を含まないドメインのサブセット を固定するホスト物理マシンの CPU を指定します。これを省略し、要素 **<vcpu>** の属性 **cpuset** が指定されていない場合、emulator はデフォルトですべての物理 CPU に固定されます。これには、固定する物理 CPU を指定する 1 つの必須属性 **cpuset** が含まれています。要素 **<vcpu>** の属性 **配置** が **auto** の場合には、**emulatorpin** は許可されません。
<共有>	ドメインの比例的な加重共有を指定します。これを省略すると、デフォルトでオペレーティングシステムに固有のデフォルト値になります。値の単位がない場合は、その他のゲスト仮想マシンの設定を基準にして計算されます。たとえば、ゲスト仮想マシンが 2048 の値で設定されている場合、1024 の値で設定されたゲスト仮想マシンの 2 倍の処理時間が得られます。
<period>	施行間隔をマイクロ秒単位で指定します。**period** を使用することにより、ドメインの各 vcpu は、割り当てられたクォータに相当する実行時間を超えて消費することはできなくなります。この値は、以下の範囲内である必要があります: **1000-1000000**。**0** の値を持つ **<period>** は、値がないことを意味します。
<quota>	最大許容帯域幅をマイクロ秒単位で指定します。**quota** が負の値のドメインは、ドメインの帯域幅が無限であることを示します。つまり、帯域幅は制御されません。値は次の範囲内である必要があります: **1000～18446744073709551**または **0**未満。**0** の値を持つ**quota** は、値がないことを意味します。この機能を使用すると、すべての vcpu で同じ速度で実行できます。
<emulator_period>	施行間隔をマイクロ秒単位で指定します。**<emulator_period>** 内では、ドメインのエミュレータースレッド (vcpu を除く) が、**<emulator_quota>** 相当の実行時間より多くを消費することはできません。**<emulator_period>** は、次の範囲内でなければなりません: **1000 - 1000000**。**0** の **<emulator_period>** は、値がないことを意味します。

要素	説明
<emulator_quota>	ドメインのエミュレータースレッド (vcpu を除く) で許可される最大帯域幅をマイクロ秒単位で指定します。**<emulator_quota>** が負の値のドメインは、ドメインがエミュレータースレッド (vcpu を除く) に無限の帯域幅を持つことを示します。これは、帯域幅が制御されないことを示しています。値は次の範囲内である必要があります: **1000～18446744073709551**、または **0**未満。**0** の**<emulator_quota>**は、値がないことを意味します。

20.6. メモリーバッキング

メモリーバッキングにより、ハイパーバイザーはゲスト仮想マシン内の大きなページを適切に管理できます。

オプションの **<memoryBacking>** 要素には、**<hugepages>** 要素が設定されている場合があります。これにより、ハイパーバイザーは、ゲスト仮想マシンのメモリーを通常のネイティブページサイズではなく、hugepages を使用して割り当てる必要があることを通知します。

図20.8 メモリーバッキング

```
<domain>
  ...
  <memoryBacking>
    <hugepages/>
  </memoryBacking>
  ...
</domain>
```

20.7. メモリーの調整

図20.9 メモリーの調整

```
<domain>
  ...
  <memtune>
    <hard_limit unit='G'>1</hard_limit>
    <soft_limit unit='M'>128</soft_limit>
    <swap_hard_limit unit='G'>2</swap_hard_limit>
    <min_guarantee unit='bytes'>67108864</min_guarantee>
  </memtune>
  ...
</domain>
```

すべてオプションですが、ドメイン XML のこのセクションのコンポーネントは次のとおりです。

表20.6 メモリーチューニング要素

要素	説明
<memtune>	ドメインのメモリー調整可能パラメーターに関する詳細を提供します。これを省略すると、デフォルトで OS が提供するデフォルトになります。パラメーターはプロセス全体に適用されるため、制限を設定する場合は、ゲスト仮想マシン RAM、ゲスト仮想マシンビデオ RAM を合計し、メモリーオーバーヘッドを考慮に入れる必要があります。最後のピースは判断が難しいので、試行錯誤を繰り返します。調整可能パラメーターごとに、**<memory>** と同じ値を使用して、入力中のユニット番号を指定できます。下位互換性のために、出力は常に KiB になります。
<hard_limit>	ゲスト仮想マシンが使用できる最大メモリー量です。この値の **単位**は、**kibibytes** (1024 バイトのブロック) で表されます
<soft_limit>	メモリーの競合中に強制されるメモリー制限です。この値の **単位**、は kibibytes (1024 バイトのブロック) で表されます
<swap_hard_limit>	最大メモリーと、ゲスト仮想マシンが使用できるスワップの合計です。この値の **単位**は、kibibytes (1024 バイトのブロック) で表されます。これは、提供された **<hard_limit>** 値より大きくなければなりません
<min_guarantee>	ゲスト仮想マシンに保証される最小メモリー割り当てです。この値の単位は kibibytes (1024 バイトのブロック) で表されます

20.8. NUMA ノードのチューニング

従来の管理ツールを使用して NUMA ノードの調整が行われると、次のドメイン XML パラメーターが有効になります。

図20.10 NUMA ノードのチューニング

```
>
<domain>
  ...
  <numatune>
    <memory mode="strict" nodeset="1-4,^3"/>
  </numatune>
  ...
</domain>
```

すべてオプションですが、ドメイン XML のこのセクションのコンポーネントは次のとおりです。

表20.7 NUMA ノードのチューニング要素

要素	説明
<numatune>	ドメインプロセスの NUMA ポリシーを制御することにより、NUMA ホストの物理マシンのパフォーマンスを調整する方法を説明します。
<memory>	NUMA ホストの物理マシンでドメインプロセスにメモリーを割り当てる方法を指定します。これにはいくつかのオプション属性が含まれます。属性 **mode** は、**interleave**、**strict**、または **preferred**のいずれかです。値が指定されていない場合、デフォルトは **strict** に設定されます。属性 **nodeset** は、要素 **<vcpu>** の属性 **cpuset** と同じ構文を使用して、NUMA ノードを指定します。属性**placement** は、ドメインプロセスのメモリー配置モードを示すために使用できます。その値は、**static** または **auto** のいずれかになります。属性 **<nodeset>**が指定されている場合、デフォルトで **<vcpu>**または **static** の **<placement>** になります。**auto** は、ドメインプロセスが numad のクエリーから返されたアドバイザリーノードセットからのみメモリーを割り当てることを示し、属性 nodeset の値が指定されている場合は無視されます。**vcpu** の属性 **placement** が **auto** で、**<numatune>** が指定されていない場合は、**<placement> auto** と mode **strict** のデフォルトの numatune が暗黙的に追加されます。

20.9. ブロック I/O チューニング

図20.11 ブロック I/O チューニング

```
<domain>
  ...
  <blkiotune>
    <weight>800</weight>
    <device>
      <path>/dev/sda</path>
      <weight>1000</weight>
    </device>
    <device>
      <path>/dev/sdb</path>
      <weight>500</weight>
    </device>
  </blkiotune>
  ...
</domain>
```

すべてオプションですが、ドメイン XML のこのセクションのコンポーネントは次のとおりです。

表20.8 ブロック I/O チューニング要素

要素	説明
<blkiotune>	このオプション要素は、ドメインの blkio cgroup 調整可能パラメーターを調整する機能を提供します。これを省略すると、デフォルトで OS が提供するデフォルトになります。
<weight>	この任意の weight 要素は、ゲスト仮想マシンの全体的な I/O ウェイトです。この値は、100 - 1000 の範囲で指定してください。
<device>	ドメインには、ドメインが使用している各ホスト物理マシンブロックデバイスのウェイトをさらに調整する複数の**<device>**要素が含まれる場合があります。複数のゲスト仮想マシンのディスクが、1 つのホストの物理マシンブロックデバイスを共有できることに注意してください。また、このパラメーターは、同じホストの物理マシンファイルシステム内のファイルでバッキングされているため、各ゲスト仮想マシンのディスクデバイスに関連付けられるのではなく、グローバルドメインレベルで設定します (1 つの**<disk>**に適用できる **<iotune>** 要素との違いを確認してください)。各デバイス要素には、2 つの必須サブ要素があります。**<path>**はデバイスの絶対パスを表し、**<weight>** はそのデバイスの相対的な重みを示します。許容範囲は 100 - 1000 になります。

20.10. リソースのパーティション設定

ハイパーバイザーを使用すると、仮想マシンをリソースパーティションに配置できます。この場合、上記パーティションのネストが考えられます。**<resource>** 要素は、リソースのパーティション設定に関連する設定をグループ化します。現在、ドメインを配置するリソースパーティションのパスを定義するコンテンツの子要素パーティションをサポートしています。リストにパーティションがない場合は、ドメインがデフォルトパーティションに置かれます。ゲスト仮想マシンを起動する前にパーティションが存在することを確認するのは、アプリ/管理者の責任です。デフォルトでは、(ハイパーバイザー固有の) デフォルトパーティションのみが存在すると想定できます。

図20.12 リソースのパーティション設定

```
<resource>
  <partition>/virtualmachines/production</partition>
</resource>
```

現在、リソースパーティションは、QEMU ドライバーおよび LXC ドライバーで対応しています。これは、マウントされたすべてのコントローラーで、パーティションパスを cgroups ディレクトリーにマッピングします。

20.11. CPU モデルとトポロジー

このセクションでは、CPU モデルの要件を説明します。すべてのハイパーバイザーには、ゲストにデフォルトで表示される CPU 機能に関する独自のポリシーがあることに注意してください。QEMU/KVM によりゲストに提示される CPU 機能のセットは、ゲスト仮想マシンの設定で選択された CPU モデルに

よって異なります。**qemu32** と **qemu64** は基本的な CPU モデルですが、他のモデル (追加機能付き) も利用できます。各モデルとそのトポロジーは、ドメイン XML の以下の要素を使用して指定されます。

図20.13 CPU モデルとトポロジーの例 1

```
<cpu match='exact'>
    <model fallback='allow'>core2duo</model>
    <vendor>Intel</vendor>
    <topology sockets='1' cores='2' threads='1'/>
    <feature policy='disable' name='lahf_lm'/>
  </cpu>
```

図20.14 CPU モデルとトポロジーの例 2

```
<cpu mode='host-model'>
   <model fallback='forbid'/>
   <topology sockets='1' cores='2' threads='1'/>
</cpu>
```

図20.15 CPU モデルとトポロジーの例 3

```
<cpu mode='host-passthrough'/>
```

CPU モデルやその機能に制限を設けない場合は、以下のような簡単な cpu 要素を使用できます。

図20.16 CPU モデルとトポロジーの例 4

```
<cpu>
   <topology sockets='1' cores='2' threads='1'/>
</cpu>
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.9 CPU モデルとトポロジー要素

要素	説明
<cpu>	この要素には、vCPU 機能セットのすべてのパラメーターが含まれています。

要素	説明
<match>	**<cpu>** 要素に示されている機能が、使用可能な vCPU とどの程度一致している必要があるかを指定します。**<topology>** が **<cpu>** 要素にネストされた唯一の要素である場合は、**match** 属性を省略できます。**match** 属性に使用できる値は、以下のとおりです。 ● **minimum** - リストされている機能は最小要件です。示されている vCPU で利用可能な機能が他にもある可能性がありますが、これは受け入れられる最小の機能です。最小要件が満たされていない場合、この値は失敗します。 ● **exact** - ゲスト仮想マシンに提供される仮想 CPU は、指定された機能と完全に一致する必要があります。一致するものが見つからない場合、エラーが発生します。 ● **strict** - ホストの物理マシンの CPU が仕様と完全に一致しない限り、ゲスト仮想マシンが作成されません。 **match** 属性が **<cpu>** 要素から省略されている場合、デフォルト設定の **match='exact'** が使用されます。

要素	説明
<mode>	このオプション属性を使用すると、ゲスト仮想マシンの CPU を、ホストの物理マシンの CPU にできるだけ近づけるように設定できます。mode 属性に使用できる値は、以下のとおりです。 ● **custom** - CPU がゲスト仮想マシンにどのように表示されるかを説明します。**mode** 属性が指定されていない場合のデフォルト設定です。このモードでは、ゲスト仮想マシンを起動しているホストの物理マシンがどのホストの物理マシンであっても、永続ゲスト仮想マシンが同じハードウェアを認識できるようになります。 ● **host-model** - これは基本的に、ホスト物理マシンの CPU 定義を機能 XML からドメイン XML にコピーするためのショートカットです。CPU 定義はドメインの起動直前にコピーされるため、同じ XML を異なるホストの物理マシンで使用することも、各ホストの物理マシンがサポートする最適なゲスト仮想マシンの CPU を提供することもできます。このモードでは、**match** 属性も機能要素も使用できません。詳細については、libvirt ドメインの XML CPU モデルを参照してください。 ● **host-passthrough** このモードでは、ゲスト仮想マシンから見える CPU は、libvirt 内のエラーの原因となる要素を含め、ホストの物理マシン CPU とまったく同じです。このモードのデメリットは、ゲストの仮想マシン環境を異なるハードウェアで再生できないため、このモードを使用する場合は十分注意して行うことが推奨されます。このモードでは、**model** 要素も **feature** 要素も許可されていません。 ● **host-model** モードと **host-passthrough** モードの両方で、現在のホスト物理マシンで使用される実際の (ホストパススルーモードに近い) CPU 定義は、virDomainGetXMLDescAPI を呼び出すときに VIR_DOMAIN_XML_UPDATE_CPU フラグを指定することで決定できることに注意してください。異なるハードウェアが提示されたときにオペレーティングシステムが再アクティブ化される可能性があり、異なる機能を持つホスト物理マシン間で移行されるゲスト仮想マシンを実行している場合、この出力を使用して XML をカスタムモードに書き換え、より堅牢な移行を行うことができます。

要素	説明
<model>	ゲスト仮想マシンが要求する CPU モデルを指定します。利用可能な CPU モデルとその定義の一覧は、libvirt の データディレクトリーにインストールされている **cpu_map.xml** ファイルを参照してください。ハイパーバイザーが、正確な CPU モデルを使用できない場合、libvirt は、CPU 機能の一覧を維持しながら、ハイパーバイザーが対応する最も近いモデルに自動的にフォールバックします。オプションの **fallback** 属性を使用すると、この動作を禁止できます。この場合、対応していない CPU モデルを要求するドメインを起動しようとすると失敗します。フォールバック属性で対応している値は、**allow** (デフォルト) と **forbid** です。オプションの **vendor_id** 属性を使用すると、ゲスト仮想マシンが認識するベンダー id を設定できます。ちょうど 12 文字の長さである必要があります。設定しない場合は、ホスト物理マシンのベンダー id が使用されます。一般的な値は、**AuthenticAMD** および **GenuineIntel** です。
<vendor>	ゲスト仮想マシンが要求する CPU ベンダーを指定します。この要素がないと、ゲスト仮想マシンは、ベンダーに関係なく、指定された機能と一致する CPU で実行します。サポートされているベンダーの一覧は、**cpu_map.xml** を参照してください。
<topology>	ゲスト仮想マシンに提供される仮想 CPU の要求されたトポロジーを指定します。ソケット、コア、およびスレッドには、それぞれゼロ以外の値を 3 つ指定する必要があります。つまり、CPU ソケットの合計数、ソケットごとのコア数、およびコアごとのスレッド数です。

要素	説明
<feature>	選択した CPU モデルが提供する機能を微調整するために使用する、ゼロ以上の要素を含むことができます。既知の機能名のリストは、CPU モデルと同じファイルにあります。各機能要素の意味は、ポリシー属性により異なります。この属性は、次のいずれかの値に設定する必要があります。 ● **force** - 仮想マシンがホスト物理マシンの CPU で実際に対応しているかどうかに関係なく、仮想マシンを強制的に対応させます。 ● **require** - この機能がホストの物理マシンの CPU で対応していない場合は、ゲスト仮想マシンの作成に失敗することを指示します。これはデフォルト設定です。 ● **optional** - この機能は仮想 CPU で対応していますが、ホストの物理マシン CPU で対応している場合に限ります。 ● **disable** - これは仮想 CPU では対応していません。 ● **forbid** - この機能がホストの物理マシンの CPU でサポートされていると、ゲスト仮想マシンの作成に失敗します。

20.11.1. ゲスト仮想マシンの NUMA トポロジー

ゲスト仮想マシンの NUMA トポロジーは、**<numa>** 要素とドメイン XML の以下を使用して指定できます。

図20.17 ゲスト仮想マシンの NUMA トポロジー

```
<cpu>
  <numa>
    <cell cpus='0-3' memory='512000'/>
    <cell cpus='4-7' memory='512000'/>
  </numa>
</cpu>
...
```

各セル要素は、NUMA セルまたは NUMA ノードを指定します。**cpus** は、ノードの一部である CPU または CPU の範囲を指定します。**memory** は、ノードメモリーを kibibytes (1024 バイトのブロック) で指定します。各セルまたはノードには、0 から始まる昇順で **cellid** または **nodeid** が割り当てられます。

20.12. イベントの設定

ドメイン XML の以下のセクションを使用すると、さまざまなイベントで実行されるデフォルトのアクションを上書きできます。

図20.18 イベントの設定

```
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>restart</on_crash>
<on_lockfailure>poweroff</on_lockfailure>
```

以下の要素のコレクションでは、ゲスト仮想マシンの OS が、ライフサイクル操作をトリガーする場合にアクションを指定できます。一般的な使用例は、OS の初期インストールを行うときに、再起動を強制的に電源オフとして扱うことです。これにより、仮想マシンをインストール後の初回起動時に再設定できます。

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.10 イベント設定要素

状態	説明
<on_poweroff>	ゲスト仮想マシンがパワーオフを要求した場合に実行するアクションを指定します。使用可能な引数は、以下の 4 つです。 ● **destroy** - この操作によりドメインが完全に終了し、すべてのリソースが解放されます。 ● **restart** - この操作によりドメインが完全に終了し、同じ設定で再起動します。 ● **preserve** - この操作によりドメインは完全に終了しますが、リソースは保持されるため、今後の解析に使用できます。 ● **rename-restart** - この操作によりドメインが完全に終了し、続いて新しい名前で再起動します。
<on_reboot>	ゲスト仮想マシンが再起動を要求した場合に実行するアクションを指定します。使用可能な引数は、以下の 4 つです。 ● **destroy** - この操作によりドメインが完全に終了し、すべてのリソースが解放されます。 ● **restart** - この操作によりドメインが完全に終了し、同じ設定で再起動します。 ● **preserve** - この操作によりドメインは完全に終了しますが、リソースは保持されるため、今後の解析に使用できます。 ● **rename-restart** - この操作によりドメインが完全に終了し、続いて新しい名前で再起動します。

状態	説明
<on_crash>	ゲスト仮想マシンがクラッシュした場合に実行するアクションを指定します。さらに、以下の追加アクションに対応します。 ● **coredump-destroy** - クラッシュしたドメインのコアがダンプされ、ドメインが完全に終了し、すべてのリソースが解放されます。 ● **coredump-restart** - クラッシュしたドメインのコアがダンプされ、同じ設定でドメインが再起動します。 使用可能な引数は、以下の 4 つです。 ● **destroy** - この操作によりドメインが完全に終了し、すべてのリソースが解放されます。 ● **restart** - この操作によりドメインが完全に終了し、同じ設定で再起動します。 ● **preserve** - この操作によりドメインは完全に終了しますが、リソースは保持されるため、今後の解析に使用できます。 ● **rename-restart** - この操作によりドメインが完全に終了し、続いて新しい名前で再起動します。
<on_lockfailure>	ロックマネージャーがリソースロックを失ったときに実行するアクションを指定します。以下のアクションは、libvirt によって認識されますが、すべてが個々のロックマネージャーによってサポートされる必要はありません。アクションを指定しないと、各ロックマネージャーはデフォルトのアクションを実行します。以下の引数を使用できます。 ● **poweroff** - ドメインの電源を強制的に切断します。 ● **restart** - ドメインを再起動して、ロックを再取得します。 ● **pause** - ロックの問題が解決したときに手動で再開できるように、ドメインを一時停止します。 ● **ignore** - 何も起こらなかったかのようにドメインを実行し続けます。

20.13. 電源管理

ドメイン XML の次のセクションに影響を与える従来の管理ツールを使用して、ゲスト仮想マシン OS への BIOS アドバタイズメントを強制的に有効または無効にすることができます。

図20.19 電源管理

