CANONICAL

PV Drivers Installation Guide

January 2016

Version 5.0

History

Version	Date	Changes
1.0	2013-06-12	Initial Version
2.0	2013-07-26	Updates for completed drivers
3.0	2013-07-31	Updates for qemu version
4.0	2015-11-11	Format change and general update
5.0	2016-01-26	Added OpenStack instructions. Removed old qemu configurations

1. Description

VirtIO is a virtualization standard for network and disk device drivers where just the guest's device driver is aware to be running in a virtual environment and uses an API from the hypervisor. This enables guests to get high performance network and disk operations, and gives most of the performance benefits of paravirtualization. This document will cover where to find the drivers from Canonical, how to install them and, optionally, how to prepare a Windows image with the installed drivers for OpenStack. For the latter, a pre-existing OpenStack environment is required.

The following drivers are available for Windows version 2008, 2008R2, 2012 and 2012R2:

- PVStorage
- PVNet
- PVSerial
- PVBalloon

2. System Requirements

VirtIO is supported from QEMU version 1.0+noroms-0ubuntu14.25, included in Ubuntu since Precise (12.04.5).

3. Driver download

The drivers are available at http://support.canonical.com, at the 'Downloads' section in the frame named 'Customer links'. The drivers currently released are:

- VirtIO Drivers (version 131): for Windows 2012R2 x64
- VirtIO Drivers (version 123): for Windows 2012 x64
- VirtIO Drivers (version 123): for Windows 2008 R2 x64
- VirtIO Drivers (version 3): for Windows 2008 x86 and x64

4. Driver installation

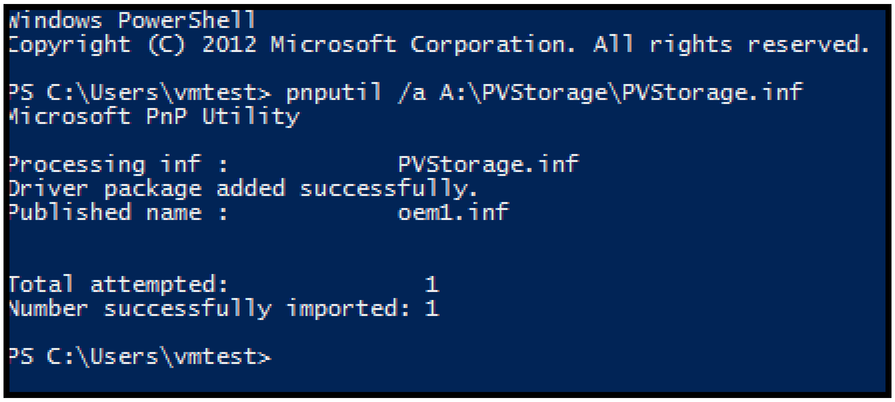
This section will cover the installation of a driver for an already existing virtIO device in a guest VM. To illustrate the installation process the next sections will cover the PVStorage driver installation, all the other drivers are installed the same way.

4.1. Using pnputil.exe

1. To install a driver using pnputil, execute the following line in a shell interface (cmd.exe or powershell):

```
pnputil /a $DRIVERPATH\$DRIVER.inf
```

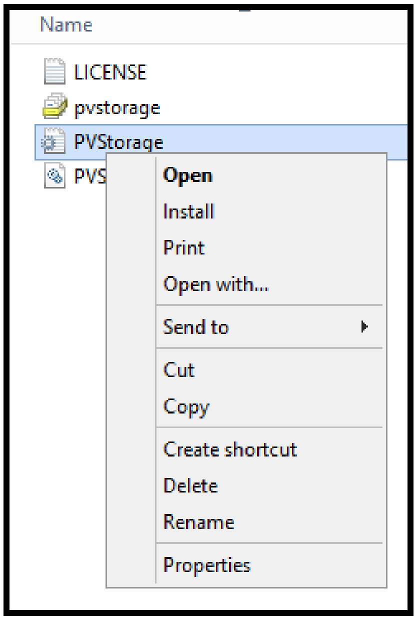
The output should be similar to:



