**Diskless booting of dom0 in Xen hypervisor with driver domain installation**

# Introduction

## Overview

This document describes the procedure to create a bootable USB drive for a diskless operating system that will be used as dom0 for Xen hypervisor. Following is the high level outline for creating such USB drive.

* Build kernel and initrd for dom0 with appropriate kernel configuration.
* Update initrd image to boot from RAM using Ubuntu core 13.04 as rootfs with necessary kernel modules.
* Create USB drive to boot using the above kernel and initrd.
* In the booted system, download necessary packages to run xcp/xapi, ssh server and basic text editor.
* Rebuild the rootfs and save the rootfs archive to the USB drive
* Regenerate initrd with the above modified rootfs inside it.
* Boot dom0 with updated initrd.
* Mount disk on the booted system and create virtual machines.

## Assumption

Ubuntu 13.04 (Raring Ringtail) has been used to illustrate the process.

# Booting from pre-built images

If you want to boot from pre-built images then checkout the files in **images/dom0\_image/**

directory and copy all the content to the USB after following section 4.1 and 4.2

# Build a custom kernel

To execute the steps described in this document, we need a machine with Ubuntu 13.04 installed in it. Hereafter, this machine will be called build machine.

On the build machine, login as root user. Install dependencies for building a new kernel

**apt-get install make libncurses5-dev build-essential**

Download source for kernel

**mkdir -p /root/mydir/**

**cd /root/mydir/**

**apt-get source linux-image-3.8.13.6**

Go to the source directory and create desired configuration using this command

**make menuconfig**

Save the default configuration. It will generate a .config file in the current directory. Update the following settings in .config file.

**CONFIG\_ACPI=y**

**CONFIG\_ACPI\_PROCFS=y**

**CONFIG\_HVC\_XEN=y**

**CONFIG\_PCI\_XEN=y**

**CONFIG\_XEN=y**

**CONFIG\_NF\_NAT=y**

**CONFIG\_XEN\_BACKEND=y**

**CONFIG\_XEN\_BALLOON=y**

**CONFIG\_XEN\_BLKDEV\_BACKEND=y**

**CONFIG\_XEN\_BLKDEV\_FRONTEND=y**

**CONFIG\_XEN\_COMPAT\_XENFS=y**

**CONFIG\_XEN\_DEV\_EVTCHN=y**

**CONFIG\_XEN\_DOM0=y**

**CONFIG\_XEN\_FBDEV\_FRONTEND=y**

**CONFIG\_XENFS=y**

**CONFIG\_XEN\_GNTDEV=y**

**CONFIG\_XEN\_MAX\_DOMAIN\_MEMORY=512**

**CONFIG\_XEN\_NETDEV\_BACKEND=y**

**CONFIG\_XEN\_NETDEV\_FRONTEND=y**

**CONFIG\_XEN\_PCIDEV\_BACKEND=y**

**CONFIG\_XEN\_PCIDEV\_FRONTEND=y**

**CONFIG\_XEN\_PRIVILEGED\_GUEST=y**

**CONFIG\_XEN\_SAVE\_RESTORE=y**

**CONFIG\_XEN\_SCRUB\_PAGES=y**

**CONFIG\_XEN\_SYS\_HYPERVISOR=y**

**CONFIG\_NF\_NAT=y**

**ONFIG\_NF\_CONNTRACK\_IPV4=y**

**CONFIG\_NF\_CONNTRACK=y**

**CONFIG\_IP\_NF\_TARGET\_MASQUERADE=y**

**CONFIG\_IP\_NF\_IPTABLES=y**

Build and install the kernel

**make**

**make modules**

**make modules\_install**

**make headers\_install**

**make install**

Assuming that we have built kernel version 3.8.13.6. The kernel modules will be installed in the direcoty **/lib/modules/3.8.13.6/ . The** kernel **vmlinuz-3.8.13.6** and initrd will be created in **/boot/** directory.

# Update initrd image to boot in RAM

We need to update initrd image such that the modules and rootfs required for dom0 are inside initrd so that we don't need to have root fs on disk. Follow these steps to create a new initrd in your current working directory.

### Regenerate initrd with only selected kernel modules.

By default the generated initrd will contain most of the disk and net modules, we will need to rebuild initrd with only required modules.

Update the file **/etc/initramfs-tools/initramfs.conf,** change the paramater **MODULE** to **MODULES=list**.

Update the file **/etc/initramfs-tools/modules,** put the required modules (one per line) in this file. (Please see Appendix for list of required modules).

Create a new initrd in the current directory.

**mkdir /root/mydir**

**cd /root/mydir**

**mkinitramfs -c gzip -o /root/mydir/initrd.img-3.8.13.6 3.8.13.6**

This will create a new file **initrd.img-3.8.13.6** in the current working directory.

Extract and update initrd image to boot from RAM

**mv initrd.img-3.8.13.6 initrd.img-3.8.13.6.gz**

**gunzip initrd.img-3.8.13.6.gz**

**mkdir initrd\_extract**

**cd initrd\_extract**

**cpio -id < ../initrd.img-3.8.13.6**

Create a script with name **ramboot** in scripts directory. (See the last section for the contents of this script). Make ramboot script executable and add required binaries**.**

**chmod a+x scripts/ramboot**

**cp /bin/tar bin/**

**cp /bin/gzip bin/**

**cp /sbin/mke2fs sbin/**

**cp /usr/lib/initramfs-tools/bin/\* bin/**

Add ubuntu core to initrd

**mkdir OS**

**wget -P OS http://cdimage.ubuntu.com/ubuntu-core/releases/13.04/release/ubuntu-core-13.04-core-amd64.tar.gz**

Generate new initrd image

**find . | cpio --quiet -H newc -o | gzip -9 -n > /root/mydir/initrd.img-3.8.13.6**

# Prepare the USB flash drive

## Create a parition in USB drive

Check dmesg command to find the USB device name after plugging in the USB. Assuming USB is on /dev/sdb , execute the following commands.

**fdisk /dev/sdb**

Command (m for help): **o**

Command (m for help): **n**

Partition type:

p primary (0 primary, 0 extended, 4 free)

e extended

Select (default p): **[Enter]**

Using default response p

Partition number (1-4, default 1): **[Enter]**

Using default value 1

First sector (2048-61767679, default 2048): **[Enter]**

Using default value 2048

Last sector, +sectors or +size{K,M,G} (2048-61767679, default 61767679): [Enter]

Using default value 61767679

Command (m for help): **t**

Selected partition 1

Hex code (type L to list codes): **c**

Command (m for help): a

Partition number (1-4): 1

Command (m for help): **w**

The partition table has been altered!

Now, we need to format the partition

**mkfs.vfat -F 32 -n "diskless\_dom0" /dev/sdb1**

## Add required files to USB disk

Install required packages on build machine.