```
...
<pm>
  <suspend-to-disk enabled='no'/>
  <suspend-to-mem enabled='yes'/>
</pm>
...
```

<pm> 要素は、引数 **yes** を使用して有効にすることも、引数 **no** を使用して無効にすることもできます。BIOS サポートは、引数 **suspend-to-disk** を使用して S3 に実装でき、引数 **suspend-to-mem** スリープ状態を使用して S4 に実装できます。何も指定しないと、ハイパーバイザーはデフォルト値のままになります。

20.14. ハイパーバイザーの機能

ハイパーバイザーを使用すると、特定の CPU/マシン機能を有効 (**state='on'**) または無効 (**state='off'**) にできます。

図20.20 ハイパーバイザーの機能

```
...
<features>
  <pae/>
  <acpi/>
  <apic/>
  <hap/>
  <privnet/>
  <hyperv>
    <relaxed state='on'/>
  </hyperv>
</features>
...
```

すべての機能は、**<features>** 要素に一覧表示されます。**<state>**を指定しないと無効になります。利用可能な機能は、**capabilities** XML を呼び出して確認できますが、完全仮想化ドメインの一般的な設定は次のとおりです。

表20.11 ハイパーバイザーの機能要素

状態	説明
<pae>	物理アドレス拡張モードでは、32 ビットのゲスト仮想マシンが 4GB を超えるメモリーにアドレスを指定できます。
<acpi>	電力管理に役立ちます。たとえば、KVM ゲスト仮想マシンでは、正常なシャットダウンが機能する必要があります。

状態	説明
<apic>	プログラム可能な IRQ 管理を使用できるようにします。この要素には、**on** および **off** の値を持つオプションの属性**eoi** があります。これは、ゲスト仮想マシンの EOI (割り込みの終了) の可用性を設定します。
<hap>	ハードウェアで利用可能な場合に、ハードウェア支援ページングの使用を有効にします。
hyperv	Microsoft Windows を実行しているゲスト仮想マシンの動作を改善するためのさまざまな機能を有効にします。値を **on** または **off** にして **緩和された**オプションの属性を使用すると、タイマーの緩和制約が有効または無効になります

20.15. 時間管理

ゲスト仮想マシンのクロックは、通常、ホストの物理マシンのクロックから初期化されます。ほとんどのオペレーティングシステムでは、ハードウェアクロックは UTC (デフォルト設定) に保持されることが想定されています。Windows ゲスト仮想マシンの場合、ゲスト仮想マシンは **localtime** に設定する必要があることに注意してください。

図20.21 時間管理

```
...
<clock offset='localtime'>
  <timer name='rtc' tickpolicy='catchup' track='guest'>
    <catchup threshold='123' slew='120' limit='10000'/>
  </timer>
  <timer name='pit' tickpolicy='delay'/>
</clock>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.12 時間管理要素

状態	説明

状態	説明
<clock>	**offset** 属性には 4 つの値を指定できます。これにより、ゲスト仮想マシンのクロックをホスト物理マシンに同期させる方法を詳細に制御できます。ハイパーバイザーは、常にすべてのポリシーをサポートする必要はないことに注意してください。 ● **utc** - 起動時にクロックを UTC に同期します。**utc** モードは、adjustment 属性を使用して制御できる **variable** モードに変換できます。値が **reset** の場合、変換は行われません。数値を入力すると、その値を初期調整として使用し、強制的に **variable** モードに変換されます。デフォルトの調整は、ハイパーバイザー固有です。 ● **localtime** - ゲスト仮想マシンのクロックを、起動時にホスト物理マシンに設定されたタイムゾーンと同期します。調整属性の動作は、utc モードと同じです。 ● **timezone** - timezone 属性を使用して、ゲスト仮想マシンのクロックを要求されたタイムゾーンに同期します。 ● **variable** - ゲスト仮想マシンクロックに、basis 属性に応じて UTC または localtime を基準にした任意のオフセットを適用します。UTC (または localtime) に対する差分は、**adjustment** 属性を使用して秒単位で指定します。ゲスト仮想マシンは、時間の経過とともに RTC を調整する自由があり、次回のシステムの再起動時に有効になると予想されます。これは、**utc** モードおよび **localtime** モード (オプション属性の **adjustment='reset'** を使用) とは対照的です。ここでは、システムの再起動ごとに RTC 調整が失われます。また、**basis** 属性は **utc** (デフォルト) または **localtime** のいずれかになります。**clock** 要素には、0 個以上の**<timer>** 要素を指定できます。
<timer>	下記を参照してください。
<frequency>	これは、**name**="**tsc**" が実行される頻度を指定する符号なし整数です。
<mode>	**mode** 属性は、**name = "tsc"<timer>** の管理方法を制御し、**auto**、**native**、**emulate**、**paravirt**、または **smpsafe** に設定できます。その他のタイマーは常にエミュレートされます。
<あり>	ゲスト仮想マシンで特定のタイマーが使用可能であるかどうかを指定します。**yes** または **no** に設定できます。

注記

各 **<timer>** 要素には **name** 属性が含まれている必要があり、指定された名前に応じて次の属性を持つ場合があります。

- **<name>** - 変更する**timer**を選択します。次の値が許容されます。**kvmclock** (QEMU-KVM), **pit**(QEMU-KVM)、または **rtc**(QEMU-KVM)、または **tsc**(libxl のみ)。**platform** は現在サポートされていないことに注意してください。
- track - タイマートラックを指定します。次の値を使用できます: **boot**、**guest**、または **wall**。**track** は **name="rtc"**に対してのみ有効です。
- **tickpolicy** - ゲスト仮想マシンにティックを挿入する期限が過ぎた場合の動作を指定します。設定可能な値は、以下のとおりです。
  - **delay** - 通常のレートでティックを配信し続けます。ゲスト仮想マシンの時間は、ティックの遅延により遅れます。
  - **catchup** - ティックの遅れを取り戻すために、高いレートでティックを配信しますキャッチアップが完了すると、ゲスト仮想マシンの時間は表示されません。また、オプションとして、threshold、slew、limit の 3 つの属性があり、それぞれ正の整数とすることができます。
  - **merge** - 遅れたティックを単一のティックにマージし、それらを挿入します。マージの実行方法によっては、ゲスト仮想マシンの時間が遅れる可能性があります。
  - **discard** - 遅れたティックを破棄し、デフォルトの間隔設定で今後の挿入を続行します。失われたティックを処理する明示的なステートメントがない限り、ゲスト仮想マシンの時間が遅延する場合があります。

20.16. DEVICES

この XML 要素のセットはすべて、ゲスト仮想マシンのドメインに提供されるデバイスを説明するために使用されます。以下のデバイスはすべて、主な devices 要素の子として示されます。

以下の仮想デバイスに対応しています。

- virtio-scsi-pci - PCI バスストレージデバイス
- virtio-9p-pci - PCI バスストレージデバイス
- virtio-blk-pci - PCI バスストレージデバイス
- virtio-net-pci - PCI バスネットワークデバイス (virtio-net としても知られる)
- virtio-serial-pci - PCI バス入力デバイス
- virtio-balloon-pci - PCI バスメモリーバルーンデバイス
- virtio-rng-pci - PCI バス仮想乱数発生器

重要

virtio デバイスを作成し、ベクトル数を 32 より大きい値に設定すると、デバイスは、Red Hat Enterprise Linux 6 ではゼロに設定されているかのように動作しますが、Enterprise Linux 7 では動作しません。生成されるベクトル設定の不一致により、いずれかのプラットフォームの virtio デバイスのベクトル数が 33 以上に設定されていると、移行エラーが発生します。したがって、**vector** の値を 32 より大きく設定することは推奨されません。virtio-balloon-pci および virtio-rng-pci を除くすべての virtio デバイスは、**vector** 引数を受け入れます。

図20.22 デバイス - 子要素

```
...
<devices>
  <emulator>/usr/lib/xen/bin/qemu-dm</emulator>
</devices>
...
```

<emulator> 要素の内容は、デバイスモデルエミュレーターバイナリーへの完全修飾パスを指定します。capabilities XML は、各特定のドメインタイプまたはアーキテクチャーの組み合わせに使用する推奨されるデフォルトエミュレーターを指定します。

20.16.1. ハードドライブ、フロッピーディスク、CDROM

ドメイン XML のこのセクションでは、フロッピー、ハードディスク、CD-ROM、準仮想化ドライバーなど、ディスクのように見えるデバイスをディスク要素で指定します。

図20.23 デバイス - ハードドライブ、フロッピーディスク、CDROM

```
...
 <devices>
   <disk type='file' snapshot='external'>
     <driver name="tap" type="aio" cache="default"/>
     <source file='/var/lib/xen/images/fv0' startupPolicy='optional'>
       <seclabel relabel='no'/>
     </source>
     <target dev='hda' bus='ide'/>
     <iotune>
       <total_bytes_sec>10000000</total_bytes_sec>
       <read_iops_sec>400000</read_iops_sec>
       <write_iops_sec>100000</write_iops_sec>
     </iotune>
     <boot order='2'/>
     <encryption type='...'>
       ...
     </encryption>
     <shareable/>
     <serial>
       ...
     </serial>
   </disk>
     ...
   <disk type='network'>
```

```
    <driver name="qemu" type="raw" io="threads" ioeventfd="on" event_idx="off"/>
    <source protocol="sheepdog" name="image_name">
      <host name="hostname" port="7000"/>
    </source>
    <target dev="hdb" bus="ide"/>
    <boot order='1'/>
    <transient/>
    <address type='drive' controller='0' bus='1' unit='0'/>
  </disk>
  <disk type='network'>
    <driver name="qemu" type="raw"/>
    <source protocol="rbd" name="image_name2">
      <host name="hostname" port="7000"/>
    </source>
    <target dev="hdd" bus="ide"/>
    <auth username='myuser'>
      <secret type='ceph' usage='mypassid'/>
    </auth>
  </disk>
  <disk type='block' device='cdrom'>
    <driver name='qemu' type='raw'/>
    <target dev='hdc' bus='ide' tray='open'/>
    <readonly/>
  </disk>
  <disk type='block' device='lun'>
    <driver name='qemu' type='raw'/>
    <source dev='/dev/sda'/>
    <target dev='sda' bus='scsi'/>
    <address type='drive' controller='0' bus='0' target='3' unit='0'/>
  </disk>
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw'/>
    <source dev='/dev/sda'/>
    <geometry cyls='16383' heads='16' secs='63' trans='lba'/>
    <blockio logical_block_size='512' physical_block_size='4096'/>
    <target dev='hda' bus='ide'/>
  </disk>
  <disk type='volume' device='disk'>
    <driver name='qemu' type='raw'/>
    <source pool='blk-pool0' volume='blk-pool0-vol0'/>
    <target dev='hda' bus='ide'/>
  </disk>
</devices>
...
```

20.16.1.1. ディスク要素

<disk> 要素は、ディスクを記述するための主要なコンテナーです。属性**type** は、**<disk>** 要素で使用できます。以下のタイプが許可されています。

- **file**
- **block**
- **dir**

- **network**

詳細については、ディスク要素を参照してください。

20.16.1.2. ソース要素

<disk type='file'> の場合、**file** 属性は、ディスクを保持しているファイルへの完全修飾パスを指定します。**<disk type='block'>** の場合、**dev** 属性は、ディスクとして機能するホスト物理マシンデバイスへのパスを指定します。**file** と **block** の両方で、以下で説明する 1 つ以上のオプションのサブ要素 **seclabel** を使用して、そのソースファイルのみのドメインセキュリティーラベル付けポリシーを上書きできます。ディスクタイプが **dir** の場合、**dir** 属性は、ディスクとして使用するディレクトリーへの完全修飾パスを指定します。ディスクタイプが **network** の場合、protocol 属性は、要求されたイメージにアクセスするためのプロトコルを指定します。可能な値は、**nbd**、**rbd**、**sheepdog**、または **gluster** です。

プロトコル属性が **rbd**、**sheepdog**、または **gluster** の場合、使用するボリュームやイメージを指定するには、追加の属性 **name** が必須です。ディスクのタイプが **network** の場合、**source** には、接続するホスト物理マシンを指定するために使用される **type='dir'** および **type='network'** を含む、ゼロ以上の **host** サブ要素が含まれることがあります。cdrom またはフロッピー (デバイス属性) を表す **file**ディスクタイプで、ソースファイルにアクセスできない場合にディスクをどうするかという動作のポリシーを定義できます。これは、**startupPolicy** 属性を次の値で操作することによって行われます。

- **mandatory** は、何らかの理由で欠落している場合に障害の原因となります。これはデフォルト設定です。
- **requisite** は、起動時に欠落している場合は失敗し、移行/復元/復帰時に欠落している場合はドロップします。
- **optional** - 起動の試行時にない場合はドロップします。

20.16.1.3. ミラー要素

この要素は、ハイパーバイザーが **BlockCopy** 操作を起動し、属性ファイルの **<mirror>** の場所が最終的にソースと同じ内容を持ち、属性形式のファイル形式 (ソースの形式とは異なる場合あり) の場合に存在します。属性 ready が存在する場合は、ディスクがピボットする準備ができていることが分かっています。存在しない場合は、ディスクが依然としてコピー中である可能性があります。現在、この要素は出力でのみ有効で、入力では無視されます。

20.16.1.4. ターゲット要素

<target> 要素は、ディスクがゲスト仮想マシン OS に公開されるバス/デバイスを制御します。dev 属性は、論理デバイスの名前を示します。実際のデバイス名が、ゲスト仮想マシンの OS のデバイス名にマッピングされる保証はありません。オプションのバス属性では、エミュレートするディスクデバイスの種類を指定します。指定できる値はドライバー固有で、一般的な値は **ide**、**scsi**、**virtio**、**xen**、**usb**、または **sata** です。省略した場合、バスタイプはデバイス名のスタイルから推測されます。たとえば、**'sda'** という名前のデバイスは通常、SCSI バスを使用してエクスポートされます。オプションの属性 **tray** は、リムーバブルディスク (CD-ROM やフロッピーディスクなど) のトレイステータスを示します。値は **open** または **closed** のいずれかです。デフォルト設定は **closed** です。詳細は、ターゲット要素を参照してください。

20.16.1.5. iotune

オプションの **<iotune>** 要素は、デバイスごとに異なる値を使用して、追加のデバイスごとの I/O チューニングを提供する機能を提供します (これを、ドメインにグローバルに適用する**blkiotune**要素と比較してください)。この要素には、次のオプションのサブ要素があります。指定されないか、または

値 **0** が指定されたサブ要素はすべて制限がないことを意味します。

- **<total_bytes_sec>** - 合計スループット制限 (バイト/秒)。この要素は、**<read_bytes_sec>** または **<write_bytes_sec>** とは併用できません。
- **<read_bytes_sec>** - 読み取りスループットの制限 (バイト/秒)。
- **<write_bytes_sec>** - 書き込みスループットの制限 (バイト/秒)。
- **<total_iops_sec>** - 1秒あたりの I/O 操作の合計です。この要素は、 **<read_iops_sec>** または **<write_iops_sec>** とは併用できません。
- **<read_iops_sec>** - 1秒あたりの読み取り I/O 操作数。
- **<write_iops_sec>** - 1秒あたりの書き込み I/O 操作数。

20.16.1.6. driver

任意の **<driver>** 要素を使用すると、ディスクの提供に使用されるハイパーバイザードライバーに関する詳細を指定できます。以下のオプションが使用できます。

- ハイパーバイザーが複数のバックエンドドライバーをサポートする場合、**name** 属性はプライマリーバックエンドドライバー名を選択し、任意の type 属性はサブタイプを提供します。可能なタイプのリストについては、ドライバー要素を参照してください。
- オプションの **cache** 属性は、キャッシュメカニズムを制御します。使用できる値は、**default**、**none**、**writethrough**、**writeback**、**directsync** (**writethrough**と同様ですが、ホスト物理マシンのページキャッシュをバイパスします) および **unsafe** (ホスト物理マシンはすべてのディスク io をキャッシュする場合があります) です。ゲスト仮想マシン仮想マシンからの同期要求は無視されます)。
- オプションの **error_policy** 属性は、ハイパーバイザーがディスクの読み取りまたは書き込みエラーでどのように動作するかを制御します。可能な値は、**stop**、**report**、**ignore**、および **enospace**です。**error_policy** のデフォルト設定は **report** です。読み取りエラーのみに対する動作を制御するオプションの**rerror_policy**もあります。**rerror_policy** が指定されていない場合、**error_policy** は読み取りおよび書き込みエラーの両方に使用されます。**rerror_policy** が指定されている場合は、読み取りエラーの **error_policy** をオーバーライドします。また、**enospace** は読み取りエラーの有効なポリシーではないため、**error_policy** が **enospace** に設定され、**no rerror_policy** が指定されていない場合は、読み取りエラーのデフォルト設定である **report** が使用される点に注意してください。
- オプションの **io** 属性は、I/O の特定のポリシーを制御します。**qemu** ゲスト仮想マシン仮想マシンは **threads** と **native** をサポートします。オプションの **ioeventfd** 属性を使用すると、ディスクデバイスのドメイン I/O の非同期処理を設定できます。デフォルトは、ハイパーバイザーの判断に任されます。指定できる値は **on** と **off** です。これを有効にすると、別のスレッドが I/O を処理している間にゲスト仮想マシンの仮想マシンを実行できます。通常、I/O の実行中にシステム CPU の使用率が高くなったゲスト仮想マシンの仮想マシンは、この恩恵を受けます。一方、ホストの物理マシンが過負荷になると、ゲスト仮想マシンの仮想マシンの I/O レイテンシーが増加します。**io** を操作する必要があることを完全に証明している場合を除き、デフォルト設定を変更せず、ハイパーバイザーが設定を指示できるようにすることを強くお勧めします。
- オプションの **event_idx** 属性は、デバイスイベント処理のいくつかの側面を制御し、**on** または **off** のいずれかに設定できます。on の場合、ゲスト仮想マシン仮想マシンの割り込みと終了の数を減らします。デフォルトではハイパーバイザーが決定し、デフォルトの設定は **on** になります。この動作が最適ではない状況がある場合、この属性は機能を強制的に **off** にする方法を

提供します。**event_idx** を操作する必要があることを完全に証明している場合を除き、デフォルト設定を変更せず、ハイパーバイザーが設定を指示できるようにすることを強くお勧めします。

- オプションの **copy_on_read** 属性は、リードバックファイルをイメージファイルにコピーするかどうかを制御します。使用できる値は、**on** または **<off>** のいずれかです。**copy-on-read** は、同じバッキングファイルセクターに繰り返しアクセスすることを回避し、バッキングファイルが低速のネットワーク上にある場合に役立ちます。デフォルトでは、**copy-on-read**は **off** です。

20.16.1.7. 追加のデバイス要素

device 要素内では、以下の属性を使用できます。

- **<boot>** - ディスクが起動可能であることを指定します。

  追加の起動値

  - **<order>** - システムの起動時にデバイスを試行する順序を指定します。
  - **<デバイスごと>**のブート要素は、BIOS ブートローダーセクションの一般的なブート要素とは併用できません。

- **<encryption>** - ボリュームの暗号化方法を指定します。詳細については、ストレージの暗号化ページを参照してください。

- **<readonly>** - ゲスト仮想マシンがデバイスを変更できないことを示します。この設定は、**attribute device='cdrom'** を使用するディスクの既定値です。

- **共有可能** デバイスがドメイン間で共有される必要があることを示しています (ハイパーバイザーと OS が対応している場合)。**shareable** を使用すると、そのデバイスに **cache='no'** が指定されます。

- **<transient>** - デバイスのコンテンツの変更は、ゲスト仮想マシンの仮想マシンの終了時に自動的に元に戻す必要があることを示します。一部のハイパーバイザーでは、ディスクに **transient** マークを付けると、ドメインが移行またはスナップショットに参加できなくなります。

- **<serial>** - ゲスト仮想マシンのハードドライブのシリアル番号を指定します。(例: **<serial>**WD-WMAP9A966149**</serial>**)

- **<wwn>** - 仮想ハードディスクまたは CD-ROM ドライブの WWN (World Wide Name) を指定します。16 桁の 16 進数で設定される必要があります。

- **<vendor>** - 仮想ハードディスクまたは CD-ROM デバイスのベンダーを指定します。印刷可能な 8 文字を超えてはなりません。

- **<product>** - 仮想ハードディスクまたは CD-ROM デバイスの製品を指定します。印刷可能な 16 文字を超えてはなりません。

- **<host>** - 4 つの属性をサポートします: **viz**、**name**、**port**、**transport**、**socket**。それぞれ、ホスト名、ポート番号、トランスポートタイプ、およびソケットへのパスを指定します。この要素の意味と要素数は、以下に示すように **protocol** 属性により異なります。