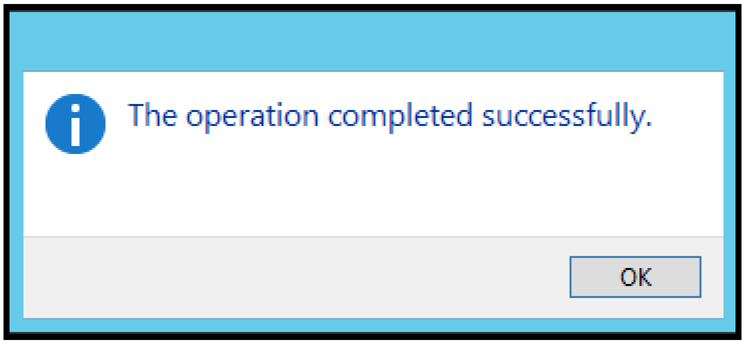
```
Windows PowerShell
Copyright (C) 2012 Microsoft Corporation. All rights reserved.

PS C:\Users\vmtest> pnputil /a A:\PVStorage\PVStorage.inf
Microsoft PnP Utility

Processing inf :            PVStorage.inf
Driver package added successfully.
Published name :            oem1.inf


Total attempted:              1
Number successfully imported: 1

PS C:\Users\vmtest>
```

4.2. Using Windows interface

To install a driver using Windows interface, go to the driver folder and right click the INF file. This will show a list of options, similar to the ones in the next image:



Click on “Install”, and once the installation has finished, a message similar to this image will appear:



The operation completed successfully.

5. Windows image creation

Canonical does not provide Windows images. This section covers the steps to create a fully functional Windows image for the deployment of VMs with VirtIO devices in OpenStack, for demonstration purposes Windows 2012R2 will be used.

5.1 Prepare a floppy disk with PV Storage driver

Windows does not include Canonical’s virtio drivers by default. To be able to install Windows on a PV disk the virtio driver must be installed for the setup wizard. An easy way to provide the driver to the Windows VM is by using a floppy disk. To create a floppy disk with the PVStorage driver for Windows 2012R2 the following steps are executed from an Ubuntu host:

1. Create and mount the floppy disk

```
cd /tmp
mkfs.vfat -C /tmp/windows2012R2_floppy.img 1440
sudo mount -o loop -t vfat /tmp/windows2012R2_floppy.img /media
```

2. Unzip the drivers file (see Driver download)

```
unzip -q windows2012R2_x64_r131.zip
```

3. Copy the driver to the floppy disk

```
sudo cp -r ./windows2012R2_x64_r131/PVStorage /media
```

4. Umount the disk

```
sync
sudo umount /media
```

5.2 Create a Windows VM with all PV devices

There are several ways to deploy a VM with PV devices and drivers, such as `virt-install`, `qemu` command line or `nova`. In this example, a libvirt domain XML file containing all PV devices and specific memory, vCPU and file paths is used because it allows a great configuration granularity. These are the steps to create an example Windows 2012R2 VM executed from an Ubuntu host:

1. Create the system disk file with `qemu-img`:

```
sudo qemu-img create -f qcow2 -o preallocation=metadata \
/tmp/windows2012R2_template.img 40G
```

2. Create a libvirt domain XML file with the following configuration, making sure the paths (disk, floppy and ISO file) and any other settings are relevant:

```
<domain type='kvm'>
  <name>windows2012R2_template</name>
  <memory unit='KiB'>4194304</memory>
  <currentMemory unit='KiB'>4194304</currentMemory>
  <vcpu placement='static'>2</vcpu>
  <os>
    <type arch='x86_64' machine='pc-i440fx-trusty'>hvm</type>
    <boot dev='cdrom'/>
    <boot dev='hd'/>
  </os>
  <features>
    <acpi/>
    <apic/>
   <pae/>
  </features>
  <clock offset='localtime'/>
  <on_poweroff>destroy</on_poweroff>
  <on_reboot>restart</on_reboot>
  <on_crash>destroy</on_crash>
  <devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
     <driver name='qemu' type='raw'/>
      <source file='/tmp/windows2012R2_template.img'/>
      <target dev='sda' bus='virtio'/>
    </disk>
    <disk type='file' device='cdrom'>
      <driver name='qemu' type='raw'/>
      <source file='/tmp/windows_2012R2.iso'/>
      <target dev='hdc' bus='ide'/>
      <readonly/>
    </disk>
    <disk type='file' device='floppy'>
      <driver name='qemu' type='raw'/>
      <source file='/tmp/windows2012R2_floppy.img'/>
      <target dev='fda' bus='fdc'/>
    </disk>
    <controller type='fdc' index='0'/>
    <controller type='ide' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
function='0x1'/>
    </controller>
    <controller type='usb' index='0'>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x01'
function='0x2'/>
    </controller>
<controller type='virtio-serial' index='1'>
</controller>
    <controller type='pci' index='0' model='pci-root'/>
    <interface type='network'>
      <source network='default'/>
      <model type='virtio'/>
    </interface>
    <input type='tablet' bus='usb'/>
    <input type='mouse' bus='ps2'/>
    <input type='keyboard' bus='ps2'/>
    <graphics type='vnc' port='-1' autoport='yes' listen='0.0.0.0'
keymap='en-us'>
      <listen type='address' address='0.0.0.0'/>
    </graphics>
    <video>
      <model type='cirrus' vram='9216' heads='1'/>
      <address type='pci' domain='0x0000' bus='0x00' slot='0x02'
function='0x0'/>
    </video>
    <memballoon model='virtio'>
    </memballoon>
  </devices>
</domain>
```