**apt-get install xcp-xapi tboot grub2**

You will need to have the SINIT file 3rd\_gen\_i5\_i7\_SINIT\_67.BIN downloaded to /boot/ from [**http://software.intel.com/en-us/articles/intel-trusted-execution-technology**](http://software.intel.com/en-us/articles/intel-trusted-execution-technology)

Copy tboot, kernel, initrd and sinit files to the USB after mounting it

**mkdir -p /media/usb/**

**mount -o rw /dev/sdb1 /media/usb**

**mkdir -p /media/usb/boot/**

**mkdir -p /media/usb/scripts/**

**mkdir -p /media/usb/OS/**

**cp /boot/tboot.gz /media/usb/boot/**

**cp /boot/xen-4.2-amd64.gz /media/usb/boot/**

**cp /boot/vmlinuz-3.8.13.6 /media/usb/boot/**

**cp /root/mydir/initrd.img-3.8.13.6 /media/usb/boot/**

**cp /boot/3rd\_gen\_i5\_i7\_SINIT\_67.BIN /media/usb/boot/**

Create a script **create\_rootfs\_archive.sh, setup\_dom0\_config.sh** in **/media/usb/scripts/** (Please check appendix for the content of these scripts). Make them executable

**chmod a+x /media/usb/scripts/\***

## 4.2 Install grub on USB

We need to make this USB drive bootable using grub2.

Now identify the USB device on which the USB setup is being done and install grub on it using the following command (assuming /dev/sdb to be your USB device)

**grub-install --recheck /dev/sdb**

Pay attention to the output. It will tell you how Grub identifies /dev/sdb, we’ll need it later. In this case it’s hd1.

Copy grub folder to USB

**cp -r /boot/grub/ /media/usb/boot/**

Some grub specific configuration

**grub**

**grub>root (hd1,0)**

**grub>setup (hd1)**

**grub> quit**

If you see error like this after doing the above steps

**Checking if "/boot/grub/stage1" exists... no**

**Checking if "/grub/stage1" exists... no**

**Error 15: File not found**

then copy the folder **images/dom0\_image/boot/grub/** from svn to **/media/usb/boot/** and redo the above steps of setting **root (hd1,0)** and **setup (hd1)**

In the USB’s boot directory create a file menu.lst (path **/media/usb/boot/grub/menu.lst**) with following contents

**color white/blue blue/yellow white/black yellow/black**

**timeout 15**

**default 0**

**fallback 1**

**title diskless dom0**

**root (hd0,0)**

**echo 'Loading Tboot 1.7.0 ...'**

**kernel --type=multiboot /boot/tboot.gz /boot/tboot.gz logging=serial,vga,memory**

**echo 'Loading Xen 4.2-amd64 ...'**

**module /boot/xen-4.2-amd64.gz /boot/xen-4.2-amd64.gz dom0\_mem=2048M,max:2048M placeholder iommu=force**

**echo 'Loading vmlinuz-3.8.13.6 ...'**

**module /boot/vmlinuz-3.8.13.6 /boot/vmlinuz-3.8.13.6 root=/dev/sdb1 boot=ramboot ramdisk\_size=3072000K elevator=deadline**

**echo 'Loading initial ramdisk initrd.img-3.8.13.6...'**

**module /boot/initrd.img-3.8.13.6 /boot/initrd.img-3.8.13.6**

**echo 'Loading sinit 3rd\_gen\_i5\_i7\_SINIT\_67.BIN ...'**

**module /boot/3rd\_gen\_i5\_i7\_SINIT\_67.BIN /boot/3rd\_gen\_i5\_i7\_SINIT\_67.BIN**

Creation of boot USB is now complete. Unmout it

**umount /dev/sdb1**

# Prepare TXT machine

1. There are two settings in BIOS that need to be enabled : TPM Security and TXT Execution.

2. In BIOS (use F2), go to Security -> TPM Security and click the check box to enable TPM Security and click on Apply.

3. This enables TPM Security but TPM is still in deactivated state. Select the Activate radio button and click on Apply to activate TPM. Click Exit.

4. Get to the BIOS again and go to Virtualization -> TXT Execution/Technology and click on the check-box to enable it. Click on Apply and then Exit to save the settings.

# Boot with the prepared USB and update rootfs

Boot the TXT machine using this USB. Default root password is set to blank.

After the system is booted, setup networking on booted system. Put appropriate values for network interface and IP addresses.

**ip link set dev eth0 up**

**ip addr add 10.35.34.17/24 dev eth0**

**ip route add default via 10.35.34.254**

**echo "nameserver 192.168.0.4" > /etc/resolv.conf**

Install openssh server

**apt-get update**

**apt-get install openssh-server**

SSH to this diskless OS from some other machine and update **/etc/apt/sources.list** (check Appendix for sources.list) and install necessary packages using **apt.** Set bridge mode for networking when the installer prompts for choosing a mode.

**apt-get install net-tools iputils-ping iptables openssh-client nano sysklogd cron unzip libxml2 libpython2.7**

**apt-get install xcp-xapi tpm-tools tpm-tools-dbg trousers libtspi1 libtspi-dev**

**apt-get clean**

**apt-get autoclean**

Set desired root password

**passwd**

Once all these are installed, check with tpm\_version and tpm\_selftest to see if you can access the TPM. The output should be something like this :

**tpm\_version**

TPM 1.2 Version Info:

Chip Version: 1.2.37.19

Spec Level: 2

Errata Revision: 3

TPM Vendor ID: ATML

TPM Version: 01010000

Manufacturer Info: 41544d4c

**tpm\_selftest**

TPM Test Results: 0000

Trousers should be running for this. Check with /etc/init.d/trousers status or ps ax | grep tcsd. Restart using /etc/init.d/trousers restart.

If TPM is working, take ownership using this command

**tpm\_takeownership -z -y**

Then run this command

**tpm\_getpubek -z**

see if you get the public encryption key. If you are not able to get the key or take ownership of TPM, then TPM needs to be reset. You can do it either by using the tpm\_clear command ( needs reboot ) or by resetting TPM is BIOS. If everything is OK, then it means TPM is setup correctly.

Add required conf file required for xen and copy rpcore and xapi proxy to /root/

**mkdir /usr/etc/**

**touch /usr/etc/xenhost.conf**

**chmod 755 /usr/etc/xenhost.conf**

**cd /root/**

**scp -r root@<build-server-ip:path>/RPCore/ /root/**

**scp -r root@<build-server-ip:path>/xapi\_proxy/ /root/**

Add the following to the /etc/bash.bashrc file to set the LD\_LIBRARY\_PATH :

**export LD\_LIBRARY\_PATH=/root/RPCore/lib:/root/xapi\_proxy/:$LD\_LIBRARY\_PATH**

Mount USB and copy script for dom0 configuration

**mount /bootflash**

**cp /bootflash/scripts/setup\_dom0\_config.sh /etc/**

**chmod a+x /etc/setup\_dom0\_config.sh**

update rc.local, add the following line

**sh /etc/setup\_dom0\_config.sh**

Create a file for containing whitelisted hashes

**touch /root/hash\_whitelist.txt**

Put one hash per line in this file

Install nova plugins. Create temporary files/directories:

**NOVA\_ZIPBALL=$(mktemp)**

**NOVA\_SOURCES=$(mktemp -d)**

Get the source from github. The example assumes the master branch is used, please amend the URL to match the version being used:

**wget -qO "$NOVA\_ZIPBALL" https://github.com/openstack/nova/archive/master.zip**

**unzip "$NOVA\_ZIPBALL" -d "$NOVA\_SOURCES"**

Copy the plugins to the xapi plugin directory

**PLUGINPATH=$(find $NOVA\_SOURCES -path '\*/xapi.d/plugins' -type d -print)**

**cp -r $PLUGINPATH/\* /usr/lib/xcp/plugins/.**

Remove the temporary files/directories:

**rm "$NOVA\_ZIPBALL"**

**rm -rf "$NOVA\_SOURCES"**

We need to make some changes in plugin to make it work with our OS. Open the file **/usr/lib/xcp/plugins/xenhost** and go to the function **def host\_data(self, arg\_dict)**Comment the following two lines

**config = \_get\_config\_dict()**

**ret\_dict.update(config)**

Some xcp related configuration and cleanup

**sed -i 's/TOOLSTACK=.\*/TOOLSTACK=xapi/g' /etc/default/xen**

**sed -i -e 's/xend\_start$/#xend\_start/' -e 's/xend\_stop$/#xend\_stop/' /etc/init.d/xen**

**update-rc.d xendomains disable**

**rm -rf /usr/share/man/\***

**rm -rf /usr/share/doc/\***

**rm -rf /usr/src/\***

We need to now save this rootfs and copy xen tar to USB

**mount /bootflash**

**cd /bootflash**

**cd /bootflash/scripts/**

**./create\_rootfs\_archive.sh**

**cp /boot/xen-4.2-amd64.gz /bootflash/boot/**

Reboot the system and go to normal Ubuntu (build machine) this time.

Now we need to put this new rootfs inside initrd. Go to the previsouly extracted initrd folder

Mount the USB and copy the new rootfs to extracted initrd

**mount -o rw /dev/sdb1 /media/usb**

**cd /root/mydir/initrd\_extract**

**rm -f OS/\***

**cp /media/usb/OS/\*tar.gz OS/**

Generate new initrd and copy it to USB drive**.** This initrd contains the updated rootfs

**find . | cpio --quiet -H newc -o | gzip -9 -n > /root/mydir/initrd.img-3.8.13.6**

**cp /root/mydir/initrd.img-3.8.13.6 /media/usb/boot/**

We need to create a file that will contain dom0 configuration related to networking and LVM. Add the following contents with appropriate values to the file **/media/usb/config/dom0\_settings.txt**

**IP\_ADDRESS=10.35.34.14**

**NETMASK=255.255.255.0**

**GATEWAY=10.35.34.254**

**BROADCAST=10.35.34.255**

**DNS\_NAMESERVER=192.168.0.4**

**DNS\_SEARCH=gslab.com**

**RPCORE\_PORT=16005**

**SR\_DEVICE=/dev/mapper/ VolumeGroup-LocalStorage NOVA\_VM\_IMAGE\_PATH=OS/nova\_vm.xvda**

Create LVM on the host machine and copy the nova vm image to the USB in OS folder. The creation of nova vm image has been covered in the document containing the setup of nova-compute VM. For creation of LVM, follow these steps:

verify that you have a LVM partition using this command

**sudo fdisk -l**

This should list a partition of type “Linux LVM”. If you don’t see a partition and you have free space on the disk, create a new partition of type “Linux LVM” (8e). If you have partition of type “Linux LVM” follow the steps below

create a physical volumes assuming the partition is /dev/sda4

**pvcreate /dev/sda4**

Now create volume group and logical volumes

**vgcreate VolumeGroup /dev/sda4**

**lvcreate --size 199G -n LocalStorage VolumeGroup**

use the following command to get the info of the device for this logical volume

**fdisk -l**

You will seen an entry like **/dev/mapper/VolumeGroup-LocalStorage**, add this to the config file mentioned above (**/media/usb/config/dom0\_settings.txt)** for the parameter **SR\_DEVICE**

Creation of USB drive is now complete. We can now use this usb to boot diskless dom0 on any machine.

# Boot dom0 in RAM with prepared USB drive

Plug the USB to the computer and boot from this USB. You can now run xe commands on the command prompt. The root password for dom0 is intelrp.

RPCore and xapi proxy will start automatically on dom0 on boot. After booting, modify the IP addresses of nova-compute, dom0 and openstack controller in the file **/etc/nova/nova.conf** and restart nova-compute service, please refer to the Openstack installtion document for more information about configuring nova.conf. This machine will now become a part of openstack.

You may install x11-utils if you wish to view VM installation progress using VNC

**apt-get install x11-utils**

## Setting up Ubuntu 12.04 LTS as domU:

This document assumes the following :

* The server is running Ubuntu 13.04 and is booted in Xen 4.2.
* The server uses xcp-xapi/xe toolstack

Please follow the instructions below to setup a Ubuntu 12.04 VM using xe :

1. Check if xcp-xapi service is running and xe toolstack is available as follows :

*root@aguna:/home# service xcp-xapi status*

*\* xapi is running*

*root@aguna:/home#*

*root@aguna:/home# xe vm-list*

*uuid ( RO) : ed9de35c-142d-8543-c9c9-5ea5b4bad988*

*name-label ( RW): Control domain on host: aguna*

*power-state ( RO): running*

*root@aguna:/home#*

2. Check if there are any templates available :

*root@aguna:~# xe template-list*

If there are no templates then invoke the following command to create templates :

*root@aguna:~# /usr/lib/xcp/lib/create\_templates*

3. Create VM

a) Create VM with Ubuntu Lucid template and later we will set it to Precise(Ubuntu 12.04)

*root@aguna:~# VM\_ID=$(xe vm-install template=Ubuntu\ Lucid\ Lynx\ 10.04\ \(64-bit\) sr-name-label=<sr-name> new-name-label=<vmname>)*

**Note:** Specify the correct *sr-name-label*  given in 2 above or check with *xe sr-list.*

b) Get UUID of xenbr0 network

*root@aguna:~# NET\_ID=$(xe network-list bridge=xenbr0 --minimal)*

c) Create virtual interface with xenbr0 network

*root@aguna:~# xe vif-create mac=random device=0 network-uuid=$NET\_ID vm-uuid=$VM\_ID*

d) Set the following vm-params :

*root@aguna:~# xe vm-param-set uuid=$VM\_ID other-config:install-repository="http://us.archive.ubuntu.com/ubuntu"*

*root@aguna:~# xe vm-param-set uuid=$VM\_ID PV-args="netcfg/get\_hostname=Ubuntu \ console=hvc0 debian-installer/locale=en\_US console-setup/layoutcode=us \ console-setup/ask\_detect=false interface=eth0 netcfg/disable\_dhcp=false \ preseed/file=/root/preseed-ubuntu-12.04.cfg"*

*root@aguna:~# xe vm-param-set uuid=${VM\_ID} other-config:disable\_pv\_vnc=1*

**Note:** Get the preseed-ubuntu-12.04.cfg from Annex 1 at the end of this document.

e) Set disk size for new VM

**Note:** Here it is being set to 8 GB - please check available size and chose the appropriate size for your deployment

*root@aguna:~# xe vm-param-set other-config:disks="<provision><disk device=\"0\" \ size=\"8589934592\" sr=\"\" bootable=\"true\" type=\"system\"/></provision>" \ uuid=$VM\_ID*

f) Set the memory parameters

*root@aguna:~# xe vm-param-set memory-static-max=1024000000 uuid=$VM\_ID*

*root@aguna:~# xe vm-param-set memory-dynamic-max=1024000000 uuid=$VM\_ID*

*root@aguna:~# xe vm-param-set memory-dynamic-min=512000000 uuid=$VM\_ID*

*root@aguna:~# xe vm-param-set memory-static-min=512000000 uuid=$VM\_ID*

g) Start VM

*root@dom0:~# xe vm-start uuid=$VM\_ID*

h) Access console and continue installation

*root@dom0:~# xe vm-lis*t

It it shows vm in running state then we can access console of VM and continue Ubuntu Precise installation. Get domain-id from the following command :

*root@dom0:/# xe vm-param-list uuid=$VM\_ID | grep dom-id*

Access the console using either :

*root@dom0:/# /usr/lib/xen-4.2/bin/xenconsole <domain-id>*

OR

*root@dom0:/# xe console vm=<VM\_NAME>*

Note : You can get the *<VM\_NAME>* from *xe vm-list.*

Then continue with OS installation.

