By jakev383 (Contact Author) (Forums)
Published: 2008-08-08 15:25

How To Remotely Install Debian Over A RH Based DistroIntroduction

Ocassionally, servers need to be retasked for various reasons. It has always been a challenge when the server has a distribution other than what I need. I do not want to drive to the data center to swap CDs around, so I decided to see if there was a way to remotely install the machine. I found some notes by Erik Jacobsen andused them to come up with an up-to-date how-to.

Hopefully this will be useful to others out there. Be warned that you can cause some serious headaches for yourself. I offer no warranties, expressednor implied, that this will work for you.

In this tutorial, I have an ancient Fedora Core 3 machine that I need to turn into a Debian Etch machine. The machine is 600 miles from my home office so traveling to put in CDs is not an economical option. The system has one 80G hard drive that is currently divided into 3 partitions: a 100M /boot partition, a 1G swap partition, and theremainder as the / partition for the Fedora Core installation. The machine has a public static IP address and will be accessed through an SSH connection.

You'll need to make sure your swap space is adequate. We will be loading the base Debian system into the space currently allocated for swap, so you need to make sure you have enough room to accomplish this task. In my case I used 371M of space out of a 1012Mswap when the Debian base was loaded, leaving me 590M of free space.

1) Install debootstrap

If you have another Debian machine available to you, you can build the debootstrap package yourself. You may download the latest debootstrap .deb file from here:

http://packages.debian.org/etch/all/debootstrap/download

On the Debian build machine, install alien:

```
apt-get install alien
```

You then need to convert the Debian .deb file to a RPM for the RH machine to use:

```
alien -rkv debootstrap*.deb
```

This will create a debootstrap RPM file that you need to install on the current RH based machine once you get it transferred over:

```
rpm -Uvh debootstrap*
```

I've made a copy of the RPM available in the event you cannot build it for whatever reason. You may download the RPM from **http://v2gnu.com/filemgmt/visit.php?lid=27**. I make this file available, but I will not be updating it.

2) Change the swap space into a usable filesystem

The remainder of this paper will be run on the RH machine that we will be converting into a Debian based machine.

Turn off your swap space so we can load Debian onto it:

```
swapoff -a
```

Use

```
fdisk -l
```

to see which partition your swap space is. On my system, the swap space is `/dev/hda2`:

```
Disk /dev/hda: 80.0 GB, 80026361856 bytes
255 heads, 63 sectors/track, 9729 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
```

```
        Device Boot      Start         End      Blocks   Id  System
/dev/hda1   *               1          13      104391   83  Linux
/dev/hda2                  14         144     1052257+  82  Linux swap
/dev/hda3                 145        9729    76991512+  83  Linux
```

Use fdisk to change the swap space into a Linux partition so we can load data into it.

*** WARNING - You are now entering the point of no return where your system may be unusable! ***

```
fdisk /dev/hda
```

```
Command (m for help): t
Partition number (1-4): 2              (change this to your swap partition number!)
Hex code (type L to list codes): 83
Changed system type of partition 2 to 83 (Linux)

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.

WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
The kernel still uses the old table.
The new table will be used at the next reboot.
Syncing disks.
```

Double check to make sure that the swap partition (/dev/hda2) is now a Linux partition:

```
fdisk -l
```

```
 Disk /dev/hda: 80.0 GB, 80026361856 bytes
```

```
      255 heads, 63 sectors/track, 9729 cylinders
      Units = cylinders of 16065 * 512 = 8225280 bytes

        Device Boot     Start        End      Blocks   Id  System
/dev/hda1  *           1         13      104391   83  Linux
/dev/hda2             14         144     1052257+  83  Linux
/dev/hda3            145         9729   76991512+  83  Linux
```

We next need to format this partition as ext3 so that we may load the Debian base onto it:

```
mke2fs -j /dev/hda2
```

```
mke2fs 1.35 (28-Feb-2004)
      max_blocks 269377536, rsv_groups = 8221, rsv_gdb = 64
      Filesystem label=
      OS type: Linux
      Block size=4096 (log=2)
      Fragment size=4096 (log=2)
      131616 inodes, 263064 blocks
      13153 blocks (5.00%) reserved for the super user
      First data block=0
      Maximum filesystem blocks=272629760
      9 block groups
      32768 blocks per group, 32768 fragments per group
      14624 inodes per group
      Superblock backups stored on blocks:
              32768, 98304, 163840, 229376

      Writing  inode tables:  done
       inode.i_blocks = 2568, i_size = 4243456
       Creating journal (8192 blocks): done
       Writing superblocks and filesystem accounting information: done
```

```
 This filesystem will be automatically checked every 32 mounts or
180 days, whichever comes first.  Use tune2fs -c or -i to override.
```