  追加のホスト属性

- **nbd** - nbd-server を実行しているサーバーを指定します。使用できるのは、1 台のホスト物理マシンだけです。
- **rbd** - RBD タイプのサーバーを監視し、1 つ以上のホスト物理マシンで使用できます。
- **sheepdog** - sheepdog サーバーのいずれかを指定し (デフォルトは localhost:7000)、ホスト物理マシンの 1 つを使用するか、またはいずれとも使用することができません。
- **gluster** - glusterd デーモンを実行しているサーバーを指定します。使用できる物理マシンは 1 つだけです。トランスポート属性の有効な値は、**tcp**、**rdma**、または **unix** です。何も指定しないと、**tcp** が想定されます。transport が **unix** の場合、**socket** 属性は unix ソケットへのパスを指定します。

- **<address>** - ディスクを、コントローラーの指定したスロットに関連付けます。実際の **<controller>** デバイスは多くの場合、推測できますが、明示的に指定することもできます。**type** 属性は必須で、通常は **pci** または **drive** です。**pci** コントローラーの場合、**bus**、**slot**、および **function** の追加属性と、オプションの **domain** および **multifunction** が存在する必要があります。**multifunction** のデフォルトは **off** です。**drive** コントローラーでは、追加の属性 **controller**、**bus**、**target**、および **unit** が利用できます。それぞれの属性のデフォルト設定は **0** です。
- **auth** - ソースへのアクセスに必要な認証情報を提供します。これには、認証時に使用するユーザー名を識別する必須属性**username**と、必須属性**type**を持つサブ要素**secret** が含まれます。詳細は、デバイス要素を参照してください。
- **geometry** - ジオメトリー設定を上書きする機能を提供します。これは、主に S390 DASD ディスクまたは古い DOS ディスクに役立ちます。
- **cyls** - シリンダーの数を指定します。
- **heads** - ヘッドの数を指定します。
- **secs** - トラックごとのセクター数を指定します。
- **trans** - BIOS-Translation-Modus を指定し、 **none**、**lba**、または **auto** の値を設定できます
- **blockio** - ブロックデバイスを、以下のブロックデバイスプロパティーのいずれかで上書きできます。

  blockio **オプション**

  - **logical_block_size** - ゲスト仮想マシン仮想マシン OS にレポートし、ディスク I/O の最小単位を記述します。
  - **physical_block_size** - ゲスト仮想マシンの仮想マシン OS を報告し、ディスクのハードウェアセクターサイズを説明します。これは、ディスクデータの調整に関連付けることができます。

20.16.2. ファイルシステム

ゲスト仮想マシン仮想マシンから直接アクセスできるホスト物理マシン上のファイルシステムディレクトリー

図20.24 デバイス - ファイルシステム

```
...
 <devices>
  <filesystem type='template'>
    <source name='my-vm-template'/>
    <target dir='/'/>
  </filesystem>
  <filesystem type='mount' accessmode='passthrough'>
    <driver type='path' wrpolicy='immediate'/>
    <source dir='/export/to/guest'/>
    <target dir='/import/from/host'/>
    <readonly/>
  </filesystem>
   ...
 </devices>
 ...
```

filesystem 属性には、次の可能な値があります。

- **type='mount'** - ゲスト仮想マシンにマウントするホスト物理マシンディレクトリーを指定します。指定されていない場合、これがデフォルトのタイプです。このモードには、属性 **type='path'** または **type='handle'** を持つオプションのサブ要素 **driver** もあります。ドライバーブロックには、ホストの物理マシンページキャッシュとの相互作用をさらに制御するオプションの属性 **wrpolicy** があります。属性を省略するとデフォルト設定に戻りますが、値を immediate に指定すると、ゲスト仮想マシンのファイル書き込み操作中にタッチされたすべてのページに対してホスト物理マシンのライトバックが即座にトリガーされます。

- **type='template'** - OpenVZ ファイルシステムテンプレートを指定し、OpenVZ ドライバーによってのみ使用されます。

- **type='file'** - ホスト物理マシンファイルがイメージとして扱われ、ゲスト仮想マシンにマウントされることを指定します。このファイルシステム形式は自動検出され、LXC ドライバーでのみ使用されます。

- **type='block'** - ゲスト仮想マシンにマウントするホスト物理マシンブロックデバイスを指定します。ファイルシステム形式は自動検出され、LXC ドライバーでのみ使用されます。

- **type='ram'** - ホスト物理マシン OS のメモリーを使用するメモリー内ファイルシステムが使用されることを指定します。source 要素には単一の属性 **usage** があり、メモリー使用量の制限をキビバイト単位で指定し、LXC ドライバーでのみ使用されます。

- **type='bind'** - ゲスト仮想マシン内の別のディレクトリーにバインドされるゲスト仮想マシン内のディレクトリーを指定します。この要素は LXC ドライバーでのみ使用されます。

- ソースにアクセスするためのセキュリティーモードを指定する **accessmode**。現在、これは QEMU/KVM ドライバーの type='mount' でのみ機能します。以下の値を使用できます。

  - **passthrough** - ゲスト仮想マシン内から設定されたユーザーのアクセス許可設定を使用してソースにアクセスすることを指定します。指定されていない場合、これがデフォルトのアクセスモードです。

  - **mapped** - ハイパーバイザーの権限設定を使用してソースにアクセスすることを指定します。

- **squash** - **'passthrough'** と同様に、例外は、**chown** などの特権操作の失敗が無視されることです。これにより、ハイパーバイザーを非ルートとして実行するユーザーがパススルーのようなモードを使用できるようになります。

- **<source>**: ゲスト仮想マシンでアクセスされるホストの物理マシンのリソースを指定します。**name** 属性は **<type='template'>** と共に使用する必要があります。**dir** 属性は **<type='mount'>** と共に使用する必要があります。**usage** 属性は、**<type='ram'>** とともに使用され、メモリー使用量を KB 単位で設定します。

- **target** - ゲスト仮想マシンでソースドライバーにアクセスできる場所を指定します。ほとんどのドライバーの場合、これは自動マウントポイントですが、QEMU-KVM の場合、これはマウントする場所のヒントとしてゲスト仮想マシンにエクスポートされる任意の文字列タグにすぎません。

- **readonly** - ファイルシステムをゲスト仮想マシンの読み取り専用マウントとしてエクスポートできるようにします。デフォルトでは、**read-write** アクセスが許可されています。

- **space_hard_limit** - このゲスト仮想マシンのファイルシステムで使用可能な最大スペースを指定します

- **space_soft_limit** - このゲスト仮想マシンのファイルシステムで使用可能な最大スペースを指定します。コンテナーは、猶予期間中、ソフト制限を超えることが許可されています。その後、ハード制限が適用されます。

20.16.3. デバイスアドレス

多くのデバイスには、仮想バスに配置されたデバイスがゲスト仮想マシンに提示される場所を説明するオプションの **<address>** サブ要素があります。入力でアドレス (またはアドレス内の任意の属性) が省略された場合、libvirt は適切なアドレスを生成します。レイアウトをさらに制御する必要がある場合は明示的なアドレスが必要になります。address 要素を含むデバイスの例は、以下を参照してください。

各アドレスには、デバイスが稼働しているバスを説明する必須の属性 ***type*** があります。特定のデバイスに使用するアドレスの選択は、デバイスやゲスト仮想マシンのアーキテクチャーによって一部が制約されます。たとえば、ディスクデバイスは **type='disk'** を使用し、コンソールデバイスは、32 ビット AMD および Intel アーキテクチャーの **type='pci'**、または AMD64 と Intel 64、ゲスト仮想マシン、または PowerPC64 pseries ゲスト仮想マシンの **type='spapr-vio'**使用します。各アドレス**<type>**には、デバイスの配置先を制御する追加のオプション属性があります。追加の属性は、以下のとおりです。

- **type='pci'** - PCI アドレスには、以下の追加属性があります。

  - **ドメイン** (2 バイトの 16 進数の整数で、現在 qemu で使用されていません)

  - **bus** (0 から 0xff までの 16 進数の値)

  - **slot** (0x0 から 0x1f までの 16 進数の値)

  - **function** (0 から 7 までの値)

  - また、**multi-function** 属性も利用できます。これは、PCI 制御レジスターの特定のスロット/機能に対して、多機能ビットをオンにすることを制御します。この多機能属性は、デフォルトでは **'off'** ですが、多機能を使用するスロットの機能 0 の場合は **'on'** に設定する必要があります。

- **type='drive'** - drive アドレスには、以下の追加属性があります。

  - **controller** - (2 桁のコントローラー番号)

- **bus** - (2 桁のバス番号)
- **target** - (2 桁のバス番号)
- **unit** - (バス上の 2 桁のユニット番号)

- **type='virtio-serial'** - 各 virtio-serial アドレスには、以下の追加属性があります。
  - **controller** - (2 桁のコントローラー番号)
  - **bus** - (2 桁のバス番号)
  - **slot** - (バス内の 2 桁のスロット)
- **type='ccid'** - スマートカードに使用される CCID アドレスには、以下の追加属性があります。
  - **bus** - (2 桁のバス番号)
  - **スロット**属性 (バス内の 2 桁のスロット)
- **type='usb'** - USB アドレスには、以下の追加属性があります。
  - **bus** - (0 から 0xfff までの 16 進数の値)
  - **port** - (1.2 または 2.1.3.1 などの最大 4 オクテットのドット表記)
- **type='spapr-vio'** - PowerPC pseries ゲスト仮想マシンでは、SPAPR-VIO バスにデバイスを割り当てることができます。フラットな 64 ビットのアドレス空間を持ちます。通常、デバイスは 0x1000 の倍数 (ゼロ以外) で割り当てられますが、その他のアドレスは有効で、libvirt で許可されています。追加の属性: reg (開始レジスタの 16 進値アドレス) をこの属性に割り当てることができます。

20.16.4. コントローラー

ゲスト仮想マシンのアーキテクチャーによっては、1 つのバスに多くの仮想デバイスを割り当てることができます。通常の状況では、libvirt はバスに使用するコントローラーを自動的に推測できます。ただし、ゲスト仮想マシン XML で明示的な **<controller>** 要素を指定する必要がある場合があります。

図20.25 コントローラー要素

```
...
<devices>
  <controller type='ide' index='0'/>
  <controller type='virtio-serial' index='0' ports='16' vectors='4'/>
  <controller type='virtio-serial' index='1'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
  <controller type='scsi' index='0' model='virtio-scsi' num_queues='8'/>
  </controller>
  ...
</devices>
...
```

各コントローラーには、必須属性**type** (**"ide", "fdc", "scsi", "sata", "usb", "ccid", or "virtio-serial"** のいずれか) と、(**address** 要素のコントローラー属性で使用するために) バスコントローラーが検出され

る順序を表す 10 進数の整数である必須属性**index** があります。**"virtio-serial"** コントローラーには、さらに 2 つのオプション属性 (**ports** および **vectors**) があり、コントローラーを介して接続できるデバイスの数を制御します。

<controller type='scsi'> には、**"auto"**、**"buslogic"**、**"ibmvscsi"**、**"lsilogic"**、**"lsias1068"**、**"virtio-scsi"**、**"vmpvscsi"** の 1 つであるオプションの属性 **model** があります。virtio-scsi コントローラーとドライバーは、KVM と Windows の両方のゲスト仮想マシンで動作することに注意してください。**<controller type='scsi'>** には、指定したキュー数のマルチキューサポートを有効にする属性 **num_queues** もあります。

"usb" コントローラーには、**"piix3-uhci"**、**"piix4-uhci"**、**"ehci"**、**"ich9-ehci1"**、**"ich9-uhci1"**、**"ich9-uhci2"**、**"ich9-uhci3"**、**"vt82c686b-uhci"**、**"pci-ohci"**、または、**"nec-xhci"** の 1 つであるオプションの属性 **model** があります。また、ゲスト仮想マシンで USB バスを明示的に無効にする必要がある場合は、**model='none'** を使用できます。PowerPC64 "spapr-vio" アドレスには、関連付けられたコントローラーがありません。

コントローラー自体が PCI バスまたは USB バスにある場合は、オプションのサブ要素 **address** は、上記のセマンティクスを使用して、コントローラーとマスターバスの正確な関係を指定できます。

USB コンパニオンコントローラーには、コンパニオンとマスターコントローラーの完全なリレーションを指定するためのオプションのサブ要素 **master** があります。コンパニオンコントローラーはマスターと同じバスにあるため、コンパニオンインデックスの値は同じである必要があります。

図20.26 デバイス - コントローラー - USB

```
...
<devices>
  <controller type='usb' index='0' model='ich9-ehci1'>
    <address type='pci' domain='0' bus='0' slot='4' function='7'/>
  </controller>
  <controller type='usb' index='0' model='ich9-uhci1'>
    <master startport='0'/>
    <address type='pci' domain='0' bus='0' slot='4' function='0' multifunction='on'/>
  </controller>
  ...
</devices>
...
```

20.16.5. デバイスリース

ロックマネージャーを使用する場合は、ゲスト仮想マシンに対してデバイスリースを記録するオプションがあります。ロックマネージャーは、リースを取得できない限り、ゲスト仮想マシンを起動しないようにします。従来の管理ツールを使用して設定すると、ドメイン xml の以下のセクションが影響を受けます。

図20.27 デバイス - デバイスリース

```
...
<devices>
  ...
  <lease>
    <lockspace>somearea</lockspace>
    <key>somekey</key>
    <target path='/some/lease/path' offset='1024'/>
  </lease>
  ...
</devices>
...
```

lease セクションには、以下の引数を指定できます。

- **lockspace** - 鍵が保持されているロック領域を識別する任意の文字列です。ロックマネージャーは、ロックスペース名の形式や長さに追加の制限を課す場合があります。
- **key** - 取得するリースを一意に識別する任意の文字列。ロックマネージャーは、キーの形式や長さに追加の制限を課す場合があります。
- **target** - ロックスペースに関連付けられたファイルの完全修飾パス。オフセットは、リースがファイル内に保存される場所を指定します。ロックマネージャーがオフセットを必要としない場合は、この値を **0** に設定します。

20.16.6. ホスト物理マシンのデバイス割り当て

このセクションでは、ホストの物理マシンデバイスの割り当てに関する情報を提供します。

20.16.6.1. USB / PCI デバイス

ホスト物理マシンの USB および PCI デバイスは、**hostdev** 要素を使用してゲスト仮想マシンに渡すことができます。管理ツールを使用してホスト物理マシンを変更することにより、ドメイン xml ファイルの以下のセクションが設定されます。

図20.28 デバイス - ホスト物理マシンのデバイス割り当て

```
...
<devices>
  <hostdev mode='subsystem' type='usb'>
    <source startupPolicy='optional'>
      <vendor id='0x1234'/>
      <product id='0xbeef'/>
    </source>
    <boot order='2'/>
  </hostdev>
</devices>
...
```

または、以下を実行することもできます。

図20.29 デバイス - ホスト物理マシンのデバイス割り当ての代替案

```
...
<devices>
  <hostdev mode='subsystem' type='pci' managed='yes'>
    <source>
      <address bus='0x06' slot='0x02' function='0x0'/>
    </source>
    <boot order='1'/>
    <rom bar='on' file='/etc/fake/boot.bin'/>
  </hostdev>
</devices>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.13 ホスト物理マシンのデバイス割り当て要素

パラメーター	説明
hostdev	これは、ホスト物理マシンデバイスを説明する主要なコンテナーです。USB デバイスの場合、パススルー **mode** は常に **subsystem** であり、**type** は USB デバイスの場合は **usb**、PCI デバイスの場合は **pci**です。PCI デバイスの **managed** が **yes** の場合、ゲスト仮想マシンに渡される前にホストの物理マシンからデタッチされ、ゲスト仮想マシンの終了後にホストの物理マシンに再割り当てられます。PCI および USB デバイスの **managed** が省略されるか、または **no** である場合、ゲスト仮想マシンを起動するか、デバイスをホットプラグする前に、**virNodeDeviceDettach** (または **virsh nodedev-dettach**) の引数を使用する必要があります。また、ゲスト仮想マシンのホットプラグまたは停止後に **virNodeDeviceReAttach** (または **virsh nodedev-reattach**) を使用します。

パラメーター	説明
source	ホストの物理マシンから見たデバイスを説明します。USB デバイスは、**vendor** 要素および **product** 要素を使用してベンダー/プロダクト id でアドレスを指定するか、**address** 要素を使用して、ホスト物理マシンのデバイスのアドレスを指定します。一方、PCI デバイスは、アドレスによってのみ記述できます。USB デバイスのソース要素には、**startupPolicy** 属性が含まれる場合があります。これを使用すると、指定したホスト物理マシンの USB デバイスが見つからない場合の対処方法に関するルールを定義できます。この属性は、次の値を受け入れます。 ● **mandatory** - 何らかの理由でない場合は失敗します (デフォルト)。 ● **requisite** - システムの起動時にない場合は失敗し、migrate/restore/revert にない場合はドロップします。 ● **optional** - 起動の試行時にない場合はドロップします。
vendor, product	このような要素には、それぞれ USB ベンダーと製品 ID を指定する **id** 属性があります。ID は、10 進数、16 進数 (0x で始まる)、または 8 進数 (0 で始まる) で指定できます。
boot	デバイスが起動可能であることを指定します。この属性の順序により、システムの起動シーケンスでデバイスが試行される順序が決定します。デバイスごとのブート要素は、BIOS ブートローダーセクションの一般的なブート要素とは併用できません。
rom	PCI デバイスの ROM がゲスト仮想マシンに表示される方法を変更する場合に使用されます。オプションの **bar** 属性は **on** または **off** に設定でき、デバイスの ROM がゲスト仮想マシンのメモリーマップで表示されるかどうかを決定します。(PCI のドキュメントでは、**rombar** 設定により、ROM のベースアドレスレジスターの存在が制御されます。)rom bar を指定しないと、デフォルト設定が使用されます。オプションの **file** 属性は、デバイスの ROM BIOS としてゲスト仮想マシンに提示されるバイナリーファイルを指定するために使用されます。これは、たとえば、sr-iov 対応イーサネットデバイスの仮想機能に PXE ブート ROM を提供する場合に役立ちます (VF にはブート ROM がありません)。

パラメーター	説明
address	また、デバイスがホストの物理マシンに表示される USB バスとデバイス番号を指定する **bus** および **bus** 属性もあります。この属性の値は、10 進数、16 進数 (0x で始まる)、または 8 進数 (0 で始まる) で指定できます。PCI デバイスの場合、この要素は、**lspci** または **virsh nodedev-list** で検出されるデバイスを指定できるように、3 つの属性を持ちます。

20.16.6.2. ブロック / キャラクターデバイス

ホストの物理マシンのブロック / キャラクターデバイスは、マネジメントツールを使用してドメイン xml **hostdev**要素を変更することで、ゲスト仮想マシンに渡すことができます。これは、コンテナーベースの仮想化でのみ可能であることに注意してください。

図20.30 デバイス - ホスト物理マシンのデバイス割り当てブロックキャラクターデバイス

```
...
<hostdev mode='capabilities' type='storage'>
  <source>
    <block>/dev/sdf1</block>
  </source>
</hostdev>
...
```

別のアプローチは以下のとおりです。

図20.31 デバイス - ホスト物理マシンのデバイス割り当てブロックキャラクターデバイスの代替案 1

```
...
<hostdev mode='capabilities' type='misc'>
  <source>
    <char>/dev/input/event3</char>
  </source>
</hostdev>
...
```

また別のアプローチは以下のとおりです。

図20.32 デバイス - ホスト物理マシンのデバイス割り当てブロックキャラクターデバイスの代替案 2

```
...
<hostdev mode='capabilities' type='net'>
  <source>
    <interface>eth0</interface>
  </source>
</hostdev>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.14 ブロック/ キャラクターデバイス要素

パラメーター	説明
hostdev	これは、ホスト物理マシンデバイスを説明する主要なコンテナーです。ブロックデバイス/キャラクターデバイスの場合、パススルー **mode** は常に **capabilities** になります。ブロックデバイスの場合は **type** は **block** に、キャラクターデバイスの場合は **char** になります。
source	これは、ホストの物理マシンから見たデバイスを説明します。ブロックデバイスの場合は、ホスト物理マシンの OS のブロックデバイスへのパスが、ネストされた **block** 要素で提供され、キャラクターデバイスの場合は **char** 要素が使用されます。

