Release Engineering 101

HOWTO Create Your Own Distribution

How to create your own add-on (third party) repository, (derivative) distribution, or worse

Publican
BOOK PUBLISHING TOOL

Jeroen van Meeuwen

Release Engineering 101 HOWTO Create Your Own Distribution
How to create your own add-on (third party) repository, (derivative) distribution, or worse
Edition 1

Author Jeroen van Meeuwen `jeroen.van.meeuwen@ergo-project.org`

The essentials of Release Engineering.

Preface **v**

1. Introduction v
    1.1. In This HOWTO v
2. Document Conventions v
    2.1. Typographic Conventions v
    2.2. Pull-quote Conventions vii
    2.3. Notes and Warnings vii
3. We Need Feedback! viii

I. Build Systems **1**

1. The Koji Build System **3**

1.1. About Koji 3
    1.1.1. CPU Architectures and Base Architectures 3
    1.1.2. External Repositories 3
    1.1.3. Using Tags 3
    1.1.4. Adding Packages 3
    1.1.5. Adding Users 3
1.2. Setting Up Koji 3
    1.2.1. Prerequisites 3
    1.2.2. Doing the Work 4
1.3. Setup the Koji Build System 7
1.4. Koji Setup for Downstream Consumers 7
    1.4.1. Creating the New Distribution Version FIXME 9
1.5. Third Party Add-on Repositories 10
    1.5.1. Example: Koji for RPM Fusion 10
    1.5.2. Creating a Custom Repository 12
    1.5.3. A more sustainable custom repository 13
    1.5.4. A buildsystem repository for Fedora 12 13

2. The Busby Build System **15**

A. Revision History **17**

Index **19**

Preface

1. Introduction

A short overview and summary of the book's subject and purpose, traditionally no more than one paragraph long. Note: the abstract will appear in the front matter of your book and will also be placed in the description field of the book's RPM spec file.

1.1. In This HOWTO

In this HOWTO, we will set up and environment that allows you to *Create Your Own Distribution*^{TM}. All the way from upstream (non-)source code, to building software packages and enabling those software packages to be checked in to version control, building the (binary) packages, pushing the packages out, distributing them through software channels, and what is the work involved.

This HOWTO also describes what exactly is necessary for the upstream distributor to allow a certain amount of flexibility and efficiency to downstream consumers, whether it be distributors of derivative distributions, add-on (third party) software distribution channels, or specific consumer needs and expectations with regards to update policies.

Ergo, this HOWTO includes chapters on the following:

- The upstream software;
- Working with the upstream software as a downstream consumer;
- The distribution software packaging version control system;
- The build system;
- The distribution mechanism;
- The work included;

2. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

2.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

[1] https://fedorahosted.org/liberation-fonts/

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press `Enter` to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press `Enter` to execute the command.
>
> Press `Ctrl`+`Alt`+`F1` to switch to the first virtual terminal. Press `Ctrl`+`Alt`+`F7` to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

2.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

2.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' won't cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

3. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

Part I. Build Systems

The Koji Build System

This chapter is about the *Koji*[1] build system suite, the successor of *Plague*.

Koji is used by, amongst others, the *Fedora Project*[2] and the *RPM Fusion*[3] third-party add-on software repository for Fedora.

1.1. About Koji

para

1.1.1. CPU Architectures and Base Architectures

i386 for the builders, exactarch for the build tags, basearches for the repositories.

1.1.2. External Repositories

Regenerating the metadata for external repositories

1.1.3. Using Tags

using tags (release, updates & updates-testing and updates-candidate)

1.1.4. Adding Packages

Packages added should not be removed anymore, but in case nothing refers to the package yet, can be removed but not easily.

1.1.5. Adding Users

In case nothing refers to the package yet, can be removed but not easily.

1.2. Setting Up Koji

1.2.1. Prerequisites

You will need at least the following resources:

- A PostgreSQL database;
- A NFS, SMB or iSCSI with GFS(2) fileserver[4];
- A build host (also called *koji builder*).