3. Create the VM using libvirt:

```
virsh define windows2012R2_template.xml
```

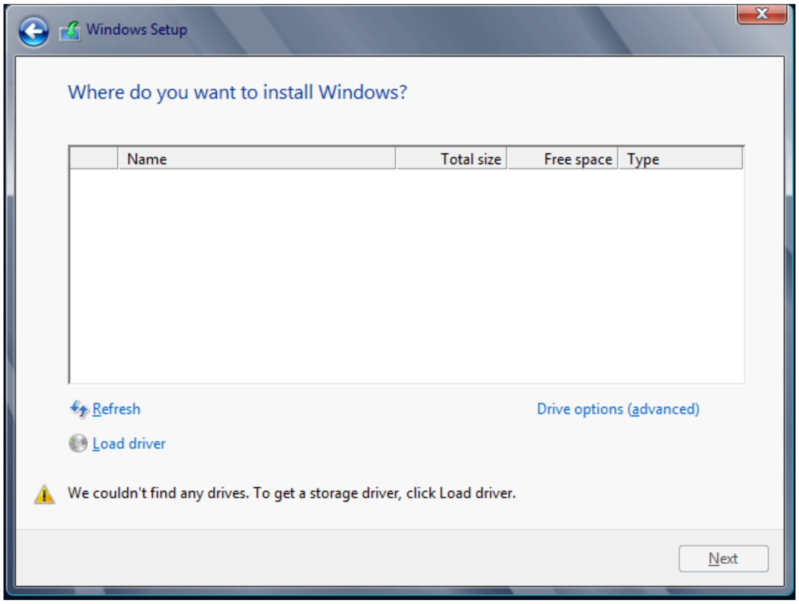
4. Start the VM:

```
virsh start windows2012R2_template
```

5.3 Install Windows on a PV disk

Once Windows installation starts, after a few welcome and keyboard configuration steps, it will show the disk partitioning wizard. The following images belong to a typical English Windows 2012R2 install:

1. Initially, the disk selection box is empty. Click on "Load driver"

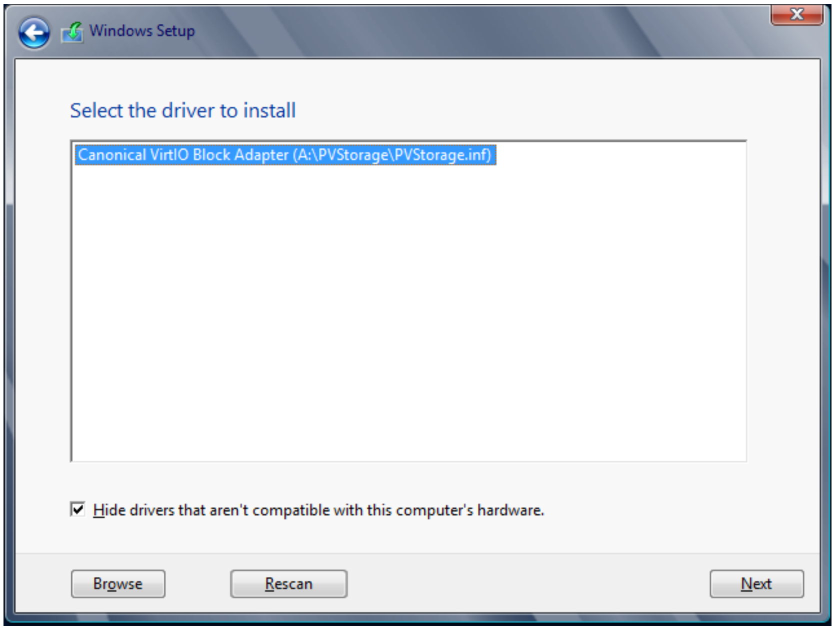


Where do you want to install Windows?

	Name	Total size	Free space	Type

We couldn't find any drives. To get a storage driver, click Load driver.

2. Click on "Browse", select the folder that contains the driver and select "Ok". If the driver has been found, you will see the driver name and its path.