Therewas an issue with the older Debian distros that balked at the dir_indexflag. This issue may or may not be present in the newer versions. We're going to turn it off to be on the safe side:

```
tune2fs -O ^dir_index /dev/hda2
```

3) Load the Debian packages onto the new partition

Nextwe create a variable name for the new partition to save on typing. We used $ASD as an arbitrary place holder since it was easier andquicker to type than /dev/hda2. Feel free to substitute anyvariable name that is convenient for you:

```
export ASD=/mnt/asd
```

We next create the directory to serve as the mount point:

```
mkdir -p $ASD
```

And lastly we mount the new partition:

```
mount /dev/hda2 $ASD
```

From here on, we can use $ASD in the place of /dev/hda2.

Nowwe run debootstrap which will take a while and populate the packagesneeded. You will be choosing the architecture (i386 here),distribution (Etch used here), directory, and URL to a Debian archive.For the URL at the end, it can be any valid Debian mirror. Youare encouraged to choose a mirror closest to your geographic location. A list of available mirrors is available on the main Debian site,specifically here: **http://www.debian.org/mirror/list**.

I am goingto use one of the more permanent mirrors so it may be a little slow,but feel free to use any mirror you wish (even your own repo!):

```
/usr/sbin/debootstrap --arch i386 etch $ASD http://mirrors.kernel.org/debian
```

This will take a while. Go have a cup of coffee or order a pizza. Once it's done you will have the base Debian packages installed into the new partition ($ASD).

4) Keep some old system files

Nextwe're going to reuse some files from the old RH based system. Chances are good that the machine name and hostname will stay thesame, as well as the name servers, so we are going to copy the configfiles for these from our RH based system into our new Debian system.

Copy resolv.conf into the new system to keep our old name server information:

```
cp /etc/resolv.conf $ASD/etc/
```

```
cp: overwrite `/mnt/asd/etc/resolv.conf'? y
```

Copy our hosts file to keep the same name mapping information:

```
cp /etc/hosts $ASD/etc/
```

Lastly, our hostname file:

```
cp /etc/hostname $ASD/etc/
```

If you're missing either of the last 2 files, don't worry. You can create them in the $ASD/etc/ directory now.

5) Begin setting up new system

Nowthat the base system is downloaded and installed, we're going to entera chroot environment so we can begin to set the new system up:

```
chroot $ASD /usr/bin/env -i HOME=/root TERM=$TERM PS1='\u:\w\$ ' PATH=/bin:/usr/bin:/sbin:/usr/sbin /bin/bash --login
```

First thing we need to do is tell the new system how to mount the filesystems when it boots up. I use vim as my editor of choice, but feel free to use whatever editor you feelcomfortable with. We're going to make a simple fstab for themoment:

```
vim /etc/fstab
```

```
# filesystem mount fs-type options dump fsck-order


/dev/hda2 / auto defaults 0 1
proc /proc proc defaults 0 0
```

Now we need to get proc mounted:

```
mount -t proc proc /proc
```

Then we set the hostname (make sure this matches what you have configured in your hosts file):

```
hostname test.example.com
```

RHbased distros will not have this next file. Debian systems configure the network interfaces in a different manner than RH based distros, so we need to create this next file and populate it:

```
vim /etc/network/interfaces
```

```
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.

auto lo
iface lo inet loopback

auto eth0 # Automatically bring eth0 up on boot
iface eth0 inet static # Define a static IP for eth0
  address 192.168.100.110 # This machine's IP address
  netmask 255.255.255.0 # The netmask for the network that this machine is on
  network 192.168.100.0 # The network that this machine is on
  broadcast 192.168.100.255 # The broadcast address for the network this machine is on
  gateway 192.168.100.1 # The gateway this machine needs to communicate through
  dns-nameservers 192.168.100.1 192.168.100.2 # Define name servers here - redundant to /etc/resolv.conf
```

Change the IP addresses to match your environment. If you do not know this information, you can always open another SSH session into the system and find out. Opening a new SSH session into the machine will bring you into the RH system, not the chroot'ed Debian system, so you may gather this information from the config files on the functioning RHmachine. Once you are done setting this information, save the file.

Let's set a few system settings now. First let's set a root password and create a user account:

```
dpkg-reconfigure passwd
```

I answered Yes to Shadow passwords, entered a root password, and created a normal user account (user1) so that I do not have to login as root via SSH.

