

VIRTUALBOX: CONVERT AN XP BOX INTO A VIRTUAL MACHINE

MARK CRUTCH

Ease the move to Linux by bringing your old Windows XP machine with you.

WHY DO THIS?

- Keep hold of your old XP installation
- Save £5.5m in support costs
- Move to Linux without the risk of losing XP functionality

On 8 April this year Microsoft issued its last update for Windows XP, leaving it vulnerable to future security exploits. Despite months of advance warning there are still millions of machines running this now obsolete operating system. Although this may seem like a perfect opportunity to migrate to Linux, many users are still reticent to give up on their old machines. One way to overcome this inertia is to convert the old Windows box into a virtual machine (VM) that will run inside the new Linux system, providing the reluctant user with a digital security blanket to cling to.

The real aim is to reduce the number of XP systems that are connected to the internet: every machine taken offline is one less that can host malware or participate in a denial of service attack. With that in mind we'll not only convert the physical box to a virtual one, but also include a few tips to ease the move to Linux and reduce the need to boot the Windows VM or put it online.

We'll be migrating a Windows XP machine, but the same approach also works with Vista and Windows 7. Due to hardware differences, licensing rules and various OEM flavours of Windows, not every machine will migrate using this approach – but we've had far more successes than failures. Although our

destination machine is a Linux box, you can use this same method to migrate your old system to a virtual machine running on a MacOS host, or even another version of Windows if you really want to.

Gather your hardware

Before starting our migration, it's worth noting a few hardware requirements. Virtual machines can quickly eat into available memory, drive space and processing power, so a capable host machine is a must. Take a look at the old XP machine to determine how much memory it has, and how much of the hard drive is in use, then ensure that the host has sufficient spare capacity to cover both the virtual machine and its own day to day usage. A large USB hard drive will also make things a bit easier, although it's not essential as long as you can move large files around using a network connection.

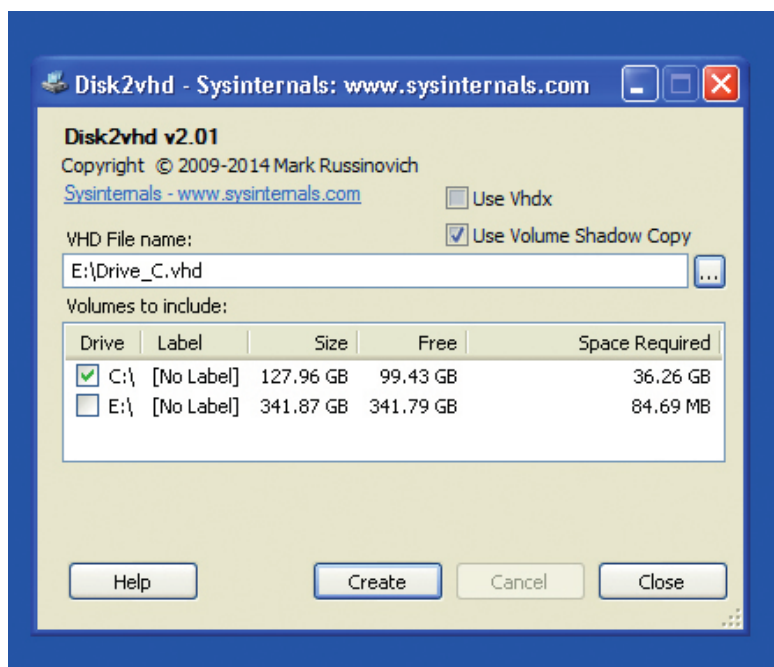
The Windows licence is probably tied to the old physical machine, so strictly speaking you should keep the hardware for as long as you have the virtual machine. You'll probably need the Windows Product Key from the sticker on the old box, but if that's too faded to read there are programs available that can extract the product key from a running Windows system. Although Microsoft's anti-piracy restrictions can sometimes cause problems, most XP machines migrate relatively smoothly. There are no guarantees, though, especially with XP Home and OEM installations, so don't go spending lots of money on extra RAM and a bigger hard drive until you're sure the migration will be a success.