20.16.7. リダイレクトされたデバイス

キャラクターデバイスを介した USB デバイスのリダイレクトは、ドメイン xml の次のセクションを変更する管理ツールを使用して設定することでサポートされます。

図20.33 デバイス - リダイレクトされたデバイス

```
...
<devices>
  <redirdev bus='usb' type='tcp'>
    <source mode='connect' host='localhost' service='4000'/>
    <boot order='1'/>
  </redirdev>
  <redirfilter>
    <usbdev class='0x08' vendor='0x1234' product='0xbeef' version='2.00' allow='yes'/>
    <usbdev allow='no'/>
  </redirfilter>
</devices>
...
```

ドメイン XML のこのセクションのコンポーネントは以下のとおりです。

表20.15 リダイレクトされたデバイス要素

パラメーター	説明
redirdev	これは、リダイレクトされたデバイスを記述するためのメインコンテナーです。USB デバイスの場合、**bus** は **usb** である必要があります。追加の属性タイプが必要で、対応しているシリアルデバイスタイプのいずれかと一致するものが、トンネルのホスト物理マシン側の説明に使用されます。通常は、**type='tcp'** または **type='spicevmc'** (SPICE グラフィックデバイスの usbredir チャンネルを使用) です。redirdev 要素には、オプションのサブ要素 **address** があります。これは、デバイスを特定のコントローラーに関連付けることができます。**target** サブ要素は必要ありませんが、**source** などのサブ要素は、指定の **type** に応じて必要になる場合があります (キャラクターデバイスのコンシューマーはゲスト仮想マシンに表示されるデバイスではなく、ハイパーバイザー自体であるため)。
boot	デバイスが起動可能であることを指定します。order 属性は、システムの起動順序においてデバイスが試行される順序を決定します。デバイスごとのブート要素は、BIOS ブートローダーセクションの一般的なブート要素とは併用できません。
redirfilter	これは、特定のデバイスをリダイレクトから除外するフィルタールールを作成するために使用されます。サブ要素 **usbdev** を使用して、各フィルタールールを定義します。**class** 属性は USB クラスコードです。

20.16.8. スマートカードデバイス

仮想スマートカードデバイスは、**smartcard** 要素を介してゲスト仮想マシンに提供することができます。ホストマシン上の USB スマートカードリーダーデバイスは、ホストとゲストの両方が使用できるようにすることはできず、ゲストから削除されたときにホストコンピューターをロックできるため、単純なデバイスパススルーを使用するゲストでは使用できません。したがって、一部のハイパーバイザーは、ゲスト仮想マシンにスマートカードインターフェイスを提示できる特別な仮想デバイスを提供します。これには、ホストマシンから、またはサードパーティーのスマートカードプロバイダーによって作成されたチャンネルから認証情報を取得する方法を説明するいくつかのモードがあります。キャラクターデバイスを介した USB デバイスリダイレクトのパラメーターを設定するには、ドメイン XML の次のセクションを編集します。

図20.34 デバイス - スマートカードデバイス

```
...
<devices>
  <smartcard mode='host'/>
  <smartcard mode='host-certificates'>
    <certificate>cert1</certificate>
    <certificate>cert2</certificate>
    <certificate>cert3</certificate>
    <database>/etc/pki/nssdb/</database>
  </smartcard>
  <smartcard mode='passthrough' type='tcp'>
    <source mode='bind' host='127.0.0.1' service='2001'/>
    <protocol type='raw'/>
    <address type='ccid' controller='0' slot='0'/>
  </smartcard>
  <smartcard mode='passthrough' type='spicevmc'/>
</devices>
...
```

smartcard 要素には必須の属性**mode** があります。次のモードがサポートされています。各モードで、ゲスト仮想マシンは、物理 USB CCID(チップ/スマートカードインターフェイスデバイス) カードのように動作する USB バス上のデバイスを認識します。

モードの属性は、以下のとおりです。

表20.16 スマートカードモードの要素

パラメーター	説明
mode='host'	このモードでは、ハイパーバイザーはゲスト仮想マシンからのすべての直接アクセス要求を、NSS を介してホスト物理マシンのスマートカードに中継します。その他の属性やサブ要素は必要ありません。オプションの **address** サブ要素の使用方法は、以下を参照してください。
mode='host-certificates'	このモードでは、スマートカードをホスト物理マシンに接続させることなく、ホスト物理マシンのデータベースに存在する 3 つの NSS 証明書名を指定できます。これらの証明書は、**certutil -d /etc/pki/nssdb -x -t CT,CT,CT -S -s CN=cert1 -n cert1,** コマンドを使用して生成できます。生成される 3 つの証明書名は、それぞれ 3 つの **certificate** サブ要素の内容として提供する必要があります。追加のサブ要素の **database** では、代替ディレクトリーの絶対パスを指定できます (証明書の作成時の **certutil** コマンドの **-d** オプションに一致)。指定しないと **/etc/pki/nssdb** にデフォルトが設定されます。

パラメーター	説明
mode='passthrough'	このモードでは、ハイパーバイザーがホスト物理マシンと直接通信するのではなく、セカンダリーキャラクターデバイスを介してすべての要求をサードパーティープロバイダーにトンネリングできます。サードパーティープロバイダーは、スマートカードと通信するか、3 つの証明書ファイルを使用します。このモードでは、トンネルのホスト物理マシン側を説明するために、対応しているシリアルデバイスタイプの 1 つに一致する追加の属性**type**が必要になります。**type='tcp'** または **type='spicevmc'** (SPICE グラフィックデバイスのスマートカードチャネルを使用) が一般的です。**target** サブ要素は必要ありませんが、**source** などのサブ要素は、指定のタイプに応じて必要になる場合があります (キャラクターデバイスのコンシューマーはゲスト仮想マシンに表示されるデバイスではなく、ハイパーバイザー自体であるため)。

各モードは、オプションのサブ要素**address**をサポートしています。これは、スマートカードと ccid バスコントローラーとの間の関連を微調整します (「デバイスアドレス」 を参照してください)。

20.16.9. Network Interfaces

ネットワークインターフェイスデバイスは、ドメイン XML の次の部分を設定する管理ツールを使用して変更されます。

図20.35 デバイス - ネットワークインターフェイス

```
...
<devices>
  <interface type='bridge'>
    <source bridge='xenbr0'/>
    <mac address='00:16:3e:5d:c7:9e'/>
    <script path='vif-bridge'/>
    <boot order='1'/>
    <rom bar='off'/>
  </interface>
</devices>
...
```

ゲスト仮想マシンに表示されるネットワークインターフェイスを指定するには、いくつかの可能性があります。以下の各サブセクションでは、一般的なセットアップオプションについて詳しく説明します。さらに、各 **<interface>** 要素には、属性 **type='pci'** を使用してインターフェイスを特定の pci スロットに関連付けることができるオプションの **<address>** サブ要素があります (「デバイスアドレス」 を参照してください)。

20.16.9.1. 仮想ネットワーク

これは、動的/ワイヤレスネットワーク設定のホスト物理マシンにおける一般的なゲスト仮想マシンの接続に推奨される設定です (または、ホスト物理マシンのハードウェアの詳細が**<ネットワーク>** 定義で

個別に説明されている複数ホストの物理マシン環境)。さらに、名前付きネットワーク定義によって詳細が記述されている接続を提供します。仮想ネットワークの **forward mode** 設定によっては、ネットワークが完全に分離している (**<forward>** 要素が指定されていない) 場合と、NAT を使用して明示的なネットワークデバイスまたはデフォルトルートに接続している (**forward mode='nat'**) 場合、NAT を使用せずにルーティングしている (**forward mode='route'**/) 場合、またはホスト物理マシンのネットワークインターフェイス (macvtap を使用) またはブリッジデバイス (**forward mode='bridge|private|vepa|passthrough'**/) のいずれかに直接接続している場合があります。

bridge、private、vepa、および passthrough の転送モードを持つネットワークでは、ホストの物理マシンに、必要な DNS サービスおよび DHCP サービスが、libvirt の範囲外に設定されていることを前提とします。分離ネットワーク、nat ネットワーク、ルーティングネットワークの場合、DHCP および DNS は、libvirt により仮想ネットワークで提供され、IP 範囲は、**virsh net-dumpxml [networkname]** で仮想ネットワーク設定を調べることで決定できます。デフォルトルートへの NAT を実行し、IP 範囲が 192.168.122.0/255.255.255.0 である、デフォルトセットアップと呼ばれる仮想ネットワークが 1 つあります。各ゲスト仮想マシンには、vnetN という名前で作成された tun デバイスが関連付けられます。これは、**<target>** 要素 (「ターゲット要素の上書き」 を参照) で上書きできます。

インターフェイスのソースがネットワークの場合、ポートグループをネットワークの名前と共に指定できます。1 つのネットワークに複数のポートグループが定義されており、各ポートグループには、ネットワーク接続のさまざまなクラスに対する若干異なる設定情報が含まれています。また、**<direct>** ネットワーク接続 (以下で説明) と同様に、タイプ **network** の接続は **<virtualport>** 要素を指定し、vepa (802.1Qbg) または 802.1Qbh 準拠のスイッチに転送される設定データと共に、Open vSwitch 仮想スイッチに転送できます。

実際のスイッチの種類は、ホストの物理マシン上の **<network>** の設定によって異なる場合があるため、仮想ポートの種類の属性を省略し、複数の異なる仮想ポートの種類の属性を指定することもできます (特定の属性を除外することもできます)。ドメインの起動時に、ネットワークで定義されたタイプと属性、およびインターフェイスによって参照されるポートグループをマージすることにより、完全な **<virtualport>** 要素が構築されます。新しく構築された仮想ポートは、両方の組み合わせです。下位の仮想ポートの属性は、上位の仮想ポートで定義された属性に変更を加えることはできません。インターフェイスが最高の優先度を取り、ポートグループが最低の優先度を取ります。

たとえば、802.1Qbh スイッチおよび Open vSwitch スイッチの両方で適切に機能するネットワークを作成する場合は、タイプを指定せず、**profileid** と **interfaceid** の両方を提供する必要があります。**managerid**、**typeid**、**profileid** などの仮想ポートから入力するその他の属性は任意です。

ゲスト仮想マシンの接続を特定タイプのスイッチのみに制限する場合は、virtualport タイプを指定でき、指定したポートタイプのスイッチのみが接続されます。追加のパラメーターを指定することで、スイッチの接続をさらに制限することもできます。その結果、ポートが指定されており、ホストの物理マシンのネットワークに別のタイプの virtualport があると、インターフェイスの接続に失敗します。仮想ネットワークパラメーターは、ドメイン XML の以下の部分を変更する管理ツールを使用して定義します。

図20.36 デバイス - ネットワークインターフェイス - 仮想ネットワーク

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
  </interface>
  ...
  <interface type='network'>
    <source network='default' portgroup='engineering'/>
    <target dev='vnet7'/>
    <mac address="00:11:22:33:44:55"/>
    <virtualport>
      <parameters instanceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f'/>
    </virtualport>

  </interface>
</devices>
...
```

20.16.9.2. LAN へのブリッジ

注記: 静的有線ネットワーク設定を使用するホスト物理マシンの一般的なゲスト仮想マシンの接続に推奨される設定設定です。

LAN へのブリッジは、ゲスト仮想マシンから LAN に直接ブリッジを提供します。ホスト物理マシン上にブリッジデバイスがあり、1 つ以上のホスト物理マシンの物理 NIC がスレーブであることを前提としています。ゲスト仮想マシンには、**<vnetN>** の名前で作成された関連する **tun** デバイスがあります。これは **<target>** 要素 (「ターゲット要素の上書き」 を参照) で上書きすることもできます。**<tun>** デバイスは、ブリッジにスレーブされます。IP 範囲/ネットワーク設定は、LAN で使用されるものです。これにより、物理マシンと同様に、ゲスト仮想マシンの完全な着信ネットアクセスと発信ネットアクセスが提供されます。

Linux システムでは、ブリッジデバイスは通常、標準的な Linux ホストの物理マシンブリッジです。Open vSwitch に対応するホストの物理マシンでは、インターフェイス定義に **virtualport type='openvswitch'/** を追加することで、Open vSwitch ブリッジデバイスに接続することもできます。Open vSwitch タイプ virtualport は、**parameters** 要素に 2 つのパラメーターを指定できます。1 つは、この特定のインターフェイスを Open vSwitch に固有に識別するために使用される標準の uuid である**interfaceid** (指定しないと、インターフェイスの最初の定義時にランダムな**interfaceid**が生成されます) で、もう 1 つは Open vSwitch にインターフェイス**<port-profile>** として送信されるオプションの**profileid** です。ブリッジを LAN 設定に設定するには、管理ツールを使用して、ドメイン XML の以下の部分を設定します。

図20.37 デバイス - ネットワークインターフェイス - LAN へのブリッジ

```
...
<devices>
  ...
  <interface type='bridge'>
    <source bridge='br0'/>
  </interface>
  <interface type='bridge'>
    <source bridge='br1'/>
    <target dev='vnet7'/>
    <mac address="00:11:22:33:44:55"/>
  </interface>
  <interface type='bridge'>
    <source bridge='ovsbr'/>
    <virtualport type='openvswitch'>
      <parameters profileid='menial' interfaceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f'/>
    </virtualport>
  </interface>
  ...
</devices>
```

20.16.9.3. ポートマスカレードの範囲の設定

ポートマスカレード範囲を設定する場合、ポートは次のように設定できます。

図20.38 ポートマスカレードの範囲

```
<forward mode='nat'>
   <address start='192.0.2.1' end='192.0.2.10'/>
</forward>  ...
```

この値は、以下のように **iptable** コマンドを使用して設定する必要があります。 「ネットワークアドレス変換モード」

20.16.9.4. ユーザー空間の SLIRP スタック

ユーザー空間の SLIRP スタックパラメーターを設定すると、外部への NAT を備えた仮想 LAN が提供されます。仮想ネットワークには DHCP サービスおよび DNS サービスがあり、ゲスト仮想マシンに 10.0.2.15 以降の IP アドレスを割り当てます。デフォルトルーターは 10.0.2.2 になり、DNS サーバーは 10.0.2.3 になります。このネットワークは、ゲスト仮想マシンに外部アクセスを行う必要がある特権のないユーザーにとって唯一のオプションです。

ユーザー空間の SLIP スタックパラメーターは、ドメイン XML の以下の部分で定義されています。

図20.39 デバイス - ネットワークインターフェイス - ユーザー空間の SLIRP スタック

```
...
<devices>
  <interface type='user'/>
  ...
  <interface type='user'>
    <mac address="00:11:22:33:44:55"/>
  </interface>
</devices>
...
```

20.16.9.5. 一般的なイーサネット接続

管理者が任意のスクリプトを実行してゲスト仮想マシンのネットワークを LAN に接続できるようになります。ゲスト仮想マシンには、**vnetN** の名前で作成された**tun** デバイスがありますが、**target** 要素で上書きすることもできます。**tun** デバイスを作成した後、必要なホスト物理マシンネットワーク統合を実行することが期待されるシェルスクリプトが実行されます。初期設定では、このスクリプトは **/etc/qemu-ifup** と呼ばれていますが、上書きできます (「ターゲット要素の上書き」 を参照)。

一般的なイーサネット接続パラメーターは、ドメイン XML の以下の部分で定義されています。

図20.40 デバイス - ネットワークインターフェイス - 汎用イーサネット接続

```
...
<devices>
  <interface type='ethernet'/>
  ...
  <interface type='ethernet'>
    <target dev='vnet7'/>
    <script path='/etc/qemu-ifup-mynet'/>
  </interface>
</devices>
...
```

20.16.9.6. 物理インターフェイスへの直接接続

<interface type='direct'> を使用すると、仮想マシンの NIC がホスト上の指定された物理インターフェイスに接続されます。

この設定では、Linux macvtap ドライバーが使用可能である必要があります。macvtap デバイスの動作モードとして次のモードのいずれかを選択できます **vepa** (仮想イーサネットポートアグリゲータ)。これはデフォルトのモード、**bridge**、または **private** です。

物理インターフェイスへの直接接続を設定するには、ドメイン XML で次のパラメーターを使用します。

図20.41 デバイス - ネットワークインターフェイス - 物理インターフェイスへの直接接続

```
...
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0' mode='vepa'/>
  </interface>
</devices>
...
```

各モードにより、パケットの配信は表20.17「物理インターフェイス要素への直接接続」 のように動作します。

表20.17 物理インターフェイス要素への直接接続

要素	説明
vepa	ゲスト仮想マシンのパケットはすべて外部ブリッジに送信されます。パケットの送信先が、パケットの送信元と同じホスト物理マシン上のゲスト仮想マシンであるパケットは、VEPA 対応のブリッジによりホスト物理マシンに返されます (現在のブリッジは、通常 VEPA 対応ではありません)。
bridge	宛先が、送信元と同じホストの物理マシンにあるパケットは、ターゲットの macvtap デバイスに直接配信されます。直接配信する場合は、作成元デバイスと宛先デバイスの両方がブリッジモードになっている必要があります。いずれかが **vepa** モードにある場合は、VEPA 対応のブリッジが必要です。
プライベート	すべてのパケットは外部ブリッジに送信されます。また、外部ルーターまたはゲートウェイを介して送信され、そのデバイスがホストの物理マシンに返す場合は、同じホストの物理マシンのターゲット VM にのみ配信されます。移行元デバイスまたは移行先デバイスのいずれかがプライベートモードの場合は、以下の手順が行われます。
パススルー	この機能は、移行機能を失うことなく、SRIOV 対応 NIC の仮想機能をゲスト仮想マシンに直接接続します。すべてのパケットは、設定したネットワークデバイスの VF/IF に送信されます。デバイスの機能によっては、追加の前提条件や制限が適用される場合があります。たとえば、これにはカーネル 2.6.38 以降が必要です。

直接接続された仮想マシンのネットワークアクセスは、ホストの物理マシンの物理インターフェイスが接続されているハードウェアスイッチにより管理できます。

スイッチが IEEE 802.1Qbg 規格に準拠している場合、インターフェイスには以下のような追加パラメー

ターを設定できます。virtualport 要素のパラメーターについては、IEEE 802.1Qbg 標準で詳細が説明されています。この値はネットワーク固有のもので、ネットワーク管理者が指定する必要があります。802.1Qbg の用語では、Virtual Station Interface (VSI) は仮想マシンの仮想インターフェイスを表します。

また、IEEE 802.1Qbg では VLAN ID にゼロ以外の値が必要です。

操作可能な追加の要素は、表20.18「物理インターフェイスの追加要素への直接接続」 で説明されています。

表20.18 物理インターフェイスの追加要素への直接接続

要素	説明
managerid	VSI Manager ID は、VSI タイプおよびインスタンス定義を含むデータベースを識別します。これは整数値で、値 **0** が予約されます。
typeid	VSI タイプ ID は、ネットワークアクセスを特徴付ける VSI タイプを識別します。VSI の種類は通常、ネットワーク管理者が管理します。これは整数値です。
typeidversion	VSI タイプバージョンでは、複数のバージョンの VSI タイプが許可されます。これは整数値です。
instanceid	VSI インスタンス ID 識別子は、VSI インスタンス (仮想マシンの仮想インターフェイス) が作成されると生成されます。これは、グローバルに一意の識別子です。
profileid	プロファイル ID には、このインターフェイスに適用されるポートプロファイルの名前が含まれます。この名前は、ポートプロファイルデータベースにより、ポートプロファイルからネットワークパラメーターに解決され、これらのネットワークパラメーターはこのインターフェイスに適用されます。

ドメイン XML の追加パラメーターには、以下が含まれます。

図20.42 デバイス - ネットワークインターフェイス - 物理インターフェイスへの直接接続追加パラメーター

```
...
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0.2' mode='vepa'/>
    <virtualport type="802.1Qbg">
      <parameters managerid="11" typeid="1193047" typeidversion="2" instanceid="09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f"/>
    </virtualport>
  </interface>
</devices>
...
```

スイッチが IEEE 802.1Qbh 規格に準拠している場合、インターフェイスには以下のような追加パラメーターを設定できます。この値はネットワーク固有のもので、ネットワーク管理者が指定する必要があります。

ドメイン XML の追加パラメーターには、以下が含まれます。

図20.43 デバイス - ネットワークインターフェイス - 物理インターフェイスへの直接接続、さらなる追加パラメーター