1.2.1.1. Performance

We recommend considering the following parameters with regards to the Koji build system environment:

[1] http://fedorahosted.org/koji
[2] http://fedoraproject.org
[3] http://rpmfusion.org

- When using the *external repositories* feature, please keep in mind the processes triggered with such usage are memory intensive.
- The storage requirements can grow very quickly, and grows exponentially with the number of builds required, the number of distribution channels used, the external repository mirroring policy, and so forth.

  Even though we need you to make your own assessment and determine how much flexibility you require in terms of being able to grow the storage volume used, we can share the storage in use today by various projects;

  - Fedora Project: approximately 2 TB.
  - RPM Fusion; 200 GB
- Building software is CPU intensive, and can be memory intensive, but most of all is very I/O intensive.

1.2.2. Doing the Work

In this section, we are going to setup a bare metal build system environment. See also *http://fedoraproject.org/wiki/Koji/ServerHowTo*.

In this manual...

In this manual, we set up Koji with SSL certificate authentication, but another way for Koji to authenticate users and builders is through Kerberos. Check *http://fedoraproject.org/wiki/Koji/ServerHowTo* for more information on the latter.

1. Make sure you've met the prerequisites in *Section 1.2.1, “Prerequisites”*.
2. In the case of Enterprise Linux 5 build systems, make sure to configure and enable the *ergo-el5-buildsys* YUM repository.
3. Create the necessary directories for SSL authentication between build nodes, clients and the hub:

```
# mkdir -p /etc/pki/koji/{certs,private}
```

4. Create **`/etc/pki/koji/ssl.cnf`** with the following content:

```
HOME                            = .
RANDFILE                        = .rand

[ca]
default_ca                      = ca_default

[ca_default]
dir                             = .
certs                           = $dir/certs
crl_dir                         = $dir/crl
database                        = $dir/index.txt
new_certs_dir                   = $dir/newcerts
certificate                     = $dir/%s_ca_cert.pem
private_key                     = $dir/private/%s_ca_key.pem
serial                          = $dir/serial
```

```
crl                             = $dir/crl.pem
x509_extensions                 = usr_cert
name_opt                        = ca_default
cert_opt                        = ca_default
default_days                    = 3650
default_crl_days                = 30
default_md                      = md5
preserve                        = no
policy                          = policy_match

[policy_match]
countryName                     = match
stateOrProvinceName             = match
organizationName                = match
organizationalUnitName          = optional
commonName                      = supplied
emailAddress                    = optional

[req]
default_bits                    = 1024
default_keyfile                 = privkey.pem
distinguished_name              = req_distinguished_name
attributes                      = req_attributes
x509_extensions                 = v3_ca # The extentions to add to the self signed cert
string_mask                     = MASK:0x2002

[req_distinguished_name]
countryName                     = Country Name (2 letter code)
countryName_default             = AT
countryName_min                 = 2
countryName_max                 = 2
stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Vienna
localityName                    = Locality Name (eg, city)
localityName_default            = Vienna
0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = My company
organizationalUnitName          = Organizational Unit Name (eg, section)
commonName                      = Common Name (eg, your name or your server\'s hostname)
commonName_max                  = 64
emailAddress                    = Email Address
emailAddress_max                = 64

[req_attributes]
challengePassword               = A challenge password
challengePassword_min           = 4
challengePassword_max           = 20
unstructuredName                = An optional company name

[usr_cert]
basicConstraints                = CA:FALSE
nsComment                       = "OpenSSL Generated Certificate"
subjectKeyIdentifier            = hash
authorityKeyIdentifier          = keyid,issuer:always

[v3_ca]
subjectKeyIdentifier            = hash
authorityKeyIdentifier          = keyid:always,issuer:always
basicConstraints                = CA:true
```