Once the setup is done and the VM boots up, login with the user *ubuntu* and password as *password.*

# Appendix

## ramboot script

**mountroot()**

**{**

**if [ ! -d /OS ]; then**

**panic "**

**ERROR: Could not find "OS" directory on ${ROOT}**

**"**

**fi**

**echo -n "Creating ramdisk on /dev/ram1: "**

**mke2fs /dev/ram1**

**mount -t ext2 /dev/ram1 ${rootmnt}**

**echo "Created ramdisk in /dev/ram1"**

**echo "Extracting OS archive(s)"**

**for i in `ls -1 /OS/\*.tar.gz`; do**

**echo -n " Extracting $i:"**

**tar zxfp $i -C ${rootmnt}**

**echo "Extracted OS archive"**

**done**

**cp -r /lib/ ${rootmnt}**

**mkdir ${rootmnt}/bootflash**

**echo "/dev/ram1 / ext2 rw 0 0" >> ${rootmnt}/etc/fstab**

**echo "${ROOT} /bootflash vfat noauto 0 0" >> ${rootmnt}/etc/fstab**

**echo "diskless-dom0" > ${rootmnt}/etc/hostname**

**sed -i 's/root:\\*:/root::/g' ${rootmnt}/etc/shadow**

**echo "" > ${rootmnt}/etc/motd.tail**

**echo " \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*" >> ${rootmnt}/etc/motd.tail**

**echo " \* Diskless dom0 using Ubuntu core 13.04 \*" >> ${rootmnt}/etc/motd.tail**

**echo " \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*" >> ${rootmnt}/etc/motd.tail**

**echo "Done"**

**echo "Booting system ..."**

**}**

## Module list

**ahci**

**arc4**

**auth\_rpcgss**

**atl1c**

**blktap**

**bridge**

**coretemp**

**cryptd**

**e1000e**

**ext2**

**fscache**

**ghash\_clmulni\_intel**

**gpio\_ich**

**hid**

**hid\_generic**

**i2c\_algo\_bit**

**libahci**

**llc**

**lp**

**lpc\_ich**

**mei**

**microcode**

**nls\_iso8859\_1**

**openvswitch**

**parport**

**parport\_pc**

**ppdev**

**sdhci**

**sdhci\_pci**

**serio\_raw**

**stp**

**tpm\_tis**

**usbhid**

**usb\_storage**

**wmi**

**ip\_tables**

**iptable\_raw**

**iptable\_filter**

**iptable\_mangle**

**iptable\_nat**

**iptable\_security**

## create\_rootfs\_archive.sh

**#!/bin/bash**

**cd /bootflash/scripts**

**mkdir -p ../OS/**

**rm /var/cache/apt/\*.bin**

**rm -rf /var/lib/apt/lists/\***

**tar \**

**--exclude="/bootflash" \**

**--exclude="/etc/udev/rules.d/\*-persistent-\*" \**

**--exclude="/home/\*" \**

**--exclude="/lib/firmware/\*" \**

**--exclude="/lib/modules/\*" \**

**--exclude="/lost+found" \**

**--exclude="/media/\*" \**

**--exclude="/mnt/\*" \**

**--exclude="/proc/\*" \**

**--exclude="/scripts" \**

**--exclude="/sys/\*" \**

**--exclude="/tmp/\*" \**

**-zcvpf ubuntu-core-13.04-core-amd64.custom.tar.gz /**

**mv ubuntu-core-13.04-core-amd64.custom.tar.gz ../OS/**

## sources.list

**deb http://in.archive.ubuntu.com/ubuntu/ raring main restricted**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring main restricted**

**deb http://in.archive.ubuntu.com/ubuntu/ raring-updates main restricted**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring-updates main restricted**

**deb http://in.archive.ubuntu.com/ubuntu/ raring universe**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring universe**

**deb http://in.archive.ubuntu.com/ubuntu/ raring-updates universe**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring-updates universe**

**deb http://in.archive.ubuntu.com/ubuntu/ raring multiverse**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring multiverse**

**deb http://in.archive.ubuntu.com/ubuntu/ raring-updates multiverse**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring-updates multiverse**

