How To Image SUSE Linux Systems Using `tbku`

This document describes how to use the TundraWare Inc. `tbku` utility to "image" or "clone" SUSE Linux systems.

It's worth noting that most/much of this will also be relevant to other Linux distributions, though some of the fine points may be different.

> **Warning**
>
> What follows is a description of activities that can (and will) clobber the contents of a hard drive. Never do any of this until you understand what's going on fully. Obviously, you should have backups of whatever machine you're targeting so that if (when) you make a mistake, you can recover your data. YOU HAVE BEEN WARNED! If you proceed, you do so at your own risk ... and no, I will *not* come to your house and help you recover your hard drive.

Why Bother Imaging?

Suppose we need to build a new instance of a `SUSE Linux` system. Perhaps we need to replace one that just had a hard drive failure. Maybe we want to build a new server that is based on our "standard" system configuration. In other words, we want to go from "bare metal" hardware to a fully running *and configured* system as quickly as possible.

There are a number of commercial and open source solutions to this problem, but they all have one thing in common: We want to minimize the amount of manual labor needed to install, configure, and otherwise customize the final system. This is especially important in large data centers where it is impractical to manually (re)install each and every server, its applications, and its customization information.

"Imaging" or "Cloning" allows us to keep a copy of the entire OS *as configured* - that means with all its applications and configuration options set up as desired. We then load or "Provision" a new hard drive with this image and *voila'*, "instant" running system.

When Does Imaging NOT Make Sense?

Imaging works best when the system you are targeting is very similar or identical to the system that made the image in the first place. For example, Imaging is a great way to restore a single machine from its own backups - say after a hard disk crash or upgrade.

Imaging is more complex when the source of the image and the target machines are different. The more different they are, the harder it will be to get the image running on the new target machine.

As a practical matter, production Data Centers tend to keep a separate restore image around *for each different system variant*. So, for example, you might find a separate image for IBM web servers, IBM applications servers, Dell database servers, Toshiba laptops, and so on.

Imaging may- or may not make sense when initially installing a new configuration. Say you have a system that is a web server, but you now want to build a separate machine that is a database server. Typically, you would initially install SUSE Linux with the installation disk, configure the database and *then* create a system image of your database server. However, this is kind of time consuming (unless you already have an `AutoYAST` configuration ready to go). It may be simpler to image the target machine with your web server image, boot it, reconfigure it as a database server, and then take an image of your newly configured server for future installations.

What Is `tbku`?

`tbku` is a shell script that makes it easy to create tarballs of some of all of your filesystems. `tbku` does not help you with *restoring* your image, it's just handy for creating the image in the first place.

If you've never used it before, take a moment to download it and read the documentation. You'll find the latest copy at:

http://www.tundraware.com/Software/tbku

There is no fee for using `tbku` in any context, personal or commercial. However, there are some licensing terms you have to abide by to use it, so take a moment to read the license in the distribution tarball.

> **Note**
>
> You don't *have* to use `tbku` to create your backup image. The description below should work fine so long as you have a backup of all the relevant files that preserves all the appropriate file information such as ownership and permissions. `tbku` just makes it easy to automate the creation of such backups.

The Big Picture

Before diving into the details, it's good to get a sense of the overall process. Imaging a system requires the following steps:

A. Create the master image:
   - Create a baseline system configured as you want it.
   - Take an "image" of it. (That's where `tbku` is helpful.)
   - Save the image somewhere (DVD, USB drive, network drive ...) you can get at when you need it to (re)install a system.

B. Use the master image to (re)provision a machine:
   - Prepare the target hard disk to receive the image.
   - Dump the image onto the hard disk.
   - Adjust the configuration if/as needed for the new hardware.

Creating The Master Image

Unlike other approaches that make an image of *the disk*, `tbku` creates an image of *files* on the disk. This means that your new target disk does not have to be physically the same as the one on which the master image (sometimes called a "snapshot") was made. You can clone systems back and forth between SCSI, IDE, and SATA. You can clone from smaller disks to larger ones or go the other way.