Next we install and configure our locales:

```
apt-get install locales
```

```
dpkg-reconfigure locales
```

Now select what type of locale you need (I selected en_US ISO-8559-1). Make sure you do NOT select 'None' as the default locale for the system environment.

Lastly, we need to set our timezone:

```
tzconfig
```

I couldn't get netselect-apt to work on my test system, so I manually configured my /etc/apt/sources.list:

```
vim /etc/apt/sources.list
```

```
deb http://mirrors.kernel.org/debian/ etch main
deb-src http://mirrors.kernel.org/debian/ etch main

deb http://security.debian.org/ etch/updates main
deb-src http://security.debian.org/ etch/updates main
```

You can try running 'netselect-apt etch' from the command line and seeing if it works for you. If not, you'll need to manually set the file. Remember to run

```
apt-get update
```

afterwards!

Now we need to install SSH so we can login after we reboot into the new Debian system:

```
apt-get -y install ssh
```

Using apt-cache we will search for a kernel appropriate for this system:

```
apt-cache search kernel-image
```

Use the SMP if you have multiple processors. I'm using the generic 686 kernel that most   everyone will use:

```
apt-get -y install kernel-image-2.6-686
```

I   chose Yes to the vmlinuz symbolic link. Make sure to choose "No" to the abort message that pops up about the bootloader.

Next we need to install a package that will help load modules, such as the Ethernet driver:

```
apt-get -y install discover
```

```
discover
```

You'llwant to compare this against an output of

```

lsmod
```

on the RH system to see   if there are any differences in identifying hardware. If there are,   you'll need to add a line to /etc/modutils/aliases to load the correct   driver. Afterwards you'll need to run

```
update-modules
```

We also need to create /etc/discover.conf to configure hardware detection settings:

```
vim /etc/discover.conf
```

```
# /etc/discover.conf: hardware detection settings
# Enable the PCI, USB, IDE, and SCSI bus scans:
enable pci,usb,ide,scsi

# Disable PCMCIA - We're not running a laptop!
disable pcmcia

# Scan for these devices at boot
boot bridge ethernet ide scsi usb
```

Let's run makedev just to be on the safe side:

```
cd /dev

./MAKEDEV generic
```

This can take a couple minutes. After it's done, log out of the chroot.

```
logout
```

6) Prepare the boot partition

Now that we're back in the RH system, let's copy the Debian kernel and initrd images to the boot partition:

```
cp $ASD/boot/vmlinuz-* /boot

cp $ASD/boot/initrd.img-* /boot
```

Now editthe grub config file to boot into the new Debian system. Makesure to add this before any of the other RH stanzas, since Grubwill load the first operating system listed if you have not configuredit to do otherwise:

```
vim /boot/grub/menu.lst
```

```
title Debian!
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.18-6-686 root=/dev/hda2 ro
    initrd /boot/initrd.img-2.6.18-6-686
```

Change (hd0,1) to whatever drive,partition you're using. Alsocheck to make sure that you match the correct versions of the kerneland initrd files. Look at $ASD/boot to double check the kernel and initrd versions that will be used.

Copy our modified grub config to the Debian system:

```
cp /etc/grub.conf $ASD/etc/

cp /boot/grub/splash.xpm.gz $ASD/boot/

cp -r /boot/grub $ASD/
```

We'll skip the fsck performed on the reboot:

```
touch $ASD/fastboot
```

Lastly, we unmount the proc and chroot environment:

```
umount $ASD/proc

umount $ASD
```

If the umount command is giving you an error about the filesystem being busy, use the -l switch:

```
umount -l $ASD
```

7) Reboot into Debian for the first time

Here comes the cliff-hanging moment. We reboot into our new Debian system for the first time.

```
reboot
```

8) Mount boot partition in Debian system

Nowif everything went well, you should be booted into the new Debian system. If it did not go well, give it a few minutes and try to loginagain. If it still doesn't work you'll need to figure out a wayto get console access to fix the problem or change to the old RHkernel.