**deb http://in.archive.ubuntu.com/ubuntu/ raring-backports main restricted universe multiverse**

**deb-src http://in.archive.ubuntu.com/ubuntu/ raring-backports main restricted universe multiverse**

**deb http://security.ubuntu.com/ubuntu raring-security main restricted**

**deb-src http://security.ubuntu.com/ubuntu raring-security main restricted**

**deb http://security.ubuntu.com/ubuntu raring-security universe**

**deb-src http://security.ubuntu.com/ubuntu raring-security universe**

**deb http://security.ubuntu.com/ubuntu raring-security multiverse**

**deb-src http://security.ubuntu.com/ubuntu raring-security multiverse**

## setup\_dom0\_config.sh

**#!/bin/bash**

**#!/bin/bash**

**# The script will do the following ############################################################**

**# 1. Read the config file and Setup dom0 networking**

**# 2. Check if we are in dom 0 and check if we have xe/xl running, then proceed to 2. Else exit.**

**# 3. If dom0 is booted, then check if PCI devices are hidden (since they will be launched in domU).**

**# If not, hide them and proceed to 3 if successful. Else exit.**

**# 4. Check if there are any previous driver domain VMs running and destroy them.**

**# Poll till the VM/s are destroyed, then proceed to next step. Else exit.**

**# 5. Check for available image in path provided as input to this script. If path and image file**

**# exists, then proceed to next step. Else exit.**

**# 6. Import vm ( from the path provided in 4.) using xe vm-import and then poll status of the VM.**

**# If successful, then we have isolated domain driver + nova + webservice**

**INPUT\_CONFIG\_FILE="/bootflash/config/dom0\_settings.txt"**

**LOG\_DIR="/var/log/dom0\_config/"**

**TMP\_INTERFACES\_FILE="/tmp/interfaces"**

**INTERFACES\_FILE="/etc/network/interfaces"**

**INTERFACES\_FILE\_BAK="/etc/network/interfaces.bak"**

**IP\_ADDR=**

**NETMASK=**

**GATEWAY=**

**BROADCAST=**

**DNS\_NAMESERVER=**

**DNS\_SEARCH=**

**SR\_DEVICE=**

**NOVA\_VM\_IMAGE\_PATH=**

**sleep 60**

**umount /bootflash/**

**mount /bootflash/**

**# Setup log file**

**if [ -d $LOG\_DIR ]; then**

**touch $LOG\_DIR**

**else**

**mkdir -p $LOG\_DIR**

**fi**

**LOG="$LOG\_DIR/idu\_`date +%m\_%d\_%y\_%H\_%M\_%S`.log"**

**if [ ! -f $INPUT\_CONFIG\_FILE ]; then**

**echo "File not found: $INPUT\_CONFIG\_FILE" >> $LOG**

**exit 1**

**fi**

**. $INPUT\_CONFIG\_FILE**

**cat > $TMP\_INTERFACES\_FILE <<INTERFACE\_INFO**

**auto lo**

**iface lo inet loopback**

**auto eth0**

**auto xenbr0**

**iface xenbr0 inet static**

**bridge\_ports eth0**

**address $IP\_ADDRESS**

**netmask $NETMASK**

**gateway $GATEWAY**

**broadcast $BROADCAST**

**dns-nameservers $DNS\_NAMESERVER**

**dns-search $DNS\_SEARCH**

**INTERFACE\_INFO**

**mv $INTERFACES\_FILE $INTERFACES\_FILE\_BAK**

**mv $TMP\_INTERFACES\_FILE $INTERFACES\_FILE**

**export IP\_ADDRESS=$IP\_ADDRESS**

**export RPCORE\_IPADDR=$IP\_ADDRESS**

**export RPCORE\_PORT=$RPCORE\_PORT**

**NOVA\_VM\_IMAGE\_PATH=/bootflash/$NOVA\_VM\_IMAGE\_PATH**

**service networking restart**

**export LD\_LIBRARY\_PATH=/root/RPCore/lib:/root/xapi\_proxy/:$LD\_LIBRARY\_PATH**

**cp -r /root/RPCore/rptmp/ /tmp/**

**cd /root/RPCore/bin/debug/**

**./rpcorex &**

**cd /root/xapi\_proxy/**

**chmod a+x xapi\_proxy.py**

**nohup python xapi\_proxy.py &**

**LSPCI=`which lspci`**

**INFO=1**

**I\_PATH=$NOVA\_VM\_IMAGE\_PATH**

**pci\_pt() {**

**LIBXL\_PATH="/usr/lib/xen-4.2/"**

**dD\_ID=$1**

**if [ -z "$dD\_ID" ];**

**then**

**echo "Please provide driver domain id\n"**

**exit 1**

**else**

**echo "Checking the state for DD now ... \n"**

**R\_dD\_ID=`xe vm-list name-label=$dD\_ID power-state=running --minimal`**

**if [ -n $R\_dD\_ID ]**

**then**

**echo "DD is in running state. Proceeding with PCI Passthrough steps ..."**

**else**

**echo "DD is provided ($dD\_ID) is not in running state."**

**echo "Please bring the DD in running state and try again. Exiting now ..."**

**exit 1**

**fi**

**fi**

**if [ -z "$LSPCI" ];**

**then**

**echo "Please install pci utils using the following command"**

**echo "sudo apt-get install pciutils"**

**exit 1**

**fi**

**unlink /usr/sbin/xl > /dev/null**

**ln -s $LIBXL\_PATH/bin/xl /usr/sbin/**

**PCI\_LIST=`$LSPCI | grep -ivE '(sata|network|smbus|isa)' | awk -F' ' '{ print $1}'`**

**for pci\_id in $PCI\_LIST**

**do**

**echo "xl pci-assignable-add $pci\_id"**

**xl pci-assignable-add $pci\_id**

**echo "xl pci-attach $dD\_ID $pci\_id"**

**xl pci-attach $dD\_ID $pci\_id**

**done**

**}**

**# 1. Check if we are in dom 0 and check if we have xe/xl running, then proceed to 2. Else exit.**

**if [ -f /proc/xen/capabilities ];**

**then**

**is\_dom0=`cat /proc/xen/capabilities`**

**if [ $is\_dom0 = 'control\_d' ];**

**then**

**echo "We are in dom0. Checking for xe ... " >> $LOG**

**xe vm-list 1> /dev/null 2> /dev/null**

**is\_xe=`echo $?`**

**if [ $is\_xe != 0 ]**

**then**

**echo "xe not running. Checking for xl ... " >> $LOG**

**xl list 1> /dev/null 2> /dev/null**

**is\_xl=`echo $?`**

**echo "is\_xl is $is\_xl\n" >> $LOG**

**if [ $is\_xl != 0 ]**

**then**

**echo "xe/xl not running. Exiting now ... " >> $LOG**

**exit 1**

**else**

**echo "xl present. Proceeding now ... " >> $LOG**

**fi**

**else**

**echo "xe present. Proceeding now ... " >> $LOG**

**fi**

**else**

**echo "We are not in dom0. Exiting now..." >> $LOG**

**exit 1**

**fi**

**fi**

**echo "Checking for logical volume to create SR on dom0" >> $LOG**

**echo "Logical Volume for domU available at $SR\_DEVICE. Proceeding to create SR now ... " >> $LOG**

**xe sr-create type=ext name-label=DD\_domU\_SR device-config:device=$SR\_DEVICE**

**is\_SR=`echo $?`**

**DD\_SRID=`xe sr-list name-label=DD\_domU\_SR --minimal`**

**POOL\_ID=`xe pool-list --minimal`**

**xe pool-param-set default-SR=$DD\_SRID uuid=$POOL\_ID**

**if [ $is\_SR -eq '0' ]**

**then**

**echo "DD domU SR created succeesfully. SR UUID is $DD\_SRID. Proceeding now ... " >> $LOG**

**else**

**echo "DD domU SR creation failed. Cannot proceed with out SR. Exiting now ... " >> $LOG**

**exit 1**

**fi**

**# 2. If dom0 is booted, then check if PCI devices are hidden (since they will be launched in domU). If not, hide them and proceed to 3 if successful. Else exit.**

**PCI\_PASS=0**

**PCI\_NUM=`$LSPCI | grep -ivE '(sata|network|smbus|isa)' | awk -F' ' '{ print $1}' | wc -l`**

**if [ $PCI\_NUM -gt 4 ]**

**then**

**echo "PCI devices available in dom0. Need to passthrough to domU." >> $LOG**

**PCI\_PASS=1**

**else**

**echo "PCI device passthrough for dom0 done." >> $LOG**

**fi**

**# 3. Check if there are any previous driver domain VMs running and destroy them. Poll till the VM/s are destroyed, then proceed to 4. Else exit.**

**DD\_UUID=`xe vm-list name-label=Ubuntu\_Compute --minimal`**

**if [ -z $DD\_UUID ]**

**then**

**echo "No driver domain on this server. Proceeding now ... " >> $LOG**

**else**

**echo "Driver domain with UUID = $DD\_UUID present. Shutting down now ... " >> $LOG**

**xe vm-shutdown uuid=$DD\_UUID**

**echo "Destroying driver domain with UUID = $DD\_UUID now ... " >> $LOG**

**xe vm-destroy uuid=$DD\_UUID**

**is\_DD=`echo $?`**

**sleep 10**

**if [ $is\_DD -eq '0' ]**

**then**

**echo "Previous Driver domain deleted. Proceeding now..." >> $LOG**

**else**

**echo "Driver domain was not deleted. Please delete the driver domain manually and re-run this script. Exiting now... " >> $LOG**