> **Note**
>
> The whole point of imaging is to avoid having to do custom configuration for each new installation. However, some configuration changes may be necessary when the target environment or hardware is different than the system on which the master image was created. This is discussed a bit more below in the Gotchas section.

Creating The Master Image

1. Select the machine whose existing SUSE Linux installation you want preserved or used as a standard installation image.
2. Image that system with `tbku` using the following fileset:

   ```
   /bin
   /boot
   /dwnl
   /etc
   /home
   /lib
   /local
   /opt
   /root
   /sbin
   /srv
   /usr
   /var
   ```

   Notice that we do *not* backup the dynamic kernel-created filesystems like `/dev` or `/proc`, nor do we backup utility mountpoints like `/mnt` or `/tmp`. Also, if you have `tbku` writing your backup to the local disk, make sure that directory is *not* included in the fileset. Doing so would create a recursive backup wherein the backup would be copied to itself.
   The exact fileset you use will vary somewhat depending on how you've laid out your directory tree and just what you want included in your image. Use the fileset above as a point of departure, and tune it for your exact needs.
3. Save the resulting `.tar.gz` (tarball) file somewhere it can be retrieved later when you want to image another machine. This can be a network server, a USB drive, a DVD or whatever makes sense in your environment. As with all backup systems, it's pretty important to make multiple copies of the backup image, and keep a couple of them off-site.

Provisioning With The Master Image

Now that we have a "snapshot" or master image, we can use it to (re)provision machines.

Provisioning Machines With A Master Image

1. Boot the SUSE Linux installation disk and load the `Rescue System`.
2. Now we have to prepare the disk to receive a Linux filesystem. The example below assumes we are installing on `/dev/hda` - a PATA master on the first IDE controller - but that the image came from a system that boots from the first SCSI drive, `/dev/sda`. Keep in mind you can do what follows with any of the drives on your system. Just substitute the device names as appropriate:

   ```
   # Partition the drive:

   fdisk /dev/hda

   # Delete and create partitions as you like
   ```

```
# Make sure the partition that will mount / is
# toggled to be bootable Be sure to use the
# 'write' option before exiting

# Suppose you end up with this:
#
#    /dev/hda1  is for swap (type 82)
#    /dev/hda2  is for your filesystem (type 83)

mkswap /dev/hda1
mkfs.reiserfs /dev/hda2
mount /dev/hda2 /mnt

# Now, let's create the top level directories
# that were not backed up and/or will be used
# by the kernel for its own filesystems:

cd /mnt
mkdir dev media mnt proc sys tmp

# Now it's time to mount your backup medium.
# Depending on your backup medium this can be
# one of several devices.  CD/DVDs are often
# found at /dev/hdc.  USB drives show up as
# SCSI drives such as /dev/sda1, and so on.
# You'll also need to know the type of the
# backup medium (see: man mount for the
# details):

mount -
tvfat /dev/sda1 /mnt/mnt  # This is a USB drive

# OK, time to dump the image previously
# created by tbku onto our shiny new
# filesystem (make sure your current directory
# is still /mnt before doing this):

tar -xzvf mnt/my-system-image.tar.gz

# Now we have to make sure that the boot
# tables and default file mounts are correct -
# Our target system may have a different drive
# type or device (SCSI, SATA, PATA) than the
# system from which tbku took the image:
#
# We need to make sure that things are mounted
# to reflect the partitioning you did with
# fdisk.  This is done by editing:
#
# /mnt/etc/fstab

# Remember that drives can be named by device
# name (/dev/xxxx) or by the drive id name
```

```
# (/dev/disk/by-id/xxxx).
#
# In our case the relevant portion of
# /mnt/etc/fstab looks like this:

/dev/sda1    swap     swap       defaults         0 0
/dev/sda2    /        reiserfs   acl,user_xattr   1 1

# But now it needs to look like this:

/dev/hda1    swap     swap       defaults         0 0
/dev/hda2    /        reiserfs   acl,user_xattr   1 1

# Be sure not to disturb the other stuff in
# the fstab file, or at least make sure it
# still makes sense.

# Now, check and fix the device map file:
#
#     /mnt/boot/grub/device.map
#
# Since we took the tbku image from a system
# that boots from SCSI, the file looks like
# this:

(fd0)   /dev/fd0
(hd0)   /dev/sda

# But our new system wants to boot from PATA
# so it now needs to look like this:

(fd0)   /dev/fd0
(hd0)   /dev/hda

# We also have to correct any differences in
# the boot menu that appears when you first
# start the system.  This is in:
#
#    /mnt/boot/grub/menu.lst
#
# Near the top of this file you'll see
# something like this:

gfxmenu (hd0,1)/boot/message

# hd0 is right - we made sure of that when we
# edited the map file above.  Make sure that
# the offset (1 in this case) is also right.
# This is the number, *counting from 0* of the
# root/boot partition within that drive.  In
# our case, (hd0,1) is correct because our
# root/boot partition is /dev/hda2.
```

```
# Following this are the individual menu entries.
# Make sure you check each line of every entry.
# Suppose we find this:

title SLED 10 - 2.6.16.54-0.2.5
    root (hd0,1)
    kernel /boot/vmlinuz-2.6.16.54-0.2.5-
default root=/dev/sda2
          resume=/dev/sda1 splash=silent showopts
    initrd /boot/initrd-2.6.16.54-0.2.5-default

# As with the previous gfxmenu statement, make
# sure root (hd0,1) is right.

# All references to /dev/sda2 have to be
# changed to /dev/hda2

# All references to /dev/sda1 have to be
# changed to /dev/hda1

# Repeat this for every menu entry.

# Finally, let's make sure that the boot
# loader is properly installed and configured:

grub-install --root-directory=/mnt /dev/hda
```