Clone your hard drive

We'll start the migration by creating a disk image that we can use directly in VirtualBox. There are a variety of ways to do this, but for this tutorial we'll be using Disk2vhd on the Windows box. This can be downloaded free of charge (<http://bit.ly/18b901i>), but remember that we're trying to keep the XP machine off the internet, so it's probably best to download the file using another machine and then copy it to the XP box via a USB drive. Unzip the file, enter the directory, and double-click on the EXE file to run it.

Once you've accepted the EULA you'll be presented with the Disk2vhd dialog. The controls always seem to be in the wrong order, in our opinion, and we usually approach them from bottom to top. First, therefore,

Disk2vhd was written to create disk images for Microsoft's own VirtualPC program, but it works just as well with VirtualBox.



is the Volumes To Include panel, which lists all the drive partitions that XP knows about. Ensure that the partitions on the internal drive are checked, and that any partitions on the USB drive are unchecked. If you have multiple physical drives in the machine it's probably best to export each of them separately – all the partitions for the first drive into one file in the first pass, then run the program again to export all the partitions for the second drive into another.

Moving up the dialog we get to the VHD Filename. Use the button on the right to browse to your USB disk. If necessary you can use the internal drive for the file destination, but performance will suffer, and you'll need a lot of free space. Finally, confirm the state of the two checkboxes at the top of the dialog: we want to create a plain VHD file, so uncheck the Use Vhdx option; we do, however, want to check the Use Volume Shadow Copy option, which utilises a feature built into XP and later versions of Windows to snapshot the hard drive for imaging. This is especially vital if you're creating the file on the source drive.

With all the options set, it's time to click on the Create button and leave it working for a while. How long will depend on the amount of data and the speed of the machine and the drives, but it typically takes hours rather than minutes.

Prepare VirtualBox

While the XP box is being imaged, we can take the time to install and configure VirtualBox on the host. Most distros' repositories tend to lag some way behind the official release, so we'll download it directly from the VirtualBox website (https://www.virtualbox.org/wiki/Linux_Downloads). The top of the downloads page has links to DEB and RPM files, but if you scroll down a little you'll find instructions for installing from the VirtualBox repositories.

The core of VirtualBox is licensed under the GPL, but there's an additional commercial extension that you'll probably want to use. This is licensed under Oracle's own "PUEL" licence, which allows for personal, academic and evaluation use at no charge. Note that "personal use" includes using it in a commercial setting if you've installed it yourself, but do check the wording of the licence (https://www.virtualbox.org/wiki/VirtualBox_PUEL) if you're using it for anything other than inarguably personal or academic reasons.



The extension pack adds various features to the base VirtualBox system, the most notable being USB 2.0 support. If you're migrating XP to support a printer or other USB hardware that has poor Linux drivers this might be an important consideration – although VirtualBox's USB support isn't perfect, especially when dealing with esoteric drivers, so make sure you test the final system thoroughly.

Installing the extension pack is as simple as downloading it from the link on the VirtualBox download page (<https://www.virtualbox.org/wiki/Downloads>), then opening it with the main VirtualBox application. If your desktop has a suitable file association set up you'll probably be offered the option to open with VirtualBox when you download the file; otherwise you can manually add it via the File > Preferences > Extensions panel in the main VirtualBox manager.

With VirtualBox installed we're going to create a new VM. At this time it won't have a hard drive – that's probably still being imaged – but we can get the rest of the machine in place. Start by launching the VirtualBox manager and clicking the New button to bring up a wizard.

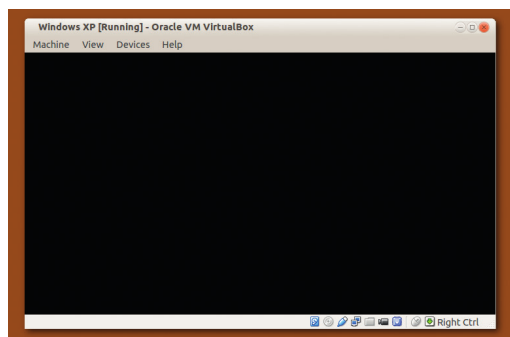
Give your VM a name: this will also be used as a directory name for holding your machine's files. Ensure that you pick the values for Type and Version that correspond to the machine you're migrating. In our case that's Microsoft Windows and Windows XP, respectively, but if your source machine is one of those rare beasts that's running the 64-bit version of XP you'll need to pick that specifically.

