Customizing Red Hat Installation

Made to Fit

If you're one of those users who just wants to slip the CD into the drive and answer the prompts, your are probably content with a Red Hat installation. But if you are someone who wants it your own way, you might be looking for a better fit. We'll show you how to get it. BY ARMIJN HEMEL

Although there are more than 300 Linux distributions, it is still the case that the most widely used distributions are members of the Red Hat family: Red Hat Linux, Fedora, Red Hat Enterprise Linux, and rebuilds of Red Hat distros such as White Box Linux.

One common complaint about Red Hat Linux is that it is inflexible – too much software is installed and the setup process doesn't allow for customizing. This view is held especially by users of distributions that are known for highly customizable installation.

However. Red Hat is, in fact, very flexible when it comes to customization. In this article, I will take a look at various methods for customizing Red Hat installation, ranging from deciding which packages get installed by default, to completely rebuilding the installer.

Before we can start customizing Red Hat Linux, it is necessary to dive a bit deeper into the Red Hat installation process. The Red Hat Linux boot process is divided into two stages.

The Red Hat Linux Installation Process

The first stage consists of booting Linux and creating a RAM disk that is large enough to perform the installation and prepare everything for the second stage.

In the second stage, the Red Hat installer, Anaconda, is started in full graphical mode for a CD install (even if all the packages are downloaded from the network) or in text mode for a full network install via floppies. In the rest of this article we will only consider the graphical installer.

Sometime during the installation process, the user is given the choice of what packages to install, broken down into categories. The user can then decide to install extra packages or categories or to leave some out.

The list of packages and categories is not fixed and compiled into the installer but is generated at runtime by reading in an XML file, called *comps.xml*, which describes the package groups and the packages that belong to them.

The structure of the XML file is fairly straightforward. There is one top level element, the *comps* tag, which has a number of children with the *group* tag.

One thing you'll notice is that not every group mentioned in the *comps.xml* file is actually displayed by Anaconda. Mandatory groups and packages, such as the *core* group, need to be installed anyway, so it it is not much use to give the user a choice.

Every group has a tag called *uservisible*, which Anaconda uses to determine whether a group should be shown or not. A program like kickstart, which we will not discuss here, can also make use of the names of these groups.

After you have done an install of a Red Hat based distribution, there will be a kickstart file called *anaconda-ks.cfg* in the home directory of the root user.

In the *%packages* section in this file, you can find group ids from *comps.xml*, prefixed with the @ symbol.

Each *packagereq* has an extra attribute called *type*, with three possible values: *mandatory*, *default*, and *optional*, for example:

```
<packagereq type="default">⮐
vim-common</packagereq>
```

Packages marked *mandatory* are installed by default but are never shown as a choice to the user, whereas the other two types are.

The packages marked *default* are already selected when a category is chosen and can be unselected. Packages marked *optional* are not selected but can be. By simply changing these attributes, you can control what gets installed from a certain category.

Package dependencies are handled internally by Anaconda, so you don't have to provide an exhaustive list of all

THE AUTHOR

Armijn Hemel is a student of computer science at Utrecht University in the Netherlands, a freelance writer/journalist, and UNIX system administrator.

dependencies to make sure that everything you need is installed. This is only true for versions of Red Hat Linux newer than Red Hat 9.

For Red Hat 9 or older, some additional work needs to be done, but since these versions are already quite old, we will not go into that. The package dependencies are described in two files, *hdlist* and *hdlist2*, which can be found in the same directory as *comps.xml* in the installation tree. Later on we will see how these files can be generated from a collection of RPM files.

After the user has finished selecting packages, Anaconda discovers the dependencies for all selected packages using *hdlist* and *hdlist2*, builds a list of packages to install, and eventually installs the packages. So just by changing *comps.xml*, you can already do a lot of tweaking.

The easiest way to test your changes is to do a network install. Simply set up an FTP site on your network with a full copy of Fedora Core (we used Fedora Core 3), in the same way the official FTP mirrors have. Fedora provides an ISO image to boot from and use for CD installs. The advantage of the CD over floppies is that with the CD, the graphical installer will be booted, but with the floppies, only the text installer will be started.

After modifying the *comps.xml* file, you should overwrite the *comps.xml* file in *$arch/Fedora/base* with your own copy – where *$arch* is either i386 or x86-64, depending on what architecture you have.

Then burn the *boot.iso* boot image to a CD to get the fully graphical installer so you can see your changes, boot from it, and, in the installer, point the download location for the files to the right directory on your FTP server.

Adding Packages

The method mentioned above works, except that it only works well as long as you limit yourself to the packages that are in the distribution. If you want to add other packages that are not in the distribution, you need to do some extra work.