5. Edit **`/etc/pki/koji/ssl.cnf`** and update it with your values.

- Set a naming convention

- Choose the architectures (per branch)
- Make sure that if you run a local mirror of the Fedora Project, *MirrorManager*[5] knows about it and points your builders in the right direction.
- Make sure that if you run a local mirror of CentOS, *MirrorManager*[6] knows about it and points your builders in the right direction.
- Optionally, choose a *dist-tag*

Using the Koji build system in FIXME, we now enable RPM Fusion to build the packages for their **`free`** and **`nonfree`** repositories.

```
koji add-tag dist-f13-release
koji add-external-repo -t dist-f13-release dist-f13-release http://download.fedoraproject.org/
pub/fedora/linux/development/13/\$arch/os/
koji add-tag --parent dist-f13-release dist-f13-release-override

koji add-tag dist-f13-updates
koji add-external-repo -t dist-f13-updates dist-f13-updates http://download.fedoraproject.org/
pub/fedora/linux/updates/13/\$arch/
koji add-tag --parent dist-f13-updates dist-f13-updates-override

koji add-tag dist-f13-updates-testing
koji add-external-repo -t dist-f13-updates-testing dist-f13-updates-testing http://
download.fedoraproject.org/pub/fedora/linux/updates/testing/13/\$arch/
koji add-tag --parent dist-f13-updates-testing dist-f13-updates-testing-override

koji add-tag --parent dist-f13-updates-override dist-f13-build
koji add-tag-inheritance --priority 1 dist-f13-build dist-f13-release-override
koji add-tag-inheritance --priority 2 dist-f13-build dist-f12-build

koji add-tag --arches="i686 x86_64" --parent dist-f13-build rpmfusion-13-free-build

koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-free-release rpmfusion-f13-free-
updates
koji add-tag --arches="i686 x86_64" rpmfusion-f13-nonfree-release
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-nonfree-release rpmfusion-f13-
nonfree-updates
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-free-updates rpmfusion-f13-free-
build
koji add-tag-inheritance --priority=2 rpmfusion-f13-free-build dist-f13-build
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-free-updates rpmfusion-f13-nonfree-
build
koji add-tag-inheritance --priority=1 rpmfusion-f13-nonfree-build rpmfusion-f13-nonfree-
updates
koji add-tag-inheritance --priority=2 rpmfusion-f13-nonfree-build dist-f13-build
koji add-tag-inheritance --priority=2 rpmfusion-f13-free-build dist-f13-build
koji add-tag-inheritance rpmfusion-f13-nonfree-release rpmfusion-f12-nonfree-updates
koji add-tag-inheritance --priority=2 rpmfusion-f13-nonfree-build dist-f13-build
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-free-updates rpmfusion-f13-free-
updates-candidate
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-free-updates rpmfusion-f13-free-
updates-testing
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-nonfree-updates rpmfusion-f13-
nonfree-updates-candidate
koji add-tag --arches="i686 x86_64" --parent rpmfusion-f13-nonfree-updates rpmfusion-f13-
nonfree-updates-testing
koji add-target rpmfusion-f13-free-release rpmfusion-f13-free-build rpmfusion-f13-free-release
koji add-target rpmfusion-f13-nonfree-release rpmfusion-f13-nonfree-build rpmfusion-f13-
nonfree-release
```

```
koji add-target rpmfusion-f13-free-updates rpmfusion-f13-free-build rpmfusion-f13-free-
updates-candidate
koji add-target rpmfusion-f13-nonfree-updates rpmfusion-f13-nonfree-build rpmfusion-f13-
nonfree-updates-candidate
```

1.3. Setup the Koji Build System

In this section, we'll initialize the Koji build system to start out with nothing but existing RPM packages.

See the other Setup sections

This section is not suitable for downstream consumers desiring to fork, override, re-compile, add-on software to an existing Linux distribution or software repository. Please see the other sections if you are not creating your own Linux distribution from scratch.