On the next page of the wizard you'll need to set the size of the virtual machine's memory. If you can

While you're at the VirtualBox website, make sure you also download the extensive user manual.

LV PRO TIP

To prevent XP connecting to the internet, uncheck the Cable Connected option in the VirtualBox network settings so that Windows simply thinks the Ethernet lead has become unplugged. That way, should you find you have to connect to the network in future, you can just re-check that option to reconnect the virtual cable.



Either your desktop is using a theme with black text on a black background, or you've got the wrong HAL.

Accessing your Windows files

If you want to copy files from your Windows machine to the Linux host, the simplest approach is probably to just attach a USB drive to Windows via the VirtualBox Devices menu, copy the files, detach it from VirtualBox to make it available to Linux and then copy the files back off it into the host.

That's fine for a one-off transfer, but for more frequent use you can set up a file share within XP using Windows' own SMB protocol, which can then be mounted on your Linux host. VirtualBox also has a Shared Folders pane in the VM's settings dialog that will let you share host folders with the Windows machine in a similar way, enabling you to "push" files to the Linux box from within XP. In our experience it's usually easier to share a Windows folder, then connect using Nautilus, Dolphin or Caja with the SMB protocol and the IP address

of the virtual machine – or use the lower-level Samba tools if your desktop environment doesn't understand the **SMB://** URL syntax.

One problem with all of these file transfer methods is that they require the Windows machine to be running. If all you want to do is get some files out of the Windows drive, there are various ways to mount the VM's disk image directly as a block device in the Linux filesystem. We've had most success with the **guestmount** command line tool, which mounts the drive using FUSE and is available in the repositories of most mainstream distributions. A word of warning: on our Linux Mint system we also had to install **libguestfs-tools** to get it working, which in turn pulls down a lot of files.

Using **guestmount** you can mount your Windows drive in read-only mode with the following

command as root (or prefix with **sudo** if your distribution needs it), replacing the VDI file and the **~/Windows** mount point as appropriate:

```
guestmount -a Windows_XP.vdi -i --ro -o allow_other ~/Windows
```

To unmount the disk image when you're finished with it, use:

```
fusermount -u ~/Windows
```

Although these are run as root, the **-o allow_other** FUSE option lets any user access the files, so you can copy files out of the Windows drive and into the Linux environment as a regular user. Despite both the VDI file and the mount point being owned by our normal user account, we have to use **sudo** on our Mint box for this approach to work. If anyone has any tips on getting **guestmount** to work as a regular user, please post them to the Linux Voice forums.

LV PRO TIP

Once the main migration is working you may wish to clone your VHD file into VirtualBox's native VDI format, as it's likely that the code for its own format is better tested and more robust. Select the File > Virtual Media Manager menu in the VirtualBox Manager, then release your VHD file from the VM. Copy it to a dynamically allocated VDI file, then attach it to the Windows machine via the Settings dialog. Check that it works, then delete the VHD file.

spare it, allocate the same amount as is present in the physical source machine. The third page is where we'll tell VirtualBox that we don't want a hard drive, and then finally create our new machine – but not until VirtualBox has offered a final warning about our lack of a disk.

Congratulations: you now have a half-imaged physical machine and a disk-deprived virtual machine. Take a well-earned break while the imaging chugs its way to completion.

Add the virtual hard drive.

With the imaging process over, you should now find a large VHD file on the USB drive. Copy any other files that you want off the XP box, then finally shut down the physical machine forever (hopefully). Mount the USB drive on the host machine, and copy the VHD file into your new virtual machine's directory. Unless you specifically chose otherwise it's probably in a folder called VirtualBox VMs in your home directory.

Return to the VirtualBox manager and select your XP VM in the list on the left. Click the heading of the Storage section in the right-hand panel to open the VM's settings dialog with the storage page showing. Select the IDE controller and click on the small icon

to add a hard disk (check the tooltips to distinguish between the two small icons, if it's not clear which one represents a hard disk). In the resultant dialog you should select Choose Existing Disk, then pick your VHD file from the VM's directory.