**exit 1**

**fi**

**fi**

**DD\_STATUS=0**

**if [ -f $I\_PATH ]**

**then**

**echo "Importing domU VM now ... " >> $LOG**

**SR\_ID=`xe sr-list name-label=DD\_domU\_SR --minimal`**

**NDD\_UUID=`xe vm-import filename=$I\_PATH sr-uuid=$SR\_ID`**

**xe vm-start uuid=$NDD\_UUID**

**echo "domU VM import successful. Checking for status ... " >> $LOG**

**while [ $DD\_STATUS -eq 0 ]**

**do**

**is\_RUN=`xe vm-list uuid=$NDD\_UUID power-state=running --minimal`**

**if [ -z $is\_RUN ]**

**then**

**echo "domU still not in running state. Wait for 5 secs before retrying ... " >> $LOG**

**sleep 5**

**else**

**echo "domU up and running. UUID is $NDD\_UUID" >> $LOG**

**DD\_STATUS=1**

**fi**

**done**

**else**

**echo "Path provided is not a file. Please provide a correct file-path and retry again. Exiting now ... " >> $LOG**

**exit 1**

**fi**

**# Check if PCI\_PASS=1 and add PCI devices to the newly created DD**

**if [ $PCI\_PASS -eq 1 ]**

**then**

**echo "PCI devices still with dom0. Need to pass these to DD with id $NDD\_UUID. Doing that now ... " >> $LOG**

**pci\_pt Ubuntu\_Compute >> $LOG**

**echo "PCI devices added successfully" >> $LOG**

**else**

**echo "PCI devices already hidden. All setup done. Exiting now." >> $LOG**

**fi**

**echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Nova vm launched successfully \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" >> $LOG**