We're DONE! Well ... maybe. If the environment or hardware of your target machine is similar/same as the machine from which you took the original image OR if the kernel you plan to boot has support for your new target hardware, you should just be able to boot and run at this point. If not, read the following Gotchas section for further explanation.

This may all seem complex the first time you do it, but after a couple of times, you'll be able to do this in your sleep. This is one of those things where describing it is more complicated than just doing it!

Depending on how large your backup image is, a complete system restore can typically be done in less than an hour. That's less than an hour to a *completely configured system* with all your applications, custom configuration, and so on as you last left them.

Gotchas

If you use the approach described above to reprovision the same machine - say after a disk failure or disk upgrade - then that's all you have to do. Your "target" machine is essentially identical to the one from which you got the backup image ... the same machine.

However, there are circumstances where you cannot avoid doing some configuration on the newly provisioned machine. This is the case where there is a significant difference between the machine that took the snapshot and the machine receiving it. This might be because the target machine has different hardware, needs a different IP address, uses a different chipset, and so on.

There is no general way to solve these sorts of problems. You'll have to dig through `YAST` (if the system boots at all) and/or the individual configuration files to correct things.

Note

Imaging is not the answer to every new installation problem. At some point, it becomes simpler to do a fresh install of SUSE Linux than to try and "tweak" an existing image to get it running properly.

As a personal preference, I like to work directly with configuration files from the command line whenever I can. If the target machine will not boot, you sort of have no other choice. You'll have to do something like this to get to those files to edit them. Boot the installation CD and select `Rescue System`, then mount the target drive(s):

```
mount /dev/hda2 /mnt
```

You can then edit the files found under `/mnt`.

What Problems Can I Expect?

So, you've decided to image a machine that is somehow different than the original source of the image. Here's what you'll possibly encounter:

A. Environmental Differences

Your newly imaged machine may work fine except that its environment needs to change. The most common thing here is the need to reconfigure the NIC with new network parameters like IP address, netmask, DNS server, default route, and so on. Similarly, you may want to change the machine name or domain name.