Remember Anaconda resolves dependencies by itself using the two files *hdlist* and *hdlist2*. Simply adding packages to *comps.xml* doesn't work. The installer explicitly uses *hdlist* and *hdlist2*. Regenerating *hdlist* and *hdlist2* is, luckily, not that hard. Before we can begin, we need to install a few packages:

- *comps-extras*
- *anaconda*
- *anaconda-runtime*

The development tools in these packages expect to work on a build tree, which needs to have the following layout:

```
i386/
i386/Fedora/
i386/Fedora/RPMS/
i386/Fedora/SRPMS/
i386/Fedora/base/
```

Next we need to set a few environment variables, which the tools expect to find:

```
export BASE=<parent directory ↩
of our buildtree>
export PYTHONPATH=/usr/lib/↩
anaconda
export PATH=$PATH:/usr/lib/↩
anaconda-runtime
```

The RPMS directory in the build tree should be filled with the Fedora RPMs and any additional RPMs you want to install. To generate *hdlist* and *hdlist2*, you need to use the following command:

```
genhdlist --productpath=↩
Fedora $BASE/i386
```

The *productpath* for nearly all tools defaults to *RedHat*, but this setting can be overridden on the command line if you want to re-brand your distribution.

Before these files can actually be used, two additional steps need to be taken. One of these steps is to determine the order of the packages for the install. Anaconda installs the most important packages first. Without this information, it cannot install and it will abort the installation process.

To generate the *pkgorder* file, you should copy your *comps.xml* to *$BASE/i386/Fedora/base* and run the following commands:

```
pkgorder $BASE/i386 i386 ↩
Fedora > $BASE/pkgorder.txt
genhdlist --fileorder $BASE/↩
pkgorder.txt --productpath=↩
Fedora $BASE/i386/
```

As a final step, we copy all RPMs plus additional packages to the right directory on our FTP site and overwrite *comps.xml*, *hdlist*, and *hdlist2* with our own copies. We can then boot a fresh install with a CD install.

One of the things you have to keep in mind is that you have to make sure that the RPMs you add will install cleanly on the system in the first place. It makes no sense to use RPMs that will not install properly. Always test that packages actually work before you start adding them!

Making Custom CDs

So far we have only used FTP to install our new distribution, but installing from a CD image is often a lot more convenient.

Creating CD images is slightly different compared to the process we saw earlier. Again, we start by creating *hdlist* and *hdlist2* and determine the *pkgorder*:

```
genhdlist --productpath=Fedora ↩
$BASE/i386/
pkgorder $BASE/i386 i386 ↩
Fedora > $BASE/pkgorder.txt
```

Next, the boot images will have to be rebuilt:

```
buildinstall --pkgorder $BASE/↩
pkgorder.txt --comp dist-3 ↩
--product Fedora --version 3 ↩
--release "applepie" --prod↩
path Fedora $BASE/i386
```

For *buildinstall* to work properly, two additional packages should be installed:

- *netpbm-progs*
- *syslinux*

Both packages contain tools that are used in one of the scripts to convert a *PNG* file (which can be found in the *fedora-logos* package) to a *.lss* file for the splash screen at boot time. If you forget to install these, the boot images will be made but they will fail to boot.

Another tool that needs to be installed is *mkisofs*, because it is used to create *boot.iso*. Unfortunately, the scripts don't check if these tools are installed and will, in some cases, happily continue if the tools can't be found.

Because the distribution is larger than one CD, it should be split accordingly:

```
splittree.py --arch i386 ↩
--total-discs 8 --bin-discs 4 ↩
--src-discs 4 --release-string ↩
"Fedora" --pkgorderfile ↩
$BASE/pkgorder.txt --distdir ↩
$BASE/i386 --srcdir ↩
$BASE/i386/Fedora/SRPMS ↩
--productpath Fedora
```

Furthermore, you will have to make absolutely sure that the files in the RPMS and SRPMS are RPM files or the scripts will fail.

Because the distribution is split, *hdlist* and *hdlist2* are not quite accurate anymore and should be rebuilt once again, this time with information specifying which CD holds which package:

```
genhdlist --fileorder ↩
$BASE/pkgorder.txt ↩
--withnumbers ↩
--productpath=Fedora ↩
$BASE/i386-disc[1-4]
```

The numbering is important ! Without it, the installer doesn't know from which CD to get a particular package and you end up swapping CDs a lot.

After that you can make ISO images and burn the CDs. The first CD should be made bootable:

```
mkisofs -b isolinux/isolinux.↩
bin -c isolinux/boot.cat  -J  ↩
-p "info@example.org" -V ↩
"Fedora disc 1" -r  -T  -v  -A ↩
"Fedora/i386 1" -o ../Fedora-↩
disc1.iso  -no-emul-boot -boot-↩
load-size  4 -boot-info-table↩
i386-disc1
```

The other CDs can be burned without the boot options.

If you have ever installed Fedora, you will probably have seen the CD check (and skipped it).

INFO

[1] Red Hat: *www.redhat.com*

[2] Fedora Project: *http://www.redhat.com/fedora/*

[3] White Box Linux: *http://www.whiteboxlinux.org/howto.html*

[4] *http://rau.homedns.org/twiki/bin/view/Anaconda/AnacondaDocumentationProject*

In some cases, such as flaky CD-players or el-cheapo media that you don't entirely trust, it might be a good idea to run the tests though. The check works by implanting an MD5 sum in the ISO file:

```
implantisomd5 Fedora-disc1.iso
```

Before you burn the ISO to CD, you can check it with the *checkisomd5* tool:

```
checkisomd5 --verbose Fedora-↩
disc1.iso
```

If you decide to use checksums (which is highly advised), you will have to make sure you burn it the right way. The checksums will fail if the CD is burned as "track at once" (even though it will install just fine), so you will have to burn it as "disk at once" or "session at once."