**mkdir -p /image\_tmpfs**

**mount -t tmpfs -o size=2G,mode=700 tmpfs /image\_tmpfs**

**9.6 preseed-ubuntu-12.04.cfg file :**

**#### Contents of the preconfiguration file (for squeeze)**

**### Localization**

**# Preseeding only locale sets language, country and locale.**

**d-i debian-installer/locale string en\_US**

**# The values can also be preseeded individually for greater flexibility.**

**#d-i debian-installer/language string en**

**#d-i debian-installer/country string NL**

**#d-i debian-installer/locale string en\_GB.UTF-8**

**# Optionally specify additional locales to be generated.**

**#d-i localechooser/supported-locales en\_US.UTF-8, nl\_NL.UTF-8**

**# Keyboard selection.**

**# Disable automatic (interactive) keymap detection.**

**d-i console-setup/ask\_detect boolean false**

**#d-i keyboard-configuration/modelcode string pc105**

**d-i keyboard-configuration/layoutcode string us**

**# To select a variant of the selected layout (if you leave this out, the**

**# basic form of the layout will be used):**

**#d-i keyboard-configuration/variantcode string dvorak**

**### Network configuration**

**# Disable network configuration entirely. This is useful for cdrom**

**# installations on non-networked devices where the network questions,**

**# warning and long timeouts are a nuisance.**

**#d-i netcfg/enable boolean false**

**# netcfg will choose an interface that has link if possible. This makes it**

**# skip displaying a list if there is more than one interface.**

**#d-i netcfg/choose\_interface select auto**

**# To pick a particular interface instead:**

**d-i netcfg/choose\_interface select eth0**

**# If you have a slow dhcp server and the installer times out waiting for**

**# it, this might be useful.**

**#d-i netcfg/dhcp\_timeout string 60**

**# If you prefer to configure the network manually, uncomment this line and**

**# the static network configuration below.**

**#d-i netcfg/disable\_autoconfig boolean true**

**# If you want the preconfiguration file to work on systems both with and**

**# without a dhcp server, uncomment these lines and the static network**

**# configuration below.**

**#d-i netcfg/dhcp\_failed note**

**#d-i netcfg/dhcp\_options select Configure network manually**

**# Static network configuration.**

**#d-i netcfg/get\_nameservers string 192.168.1.1**

**#d-i netcfg/get\_ipaddress string 192.168.1.42**

**#d-i netcfg/get\_netmask string 255.255.255.0**

**#d-i netcfg/get\_gateway string 192.168.1.1**

**#d-i netcfg/confirm\_static boolean true**

**# Any hostname and domain names assigned from dhcp take precedence over**

**# values set here. However, setting the values still prevents the questions**

**# from being shown, even if values come from dhcp.**

**d-i netcfg/get\_hostname string unassigned-hostname**

**d-i netcfg/get\_domain string unassigned-domain**

**# Disable that annoying WEP key dialog.**

**d-i netcfg/wireless\_wep string**

**# The wacky dhcp hostname that some ISPs use as a password of sorts.**

**#d-i netcfg/dhcp\_hostname string radish**

**# If non-free firmware is needed for the network or other hardware, you can**

**# configure the installer to always try to load it, without prompting. Or**

**# change to false to disable asking.**

**#d-i hw-detect/load\_firmware boolean true**

**### Network console**

**# Use the following settings if you wish to make use of the network-console**

**# component for remote installation over SSH. This only makes sense if you**

**# intend to perform the remainder of the installation manually.**

**#d-i anna/choose\_modules string network-console**

**#d-i network-console/password password r00tme**

**#d-i network-console/password-again password r00tme**

**### Mirror settings**

**# If you select ftp, the mirror/country string does not need to be set.**

**#d-i mirror/protocol string ftp**

**d-i mirror/country string manual**

**d-i mirror/http/hostname string archive.ubuntu.com**

**d-i mirror/http/directory string /ubuntu**

**d-i mirror/http/proxy string**

**# Alternatively: by default, the installer uses CC.archive.ubuntu.com where**

**# CC is the ISO-3166-2 code for the selected country. You can preseed this**

**# so that it does so without asking.**

**#d-i mirror/http/mirror select CC.archive.ubuntu.com**

**# Suite to install.**

**#d-i mirror/suite string squeeze**

**# Suite to use for loading installer components (optional).**

**#d-i mirror/udeb/suite string squeeze**

**# Components to use for loading installer components (optional).**

**#d-i mirror/udeb/components multiselect main, restricted**

**### Clock and time zone setup**

**# Controls whether or not the hardware clock is set to UTC.**

**d-i clock-setup/utc boolean true**

**# You may set this to any valid setting for $TZ; see the contents of**

**# /usr/share/zoneinfo/ for valid values.**

**d-i time/zone string US/Pacific**

**# Controls whether to use NTP to set the clock during the install**

**d-i clock-setup/ntp boolean true**

**# NTP server to use. The default is almost always fine here.**

**#d-i clock-setup/ntp-server string ntp.example.com**

**### Partitioning**

**## Partitioning example**

**# If the system has free space you can choose to only partition that space.**

**# This is only honoured if partman-auto/method (below) is not set.**

**# Alternatives: custom, some\_device, some\_device\_crypto, some\_device\_lvm.**

**#d-i partman-auto/init\_automatically\_partition select biggest\_free**

**# Alternatively, you may specify a disk to partition. If the system has only**

**# one disk the installer will default to using that, but otherwise the device**

**# name must be given in traditional, non-devfs format (so e.g. /dev/hda or**

**# /dev/sda, and not e.g. /dev/discs/disc0/disc).**

**# For example, to use the first SCSI/SATA hard disk:**

**#d-i partman-auto/disk string /dev/sda**

**# In addition, you'll need to specify the method to use.**

**# The presently available methods are:**

**# - regular: use the usual partition types for your architecture**

**# - lvm: use LVM to partition the disk**

**# - crypto: use LVM within an encrypted partition**

**d-i partman-auto/method string regular**

**# If one of the disks that are going to be automatically partitioned**

**# contains an old LVM configuration, the user will normally receive a**

**# warning. This can be preseeded away...**

**d-i partman-lvm/device\_remove\_lvm boolean true**

**# The same applies to pre-existing software RAID array:**

**d-i partman-md/device\_remove\_md boolean true**

**# And the same goes for the confirmation to write the lvm partitions.**

**d-i partman-lvm/confirm boolean true**

**# For LVM partitioning, you can select how much of the volume group to use**

**# for logical volumes.**

**#d-i partman-auto-lvm/guided\_size string max**

**#d-i partman-auto-lvm/guided\_size string 10GB**

**#d-i partman-auto-lvm/guided\_size string 50%**

**# You can choose one of the three predefined partitioning recipes:**

**# - atomic: all files in one partition**

**# - home: separate /home partition**

**# - multi: separate /home, /usr, /var, and /tmp partitions**

**d-i partman-auto/choose\_recipe select atomic**

**# Or provide a recipe of your own...**

**# If you have a way to get a recipe file into the d-i environment, you can**

**# just point at it.**

**#d-i partman-auto/expert\_recipe\_file string /hd-media/recipe**

**# If not, you can put an entire recipe into the preconfiguration file in one**

**# (logical) line. This example creates a small /boot partition, suitable**

**# swap, and uses the rest of the space for the root partition:**

**#d-i partman-auto/expert\_recipe string \**

**# boot-root :: \**

**# 40 50 100 ext3 \**

**# $primary{ } $bootable{ } \**

**# method{ format } format{ } \**

**# use\_filesystem{ } filesystem{ ext3 } \**

**# mountpoint{ /boot } \**

**# . \**

**# 500 10000 1000000000 ext3 \**

**# method{ format } format{ } \**

**# use\_filesystem{ } filesystem{ ext3 } \**

**# mountpoint{ / } \**

**# . \**

**# 64 512 300% linux-swap \**

**# method{ swap } format{ } \**

**# .**

**# If you just want to change the default filesystem from ext3 to something**

**# else, you can do that without providing a full recipe.**

**#d-i partman/default\_filesystem string ext4**

**# The full recipe format is documented in the file partman-auto-recipe.txt**

**# included in the 'debian-installer' package or available from D-I source**

**# repository. This also documents how to specify settings such as file**

**# system labels, volume group names and which physical devices to include**

**# in a volume group.**

**# This makes partman automatically partition without confirmation, provided**

**# that you told it what to do using one of the methods above.**

**d-i partman-partitioning/confirm\_write\_new\_label boolean true**

**d-i partman/choose\_partition select finish**

**d-i partman/confirm boolean true**

**d-i partman/confirm\_nooverwrite boolean true**

**## Partitioning using RAID**

**# The method should be set to "raid".**

**#d-i partman-auto/method string raid**

**# Specify the disks to be partitioned. They will all get the same layout,**

**# so this will only work if the disks are the same size.**

**#d-i partman-auto/disk string /dev/sda /dev/sdb**

**# Next you need to specify the physical partitions that will be used.**

**#d-i partman-auto/expert\_recipe string \**

**# multiraid :: \**

**# 1000 5000 4000 raid \**

**# $primary{ } method{ raid } \**

**# . \**

**# 64 512 300% raid \**

**# method{ raid } \**

**# . \**

**# 500 10000 1000000000 raid \**

**# method{ raid } \**

**# .**

**# Last you need to specify how the previously defined partitions will be**

**# used in the RAID setup. Remember to use the correct partition numbers**

**# for logical partitions. RAID levels 0, 1, 5, 6 and 10 are supported;**

**# devices are separated using "#".**

**# Parameters are:**

**# <raidtype> <devcount> <sparecount> <fstype> <mountpoint> \**

**# <devices> <sparedevices>**

**#d-i partman-auto-raid/recipe string \**

**# 1 2 0 ext3 / \**

**# /dev/sda1#/dev/sdb1 \**

**# . \**

**# 1 2 0 swap - \**

**# /dev/sda5#/dev/sdb5 \**

**# . \**

**# 0 2 0 ext3 /home \**

**# /dev/sda6#/dev/sdb6 \**

**# .**

**# For additional information see the file partman-auto-raid-recipe.txt**

**# included in the 'debian-installer' package or available from D-I source**

**# repository.**

**# This makes partman automatically partition without confirmation.**

**d-i partman-md/confirm boolean true**

**d-i partman-partitioning/confirm\_write\_new\_label boolean true**

**d-i partman/choose\_partition select finish**

**d-i partman/confirm boolean true**

**d-i partman/confirm\_nooverwrite boolean true**

**## Controlling how partitions are mounted**

**# The default is to mount by UUID, but you can also choose "traditional" to**

**# use traditional device names, or "label" to try filesystem labels before**

**# falling back to UUIDs.