Procedure 1.1. Starting from Scratch

1. Create the **`-release`** tag, and prefix it with **`devel-`**.
2. Create the **`-release-candidate`** tag, again prefixing it with **`devel-`**.
3. Create the **`-build`** tag, again prefixing it with **`devel-`**
4. Create the **`build`** and **`srpm-build`** groups.
5. Add the packages to the groups mentioned in step #4, including all of their dependencies (minimal build root)
6. Create the build target and prefix it with **`devel-`**

1.4. Koji Setup for Downstream Consumers

In this section, we'll set up the skeleton of a Koji build system using *external repositories*. Please read through the other sections as well, which explain how to set up Koji for different purposes. Yet, this type of skeleton setup enables you to use and reuse readily available software packages, yet allows for a certain amount of flexibility downstream –that is if you set it up right. This functionality is ideal for several intended and/or unintended purposes, including:

- Building your own software packages on top of the set of software packages available, making available extra software while not changing anything to the distribution;
- Building newer versions of software packages to be used instead of the software packages available from upstream, also known as fast-tracking. For instance, compiling **php-5.2.x** for Enterprise Linux 5 would be considered fast-tracking;
- Rebuild the software packages available from upstream differently hence creating a different version of the same package, possibly with a different name, a different set of compilation options, or against a different version of a build requirement.
- Rebuild the software packages available from upstream as-is, for either of several purposes;
  - For fun, or to learn from the experience;

  - Proof of Concept
  - Quality Assurance, including *Fails To Build From Source*, or FTBFS.
  - Security
  - Verification
- Building packages in development for the current release, such as a new package to be included in the distribution, such as software packages not yet accepted by the review process;
- Building packages for (future) feature development purposes, based on the current release, so that the resulting packages may be installed by users running a stable base operating system. For instance, building **ruby-1.9.1** for Fedora 12;
- Building packages for future releases, way beyond what could be done within the current development environment of the distribution. This includes, for example, re-building all software packages against **gcc-4.4** FIXME;
- rebuild packages possibly with extra patches or different options relevant to how the software compiles (ranging from **`./configure --with`** or **`./configure --without`** to **`%patch1000`**)
- fork an existing distribution and not have to redo all the work.

Procedure 1.2. Initialize a Koji Build System Environment with External Repositories

1. Create the distribution tags to represent the external repositories used. In this example, we use the **`dist`** prefix for the tag name to indicate it is a (complete) distribution, the **`f11`** to indicate it is Fedora 11[7], and the **`-release`**[8], **`-updates`** and **`-updates-testing`** suffices to indicate the upstream software distribution channel (or *repository*).

```
koji add-tag dist-f11-release
koji add-external-repo \
    -t dist-f11-release dist-f11-release \
    http://download.fedoraproject.org/pub/fedora/linux/releases/11/Everything/\$arch/os/

koji add-tag dist-f11-updates
koji add-external-repo \
    -t dist-f11-updates dist-f11-updates \
    http://download.fedoraproject.org/pub/fedora/linux/updates/11/\$arch/

koji add-tag dist-f11-updates-testing
koji add-external-repo \
    -t dist-f11-updates-testing dist-f11-updates-testing \
    http://download.fedoraproject.org/pub/fedora/linux/updates/testing/11/\$arch/
```

2. In order to allow you to override what is available from the external repositories, create override tags for each of the distribution tags:

```
koji add-tag --parent dist-f11-release dist-f11-release-override
koji add-tag --parent dist-f11-updates dist-f11-updates-override
koji add-tag --parent dist-f11-updates-testing dist-f11-updates-testing-override
```

   This may seem unnecessary, but rest assured you will need them later on.