Rebuilding *comps.rpm*

When we mentioned that a few files should be overwritten in the *$arch/Fedora/base* directory on our FTP server, we deliberately ignored the *comps.rpm* file. Even though it is used by Anaconda, it isn't that important for an install. However, when the system is up and running, it is used for installing software via one of the menu entries to add and remove software.

You can verify this by adding a few packages to the distribution as described earlier, launching *system-config-packages* from the command line or via the menu, and seeing that your extra software doesn't show up in the menu. In fact, we have used the default Fedora *comps.rpm* file.

The *comps* RPM is created in a few steps. The reason that it can't be done in one step is because not every bit of information that is needed is available beforehand, such as which package is on which CD.

Because the *comps* RPM file has to be on the CDs itself (and thus influences things like how the packages are distributed over the CDs), there is a bit of a chicken and egg problem.

Fedora provides a so called *spec-file* with which the comps RPM file can be regenerated.

The *spec-file* is not quite ready to use. Two variables need to be defined:

```
%define basedir ↩
/path/to/$BASE/i386/Fedora/base
%define compsversion <version>
```

In the case of Fedora Core 3, *compsversion* would be 3. Obviously, the RPM should be on the CDs, so it should be mentioned in *hdlist* and *hdlist2* for Anaconda to find it during the install. These files need to be in the RPM itself, so we have to create a placeholder RPM.

First we generate *hdlist* and *hdlist2*:

```
genhdlist --productpath=Fedora ↩
$BASE/i386
```

The build script inside the *spec-file* expects a few files to be there, including a file called *.discinfo*.

The problem is that this file is not created until *splittree.py* is run. A simple workaround is to touch this file:

```
touch $BASE/i386/.discinfo
```

Next we build the RPM:

```
rpmbuild -bb comps-fedora.↩
spec
```

copy the resulting RPM to *$BASE/i386/Fedora/RPMS* and rerun:

```
genhdlist --productpath=Fedora ↩
$BASE/i386
```

to add it to *hdlist* and *hdlist2*.

Of course, this is just a dummy and it cannot be shipped; in fact, *system-config-packages* will even refuse to run because of the empty *.discinfo* file.

After splitting the distribution and rebuilding *hdlist* and *hdlist2* for inclusion on the first CD, as shown earlier, the *comps* RPM should be rebuilt, but with the *basedir* set to */path/to/$BASE/i386-disc1/Fedora/base*:

```
rpmbuild -bb comps-fedora.spec
```

and copied over the old placeholder RPM, as well as to the base directory for the first CD:

```
cp comps-<somedate>.rpm ↩
$BASE/i386-disc1/Fedora/↩
base/comps.rpm
```

Note that it is renamed to *comps.rpm* because Anaconda expects it to be there during install time.

Rebuilding the Installer

Even though the installation can already be tweaked just by editing the *comps.xml* file and adding/removing packages to the distribution, there are still cases where this is not enough. An example would be if you have two different types of clients that are very similar, but differ slightly, for example, a desktop without compiler and kernel sources and a desktop with a compiler and kernel sources. Even though this might be possible by using just *comps.xml*, this solution is cumbersome and a lot of work.

The much easier solution is to add (or remove) installation targets in Anaconda itself. The source for Anaconda is available as a source RPM, or via CVS. In the source tree, there is a directory called *installclasses* with a few Python files. These files describe the installation targets, such as *Personal Workstation* or *Server*, and are used at runtime to generate the list of classes you can choose from during an install.

Adding your own class is fairly straightforward: you simply copy or adapt an existing class. There are quite a few options you can tweak in these classes, such as whether or not the package selection should be skipped altogether and whether to use the default package selection or not. In an *installclass*, there are a few things that can be overridden.

The *setGroupSelection* method is used for package selection. To add a group, supply its group ID from *comps.xml*.

The *sortPriority* variable is used to determine the place the install class gets in the overview of available install classes. The *id* and *name* should also be changed for your *installclass*.

Finally, the RPM has to be rebuilt. The easiest way to do this is to pack the (modified) sources into a tarball and run *rpmbuild* on it:

```
tar jcvf ⮐
anaconda-10.1.0.2.tar.bz2 ⮐
anaconda-10.1.0.2

rpmbuild -tb ⮐
anaconda-10.1.0.2.tar.bz2
```

To be able to use the new installer classes, you need to copy the RPMs that you just created to the RPM directory before building the complete install images.

Releasing your own Distro

The ultimate act of customization is to create your own distribution. If you really want to roll your own distribution and actually distribute it, you will have to keep in mind that some files in Fedora, such as the artwork, are copyrighted by Red Hat and should be replaced.

Apart from the various legal issues related to trademarks and copyrights, there are also other things you'll need to consider before you start your own distribution, such as how you are going to handle things like updates and support. ■

Advertisment