```
...
<devices>
  ...
  <interface type='direct'>
    <source dev='eth0' mode='private'/>
    <virtualport type='802.1Qbh'>
      <parameters profileid='finance'/>
    </virtualport>
  </interface>
</devices>
...
```

profileid 属性には、このインターフェイスに適用されるポートプロファイルの名前が含まれます。この名前は、ポートプロファイルデータベースにより、ポートプロファイルからネットワークパラメーターに解決され、これらのネットワークパラメーターはこのインターフェイスに適用されます。

20.16.9.7. PCI パススルー

source 要素で指定される PCI ネットワークデバイスは、ジェネリックデバイスパススルーを使用してゲスト仮想マシンに直接割り当てられます。このデバイスの MAC アドレスは、最初にオプションで設定された値に設定され、そのデバイスの MAC アドレスがオプションで指定された **virtualport** 要素を使用して 802.1Qbh 対応スイッチに関連付けられます (type='direct' ネットワークデバイスの場合は、上記の仮想ポートの例を参照してください)。標準的なシングルポートの PCI イーサネットカードドライバー設計の制限により、この方法で割り当てることができるのは Single Root I/O Virtualization (SR-IOV) virtual function (VF) デバイスのみとなることに注意してください。標準的なシングルポートの PCI または PCIe イーサネットカードをゲスト仮想マシンに割り当てる場合は、従来の **hostdev** デバイス定義を使用します。

ネットワークデバイスにおける "intelligent passthrough" は、標準の **hostdev** デバイスの機能と非常に似ています。相違点は、パススルーデバイスに MAC アドレスと**virtualport**を指定できることです。この機能が不要な場合、SR-IOV に対応していない標準のシングルポート PCI カード、PCIe カード、または USB ネットワークカードがある場合 (そのため、ゲスト仮想マシンドメインに割り当てられた後にリセット中に設定済みの MAC アドレスを失います)、または 0.9.11 よりも古い libvirt のバージョンを使用している場合は、標準の **hostdev** を使用して、**interface type='hostdev'**/ の代わりに、ゲスト仮想マシンにデバイスを割り当てます。

図20.44 デバイス - ネットワークインターフェイス - PCI パススルー

```
...
<devices>
  <interface type='hostdev'>
    <driver name='vfio'/>
    <source>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
    </source>
    <mac address='52:54:00:6d:90:02'>
    <virtualport type='802.1Qbh'>
      <parameters profileid='finance'/>
    </virtualport>
  </interface>
</devices>
...
```

20.16.9.8. マルチキャストトンネル

マルチキャストグループは、仮想ネットワークを表すために使用できます。ネットワークデバイスが同じマルチキャストグループ内にあるゲスト仮想マシンは、複数の物理ホスト物理マシンにまたがって存在する場合でも、相互に通信します。このモードは、非特権ユーザーとして使用できます。デフォルトの DNS または DHCP に対応せず、発信ネットワークアクセスもありません。外部へのネットワークアクセスを提供するには、ゲスト仮想マシンの 1 つに 2 番目の NIC が必要です。この NIC は、最初の 4 つのネットワークタイプのいずれかに接続し、適切なルーティングを提供します。マルチキャストプロトコルは、**user mode** linux ゲスト仮想マシンが使用するプロトコルと互換性があります。使用するソースアドレスは、マルチキャストアドレスブロックからのものである必要があることに注意してください。マルチキャストトンネルは、マネジメントツールを使用して **interface type** を操作し、これを **mcast** に設定/作成されます。そして、mac アドレスおよびソースアドレスを提供します。結果は、ドメイン XML に加えられた変更に表示されます。

図20.45 デバイス - ネットワークインターフェイス - マルチキャストトンネル

```
...
<devices>
  <interface type='mcast'>
    <mac address='52:54:00:6d:90:01'>
    <source address='230.0.0.1' port='5558'/>
  </interface>
</devices>
...
```

20.16.9.9. TCP トンネル

TCP クライアント/サーバーアーキテクチャーを作成することは、あるゲスト仮想マシンがネットワー

クのサーバー側を提供し、他のすべてのゲスト仮想マシンがクライアントとして設定される仮想ネットワークを提供するもう 1 つの方法になります。ゲスト仮想マシン間のすべてのネットワークトラフィックは、サーバーとして設定されているゲスト仮想マシンを介してルーティングされます。このモデルは、権限のないユーザーも使用できます。デフォルトの DNS または DHCP に対応せず、発信ネットワークアクセスもありません。外部へのネットワークアクセスを提供するには、ゲスト仮想マシンの 1 つに 2 番目の NIC が必要です。この NIC は、最初の 4 つのネットワークタイプのいずれかに接続し、適切なルーティングを提供します。TCP トンネルは、管理ツールを使用して **interface type** を操作し、**server** または **client** に設定/変更し、Mac と送信元アドレスを提供することによって作成されます。結果は、ドメイン XML に加えられた変更に表示されます。

図20.46 デバイス - ネットワークインターフェイス - TCP トンネル

```
...
<devices>
  <interface type='server'>
    <mac address='52:54:00:22:c9:42'>
    <source address='192.168.0.1' port='5558'/>
  </interface>
  ...
  <interface type='client'>
    <mac address='52:54:00:8b:c9:51'>
    <source address='192.168.0.1' port='5558'/>
  </interface>
</devices>
...
```

20.16.9.10. NIC ドライバー固有のオプションの設定

NIC によっては、調整可能なドライバー固有のオプションが含まれる場合があります。このオプションは、インターフェイス定義の **driver** サブ要素の属性として設定されます。このようなオプションは、管理ツールを使用して設定し、ドメイン XML の以下のセクションを設定します。

図20.47 デバイス - ネットワークインターフェイス - NIC ドライバー固有のオプションの設定

```
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet1'/>
    <model type='virtio'/>
    <driver name='vhost' txmode='iothread' ioeventfd='on' event_idx='off'/>
  </interface>
</devices>
...
```

現在、virtio の NIC ドライバーでは、以下の属性を使用できます。

表20.19 virtio NIC ドライバー要素

パラメーター	説明

パラメーター	説明
name	オプションの **name** 属性では、使用するバックエンドドライバーの種類が強制されます。値は、**qemu** (ユーザー空間のバックエンド) または **vhost** (カーネルにより vhost モジュールが提供されることが必要なカーネルバックエンド) のいずれかになります。カーネルサポートのない vhost ドライバーを要求しようとする場合は拒否されます。デフォルト設定は、vhost ドライバーが存在する場合は **vhost** になりますが、存在しない場合は警告なしに **qemu** に戻ります。
txmode	送信バッファーが満杯になった場合にパケットの送信を処理する方法を指定します。値は、**iothread** または **timer** のいずれかになります。**iothread** に設定すると、パケット tx はドライバーの下半分の iothread ですべて実行されます (このオプションは、**qemu** コマンドラインの "-device" virtio-net-pci オプションに **"tx=bh"** を追加することに変換されます)。**timer** に設定すると、qemu で tx 処理が行われます。現在送信可能な tx データよりも多くの tx データが存在する場合は、qemu が他の処理を行うために移動する前にタイマーが設定されます。タイマーが実行されると、さらなるデータを送信するために別の試みが行われます。一般に、このオプションを変更することが絶対に必要であると確信している場合を除いて、このオプションはそのままにしておく必要があります。
ioeventfd	ユーザーがインターフェイスデバイスのドメイン I/O 非同期処理を設定できるようにします。デフォルトは、ハイパーバイザーの判断に任されます。許可される値は **on** と **off** です。このオプションを有効にすると、qemu は、別のスレッドが I/O を処理している間にゲスト仮想マシンを実行できます。通常、I/O の実行中にシステム CPU の使用率が高くなったゲスト仮想マシンは、この恩恵を受けます。一方、物理ホストマシンのオーバーロードは、ゲスト仮想マシンの I/O レイテンシーを増加させる可能性もあります。したがって、このオプションを変更することが絶対に必要であると確信している場合を除いて、このオプションはそのままにしておく必要があります。

パラメーター	説明
event_idx	event_idx 属性は、デバイスイベント処理の一部の側面を制御します。値は **on** または **off** のいずれかになります。**on** を選択すると、ゲスト仮想マシンの割り込みと終了の数が減ります。デフォルトは **on** です。この動作が最適ではない状況がある場合、この属性は機能を強制的にオフにする方法を提供します。変更が絶対に必要であると確信している場合を除いて、このオプションはそのままにしておく必要があります。

20.16.9.11. ターゲット要素の上書き

ターゲット要素を上書きする場合は、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.48 デバイス - ネットワークインターフェイス - ターゲット要素の上書き

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet1'/>
  </interface>
</devices>
...
```

ターゲットを指定しないと、特定のハイパーバイザーにより、作成された tun デバイスの名前が自動的に生成されます。この名前は手動で指定できますが、vnet または vif で始まる名前は使用できません。これらの接頭辞は、libvirt および特定のハイパーバイザーが予約するものです。この接頭辞を使用して手動で指定したターゲットは無視されます。

20.16.9.12. 起動順序の指定

起動順序を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.49 起動順序の指定

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet1'/>
    <boot order='1'/>
  </interface>
</devices>
...
```

これに対応するハイパーバイザーでは、ネットワーク起動に使用する特定の NIC を設定できます。属性の順序により、システムの起動シーケンスでデバイスが試行される順序が決定します。デバイスごとの

ブート要素は、BIOS ブートローダーセクションの一般的なブート要素とは併用できないことに注意してください。

20.16.9.13. インターフェイス ROM BIOS 設定

ROM BIOS 設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を加えます。

図20.50 インターフェイス ROM BIOS 設定

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet1'/>
    <rom bar='on' file='/etc/fake/boot.bin'/>
  </interface>
</devices>
...
```

これをサポートするハイパーバイザーの場合は、PCI Network デバイスの ROM がゲスト仮想マシンにどのように提供されるかを変更できます。**bar** 属性は **on** または **off** に設定でき、デバイスの ROM がゲスト仮想マシンのメモリーマップで表示されるかどうかを決定します。(PCI のドキュメントでは、rombar 設定により、ROM のベースアドレスレジスターの存在が制御されます。)ROM バーが指定されていない場合、qemu のデフォルトが使用されます (古いバージョンの qemu はデフォルトの **off**を使用していましたが、新しい qemus のデフォルトは **on**です)。オプションの **file** 属性は、デバイスの ROM BIOS としてゲスト仮想マシンに提示されるバイナリーファイルを指定するために使用されます。これは、ネットワークデバイスに別のブート ROM を提供する際に役立ちます。

20.16.9.14. QoS (Quality of Service)

ドメイン XML のこのセクションは、サービス品質の設定を提供します。送受信トラフィックは独立してシェイプできます。**bandwidth** 要素には、最大で 1 つのインバウンドおよび最大で 1 つのアウトバウンド子要素を設定できます。このような子要素のいずれかを外した場合は、トラフィックの方向に QoS が適用されません。したがって、ドメインの受信トラフィックのみを形成する場合は、受信のみを使用し、その逆も同様です。

これらの各要素には、必須の属性**average** (または以下のような **floor**) が 1 つずつあります。**average** は、シェイプされるインターフェイスの平均ビットレートを指定します。次に、2 つのオプションの属性があります。インターフェイスがデータを送信できる最大速度を指定する **peak**と、ピーク速度で burst できるバイト数を指定する**burst** です。属性に使用できる値は整数です。

average 属性および **peak** 属性の単位は キロバイト/秒 で、**burst** はキロバイト単位でのみ設定されます。また、着信トラフィックには**floor** 属性を指定できます。これにより、シェーピングのインターフェイスでは最小限のスループットが保証されます。**floor** を使用する場合は、すべてのトラフィックが QoS の決定が行われる 1 つのポイントを通過する必要があります。このため、**forward**タイプが **route**、**nat**、または no forward at all である **interface type='network'**/ の場合のみに使用できます。仮想ネットワーク内では、接続されているすべてのインターフェイスに、少なくとも着信 QoS セット (**average** 以上) が必要ですが、floor 属性では **average** を指定する必要がありません。ただし、**peak** 属性および **burst** 属性には依然として**average** が必要です。現時点では、Ingress qdiscs にクラスがないため、**floor** は受信トラフィックにのみ適用でき、送信トラフィックには適用されません。

QoS 設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.51 QoS (Quality of Service)

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet0'/>
    <bandwidth>
      <inbound average='1000' peak='5000' floor='200' burst='1024'/>
      <outbound average='128' peak='256' burst='256'/>
    </bandwidth>
  </interface>
<devices>
...
```

20.16.9.15. VLAN タグの設定 (サポートされているネットワークタイプのみ)

VLAN タグ設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.52 VLAN タグの設定 (サポートされているネットワークタイプのみ)

```
...
<devices>
  <interface type='bridge'>
    <vlan>
      <tag id='42'/>
    </vlan>
    <source bridge='ovsbr0'/>
    <virtualport type='openvswitch'>
      <parameters interfaceid='09b11c53-8b5c-4eeb-8f00-d84eaa0aaa4f'/>
    </virtualport>
  </interface>
<devices>
...
```

ゲスト仮想マシンが使用するネットワーク接続がゲスト仮想マシンに対して透過的な vlan タグ付けをサポートしている場合 (およびその場合のみ)、オプションの **vlan** 要素でゲスト仮想マシンのネットワークトラフィックに適用する 1 つ以上の vlan タグを指定できます (openvswitch および **type='hostdev'** SR-IOV インターフェイスは、ゲスト仮想マシントラフィックの透過的な VLAN タグ付けをサポートします。標準の Linux ブリッジや libvirt 独自の仮想ネットワークを含む他のすべては、これをサポートしません。802.1 Qbh (vn-link) スイッチおよび 802.1Qbg (VEPA) スイッチは、(libvirt の外部で) ゲスト仮想マシンのトラフィックを特定の vlan にタグ付けする独自の方法を提供します。 複数のタグを指定できるようにするには (vlan トランクの場合)、**tag** サブ要素で使用する vlan タグを指定します (**tag id='42'**/ など)。インターフェイスに複数の **vlan** 要素が定義されている場合は、指定されたすべてのタグを使用して VLAN トランクを実行することが想定されます。単一のタグを使用した vlan トランキングが必要な場合は、オプションの属性 **trunk='yes'** をトップレベルの vlan 要素に追加できます。

20.16.9.16. 仮想リンクの状態の変更

この要素は、仮想ネットワークリンクの状態を設定する手段を提供します。属性 **state** に指定できる値は、**up** および **down** です。値として **down** が指定されている場合、インターフェイスはネットワークケーブルが切断されているかのように動作します。この要素が指定されていない場合のデフォルトの動作は、リンク状態を **up** にすることです。

仮想リンク状態の設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.53 仮想リンクの状態の変更

```
...
<devices>
  <interface type='network'>
    <source network='default'/>
    <target dev='vnet0'/>
    <link state='down'/>
  </interface>
<devices>
...
```

20.16.10. 入力デバイス

入力デバイスを使用すると、ゲスト仮想マシンの仮想マシンのグラフィカルフレームバッファーとの相互作用が可能になります。フレームバッファーを有効にすると、入力デバイスが自動的に提供されます。デバイスを明示的に追加することもできます。たとえば、絶対的なカーソルの動きにグラフィックタブレットを使用することもできます。

入力デバイスの設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.54 入力デバイス

```
...
<devices>
  <input type='mouse' bus='usb'/>
</devices>
...
```

<input> 要素には 1 つの必須属性があります。**type** は次のように設定できます: **mouse** または **tablet**。後者は絶対的なカーソル移動を提供し、前者は相対的な移動を使用します。オプションの **bus** 属性を使用すると、デバイスタイプを詳細に指定できます。また、**xen** (準仮想化)、**ps2**、および **usb** に設定できます。

インプット要素にはオプションのサブ要素**<address>**があります。これは、上記で説明されているように、デバイスを特定の PCI スロットに関連付けることができます。

20.16.11. ハブデバイス

ハブは、1 つのポートを複数の場所に拡張し、デバイスをホスト物理マシンシステムに接続するために利用可能なデバイスです。

ハブデバイスの設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.55 ハブデバイス

```
...
<devices>
  <hub type='usb'/>
</devices>
...
```

ハブ要素には 1 つの必須属性があり、その値は **usb** のみになります。hub 要素には、**type='usb'** をオプションに持つオプションのサブ要素 **address** があり、デバイスを特定のコントローラーに関連付けることができます。

20.16.12. グラフィカルフレームバッファー

グラフィックデバイスを使用すると、ゲスト仮想マシン OS とのグラフィカルな対話が可能になります。ゲスト仮想マシンには通常、admin との対話を許可するようにフレームバッファーまたはテキストコンソールのいずれかが設定されます。

グラフィカルフレームバッファーデバイスの設定を指定するには、管理ツールを使用して、ドメイン XML に次の変更を行います。

図20.56 グラフィカルフレームバッファー

```
...
<devices>
  <graphics type='sdl' display=':0.0'/>
  <graphics type='vnc' port='5904'>
    <listen type='address' address='192.0.2.1'/>
  </graphics>
  <graphics type='rdp' autoport='yes' multiUser='yes' />
  <graphics type='desktop' fullscreen='yes'/>
  <graphics type='spice'>
    <listen type='network' network='rednet'/>
  </graphics>
</devices>
...
```

graphics 要素には必須の**type** 属性があり、以下の説明に従って、**sdl**、**vnc**、**rdp**、または **desktop** の値をとります。

表20.20 グラフィカルフレームバッファー要素

パラメーター	説明
sdl	これにより、ホスト物理マシンのデスクトップにウィンドウが表示されます。3 つのオプションの引数を取ることができます。使用するディスプレイの **display** 属性、認証 ID の **xauth** 属性、および **yes** または **no** の値を受け入れるオプションの **fullscreen** 属性です。

パラメーター	説明
vnc	VNC サーバーを起動します。**port** 属性は、TCP ポート番号を指定します (**-1** は、自動割り当てが必要であることを示す従来の構文)。**autoport** 属性は、使用する TCP ポートの自動割り当てを指定する際に推奨される新しい構文です。**listen** 属性は、サーバーがリッスンする IP アドレスです。**passwd** 属性は、クリアテキストで VNC パスワードを提供します。**keymap** 属性は、使用するキーマップを指定します。**timestamp passwdValidTo='2010-04-09T15:51:00'** が UTC であると想定し、パスワードの有効性に制限を設定することができます。**connected** 属性を使用すると、パスワードの変更時に接続先を制御できます。VNC は **keep** 値のみを受け入れます。そして、すべてのハイパーバイザーでサポートされていないことに注意してください。QEMU は、listen/port を使用する代わりに、unix ドメインソケットパスをリッスンするソケット属性をサポートします。
spice	SPICE サーバーを起動します。**port** 属性は TCP ポート番号を指定します (自動割り当ての必要があることを示す従来の構文として -1 を使用)。**tlsPort** は代替のセキュアポート番号を指定します。**autoport** 属性は、両方のポート番号の自動割り当てを指定する際に推奨される新しい構文です。**listen** 属性は、サーバーがリッスンする IP アドレスです。**passwd** 属性は、クリアテキストで SPICE パスワードを提供します。**keymap** 属性は、使用するキーマップを指定します。**timestamp passwdValidTo='2010-04-09T15:51:00'** が UTC であると想定し、パスワードの有効性に制限を設定することができます。**connected** 属性を使用すると、パスワードの変更時に接続先を制御できます。SPICE は、クライアントとの接続を維持するための keep、クライアントとの接続を解除する disconnect、そしてパスワードの変更に失敗する fail を受け入れます。すべてのハイパーバイザーでサポートされているわけではないことに注意してください。**defaultMode** 属性は、デフォルトのチャネルセキュリティーポリシーを設定します。有効な値は、**secure**、**insecure**、およびデフォルトの **any** になります (可能な場合は **secure** になりますが、安全なパスが利用できない場合にエラーになるのではなく、**insecure** にフォールバックします)。

SPICE に通常の TCP ポートと、TLS で保護された TCP ポートの両方が設定されている場合は、各ポートで実行できるチャネルを制限することが望ましい場合があります。これは、メインの **graphics** 要素内に 1 つ以上の **channel** 要素を追加することによって実現されます。有効なチャンネル名は、**main**、**display**、**inputs**、**cursor**、**playback**、**record**、**smartcard**、および **usbredir** などです。

SPICE 設定を指定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.57 SPICE 設定

