Slackware as a Xen DomU Guest

Introduction

This document explains how to create a guest virtual machine for a Xen environment using HVM (with PV drivers) virtualisation mode. Such a VM could be used on AWS (Amazon Web Services) or with an on-premise Xen setup but has the added advantage that it will still boot into Desktop virtualisation software like VirtualBox, KVM, VMWare as it uses a conventional MBR. There is no requirement to use Grub and we can stick with the familiar LILO.

Installation

First, we must prepare a Slackware install in a virtual machine. You could always do this in Xen itself, however I'm using VirtualBox because it's easier to setup.

Select at a minimum disk sets A, AP, D, K, L and N sets. Install everything. You can try with less if you like, but this is tested and working.

Make an Initrd

Now we must create an initrd. Any initrd. Yes, that's right it doesn't has to contain any modules just exist.

```
# cd /boot
# mkinitrd -c
```

And of course, add it to LILO

```
image = /boot/vmlinuz
  root = /dev/sda1
  label = Linux
  read-only
  initrd = /boot/initrd.gz
```

UUID Root device

Next, we must setup lilo to boot using the UUID of the root partition instead of the device name (/dev/sda1 etc...)

Run the blkid program to list the UUIDs of the partitions, copy the one that matches the above partition, e.g

```
# blkid
/dev/sda1: UUID="43b8f058-1f75-4944-af9a-ee33ecc297aa" BLOCK_SIZE="4096"
TYPE="ext4" PARTUUID="c0baa0b2-01"
```

Add that UUID into your lilo.conf taking care to quote it correctly (the quotes go around the whole thing!)

```
image = /boot/vmlinuz
  root = "UUID=43b8f058-1f75-4944-af9a-ee33ecc297aa"
  label = Linux
  read-only
  initrd = /boot/initrd.gz
```

In /etc/fstab, change /dev/sda1 (or whatever your root partition was) in similar way, but you don't need any quotes this time:

```
UUID=43b8f058-1f75-4944-af9a-ee33ecc297aa     /               ext4
defaults
#/dev/cdrom      /mnt/cdrom       auto        noauto,owner,ro
...
```

After this you may wish to run lilo, check that your system still boots. All we have done is change the root device specification such that it's in terms of UUID, nothing else. This is explained more fully elsewhere.

Configure the kernel

Get ready to recompile the kernel using the current config as a starting point.

```
# cd /usr/src/linux
# zcat /proc/config.gz > .config
# make menuconfig
```

Then select the following kernel options

```
Processor type
 Processor type and features  --->
  [*] Linux guest support  --->
   [*] Xen guest support                 # sets CONFIG_XEN  and several
others.

Device Drivers  --->
 [*] PCI support  --->
  <*>   Xen PCI Frontend (NEW)               # sets
CONFIG_XEN_PCIDEV_FRONTEND
 [*] Block devices  --->
  <*>   Xen virtual block device support (NEW)        # sets
CONFIG_XEN_BLKDEV_FRONTEND
```

```
 SCSI device support  --->
  [*] SCSI low-level drivers  --->
   <*>   XEN SCSI frontend driver        # sets CONFIG_XEN_SCSI_FRONTEND

 -*- Network device support  --->
  <*>   Xen network device frontend driver (NEW)      # sets
CONFIG_XEN_NETDEV_FRONTEND
```

Those selections should give you the following extra kernel options. You may have more or less depending on your exact kernel version (similar options will work with the stock 14.2 kernel). Make sure you have at least the above five kernel options.

```
# cat .config | grep XEN | grep =y
CONFIG_XEN=y
CONFIG_XEN_PV=y
CONFIG_XEN_PV_SMP=y
CONFIG_XEN_DOM0=y
CONFIG_XEN_PVHVM=y
CONFIG_XEN_PVHVM_SMP=y
CONFIG_XEN_512GB=y
CONFIG_XEN_SAVE_RESTORE=y
CONFIG_PCI_XEN=y
CONFIG_XEN_PCIDEV_FRONTEND=y
CONFIG_XEN_BLKDEV_FRONTEND=y
CONFIG_XEN_SCSI_FRONTEND=y
CONFIG_XEN_NETDEV_FRONTEND=y
CONFIG_INPUT_XEN_KBDDEV_FRONTEND=y
CONFIG_HVC_XEN=y
CONFIG_HVC_XEN_FRONTEND=y
CONFIG_XEN_FBDEV_FRONTEND=y
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES_DEFAULT=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XEN_BACKEND=y
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_SYS_HYPERVISOR=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_SWIOTLB_XEN=y
CONFIG_XEN_PRIVCMD=y
CONFIG_XEN_HAVE_PVMMU=y
CONFIG_XEN_EFI=y
CONFIG_XEN_AUTO_XLATE=y
CONFIG_XEN_ACPI=y
CONFIG_XEN_SYMS=y
CONFIG_XEN_HAVE_VPMU=y
```

Recompile the kernel

Compile the kernel with

```
make bzImage
```

Note we don't need the modules unless you formatted rootfs with some obscure filesystem :).

Test the kernel

Copy the new kernel to /boot, e.g.

```
cp arch/x86/boot/bzImage /boot/xen
```

Now create an extra /etc/lilo.conf section for the new kernel. It will be mostly a copy of the other section with the same initrd but different kernel.

```
image = /boot/xen
  root = "UUID=43b8f058-1f75-4944-af9a-ee33ecc297aa"
  label = Xen
  read-only
  initrd = /boot/initrd.gz
```

run lilo

```
# lilo
Warning: LBA32 addressing assumed
Added Linux  +  *
Added Xen  +
One warning was issued.
```

Now test that your system still boots when using the Xen kernel. Nothing much should have changed, but you can now import this virtual machine into a DomU environment, and it should detect and use paravirtualised hardware if available. You'll know if it has succeeded because device node /dev/xvda exists and various files are found in /sys/bus/xen/devices/.

Sources

* Originally written by User Bifferos

howtos, virtualisation, xen, domu