Let's mount our old /boot partition since our skeletonfstab did not account for it. We'll erase what's in there, butdon't worry - we have a copy in our own /boot directory right nowanyway so we'll just copy it back in:

```
mke2fs -j /dev/hda1
```

We create a temporary place to mount the partition:

```
mkdir /boot2
```

Then we mount it:

```
mount /dev/hda1 /boot2
```

Now we copy our boot files into the new partition:

```
cp /boot/* /boot2/

mv /grub /boot2/
```

Since that is done, let's unmount it and remove the mount point:

```
umount /boot2

rmdir /boot2
```

Now remove the old directory, since it will now be mounting a partition there:

```
rm -rf /boot/*
```

Add the new boot partition into fstab:

```
vim /etc/fstab
```

```
# filesystem mount fs-type options dump fsck-order

/dev/hda1 /boot ext3 defaults 1 2
/dev/hda2 / auto defaults 0 1
proc /proc proc defaults 0 0
```

This next command mounts all the mount points located in the fstab:

```
mount -a
```

Now we need to modify the grub config file, since our paths are now different:

```
vim /boot/grub/menu.lst
```

```
title Debian!
    root (hd0,0)
    kernel /vmlinuz-2.6.18-6-686 root=/dev/hda2 ro
    initrd /initrd.img-2.6.18-6-686
```

Notethat we're changing hd0,1 to hd0,0 since we're now booting from thefirst partition. We need to remove the /boot path fromthe vmlinuz and initrd images since they'll be in the root of the firstpartition.

9) Copy base Debian system into new partition

We need to move the system back onto the original partition that held the RH distro. Look back over your notes, select the partition the RH distro was located in. In our case the partition was /dev/hda3.

First do a quick format of the system to clear it off:

```
mke2fs -j /dev/hda3
```

Once that's done, we'll move the filesystem over using dd:

```
dd if=/dev/hda2 of=/dev/hda3 bs=1024
```

'if' is the current partition you want to copy the data FROM.
'of' is the partition you want to copy the data TO.
'bs' defines the block size. You can use a value larger than 1024 if you want, but 1024 works fine for most.

Thiscommand can take 3-30 minutes depending on your system and harddrive speeds, so have another slice of pizza before it gets toocold.

When that finishes, the new filesystem isn't aware of the new partition size. That will need to be adjusted:

```
e2fsck -f -y /dev/hda3
```

Run this command until you do not get any more errors (usually twice should do it).

Now resize the partition:

```
resize2fs -p /dev/hda3
```

```
resize2fs 1.40-WIP (14-Nov-2006)
Resizing the filesystem on /dev/hda3 to 19247878 (4k) blocks.
Begin pass 1 (max = 579)
Extending the inode table     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
The filesystem on /dev/hda3 is now 19247878 blocks long.
```

Run fsck on it again to make sure there were no hiccups:

```
e2fsck -f -y /dev/hda3
```

Then go back into grub and tell it to boot into the new partition:

```
vim /boot/grub/menu.lst
```

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
#hiddenmenu
title Debian!
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.18-6-686 root=/dev/hda3 ro
    initrd /boot/initrd.img-2.6.18-6-686
```

Note that we changed /dev/hda2 to /dev/hda3.

Change the /etc/fstab to point / (root) at the new partition (/dev/hda3):

```
vim /etc/fstab
```

```
# filesystem mount fs-type options dump fsck-order

/dev/hda1 /boot ext3 defaults 1 2
/dev/hda3 / auto defaults 0 1
proc /proc proc defaults 0 0
```

And a reboot to bring the new partition up......

```
reboot
```

Youwill now be in your full Debian system. There are still a couple details to be attended to, but the system is now fully functional.

10) Get our swap partition back

One of the final steps is to convert the temporary filesystem we loaded the Debian base onto (/dev/hda2) back into swap space to be utilized by the Debian system.

We need to use fdisk to change the partition type from ext3 to swap:

```
fdisk /dev/hda
```

```
 Command (m for help): t
 Partition number (1-4): 2          (change this to your swap partition number!)
 Hex code (type L to list codes): 82
 Changed system type of partition 2 to 82 (Linux swap / Solaris)

 Command (m for help): w
 The partition table has been altered!

 Calling ioctl() to re-read partition table.

 WARNING: Re-reading the partition table failed with error 16: Device or resource busy.
 The kernel still uses the old table.
 The new table will be used at the next reboot.
Syncing disks.
```

Add an entry into fstab for the swap partition:

```
vim /etc/fstab
```

```
# filesystem mount fs-type options dump fsck-order

/dev/hda1 /boot ext3 defaults 1 2
/dev/hda2 swap swap defaults 0 0
/dev/hda3 / auto defaults 0 1
proc /proc proc defaults 0 0
```

Have the system turn it back into swap space:

```
mkswap /dev/hda2
```

Activate and sync the swap space for the running system to use:

```
swapon -a
```

```
sync;sync;sync
```

Andthat was the last step! You've now converted a RH based machineinto a Debian machine without ever physically touching it. I like to do 1 more reboot to make sure everything is working correctly - I'd recommend you do the same.