```
<graphics type='spice' port='-1' tlsPort='-1' autoport='yes'>
  <channel name='main' mode='secure'/>
  <channel name='record' mode='insecure'/>
  <image compression='auto_glz'/>
  <streaming mode='filter'/>
  <clipboard copypaste='no'/>
  <mouse mode='client'/>
</graphics>
```

SPICE は、オーディオ、イメージ、およびストリーミングの変数圧縮設定に対応しています。これらの設定には、次のすべての要素で圧縮属性を使用してアクセスできます: **image** 圧縮を設定するイメージ (auto_glz、auto_lz、quick、glz、lz、off を受け入れる)、wan 上のイメージの JPEG 圧縮用の **jpeg** (auto、never、always を受け入れる)、wan イメージ圧縮を設定するための **zlib** (auto、never、always を受け入れる) およびオーディオストリーム圧縮を有効にするための **playback** (on または off を受け入れる)。

ストリーミングモードは **streaming** 要素によって設定され、その **mode** 属性を **filter**の 1 つ、**all** または **off** に設定します。

さらに、(SPICE エージェントを使用して) コピーアンドペースト機能は、**clipboard** 要素により設定されます。これはデフォルトで有効になっており、**copypaste** プロパティーを **no** に設定することで無効にできます。

マウスモードは、**mouse** 要素によって設定され、その **mode** 属性を **server** または **client** のいずれかに設定します。mode を指定しない場合は、qemu のデフォルトが使用されます (**client** モード)。

追加の要素は以下のとおりです。

表20.21 追加のグラフィカルフレームバッファー要素

パラメーター	説明
rdp	RDP サーバーを起動します。port 属性は、TCP ポート番号を指定します (自動割り当ての必要があることを示す従来の構文として -1 を使用)。autoport 属性は、使用する TCP ポートの自動割り当てを指定する際に推奨される構文です。replaceUser 属性は、VM への複数の同時接続を許可するかどうかを決定するブール値です。新規クライアントがシングル接続モードで接続する場合に、multiUser 属性で既存の接続をドロップする必要があるか、また、新規の接続を VRDP サーバーで確立する必要があるかを指定します。
desktop	この値は、現時点では VirtualBox ドメイン用に予約されています。sdl と同様に、ホスト物理マシンデスクトップにウィンドウを表示しますが、VirtualBox ビューアーを使用します。sdl と同様に、オプション属性の display および fullscreen を受け入れます。

パラメーター	説明
listen	グラフィックタイプ vnc および spice のリスニングソケットを設定するために使用されるアドレス情報を **graphics** に配置する代わりに、**listen** と呼ばれる **graphics** の個別のサブ要素である **listen** 属性を指定できます (上記の例を参照)。**listen** は次の属性を受け入れます。 ● **type** - address または network に設定します。これは、このリッスン要素が、直接使用するアドレスを指定するか、ネットワークに名前を付けるか (これを使用して、リッスンする適切なアドレスを判断します) を指定します。 ● **address** - この属性には、リッスンする IP アドレスまたはホスト名 (DNS クエリーで IP アドレスに解決されます) のいずれかが含まれます。実行中のドメインの"live" XML では、この属性は、たとえ**type='network'**であっても、リッスンに使用される IP アドレスに設定されます。 ● network - **type='network'** の場合、network 属性には、libvirt の設定済みのネットワーク一覧にネットワークの名前が含まれます。名前の付いたネットワーク設定を検証し、適切なリッスンアドレスを決定します。たとえば、ネットワークの設定に IPv4 アドレスがある場合 (正引きタイプのルート、nat、または no forward タイプ (isolated) がある場合など) は、ネットワークの設定に一覧表示されている最初の IPv4 アドレスが使用されます。ネットワークがホスト物理マシンブリッジを記述している場合、そのブリッジデバイスに関連付けられている最初の IPv4 アドレスが使用され、ネットワークが直接 (macvtap) モードの 1 つを記述している場合、最初の転送 dev の最初の IPv4 アドレス使用されます。

20.16.13. ビデオデバイス

ビデオデバイス。

ビデオデバイスの設定を指定するには、管理ツールを使用して、ドメイン XML に次の変更を行います。

図20.58 ビデオデバイス

```
...
<devices>
  <video>
    <model type='vga' vram='8192' heads='1'>
      <acceleration accel3d='yes' accel2d='yes'/>
    </model>
  </video>
</devices>
...
```

graphics 要素には必須の**type** 属性があり、以下の説明に従って、"sdl"、"vnc"、"rdp"、または "desktop" の値をとります。

表20.22 グラフィカルフレームバッファー要素

パラメーター	説明
video	**video** 要素は、ビデオデバイスを説明するコンテナーです。後方互換性のために、動画が設定されておらず、ドメイン XML に **graphics** 要素がある場合は、ゲスト仮想マシンの種類に応じたデフォルトの **video** が追加されます。ram または vram が指定されていない場合は、デフォルト値が使用されます。
model	これには、利用可能なハイパーバイザー機能に応じて、**vga**、**cirrus**、**vmvga**、**xen**、**vbox**、または **qxl** の値を取る必須の **type** 属性があります。また、vram と、ヘッドを使用した数値を使用して、ビデオメモリーの量をキビバイト (1024 バイトのブロック) で提供することもできます。
アクセラレーション	アクセラレーションに対応している場合は、**acceleration** 要素の **accel3d** 属性および **accel2d** 属性を使用して有効にする必要があります。
address	オプションの address サブ要素を使用すると、ビデオデバイスを特定の PCI スロットに関連付けることができます。

20.16.14. コンソール、シリアル、パラレル、およびチャネルデバイス

キャラクターデバイスは、仮想マシンと対話する方法を提供します。準仮想化コンソール、シリアルポート、パラレルポート、およびチャネルはすべて文字デバイスとして分類されるため、同じ構文を使用して表されます。

コンソール、チャネル、およびその他のデバイス設定を指定する場合は、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.59 コンソール、シリアル、パラレル、およびチャネルデバイス

```
...
<devices>
  <parallel type='pty'>
    <source path='/dev/pts/2'/>
    <target port='0'/>
  </parallel>
  <serial type='pty'>
    <source path='/dev/pts/3'/>
    <target port='0'/>
  </serial>
  <console type='pty'>
    <source path='/dev/pts/4'/>
    <target port='0'/>
  </console>
  <channel type='unix'>
    <source mode='bind' path='/tmp/guestfwd'/>
    <target type='guestfwd' address='10.0.2.1' port='4600'/>
  </channel>
</devices>
...
```

このディレクティブの各々では、トップレベル要素の名前 (parallel、serial、console、channel) でデバイスがゲスト仮想マシンにどのように提示されるかを説明します。ゲスト仮想マシンインターフェイスは、target 要素で設定されます。ホスト物理マシンに提示されるインターフェイスは、トップレベル要素の type 属性で指定されます。ホスト物理マシンインターフェイスは、source 要素で設定されます。source 要素には、ソケットパスでのラベリングの実行方法を上書きするオプションの seclabel を指定できます。この要素がない場合は、ドメインごとの設定からセキュリティーラベルが継承されます。各キャラクターデバイス要素には、オプションのサブ要素**address** があり、デバイスを特定のコントローラーまたは PCI スロットに関連付けることができます。

20.16.15. ゲスト仮想マシンのインターフェイス

キャラクターデバイスは、次のいずれかの形式でゲスト仮想マシンに表示されます。

パラレルポートを設定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.60 ゲスト仮想マシンインターフェイスパラレルポート

```
...
<devices>
  <parallel type='pty'>
    <source path='/dev/pts/2'/>
    <target port='0'/>
  </parallel>
</devices>
...
```

<target> には、ポート番号を指定する **port** 属性を指定できます。ポートには 0 から始まる番号が付けられています。通常、0、1、または 2 つのパラレルポートがあります。

シリアルポートを設定するには、管理ツールを使用して、ドメイン XML に以下の変更を行います。

図20.61 ゲスト仮想マシンインターフェイスのシリアルポート

```
...
<devices>
  <serial type='pty'>
    <source path='/dev/pts/3'/>
    <target port='0'/>
  </serial>
</devices>
...
```

<target> には、ポート番号を指定する **port** 属性を指定できます。ポートには 0 から始まる番号が付けられています。通常、シリアルポートは 0、1、または 2 です。オプションの **type** 属性もあり、その値には 2 つの選択肢があります。1 つは **isa-serial** で、もう 1 つは **usb-serial** です。**type** がない場合は、**isa-serial** がデフォルトで使用されます。usb-serial の場合、**type='usb'** の任意のサブ要素 **<address>** で、上記の特定のコントローラーにデバイスを関連付けることができます。

<console> 要素は、対話式コンソールを表すために使用されます。以下のルールに従い、使用中のゲスト仮想マシンのタイプによって、コンソールは準仮想化デバイスである場合と、シリアルデバイスのクローンである場合があります。

- **targetType** 属性が設定されていない場合、デフォルトのデバイス**type**はハイパーバイザーのルールに従います。libvirt に渡された XML を再クエリーする際に、デフォルトの**type**が追加されます。完全に仮想化されたゲスト仮想マシンの場合、デフォルトのデバイスタイプは通常、シリアルポートです。

- **targetType** 属性が **serial** で、**<serial>** 要素が存在しない場合は、console 要素が **<serial>** 要素にコピーされます。**<serial>** 要素がすでに存在する場合は、console 要素は無視されます。

- **targetType** 属性が **serial** でない場合は、通常処理されます。

- 最初の **<console>** 要素のみが、**serial** の**targetType**を使用できます。セカンダリーコンソールはすべて準仮想化する必要があります。

- s390 では、console 要素が sclp または sclplm (ラインモード) の targetType を使用できます。SCLP は、s390 のネイティブのコンソールタイプです。SCLP コンソールには関連付けられたコントローラーがありません。

以下の例では、virtio コンソールデバイスがゲスト仮想マシンに /dev/hvc[0-7] として公開されています (詳細については、http://fedoraproject.org/wiki/Features/VirtioSerial を参照してください)。

図20.62 ゲスト仮想マシンインターフェイス - virtio コンソールデバイス

```
...
<devices>
  <console type='pty'>
    <source path='/dev/pts/4'/>
    <target port='0'/>
  </console>

  <!-- KVM virtio console -->
  <console type='pty'>
    <source path='/dev/pts/5'/>
    <target type='virtio' port='0'/>
  </console>
</devices>
...

...
<devices>
  <!-- KVM s390 sclp console -->
  <console type='pty'>
    <source path='/dev/pts/1'/>
    <target type='sclp' port='0'/>
  </console>
</devices>
...
```

コンソールがシリアルポートとして提示される場合、**<target>** 要素にはシリアルポートと同じ属性があります。通常、コンソールは 1 つだけです。

20.16.16. チャネル

これは、ホスト物理マシンとゲスト仮想マシン間のプライベート通信チャネルを表し、管理ツールを使用してゲスト仮想マシン仮想マシンに変更を加えることで操作され、ドメイン xml の次のセクションに変更が加えられます。

図20.63 チャネル

```
...
<devices>
  <channel type='unix'>
    <source mode='bind' path='/tmp/guestfwd'/>
    <target type='guestfwd' address='10.0.2.1' port='4600'/>
  </channel>

  <!-- KVM virtio channel -->
  <channel type='pty'>
    <target type='virtio' name='arbitrary.virtio.serial.port.name'/>
  </channel>
  <channel type='unix'>
    <source mode='bind' path='/var/lib/libvirt/qemu/f16x86_64.agent'/>
    <target type='virtio' name='org.qemu.guest_agent.0'/>
  </channel>
  <channel type='spicevmc'>
    <target type='virtio' name='com.redhat.spice.0'/>
  </channel>
</devices>
...
```

これは、さまざまな方法で実装できます。**<channel>** の特定のタイプは、**<target>** 要素の **type** 属性で指定します。各チャンネルタイプには、以下のように異なるターゲット属性があります。

- **guestfwd** - ゲスト仮想マシンが指定の IP アドレスおよびポートに送信する TCP トラフィックが、ホスト物理マシンのチャネルデバイスに転送されることを指定します。**target** 要素には、address 属性および port 属性が必要です。

- **virtio** - 準仮想化された virtio チャネル。 **<channel>** は **/dev/vport*** の下のゲスト仮想マシンで公開され、オプションの要素 **name** が指定されている場合は、**/dev/virtio-ports/$name** (詳細は、http://fedoraproject.org/wiki/Features/VirtioSerial を参照してください)。オプションの要素 **address** では、チャンネルを特定の**type='virtio-serial'** コントローラーに関連付けることができます (上記を参照)。QEMU では、名前が org.qemu.guest_agent.0 の場合、libvirt は、ゲスト仮想マシンのシャットダウンやファイルシステムの静止などのアクションのために、ゲスト仮想マシンにインストールされているゲスト仮想マシンエージェントと対話できます。

- **spicevmc** - 準仮想化 SPICE チャンネル。ドメインには、グラフィックデバイスとしての SPICE サーバーも必要です。この時点で、ホストの物理マシンがメインチャネル全体でメッセージをピギーバックします。属性が **type='virtio';** の **target** 要素が必要です。オプションの属性 **name** は、ゲスト仮想マシンがチャネルにアクセスする方法を制御し、デフォルトは **name='com.redhat.spice.0'** になります。オプションの **<address>** 要素は、チャンネルを特定の **type='virtio-serial'** コントローラーに関連付けることができます。

20.16.17. ホストの物理マシンインターフェイス

キャラクターデバイスは、ホストの物理マシンに、次のいずれかの形式で表示されます。

表20.23 キャラクターデバイスの要素

パラメーター	説明	XML スニペット

パラメーター	説明	XML スニペット
ドメインログファイル	キャラクターデバイスのすべての入力を無効にし、仮想マシンのログファイルに出力を送信します	<devices> <console type='stdio'> <target port='1'/> </console> </devices>
デバイスログファイル	ファイルが開かれ、キャラクターデバイスに送信したすべてのデータがファイルに書き込まれます。	<devices> <serial type="file"> <source path="/var/log/vm/vm-serial.log"/> <target port="1"/> </serial> </devices>
仮想コンソール	キャラクターデバイスを仮想コンソールのグラフィカルフレームバッファーに接続します。通常、これには "ctrl+alt+3" のような特別なホットキーシーケンスを介してアクセスします。	<devices> <serial type='vc'> <target port="1"/> </serial> </devices>
Null デバイス	キャラクターデバイスをボイドに接続します。入力にデータが提供されることはありません。書き込まれたデータはすべて破棄されます。	<devices> <serial type='null'> <target port="1"/> </serial> </devices>

パラメーター	説明	XML スニペット
擬似 TTY	疑似 TTY は、**/dev/ptmx** を使用して割り当てられます。**virsh console** などの適切なクライアントは、接続してローカルのシリアルポートと対話できます。	<devices> <serial type="pty"> <source path="/dev/pts/3"/> <target port="1"/> </serial> </devices>
NB 特殊ケース	NB 特殊なケースで、**<console type='pty'>** の場合、TTY パスもトップレベルの **<console>** タグの属性 **tty='/dev/pts/3'** として複製されます。これにより、**<console>** タグの構文が簡略化されました。	
ホスト物理マシンのデバイスプロキシー	キャラクターデバイスは、基本となる物理キャラクターデバイスに渡されます。エミュレートシリアルポートはホスト物理マシンのシリアルポートにのみ接続するなど、デバイスの種類が一致する必要があります。シリアルポートをパラレルポートに接続しないでください。	<devices> <serial type="dev"> <source path="/dev/ttyS0"/> <target port="1"/> </serial> </devices>
名前付きパイプ	キャラクターデバイスは、名前付きパイプに出力を書き込みます。詳細については、pipe(7) の man ページを参照してください。	<devices> <serial type="pipe"> <source path="/tmp/mypipe"/> <target port="1"/> </serial> </devices>
TCP クライアント/サーバー	キャラクターデバイスは、リモートサーバーに接続する TCP クライアントとして機能します。	<devices> <serial type="tcp"> <source mode="connect" host="0.0.0.0" service="2445"/> <protocol

パラメーター	説明	XML スニペット
		type="raw"/> <target port="1"/> </serial> </devices> または、クライアント接続を待機している TCP サーバーとして機能します。 <devices> <serial type="tcp"> <source mode="bind" host="127.0.0.1" service="2445"/> <protocol type="raw"/> <target port="1"/> </serial> </devices> または、raw の TCP の代わりに telnet を使用できます。また、telnets (セキュアな telnet) および tls も使用できます。 <devices> <serial type="tcp"> <source mode="connect" host="0.0.0.0" service="2445"/> <protocol type="telnet"/> <target port="1"/> </serial> <serial type="tcp"> <source mode="bind" host="127.0.0.1" service="2445"/> <protocol type="telnet"/> <target port="1"/> </serial> </devices>

パラメーター	説明	XML スニペット
UDP ネットワークコンソール	キャラクターデバイスは UDP netconsole サービスとしてパケットの送受信を行います。これは不可逆のサービスです。	<devices> <serial type="udp"> <source mode="bind" host="0.0.0.0" service="2445"/> <source mode="connect" host="0.0.0.0" service="2445"/> <target port="1"/> </serial> </devices>
UNIX ドメインソケットクライアント/サーバー	キャラクターデバイスは、UNIX ドメインソケットサーバーとして機能し、ローカルクライアントからの接続を受け入れます。	<devices> <serial type="unix"> <source mode="bind" path="/tmp/foo"/> <target port="1"/> </serial> </devices>

20.17. サウンドデバイス

仮想サウンドカードは、sound 要素を使用してホスト物理マシンに接続できます。

図20.64 仮想サウンドカード

```
...
<devices>
  <sound model='es1370'/>
</devices>
...
```

sound 要素には、必須の属性 **model** が 1 つあります。これは、実際のサウンドデバイスをエミュレートするものを指定します。有効な値は、基本となるハイパーバイザーに固有のものですが、一般的な選択肢は **'es1370'**、**'sb16'**、**'ac97'**、および **'ich6'** です。また、ich6 モデルセットを持つ sound 要素には、さまざまなオーディオコーデックをオーディオデバイスに割り当てるためのオプションの **codec** サブ要素を追加できます。指定しない場合は、再生および録画を許可するために、デフォルトのコーデックが割り当てられます。有効な値は **'duplex'** (ラインインおよびラインアウトをアドバタイズ) および **'micro'** (スピーカーおよびマイクをアドバタイズ) です。

図20.65 サウンドデバイス

```
...
<devices>
  <sound model='ich6'>
    <codec type='micro'/>
  <sound/>
</devices>
...
```

各サウンド要素には、オプションのサブ要素**<address>**があります。このサブ要素は、デバイスを特定の PCI スロットに接続できます (上記を参照)。

20.18. ウォッチドッグデバイス

仮想ハードウェアウォッチドッグデバイスは、**<watchdog>** 要素を介してゲスト仮想マシンに追加できます。ウォッチドッグデバイスには、ゲスト仮想マシンに追加のドライバーおよび管理デーモンが必要です。libvirt 設定でウォッチドッグを有効にするだけでは、それ自体では何の役にも立ちません。現在、ウォッチドッグの実行時にサポート通知はありません。

図20.66 ウォッチドッグデバイス

```
...
<devices>
  <watchdog model='i6300esb'/>
</devices>
...