**

**#d-i partman/mount\_style select uuid**

**### Base system installation**

**# Configure APT to not install recommended packages by default. Use of this**

**# option can result in an incomplete system and should only be used by very**

**# experienced users.**

**#d-i base-installer/install-recommends boolean false**

**# The kernel image (meta) package to be installed; "none" can be used if no**

**# kernel is to be installed.**

**#d-i base-installer/kernel/image string linux-generic**

**### Account setup**

**# Skip creation of a root account (normal user account will be able to**

**# use sudo). The default is false; preseed this to true if you want to set**

**# a root password.**

**#d-i passwd/root-login boolean false**

**# Alternatively, to skip creation of a normal user account.**

**#d-i passwd/make-user boolean false**

**# Root password, either in clear text**

**#d-i passwd/root-password password r00tme**

**#d-i passwd/root-password-again password r00tme**

**# or encrypted using an MD5 hash.**

**#d-i passwd/root-password-crypted password [MD5 hash]**

**# To create a normal user account.**

**d-i passwd/user-fullname string Ubuntu User**

**d-i passwd/username string ubuntu**

**# Normal user's password, either in clear text**

**d-i passwd/user-password password password**

**d-i passwd/user-password-again password password**

**# or encrypted using an MD5 hash.**

**#d-i passwd/user-password-crypted password [MD5 hash]**

**# Create the first user with the specified UID instead of the default.**

**#d-i passwd/user-uid string 1010**

**# The installer will warn about weak passwords. If you are sure you know**

**# what you're doing and want to override it, uncomment this.**

**d-i user-setup/allow-password-weak boolean true**

**# The user account will be added to some standard initial groups. To**

**# override that, use this.**

**#d-i passwd/user-default-groups string audio cdrom video**

**# Set to true if you want to encrypt the first user's home directory.**

**d-i user-setup/encrypt-home boolean false**

**### Apt setup**

**# You can choose to install restricted and universe software, or to install**

**# software from the backports repository.**

**#d-i apt-setup/restricted boolean true**

**#d-i apt-setup/universe boolean true**

**#d-i apt-setup/backports boolean true**

**# Uncomment this if you don't want to use a network mirror.**

**#d-i apt-setup/use\_mirror boolean false**

**# Select which update services to use; define the mirrors to be used.**

**# Values shown below are the normal defaults.**

**#d-i apt-setup/services-select multiselect security**

**#d-i apt-setup/security\_host string security.ubuntu.com**

**#d-i apt-setup/security\_path string /ubuntu**

**# Additional repositories, local[0-9] available**

**#d-i apt-setup/local0/repository string \**

**# http://local.server/ubuntu squeeze main**

**#d-i apt-setup/local0/comment string local server**

**# Enable deb-src lines**

**#d-i apt-setup/local0/source boolean true**

**# URL to the public key of the local repository; you must provide a key or**

**# apt will complain about the unauthenticated repository and so the**

**# sources.list line will be left commented out**

**#d-i apt-setup/local0/key string http://local.server/key**

**# By default the installer requires that repositories be authenticated**

**# using a known gpg key. This setting can be used to disable that**

**# authentication. Warning: Insecure, not recommended.**

**#d-i debian-installer/allow\_unauthenticated boolean true**

**### Package selection**

**tasksel tasksel/first multiselect ubuntu-desktop**

**#tasksel tasksel/first multiselect lamp-server, print-server**

**#tasksel tasksel/first multiselect kubuntu-desktop**

**# Individual additional packages to install**

**#d-i pkgsel/include string openssh-server build-essential**

**# Whether to upgrade packages after debootstrap.**

**# Allowed values: none, safe-upgrade, full-upgrade**

**#d-i pkgsel/upgrade select none**

**# Language pack selection**

**#d-i pkgsel/language-packs multiselect de, en, zh**

**# Policy for applying updates. May be "none" (no automatic updates),**

**# "unattended-upgrades" (install security updates automatically), or**

**# "landscape" (manage system with Landscape).**

**d-i pkgsel/update-policy select none**

**# Some versions of the installer can report back on what software you have**

**# installed, and what software you use. The default is not to report back,**

**# but sending reports helps the project determine what software is most**

**# popular and include it on CDs.**

**#popularity-contest popularity-contest/participate boolean false**

**# By default, the system's locate database will be updated after the**

**# installer has finished installing most packages. This may take a while, so**

**# if you don't want it, you can set this to "false" to turn it off.**

**#d-i pkgsel/updatedb boolean true**

**### Boot loader installation**

**# Grub is the default boot loader (for x86). If you want lilo installed**

**# instead, uncomment this:**

**#d-i grub-installer/skip boolean true**

**# To also skip installing lilo, and install no bootloader, uncomment this**

**# too:**

**#d-i lilo-installer/skip boolean true**

**# With a few exceptions for unusual partitioning setups, GRUB 2 is now the**

**# default. If you need GRUB Legacy for some particular reason, then**

**# uncomment this:**

**#d-i grub-installer/grub2\_instead\_of\_grub\_legacy boolean false**

**# This is fairly safe to set, it makes grub install automatically to the MBR**

**# if no other operating system is detected on the machine.**

**d-i grub-installer/only\_debian boolean true**

**# This one makes grub-installer install to the MBR if it also finds some other**

**# OS, which is less safe as it might not be able to boot that other OS.**

**d-i grub-installer/with\_other\_os boolean true**

**# Alternatively, if you want to install to a location other than the mbr,**

**# uncomment and edit these lines:**

**#d-i grub-installer/only\_debian boolean false**

**#d-i grub-installer/with\_other\_os boolean false**

**#d-i grub-installer/bootdev string (hd0,0)**

**# To install grub to multiple disks:**

**#d-i grub-installer/bootdev string (hd0,0) (hd1,0) (hd2,0)**

**# Optional password for grub, either in clear text**

**#d-i grub-installer/password password r00tme**

**#d-i grub-installer/password-again password r00tme**

**# or encrypted using an MD5 hash, see grub-md5-crypt(8).**

**#d-i grub-installer/password-crypted password [MD5 hash]**

**# Use the following option to add additional boot parameters for the**

**# installed system (if supported by the bootloader installer).**

**# Note: options passed to the installer will be added automatically.**

**#d-i debian-installer/add-kernel-opts string nousb**

**### Finishing up the installation**

**# During installations from serial console, the regular virtual consoles**

**# (VT1-VT6) are normally disabled in /etc/inittab. Uncomment the next**

**# line to prevent this.**

**#d-i finish-install/keep-consoles boolean true**

**# Avoid that last message about the install being complete.**

**d-i finish-install/reboot\_in\_progress note**

**# This will prevent the installer from ejecting the CD during the reboot,**

**# which is useful in some situations.**

**#d-i cdrom-detect/eject boolean false**

**# This is how to make the installer shutdown when finished, but not**

**# reboot into the installed system.**

**#d-i debian-installer/exit/halt boolean true**

**# This will power off the machine instead of just halting it.**

**#d-i debian-installer/exit/poweroff boolean true**

**### X configuration**

**# X can detect the right driver for some cards, but if you're preseeding,**

**# you override whatever it chooses. Still, vesa will work most places.**

**#xserver-xorg xserver-xorg/config/device/driver select vesa**

**# A caveat with mouse autodetection is that if it fails, X will retry it**

**# over and over. So if it's preseeded to be done, there is a possibility of**

**# an infinite loop if the mouse is not autodetected.**

**#xserver-xorg xserver-xorg/autodetect\_mouse boolean true**

**# Monitor autodetection is recommended.**

**xserver-xorg xserver-xorg/autodetect\_monitor boolean true**

**# Uncomment if you have an LCD display.**

**#xserver-xorg xserver-xorg/config/monitor/lcd boolean true**

**# X has three configuration paths for the monitor. Here's how to preseed**

**# the "medium" path, which is always available. The "simple" path may not**

**# be available, and the "advanced" path asks too many questions.**

**xserver-xorg xserver-xorg/config/monitor/selection-method \**

**select medium**

**xserver-xorg xserver-xorg/config/monitor/mode-list \**

**select 1024x768 @ 60 Hz**

**### Preseeding other packages**

**# Depending on what software you choose to install, or if things go wrong**

**# during the installation process, it's possible that other questions may**

**# be asked. You can preseed those too, of course. To get a list of every**

**# possible question that could be asked during an install, do an**

**# installation, and then run these commands:**

**# debconf-get-selections --installer > file**

**# debconf-get-selections >> file**

**#### Advanced options**

**### Running custom commands during the installation**

**# d-i preseeding is inherently not secure. Nothing in the installer checks**

**# for attempts at buffer overflows or other exploits of the values of a**

**# preconfiguration file like this one. Only use preconfiguration files from**

**# trusted locations! To drive that home, and because it's generally useful,**

**# here's a way to run any shell command you'd like inside the installer,**

**# automatically.**

**# This first command is run as early as possible, just after**

**# preseeding is read.**

**#d-i preseed/early\_command string anna-install some-udeb**

**# This command is run immediately before the partitioner starts. It may be**

**# useful to apply dynamic partitioner preseeding that depends on the state**

**# of the disks (which may not be visible when preseed/early\_command runs).**

**#d-i partman/early\_command \**

**# string debconf-set partman-auto/disk "$(list-devices disk | head -n1)"**

**# This command is run just before the install finishes, but when there is**

**# still a usable /target directory. You can chroot to /target and use it**

**# directly, or use the apt-install and in-target commands to easily install**

**# packages and run commands in the target system.**

**#d-i preseed/late\_command string apt-install zsh; in-target chsh -s /bin/zsh**

**d-i preseed/late\_command string in-target perl -pi -e 's/(errors=remount-ro)/noatime,nodiratime,$1,barrier=0/' /etc/fstab**