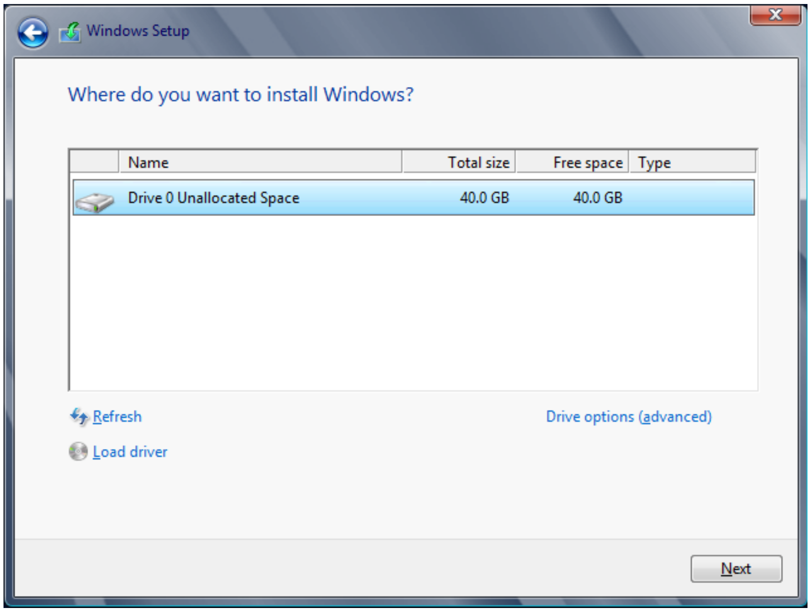


Select the driver to install

Canonical VirtIO Block Adapter (A:\PVStorage\PVStorage.inf)

☑ Hide drivers that aren't compatible with this computer's hardware.

3. Click on “Select” and wait until the selection box appears. The VirtIO disk should be in it.



Windows Setup

Where do you want to install Windows?

Name	Total size	Free space	Type
Drive 0 Unallocated Space	40.0 GB	40.0 GB	

4. From this point on, the disk can be partitioned the same way an IDE disk would be.

5.4 Install the remaining drivers

The next step after the OS installation is to install the remaining PV drivers (balloon, serial and network) as suggested in Driver installation. There are several ways to transfer the drivers to the VM, in this example a virtual USB drive is used.

1. Create and mount the USB device

```
qemu-img create -f raw /tmp/usb_disk.img 32M
mkfs.ntfs -F /tmp/usb_disk.img
sudo mount -o loop -t ntfs /tmp/usb_disk.img /media
```

2. Copy the drivers to the USB device

```
sudo cp windows2012R2_x64_r131.zip /media/
sudo unzip windows2012R2_x64_r131.zip -d /media
```

3. Umount the device and change the owner

```
sudo umount /media
sudo chown libvirt-qemu:libvirtd /tmp/usb_disk.img
```

4. Attach the USB device to the VM

```
sudo virsh qemu-monitor-command --hmp windows2012R2_template \
  drive_add 0 id=my_usb_disk,if=none,file=/tmp/usb_disk.img
sudo virsh qemu-monitor-command --hmp windows2012R2_template \
  device_add usb-storage,id=my_usb_disk,drive=my_usb_disk
```

5. See Driver Install for details on how to install the rest of the drivers.

5.5 Upload the image to Glance

A Glance image is a VM disk with metadata: the image name, device types used and memory or cpu requirements. Once finished with the VM installation and configuration of the PV devices the VM is ready to be converted into an image: stop the VM (either using virsh or shutting it down from the guest OS) and upload the image to Glance setting the minimum amount of memory and disk required for its deployment. See example:

```
glance image-create --name Windows2012R2 --disk-format qcow2 \
 --container-format bare --is-public True \
 --min-disk 40 --min-ram 4 \
 --file /tmp/windows2012R2_template.img
```

5.6 Device types in Glance images

Nova decides the device types to be used by a VM from the metadata related to that image that can be found from Glance. If this metadata is missing, the default value will be virtio, to optimize performance. To change the default values, the following metadata properties should be used:

- hw_disk_bus - type of disk controller to attach disk devices to
- hw_cdrom_bus - type of disk controller to attach cdrom devices to
- hw_nic_model - model of the network interface device

The available options for hw_disk_bus and hw_cdrom_bus are:

- scsi
- virtio
- uml
- xen
- ide
- usb

The available options for hw_nic_model are:

- e1000
- rtl8139
- xen
- virtio
- uml

For instance, to use an IDE disk and an e1000 NIC with a particular image (Windows 2012R2 in this example):

```
glance image-update\
 --property hw_disk_bus=ide\
 --property hw_cdrom_bus=ide\
 --property hw_vif_model=e1000\
 Windows2012R2
```

To revert any particular device back to virtIO (network for Windows 2012R2 in this example):

```
glance image-update --property hw_vif_model=virtio Windows2012R2
```