...
<devices>
  <watchdog model='i6300esb' action='poweroff'/>
</devices>
</domain>
```

この XML では、以下の属性が宣言されています。

- **model** - 必要な **model** 属性は、実際のウォッチドッグデバイスをエミュレートするものを指定します。有効な値は、基になるハイパーバイザーに固有のものです。
- **model** 属性は、以下の値をとることができます。
  - **i6300esb** - 推奨されるデバイスで、PCI Intel 6300ESB をエミュレートします。
  - **ib700** - ISA iBase IB700 をエミュレートします。
- **action** - オプションの **action** 属性は、ウォッチドッグの期限が切れたときに行うアクションを説明します。有効な値は、基になるハイパーバイザーに固有のものです。**action** 属性には、以下の値を指定できます。
  - **reset**: デフォルト設定で、ゲスト仮想マシンを強制的にリセットします。
  - **shutdown** - ゲスト仮想マシンを正常にシャットダウンします (推奨されません)。

- **poweroff**: ゲスト仮想マシンを強制的にオフにします。
- **pause**: ゲスト仮想マシンを一時停止します。
- **none** - 何も実行しません。
- **dump** - ゲスト仮想マシンを自動的にダンプします。

'shutdown' アクションでは、ゲスト仮想マシンが ACPI シグナルに応答することが必要になります。ウォッチドッグが期限切れとなった状況では、ゲスト仮想マシンは通常、ACPI シグナルに応答できません。したがって、'shutdown' の使用は推奨されません。また、ダンプファイルを保存するディレクトリーは、/etc/libvirt/qemu.conf.ファイルの auto_dump_path で設定できます。

20.19. メモリーバルーンデバイス

仮想メモリーバルーンデバイスは、すべての Xen および KVM/QEMU ゲスト仮想マシンに追加されます。**<memballoon>** 要素として表示されます。必要に応じて自動的に追加されるため、特定の PCI スロットを割り当てる必要がない限り、ゲスト仮想マシン XML にこの要素を明示的に追加する必要はありません。memballoon デバイスを明示的に無効にする必要がある場合は、**model='none'** を使用できることに注意してください。

次の例では、KVM を使用してデバイスを自動的に追加しました

図20.67 メモリーバルーンデバイス

```
...
<devices>
  <memballoon model='virtio'/>
</devices>
...
```

これは、静的 PCI スロット 2 が要求された状態でデバイスを手動で追加する例です。

図20.68 手動で追加したメモリーバルーンデバイス

```
...
<devices>
  <memballoon model='virtio'>
    <address type='pci' domain='0x0000' bus='0x00' slot='0x02' function='0x0'/>
  </memballoon>
</devices>
</domain>
```

必要な **model** 属性は、どのようなバルーンデバイスが提供されるかを指定します。有効な値は、仮想化プラットフォームに固有です。**'virtio'** は KVM ハイパーバイザーのデフォルト設定であり、**'xen'** は Xen ハイパーバイザーのデフォルト設定です。

20.20. セキュリティーラベル

<seclabel> 要素により、セキュリティードライバーの動作を制御できます。基本的な動作モードは、**'dynamic'** で、libvirt が一意のセキュリティーラベルを自動的に生成します。**'static'** では、アプリケーションまたは管理者がラベルを選択します。**'none'** では、制限が無効になっています。動的ラベル生成では、libvirt は、仮想マシンに関連付けられたリソースに常に自動的に再ラベル付けします。静的

ラベル割り当てでは、デフォルトで、管理者またはアプリケーションは、すべてのリソースにラベルが正しく設定されていることを確認する必要があります。ただし、必要に応じて自動再ラベル付けを有効にすることができます。

libvirt で複数のセキュリティードライバーを使用する場合は、複数の seclabel タグを使用できます。ドライバーごとに 1 つ、各タグで参照されるセキュリティードライバーは、属性 **model** を使用して定義できます。最上位のセキュリティーラベルの有効な入力 XML 設定は次のとおりです。

図20.69 セキュリティーラベル

```
<seclabel type='dynamic' model='selinux'/>

<seclabel type='dynamic' model='selinux'>
  <baselabel>system_u:system_r:my_svirt_t:s0</baselabel>
</seclabel>

<seclabel type='static' model='selinux' relabel='no'>
  <label>system_u:system_r:svirt_t:s0:c392,c662</label>
</seclabel>

<seclabel type='static' model='selinux' relabel='yes'>
  <label>system_u:system_r:svirt_t:s0:c392,c662</label>
</seclabel>

<seclabel type='none'/>
```

入力 XML に**'type'** 属性が指定されていない場合は、セキュリティードライバーのデフォルト設定 (**'none'** または **'dynamic'**) が使用されます。**<baselabel>** を設定し、**'type'** を設定しないと、型が **'dynamic'** であると想定されます。自動リソース再ラベル付けが有効な実行中のゲスト仮想マシンの XML を表示する場合は、追加の XML 要素 **imagelabel** が含まれます。これは出力専用の要素であるため、ユーザー提供の XML ドキュメントでは無視されます。

以下の要素は、以下の値で操作できます。

- **type** - libvirt が一意のセキュリティーラベルを自動的に生成するかどうかを判断するために、**static**、**dynamic**、または **none** を指定します。
- **model** - 有効なセキュリティーモデル名で、現在アクティブなセキュリティーモデルと一致します
- **relabel** - **yes** または **no** のいずれかです。動的ラベル割り当てを使用する場合は、これを常に**yes** にする必要があります。静的ラベル割り当てでは、デフォルトは **no** になります。
- **<label>** - 静的ラベリングを使用する場合は、仮想ドメインに割り当てる完全なセキュリティーラベルを指定する必要があります。コンテンツの形式は、使用されているセキュリティードライバーによって異なります。
  - **SELinux** - SELinux のコンテキストです。
  - **AppArmor** - AppArmor プロファイル。
  - **DAC** - 所有者とグループをコロンで区切ります。ユーザー/グループ名または uid/gid の両方として定義できます。ドライバーはまずこれらの値を名前として解析しようとしますが、先頭にあるプラス記号を使用すると、ドライバーが値を uid または gid として解析するように強制できます。

- **<baselabel>** - 動的ラベル付けを使用する場合は、ベースセキュリティーラベルを指定するためにオプションで使用できます。コンテンツの形式は、使用されているセキュリティードライバーによって異なります。

- **<imagelabel>** - 出力専用の要素です。仮想ドメインに関連付けられたリソースで使用されるセキュリティーラベルを示します。コンテンツの形式は、使用しているセキュリティードライバーによって異なります。再ラベル付けが有効な場合、ラベル付けを無効にすることで、特定のソースファイル名に対して行われるラベル付けを微調整することもできます (ファイルが NFS またはその他に存在する場合に便利です)。セキュリティーラベル付けがないファイルシステム) または代替ラベルの要求 (管理アプリケーションがドメイン間ですべてではないが一部のリソースを共有できるようにする特別なラベルを作成する場合に便利です)。seclabel 要素がトップレベルドメイン割り当てではなく特定のパスに割り当てられている場合は、属性の relabel または sub-element ラベルのみがサポートされます。

20.21. ドメイン XML 設定の例

QEMU は AMD64 および Intel でゲスト仮想マシンをエミュレートしました

図20.70 ドメイン XML 設定の例

```
<domain type='qemu'>
  <name>QEmu-fedora-i686</name>
  <uuid>c7a5fdbd-cdaf-9455-926a-d65c16db1809</uuid>
  <memory>219200</memory>
  <currentMemory>219200</currentMemory>
  <vcpu>2</vcpu>
  <os>
    <type arch='i686' machine='pc'>hvm</type>
    <boot dev='cdrom'/>
  </os>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='cdrom'>
      <source file='/home/user/boot.iso'/>
      <target dev='hdc'/>
      <readonly/>
    </disk>
    <disk type='file' device='disk'>
      <source file='/home/user/fedora.img'/>
      <target dev='hda'/>
    </disk>
    <interface type='network'>
      <source network='default'/>
    </interface>
    <graphics type='vnc' port='-1'/>
  </devices>
</domain>
```

i686 上の KVM ハードウェアアクセラレーションゲスト仮想マシン

図20.71 ドメイン XML 設定の例

```
<domain type='kvm'>
  <name>demo2</name>
  <uuid>4dea24b3-1d52-d8f3-2516-782e98a23fa0</uuid>
  <memory>131072</memory>
  <vcpu>1</vcpu>
  <os>
    <type arch="i686">hvm</type>
  </os>
  <clock sync="localtime"/>
  <devices>
    <emulator>/usr/bin/qemu-kvm</emulator>
    <disk type='file' device='disk'>
      <source file='/var/lib/libvirt/images/demo2.img'/>
      <target dev='hda'/>
    </disk>
    <interface type='network'>
      <source network='default'/>
      <mac address='24:42:53:21:52:45'/>
    </interface>
    <graphics type='vnc' port='-1' keymap='de'/>
  </devices>
</domain>
```

第21章 トラブルシューティング

本章では、Red Hat Enterprise Linux 6 の仮想化に関する一般的な問題と解決策を説明します。

この章を読むことで、仮想化テクノロジーに関連する一般的な問題の一部を理解することができます。トラブルシューティングでは実用と経験を要し、本ガイドで学習するのが困難です。トラブルシューティングスキルを向上させるために、Red Hat Enterprise Linux 6 で仮想化を実験してテストすることが推奨されます。

本書で回答が見つからない場合は、オンライン上の仮想化コミュニティーに回答がある場合があります。Linux 仮想化 Web サイトの一覧については、「オンラインリソース」 を参照してください。

21.1. デバッグおよびトラブルシューティングツール

本セクションでは、システム管理者アプリケーション、ネットワークユーティリティー、およびデバッグツールの概要を説明します。標準のシステム管理ツールおよびログを使用して、トラブルシューティングを支援できます。

- **kvm_stat** - 参照 「kvm_stat」
- **trace-cmd**
- **ftrace** 『Red Hat Enterprise Linux 開発者ガイド 』を参照してください。
- **vmstat**
- **iostat**
- **lsof**
- **systemtap**
- **crash**
- **sysrq**
- **sysrq t**
- **sysrq w**

これらのネットワークツールは、以下の仮想化ネットワークの問題のトラブルシューティングに役立ちます。

- **ifconfig**
- **tcpdump**

  tcpdump コマンドは、ネットワークパケットをスニッフィングします。**tcpdump** は、ネットワーク異常やネットワーク認証の問題を見つける際に役立ちます。**tcpdump** のグラフィカルバージョンでは、**wireshark** という名前のグラフィカルバージョンがあります。

- **brctl**

  brctlは、Linux カーネル内のイーサネットブリッジ設定を検証して設定するネットワークツールです。次のコマンド例を実行する前に、root アクセス権が必要です。

```
# brctl show
bridge-name    bridge-id          STP  enabled  interfaces
-----------------------------------------------------------------------------
virtbr0            8000.feffffff       yes        eth0

# brctl showmacs virtbr0
port-no          mac-addr                local?       aging timer
1                fe:ff:ff:ff:ff:          yes           0.00
2                fe:ff:ff:fe:ff:          yes           0.00
# brctl showstp virtbr0
virtbr0
bridge-id              8000.fefffffffff
designated-root        8000.fefffffffff
root-port              0                path-cost            0
max-age                20.00            bridge-max-age       20.00
hello-time             2.00             bridge-hello-time    2.00
forward-delay          0.00             bridge-forward-delay 0.00
aging-time             300.01
hello-timer            1.43             tcn-timer            0.00
topology-change-timer  0.00             gc-timer             0.02
```

仮想化のトラブルシューティングに役立つコマンドを以下に示します。

- **strace** は、システムコールと、別のプロセスが受信して使用したイベントを追跡するコマンドです。
- **vncviewer** は、サーバーまたは仮想マシンで実行している VNC サーバーに接続します。**yum install tigervnc** を使用して **vncviwer** をインストールします。
- **vncserver** は、サーバーでリモートデスクトップを起動します。リモートセッションを介して、virt-manager などのグラフィカルユーザーインターフェイスを実行できます。**yum install tigervnc-server** を使用して **vncserver** をインストールします。

21.2. ディザスターリカバリーの準備

可能であれば、天候やその他の理由で機器が危険にさらされる状況に備えておくのが最善です。ホストの物理マシンで次のファイルとディレクトリーのバックアップを実行することを強くお勧めします。

- **/etc/libvirt** ディレクトリーからのすべてのファイル。
- **/var/lib/libvirt** ディレクトリーから、以下の項目をバックアップします。
  - **/var/lib/libvirt/dnsmasq** にある現在の dnsmasqDHCP リース
  - **/var/lib/libvirt/network** にある実行中の仮想ネットワーク設定ファイル
  - ゲストの現在の状態を保存するときに **virt-manager** によって作成されるゲスト仮想マシンファイル (存在する場合)。これらは **/var/lib/libvirt/qemu/save/** ディレクトリーにあります。virsh **save command** を使用して仮想マシンを作成した場合、ファイルは、ユーザーが **virsh save** 用に指定した場所にあります。
  - **qemu-img create** および **virsh snapshot-create** コマンドによって作成され、ユーザーがコマンドに指定した場所にあるゲスト仮想マシンのスナップショットファイル。
  - **virt-manager** によって作成されたゲスト仮想マシンのディスクイメージ (存在する場合)。これは、**/var/lib/libvirt/images/** ディレクトリーにあります。**virsh pool-define** コマンドを

使用して仮想ストレージを作成した場合、イメージファイルはユーザーが **virsh pool-define** に指定した場所にあります。ゲストイメージファイルをバックアップする方法については、次の手順で説明されている手順を使用してください。手順21.1「ディザスターリカバリーの目的でゲスト仮想マシンのディスクイメージのバックアップを作成する」。

- ブリッジを使用している場合は、**/etc/sysconfig/network-scripts/ifcfg-<bridge_name>** にあるファイルもバックアップする必要があります。
- オプションで、**/var/lib/libvirt/qemu/dump** にあるゲスト仮想マシンのコアダンプファイルをバックアップして、障害の原因を分析するために使用することもできます。ただし、一部のシステムでは、これらのファイルが非常に大きくなる可能性があることに注意してください。

手順21.1 ディザスターリカバリーの目的でゲスト仮想マシンのディスクイメージのバックアップを作成する

この手順では、いくつかの異なるディスクイメージタイプをバックアップする方法について説明します。

1. ゲスト仮想マシンのディスクイメージのみをバックアップするには、**/var/lib/libvirt/images** にあるファイルをバックアップします。LVM 論理ボリュームを使用してゲスト仮想マシンのディスクイメージをバックアップするには、次のコマンドを実行します。

   ```
   # lvcreate --snapshot --name snap --size 8G /dev/vg0/data
   ```

   このコマンドは、64G ボリュームの一部として 8G のサイズの snap という名前のスナップショットボリュームを作成します。

2. 次のようなコマンドを使用して、スナップショットのファイルを作成します。

   ```
   # mkdir /mnt/virt.snapshot
   ```

3. 次のコマンドを使用して、作成したディレクトリーとスナップショットボリュームをマウントします。

   ```
   # mount /dev/vg0/snap /mnt/virt.snapshot
   ```

4. 次のいずれかのコマンドを使用して、ボリュームをバックアップします。

   a.
   ```
   # tar -pzc -f /mnt/backup/virt-snapshot-MM-DD-YYYY.tgz
   /mnt/virt.snapshot++++++++++++
   ```

   b.
   ```
   # rsync -a /mnt/virt.snapshot/ /mnt/backup/virt-snapshot.MM-DD-YYYY/
   ```

21.3. VIRSH ダンプファイルの作成

virsh dump コマンドを実行すると、ゲスト仮想マシンのコアをファイルにダンプする要求が送信され、仮想マシンのエラーを診断できます。このコマンドを実行すると、引数 **corefilepath** で指定したファイルおよびパスに対する適切なパーミッションを手動で確認する必要があります。**virsh dump** コマンドは、コアダンプ (または **crash** ユーティリティー) と似ています。**virsh dump** ファイルを作成するには、次のコマンドを実行します。

```
#virsh dump <domain> <corefilepath> [--bypass-cache] { [--live] | [--crash] | [--reset] } [--verbose] [--memory-only]
```

ドメイン (ゲスト仮想マシンのドメイン名) と corefilepath (新しく作成されたコアダンプファイルの場所) は必須ですが、次の引数はオプションです。

- **--live** は、実行中のマシンにダンプファイルを作成し、一時停止しません。
- **--crash** はゲスト仮想マシンを停止し、ダンプファイルを生成します。主な違いは、ゲスト仮想マシンが停止として表示されず、理由がクラッシュしたことです。**virt-manager** では、ステータスが一時停止として表示されることに注意してください。
- **--reset** は、ダンプが成功した後にゲスト仮想マシンをリセットします。これらの 3 つのスイッチは相互に排他的であることに注意してください。
- **--bypass-cache** は、O_DIRECT を使用してファイルシステムキャッシュをバイパスします。
- **--memory-only** ダンプファイルは elf ファイルとして保存され、ドメインのメモリーと cpu の一般的なレジスター値のみが含まれます。このオプションは、ドメインがホストデバイスを直接使用する場合に非常に役立ちます。
- **--verbose** - ダンプの進捗を表示します。

ダンププロセス全体は、**virsh domjobinfo**z コマンドを使用して監視でき、**virsh domjobabort**を実行することでキャンセルできます。

21.4. KVM_STAT

kvm_stat コマンドは、**kvm** カーネルモジュールからランタイム統計を取得する python スクリプトです。**kvm_stat** コマンドは、**kvm** に表示されるゲストの動作を診断するために使用できます。特に、ゲストのパフォーマンスに関連する問題です。現在、報告されている統計はシステム全体を対象としており、実行中のすべてのゲストの動作が報告されます。このスクリプトを実行するには、qemu-kvm-tools パッケージをインストールする必要があります。

kvm_stat コマンドでは、**kvm** カーネルモジュールが読み込まれ、**debugfs** がマウントされている必要があります。上記のいずれかの機能が有効になっていない場合は、**debugfs** または **kvm** モジュールを有効にするために必要な手順が表示されます。以下に例を示します。

```
# kvm_stat
Please mount debugfs ('mount -t debugfs debugfs /sys/kernel/debug')
and ensure the kvm modules are loaded
```

必要に応じて**debugfs** をマウントします。

```
# mount -t debugfs debugfs /sys/kernel/debug
```

kvm_stat 出力

kvm_stat コマンドは、すべてのゲストとホストの統計情報を出力します。(**Ctrl**+**c** または **q** を使用して) コマンドが終了するまで、出力は更新されます。

```
# kvm_stat

kvm statistics
```

```
efer_reload                94       0
exits                   4003074   31272
fpu_reload              1313881   10796
halt_exits                14050     259
halt_wakeup                4496     203
host_state_reload       1638354   24893
hypercalls                    0       0
insn_emulation          1093850    1909
insn_emulation_fail           0       0
invlpg                    75569       0
io_exits                1596984   24509
irq_exits                 21013     363
irq_injections            48039    1222
irq_window                24656     870
largepages                    0       0
mmio_exits                11873       0
mmu_cache_miss            42565       8
mmu_flooded               14752       0
mmu_pde_zapped            58730       0
mmu_pte_updated               6       0
mmu_pte_write            138795       0
mmu_recycled                  0       0
mmu_shadow_zapped         40358       0
mmu_unsync                  793       0
nmi_injections                0       0
nmi_window                    0       0
pf_fixed                 697731    3150
pf_guest                 279349       0
remote_tlb_flush              5       0
request_irq                   0       0
signal_exits                  1       0
tlb_flush                200190       0
```

変数の説明

efer_reload

拡張機能イネーブルレジスタ (EFER) のリロードの数。

終了

すべての **VMEXIT** 呼び出しの数。

fpu_reload

VMENTRY が FPU の状態を再読み込みした回数。ゲストが浮動小数点ユニット (FPU) を使用している場合、**fpu_reload** は増分されます。

halt_exits

halt 呼び出しによるゲストの終了数。通常、このタイプの終了は、ゲストがアイドル状態になると表示されます。

halt_wakeup

halt からのウェイクアップの数。

host_state_reload
ホスト状態の完全なリロードの数 (現時点では tallies MSR 設定およびゲスト MSR 読み込み)。

ハイパーコール
ゲストハイパーバイザーサービス呼び出しの数。

insn_emulation
ホストによってエミュレートされたゲスト命令の数。

insn_emulation_fail
insn_emulation の試行に失敗した回数。

io_exits
I/O ポートアクセスから終了するゲストの数。

irq_exits
外部割り込みによるゲストの終了回数。

irq_injections
ゲストに送信された割り込みの数。

irq_window
未処理の割り込みウィンドウから終了するゲストの数。

ラージページ
現在使用中の大きなページの数。

mmio_exits
メモリーマップ I/O(MMIO) アクセスにより、ゲストが終了する回数。

mmu_cache_miss
作成された KVM MMU シャドウページの数。

mmu_flooded
MMU ページへの過剰な書き込み操作の検出数。これは、個々の書き込み操作ではなく、検出された書き込み操作をカウントします。

mmu_pde_zapped
ページディレクトリーエントリー (PDE) 破棄操作の数。

mmu_pte_updated
ページテーブルエントリー (PTE) 破棄操作の数。

mmu_pte_write
ゲストページテーブルエントリー (PTE) の書き込み操作の数。

mmu_recycled
再利用できるシャドウページの数。

mmu_shadow_zapped
無効化されたシャドウページの数。

mmu_unsync
まだリンクが解除されていない同期されていないページの数。

nmi_injections
ゲストへのマスク不可割り込み (NMI) インジェクションの数。

nmi_window
(未処理の) マスク不可割り込み (NMI) ウィンドウからのゲストの終了回数。

pf_fixed
固定 (非ページング) ページテーブルエントリー (PTE) マップの数。

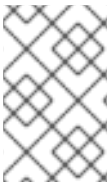
pf_guest
ゲストに挿入されたページフォールトの数。

remote_tlb_flush
リモート (sibling CPU) トランスレーションルックアサイドバッファー (TLB) フラッシュ要求の数。

request_irq
ゲスト割り込みウィンドウ要求が終了する数。

signal_exits
ホストからの保留中の信号によってゲストが終了する回数。

tlb_flush
ハイパーバイザーが実行する **tlb_flush** 操作の数。



注記

kvm_stat コマンドからの出力情報は、KVM ハイパーバイザーによって **/sys/kernel/debug/kvm/** ディレクトリーにある疑似ファイルとしてエクスポートされます。

21.5. ゲスト仮想マシンのシャットダウンに失敗する

従来は、**virsh shutdown** コマンドを実行すると、電源ボタンの ACPI イベントが送信されるため、物理マシンの電源ボタンを押したときと同じ動作がコピーされていました。すべての物理マシン内で、このイベントを処理するかどうかは OS 次第になります。以前は、オペレーティングシステムは単にサイレントシャットダウンしていました。現在、最も一般的なアクションは、実行すべき内容を尋ねるダイアログを表示することです。オペレーティングシステムによっては、特にユーザーがログインしていない場合に、このイベントを完全に無視するものもあります。このようなオペレーティングシステムがゲ

スト仮想マシンにインストールされていると、**virsh shutdown** は機能しません (無視されるか、仮想ディスプレイにダイアログが表示されます)。ただし、ゲスト仮想マシンに qemu-guest-agent チャンネルが追加されており、このエージェントがゲスト仮想マシンの OS 内で実行している場合は、**virsh shutdown** コマンドにより ACPI イベントを送信する代わりに、ゲスト OS をシャットダウンするように求められます。エージェントはゲスト仮想マシン OS 内からシャットダウンを呼び出し、すべてが想定どおりに機能します。

手順21.2 ゲスト仮想マシンでのゲストエージェントチャネルの設定

1. ゲスト仮想マシンを停止します。
2. ゲスト仮想マシンの ドメイン XML を開き、以下のスニペットを追加します。

   図21.1 ゲストエージェントチャネルの設定