We're close to starting our virtual machine, but it's worth taking a couple of minutes to step through the other settings pages. If you've installed the VirtualBox extension pack then it's worth checking that USB 2.0 is enabled. We also like to enable the Remote Display option, which lets you access the screen of your running VM from another machine using a remote desktop client such as Remmina or Rdesktop. On the Advanced tab of the General Settings panel it's usually worth enabling the Shared clipboard feature. Set it to bidirectional to let you copy and paste text between the Linux host and the virtual machine. We tend to use the Description tab in the same panel to hold a copy of the Windows Product Key, to save me rummaging around the loft for the physical box if I need to enter it in future.

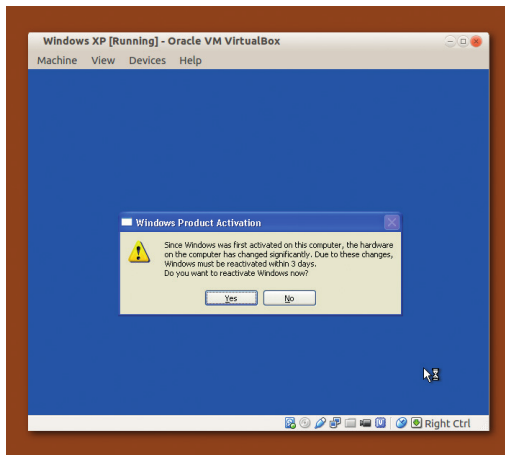
One final thing to set up is networking. We don't really want this machine on the internet, but you'll probably want it connected during its first boot to deal with Windows' activation requirements. On the Network panel, enable the first adaptor, and choose NAT from the first pop-up menu. Ensure the Cable Connected setting is checked.

With all that done it's time for the big moment. Close the settings dialog, ensure the XP VM is selected on the left, click the Start button in the VirtualBox toolbar, and watch your new virtual machine boot...

I'm sorry, Dave. I'm afraid I can't do that.

If you're very lucky you're now sitting in front of a VirtualBox window showing the XP login screen, or a Windows activation screen. More probably you're looking at a black window, with the icons in the VirtualBox status bar showing no disk or network activity. You've just been halted by HAL.

You've changed your hardware. Pirates sometimes change their hardware. Ergo, you are a pirate until you prove otherwise



HAL is the Hardware Abstraction Layer, a system component in Windows that exists as a number of different variants. The exact version that is on a given machine depends on the hardware that was present when Windows was installed. A mismatch between the installed HAL and the fake hardware presented by VirtualBox is the most common reason for a failure to boot at this time, and manifests itself as the VM hanging on a black screen during the boot process.

The solution to this predicament is to fix the mismatch between the HAL and the hardware, either by changing the HAL or by changing the emulated components of the machine. Most instances of this problem can be solved by shutting down the VM, opening the System panel of the settings dialog, and setting the Enable IO APIC option. Start the VM once more, and usually the machine will boot as expected.

If you're one of the unlucky few for whom this little checkbox doesn't fix the boot problem, you'll have little choice but to change the HAL. That process falls outside the scope of this tutorial, but there's lots of information online about the process. Most techniques involve replacing the HAL on your running Windows installation, so you'll probably find that you have to make the change on the physical XP box and then re-image the hard drive to create a fresh VHD file. Bear in mind that changing the HAL can result in an un-bootable system, so you might want to consider making a full disk image of the machine using something like **dd** or Clonezilla before you start messing around with Windows' system files.

Final steps

Once you're able to boot the Windows VM, you'll probably get to a familiar looking login screen. Avoid the temptation to press Ctrl+Alt+Del, even if Windows claims you need to. Doing that will send the keystrokes to the Linux host first, possibly shutting it down. Instead you need to send the keys to the XP box inside the VM, either by choosing the option from VirtualBox's Machine menu, or by pressing the VirtualBox "host" key (usually the Right Ctrl key) plus Delete.

Given that we've essentially replaced the machine's motherboard, Windows is almost certain to throw up a prompt telling you that you need to activate the machine. The easiest way to do this is online, and this is the one part of this tutorial where I'm going to recommend actually putting your XP box back onto the internet. Choose to activate over the internet and be ready with your Product Key in case it's requested. If you have problems activating over the internet there is also an option to telephone customer services. We've never had to take this approach, but if you do end up talking to a real person it's probably best to avoid mentioning virtual machines and instead just tell them that you've replaced your motherboard.

You should now be at a small Windows desktop, probably being pestered with the Found New Hardware wizard. Cancel each wizard that pops up until you're back to just the desktop, then move your