This is all easily done via `YAST` or by editing the relevant configuration files directly. Keep in mind that changing the OS environment may also require changes in your applications' configuration. For instance, changing your machine name, IP, and so forth can break Apache.

B. Different Hardware

This is the tougher situation to handle after a machine has been newly imaged. Modern SUSE Linux kernels come with enough standard driver support built-in that they should boot on most standard hardware ... unless you've hand tuned the kernel on the machine where the image was taken.

However "booting" and "running properly" are two different things. In the process of preparing this documentation, I discovered that my newly imaged test machine *refused* to set the PATA drive into UDMA modes 5/6. Why? Because the machine used to create the original image had an older (different) chipset than the newly imaged machine. I had to figure out which additional drivers the kernel needed to load for it to work properly on the new hardware.

Hardware differences show up in a number of places:

1. CPU Architecture
   If you built your image on a machine that is configured exclusively to run, say, on Xeon chipsets, and then try to image another machine with a Pentium 4, um ... it's not going to work. The kernels in your image have to be compatible with the CPU architecture on your target machine
2. Motherboard Chipset
   Motherboards have so-called "Northbridge" and "Southbridge" chipsets. The Northbridge chip(s) control memory and high speed graphics (like AGP). The Southbridge chip(s) control the slower I/O functions and peripherals of the motherboard. If the machine you're

imaging uses wildly different chipsets than the machine where the image was taken, you're going to probably have problems.
This was the case in the example above. By default, SUSE Linux could boot IDE in its slowest possible mode, but it could not exploit the higher speed UDMA features of the new Southbridge chipset - that required the installation of a driver specific for that chipset.
If you have different Southbridges, you'll run into this with any of the on-board controllers:

- Audio
- Buses
- Disk
- Joystick
- Network
- Video

3. Peripheral Cards
   If your newly imaged machine has different PCI and/or video cards than the machine that produced the image, you may, again, have to install additional or different drivers.

Configuring Drivers

Let's assume you can boot your machine fine, but you need to get additional or different drivers to load for the machine to run optimally. The kernel configuration is in this file:

```
/etc/sysconfig/kernel
```

In that file, you'll see a line something like this:

```
INITRD_MODULES="piix aic7xxx processor thermal fan reiserfs edd"
```

Now, suppose we want to add the drivers for, say, a VIA chipset. We'd edit that line as follows:

```
INITRD_MODULES="piix aic7xxx sata_via via82cxxx processor thermal fan reiserfs edd"
```

Then we have to create a new `initrd` like this:

```
mkinitrd
```

Now unmount the drive and reboot.

> **Note**
>
> If you *cannot* boot your new system, boot the installation CD as before and get into the `Rescue System`. Mount the target drive under `/mnt` as we did previously. This will allow you to edit `/mnt/etc/sysconfig/kernel` as needed. You can then run `mkinitrd` with options to write the updated file onto your target drive. See `man mkinitrd` for the details.

The trick here is know *which* drivers you'll need. That's going to take some digging on your part. Generally, you'll find the compiled driver modules under:

```
/lib/modules/<kernel-name>/kernel
```

But, it's going to be up to you to figure out which of these your particular hardware actually needs.

In the end, unless the differences in source and target hardware are fairly small/simple, you're typically better off to do a new installation for each class of hardware you run, and create separate image for each of them.

Author

Tim Daneliuk - tbku@tundraware.com

Comments and/or improvements welcome!

Acknowledgements

Several of Novell's terrific Dedicated Support Engineers answered my (often stupid) questions. These guys know SUSE Linux inside out and were most generous with their time and advice. Anything you see here that's right is probably mostly due to them. Anything that's wrong is likely my malfunction. In any case, I owe Aaron Gresko and Jared Hudson a big "Thanks!" for their help.

Document Information

This document was produced using the very useful `reStructuredText` tools in the `docutils` package. For more information, see:

http://docutils.sourceforge.net/rst.html

`$Id: Imaging-SUSE-Linux-With-tbku.txt,v 1.119 2008/03/23 15:41:09 tundra Exp $`