```
<channel type='unix'>
   <source mode='bind'/>
   <target type='virtio' name='org.qemu.guest_agent.0'/>
</channel>
```

3. **virsh start [domain]** を実行して、ゲスト仮想マシンを起動します。
4. ゲスト仮想マシン (**yum install qemu-guest-agent**) に qemu-guest-agent をインストールし、システムの起動時にサービス (qemu-guest-agent.service) として自動的に実行されるようにします。詳細は、10章*qemu-img および QEMU ゲストエージェント* を参照してください。

21.6. シリアルコンソールのトラブルシューティング

Linux カーネルは、情報をシリアルポートに出力できます。これは、ビデオデバイスまたはヘッドレスサーバーでカーネルパニックおよびハードウェアの問題のデバッグに役立ちます。このセクションのサブセクションでは、KVM ハイパーバイザーを使用したホスト物理マシンのシリアルコンソール出力の設定について説明します。

本セクションでは、完全に仮想化したゲストに対してシリアルコンソールの出力を有効にする方法を説明します。

virsh console コマンドで、完全に仮想化されたゲストシリアルコンソール出力を表示できます。

完全に仮想化されたゲストのシリアルコンソールにはいくつかの制限があります。制限事項は次のとおりです。

- 出力データは削除またはスクラブルされます。

シリアルポートは、Linux では **ttyS0**、Windows では **COM1** と呼ばれます。

仮想オペレーティングシステムを設定して、情報を仮想シリアルポートに出力する必要があります。

完全に仮想化された Linux ゲストからドメインにカーネル情報を出力するには、**/boot/grub/grub.conf** ファイルを変更します。***kernel*** の行に ***console=tty0 console=ttyS0,115200*** を追加します。

```
title Red Hat Enterprise Linux Server (2.6.32-36.x86-64)
 root (hd0,0)
 kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/volgroup00/logvol00 \
```

```
console=tty0 console=ttyS0,115200
initrd /initrd-2.6.32-36.x86-64.img
```

ゲストを再起動します。

ホストで、以下のコマンドでシリアルコンソールにアクセスします。

```
# virsh console
```

virt-manager を使用して、仮想テキストコンソールを表示することもできます。ゲストコンソールウィンドウで、**View** メニューから **Text Consoles** の **Serial 1** を選択します。

21.7. 仮想化ログファイル

- 完全に仮想化された各ゲストログは、**/var/log/libvirt/qemu/** ディレクトリーにあります。各ゲストログは *GuestName*.log という名前で、サイズ制限に達すると定期的に圧縮されます。

仮想マシンマネージャーでエラーが発生した場合は、**$HOME/.virt-manager** ディレクトリーにある **virt-manager.log** ファイルで生成されたデータを確認できます。

21.8. ループデバイスエラー

ファイルベースのゲストイメージを使用する場合は、設定されているループデバイスの数を増やす必要がある場合があります。デフォルト設定では、最大 8 つのアクティブループデバイスが使用できます。8 個以上のファイルベースのゲストまたはループデバイスが必要な場合は、設定されたループデバイスを **/etc/modprobe.d/**ディレクトリーで調整できます。以下の行を追加します。

```
options loop max_loop=64
```

この例では 64 を使用しますが、別の数値を指定して最大ループ値を設定できます。また、システムにループデバイスバッキングゲストを実装する必要がある場合もあります。完全仮想化システムでループデバイスバッキングゲストを使用するには、**phy: device** コマンドまたは **file: file** コマンドを使用します。

21.9. ライブマイグレーションエラー

ライブマイグレーションにより、メモリーコンテンツが再度転送される可能性があります。このプロセスにより、ゲストはメモリーを常に書き込む状態であるため、移行が遅くなります。これが発生し、ゲストが 1 秒あたり数十 MB を超える書き込みを行っている場合、ライブマイグレーションが完了 (収束) しない可能性があります。この問題は、Red Hat Enterprise Linux 6 の時点で解決される予定ですが、Red Hat Enterprise Linux 7 では修正される予定です。

現在のライブマイグレーション実装では、デフォルトのマイグレーション時間が 30ms に設定されています。この値は、残りを転送するために、移行の最後にゲストの一時停止時間を決定します。値を大きくすると、ライブマイグレーションが収束する可能性が高くなります

21.10. BIOS での INTEL VT-X および AMD-V VIRTUALIZATION ハードウェア拡張の有効化

注記

専門知識を深めるには、Red Hat Virtualization (RH318) のトレーニングコースの受講をお勧めします。

本セクションでは、ハードウェア仮想化拡張機能を識別し、無効になっている場合に、その拡張機能を BIOS で有効にする方法を説明します。

Intel VT-x 拡張機能は、BIOS で無効にできます。特定のラップトップベンダーは、CPU で Intel VT-x 拡張機能をデフォルトで無効にしています。

AMD-V の BIOS では、仮想化拡張機能を無効にできません。

無効にした仮想化拡張機能を有効にする方法については、以下のセクションを参照してください。

BIOS で、仮想化拡張機能が有効になっていることを確認します。Intel VT または AMD-V の BIOS 設定は、通常 **Chipset** メニューまたは **Processor** メニューにあります。メニュー名は、このガイドとは異なる場合があります。仮想化拡張機能の設定は、**Security Settings** や、他の非標準のメニュー名などに記載されている場合があります。

手順21.3 BIOS での仮想化拡張機能の有効化

1. コンピューターを再起動し、システムの BIOS メニューを開きます。通常、これは **delete** キー、**F1** キー、または **Alt** キーと **F4** キーを押して実行できます。
2. **BIOS での仮想化拡張機能の有効化**

注記

以下の手順の多くは、使用しているマザーボード、プロセッサーのタイプ、チップセット、および OEM により異なる場合があります。システムの設定に関する正しい情報は、お使いのシステムの付随するドキュメントを参照してください。

   a. **Processor** サブメニューを開きます。プロセッサー設定メニューは、**Chipset**、**Chipset**、または **Northbridge** で非表示にすることができます。

   b. **Intel Virtualization Technology** を有効にします (Intel VT-x としても知られています)。**AMD-V** 拡張機能は、BIOS では無効にできないため、事前に有効にしておく必要があります。仮想化拡張機能は、OEM やシステム BIOS によっては、**Virtualization Extensions**、**Vanderpool**、またはその他の名前にラベルが付けられている場合があります。

   c. オプションが利用できる場合は、Intel VT-d または AMD IOMMU を有効にします。PCI デバイスの割り当てには、Intel VT-d および AMD IOMMU を使用します。

   d. **Save & Exit**を選択します。

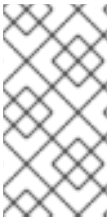
3. マシンを再起動します。
4. マシンが起動したら、**cat /proc/cpuinfo |grep -E "vmx|svm"** を実行します。***--color*** の指定は任意ですが、検索用語を強調表示したい場合は便利です。コマンドが出力すると、仮想化拡張機能が有効になります。出力がない場合は、システムの仮想化拡張機能や、正しい BIOS 設定が有効になっていない可能性があります。

21.11. KVM ネットワークパフォーマンス

デフォルトでは、KVM 仮想マシンには、仮想 Realtek 8139 (rtl8139) の NIC (ネットワークインターフェイスコントローラー) が割り当てられています。Red Hat Enterprise Linux ゲストにはデフォルトで virtio NIC が割り当てられていますが、Windows ゲストまたはゲストタイプは指定されていません。

rtl8139 仮想化 NIC はほとんどの環境で問題なく機能しますが、このデバイスは、10 ギガバイトイーサネットなど、一部のネットワークでパフォーマンス低下の問題が発生する可能性があります。

パフォーマンスを向上させるために、準仮想化ネットワークドライバーに切り替えることができます。



注記

仮想化 Intel PRO/1000 (**e1000**) ドライバーも、エミュレートされたドライバーとしてサポートされていることに注意してください。**e1000** ドライバーを使用するには、以下の手順の **virtio** を **e1000** に置き換えます。最適なパフォーマンスを得るには、**virtio** ドライバーを使用することを推奨します。

手順21.4 virtio ドライバーへの切り替え

1. ゲストオペレーティングシステムをシャットダウンします。
2. **virsh** コマンドでゲストの設定ファイルを編集します (***GUEST*** はゲストの名前です)。

```
# virsh edit GUEST
```

   virsh edit コマンドは、**$EDITOR** シェル変数を使用して、使用するエディターを決定します。

3. 設定のネットワークインターフェイスセクションを見つけます。このセクションは、以下のスニペットと似ています。

```
<interface type='network'>
  [output truncated]
  <model type='rtl8139' />
</interface>
```

4. model 要素の type 属性を ***'rtl8139'*** から ***'virtio'*** に変更します。これにより、ドライバーが rtl8139 ドライバーから e1000 ドライバーに変更されます。

```
<interface type='network'>
  [output truncated]
  <model type='virtio' />
</interface>
```

5. 変更を保存し、テキストエディターを終了します。
6. ゲストオペレーティングシステムを再起動します。

その他のネットワークドライバーを使用した新しいゲストの作成

別のネットワークドライバーを使用して、新しいゲストを作成することもできます。ネットワーク接続でゲストをインストールするのが困難な場合に必要になることがあります。この方法では、テンプレートとして使用するために、少なくとも 1 つのゲストが作成されている (CD または DVD からインストールされている可能性あり) 必要があります。

1. 既存のゲスト (この例では *Guest1*) から XML テンプレートを作成します。

```
# virsh dumpxml Guest1 > /tmp/guest-template.xml
```

2. XML ファイルをコピーして編集し、一意のフィールド (仮想マシン名、UUID、ディスクイメージ、MAC アドレス、およびその他の一意のパラメーター) を更新します。UUID および MAC アドレス行を削除できることに注意してください。virsh を実行すると、UUID と MAC アドレスが生成されます。

```
# cp /tmp/guest-template.xml /tmp/new-guest.xml
# vi /tmp/new-guest.xml
```

ネットワークインターフェイスセクションにモデル行を追加します。

```
 <interface type='network'>
  [output truncated]
  <model type='virtio' />
</interface>
```

3. 新しい仮想マシンを作成します。

```
# virsh define /tmp/new-guest.xml
# virsh start new-guest
```

21.12. LIBVIRT を使用して外部スナップショットを作成するための回避策

QEMU ゲストのスナップショットには 2 つのクラスがあります。Internal snapshots は qcow2 ファイルに完全に含まれ、libvirt で完全に対応しているため、スナップショットの作成、削除、および復帰が可能になります。これは、特にオプションが指定されていない場合に、スナップショットの作成時に libvirt が使用するデフォルトの設定です。このファイルタイプは、スナップショットの作成時には少し長くなりますが、libvirt で qcow2 ディスクを使用する必要があります。このファイルタイプのもう 1 つの欠点は、qcow2 ディスクが QEMU からの改善を受けないことです。

一方、外部スナップショットは、あらゆるタイプの元のディスクイメージで機能し、ゲストのダウンタイムなしで取得でき、QEMU から積極的な改善を受けることができます。libvirt では、**--disk-only** オプションを使用して **snapshot-create-as** (または同じように **snapshot-create** に明示的な XML ファイルを指定する場合) に作成されます。現時点では、libvirt が作成できるものの、それ以上操作を行うことができないため、外部スナップショットは一方向操作となります。

21.13. 日本語キーボードを使用したゲストコンソールでの文字の欠落

Red Hat Enterprise Linux 6 ホストで、日本語キーボードをマシンにローカルに接続すると、ゲストコンソールでアンダースコア (_文字) などの文字が正しく表示されない場合があります。これは、必要なキーマップがデフォルトで正しく設定されていないために発生します。

Red Hat Enterprise Linux 3 および Red Hat Enterprise Linux 6 ゲストでは、通常、関連するキーを押したときにエラーメッセージが表示されません。ただし、Red Hat Enterprise Linux 4 および Red Hat Enterprise Linux 5 のゲストでは、以下のようなエラーが表示される場合があります。

```
atkdb.c: Unknown key pressed (translated set 2, code 0x0 on isa0060/serio0).
atkbd.c: Use 'setkeycodes 00 <keycode>' to make it known.
```

virt-manager でこの問題を修正するには、次の手順を実行します。

- 影響を受けるゲストを **virt-manager** で開きます。

- **View → Details** をクリックします。
- 一覧で **Display VNC** を選択します。
- **Keymap** プルダウンメニューで、**Auto** を **ja** に変更します。
- **Apply** ボタンをクリックします。

または、ターゲットゲストで **virsh edit** コマンドを使用してこの問題を修正する場合は、次のコマンドを実行します。

- **virsh edit <target guest>** を実行します。
- <graphics> タグに以下の属性を追加します。**keymap='ja'**。以下に例を示します。

```
<graphics type='vnc' port='-1' autoport='yes' keymap='ja'/>
```

21.14. 仮想化拡張機能の検証

このセクションを使用して、システムにハードウェア仮想化拡張機能があるかどうかを判断します。完全仮想化には、仮想化拡張機能 (Intel VT-x または AMD-V) が必要です。

1. 次のコマンドを実行して、CPU 仮想化拡張機能が利用可能であることを確認します。

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

2. 出力を分析します。

   - 以下の出力には、Intel VT-x 拡張機能が含まれる Intel プロセッサーを示す **vmx** エントリーが含まれます。

```
flags   : fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
 dts acpi mmx fxsr sse sse2 ss ht  tm syscall lm constant_tsc pni monitor ds_cpl
 vmx est tm2 cx16 xtpr lahf_lm
```

   - 以下の出力には、AMD-V 拡張機能が含まれる AMD プロセッサーを示す **svm** エントリーが含まれます。

```
flags   :  fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
 mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt lm 3dnowext 3dnow pni cx16
 lahf_lm cmp_legacy svm cr8legacy ts fid vid ttp tm stc
```

   出力を受信した場合、プロセッサーにはハードウェア仮想化拡張機能があります。ただし、状況によっては、製造元が BIOS で仮想化拡張機能を無効にする場合があります。

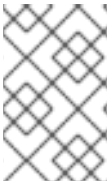
   flags:出力コンテンツは、システム上のハイパースレッド、コア、または CPU ごとに 1 回ずつ、複数回表示される場合があります。

   BIOS で仮想化拡張機能を無効にすることができます。拡張機能が表示されない場合、または完全仮想化が機能しない場合は、手順21.3「BIOS での仮想化拡張機能の有効化」 を参照してください。

3. **KVM サブシステムがロードされていることを確認します**
   追加のチェックとして、**kvm** モジュールがカーネルに読み込まれていることを確認します。

```
# lsmod | grep kvm
```

出力に **kvm_intel** または **kvm_amd** が含まれる場合は、**kvm** ハードウェア仮想化モジュールが読み込まれ、システムが要件を満たしています。



注記

libvirt パッケージがインストールされている場合は、**virsh** コマンドは、仮想化システム機能の一覧を出力します。完全なリストを受け取るには、root として **virsh capabilities** を実行します。

付録A 仮想ホストメトリクスデーモン (VHOSTMD)

vhostmd (仮想ホストメトリックデーモン) を使用すると、仮想マシンは、実行されているホストに関する限られた情報を確認できます。このデーモンは、Red Hat Enterprise Linux for SAP でのみ提供されます。

ホストでは、デーモン (**vhostmd**) が実行され、定期的にメトリックをディスクイメージに書き込みます。このディスクイメージは、ゲスト仮想マシンに読み取り専用でエクスポートされます。ゲスト仮想マシンは、ディスクイメージを読み取ってメトリックを確認できます。単純な同期により、ゲスト仮想マシンが古いメトリックや破損したメトリックを認識しないようにします。

システム管理者は、ゲスト仮想マシンごとに使用できるメトリックを選択します。さらに、システム管理者は、1 つ以上のゲスト仮想マシンがメトリック設定にアクセスできないようにブロックする場合があります。

したがって、**vhostmd** および **vm-dump-metrics** を使用するお客様は、RHEL for SAP Business Applications のサブスクリプションが必要になります。これにより、SAP を実行する RHEL システムをカスタマーポータルまたは Red Hat Subscription Management の RHEL for SAP チャンネルにサブスクライブしてパッケージをインストールします。カスタマーポータルの次の kbase の記事では、RHEL での vhostmd の設定について説明しています。https://access.redhat.com/knowledge/solutions/41566

付録B 関連情報

仮想化および Red Hat Enterprise Linux の詳細は、以下のリソースを参照してください。

B.1. オンラインリソース

- http://www.libvirt.org/ は、**libvirt** 仮想化 API の公式 Web サイトです。
- https://virt-manager.org/ は、仮想マシンを管理するグラフィカルアプリケーションである **Virtual Machine Manager** (virt-manager) のプロジェクトの Web サイトです。
- Red Hat Virtualization - http://www.redhat.com/products/cloud-computing/virtualization/
- Red Hat 製品ドキュメント - https://access.redhat.com/documentation/en/
- 仮想化テクノロジーの概要 - http://virt.kernelnewbies.org

B.2. インストールされているドキュメント

- **man virsh** および **/usr/share/doc/libvirt-*<version-number>*** - **virsh** 仮想マシン管理ユーティリティーのサブコマンドとオプション、および **libvirt** 仮想ライブラリー API の包括的な情報が含まれます。
- **/usr/share/doc/gnome-applet-vm-*<version-number>*** - ローカルで実行している仮想マシンを監視および管理する GNOME グラフィカルパネルアプレットのドキュメント。
- **/usr/share/doc/libvirt-python-*<version-number>*** - **libvirt** ライブラリーの Python バインディングの詳細を説明します。**libvirt-python** パッケージにより、python の開発者は、**libvirt** 仮想化管理ライブラリーとインターフェイスするプログラムを作成できます。
- **/usr/share/doc/python-virtinst-*<version-number>*** - 仮想マシン内で Fedora および Red Hat Enterprise Linux 関連のディストリビューションのインストールを開始する際に役に立つ、**virt-install** コマンドに関するドキュメントを提供します。
- **/usr/share/doc/virt-manager-*<version-number>*** - Virtual Machine Manager に関するドキュメントを提供します。このドキュメントは、仮想マシンを管理するグラフィカルツールを提供します。

付録C 更新履歴

改訂 1-502 6.9 GA リリースの更新	Mon Mar 08 2017	Jiri Herrmann
改訂 1-501 6.8 GA リリースの更新	Mon May 02 2016	Jiri Herrmann
改訂 1-500 6.8 ベータリリースの複数の更新	Thu Mar 01 2016	Jiri Herrmann
改訂 1-449 改訂履歴の整理	Thu Oct 08 2015	Jiri Herrmann
改訂 1-447 6.7 GA リリースの更新。	Fri Jul 10 2015	Dayle Parker