3. Now, create a build tag that uses the **`-updates-override`** and **`-release-override`** distribution tags, in that order:

```
koji add-tag --arches="i586 x86_64" --parent=dist-f11-updates-override dist-f11-build
koji add-tag-inheritance --priority 1 dist-f11-build dist-f11-release-override
```

The Inheritance is Important

The aforementioned inheritance is important. Much like with a consuming (end-user) system using **yum-plugin-priorities** or **yum-priorities**, the priority attached to the inheritance of tags tells the build system which external repository has the upper hand when building packages using these external repositories.

4. Now that the *build tag* exists, create two *build groups* in the *build tag*: **`build`** and **`srpm-build`**. Then, associate some packages with it (relating to the *minimal buildroot*):

```
koji add-group dist-f11-build build
koji add-group-pkg dist-f11-build build \
    bash bzip2 coreutils cpio \
    diffutils findutils gawk gcc \
    gcc-c++ grep gzip info \
    make patch redhat-rpm-config rpm-build \
    sed shadow-utils tar unzip \
    util-linux-ng which

koji add-group dist-f11-build srpm-build
koji add-group-pkg dist-f11-build srpm-build \
    bash curl cvs gnupg \
    make redhat-rpm-config rpm-build shadow-utils
```

Not Included In The Build Groups

Not included in the build groups is the package **fedora-release**. This is on purpose, as you may have suspected already. The **fedora-release** package distributes **`/etc/rpm/macros.dist`** which can be used in many cases (such as genuine add-on repositories), but sometimes just doesn't apply (such as fasttrack repositories). The **`%{dist}`** tag makes the **`%{release}`** unique across Koji tags. Hence, we add this package to the appropriate group later on.

Congratulations, you now have the skeleton of a Koji build system environment with external repositories for Fedora 11. But you cannot yet build anything ;-)

1.4.1. Creating the New Distribution Version FIXME

Fedora 11 doesn't last forever.

1. Create the **`-release`**, **`-updates`** and **`-updates-testing`** distribution tags as you would creating the skeleton of a new Koji build system environment setup:

```
koji add-tag dist-f12-release
koji add-external-repo -t dist-f12-release dist-f12-release \
    http://download.fedoraproject.org/pub/fedora/linux/releases/12/Everything/\$arch/os/
```

```
koji add-tag --parent dist-f12-release dist-f12-release-override

koji add-tag dist-f12-updates
koji add-external-repo -t dist-f12-updates dist-f12-updates \
    http://download.fedoraproject.org/pub/fedora/linux/updates/12/\$arch/

koji add-tag --parent dist-f12-updates dist-f12-updates-override

koji add-tag dist-f12-updates-testing
koji add-external-repo -t dist-f12-updates-testing dist-f12-updates-testing \
    http://download.fedoraproject.org/pub/fedora/linux/updates/testing/12/\$arch/

koji add-tag --parent dist-f12-updates-testing dist-f12-updates-testing-override
```

2. Now, add the **`-build`** inheriting the **`-release-override`** and **`-updates-override`**

```
koji add-tag --arches="i686 x86_64" --parent dist-f12-updates-override dist-f12-build
koji add-tag-inheritance --priority 1 dist-f12-build dist-f12-release-override
```

3. As an extra step, inherit the **`dist-f11-build`** tag.

```
koji add-tag-inheritance --maxdepth 1 --priority 2 dist-f12-build dist-f11-build
```

If you set up the tags as described above, the **`dist-f12-build`** build tag now inherits the *build groups* from the **`dist-f11-build`** build tag. It also inherits the **`-release-override`** and **`-updates-override`** tags from **`dist-f11`**, so blocked packages remain blocked. If you want a blocked package from **`dist-f11`** to still appear in **`dist-f12`**, you can unblock the package. Same goes for the ownership of a package (someone other then the owner in **`dist-f11`** may be the owner of the package in **`dist-f12`**) FIXME.

1.5. Third Party Add-on Repositories

In this section, we describe the way to add tags and targets to the Koji environment set up in *Section 1.4, “Koji Setup for Downstream Consumers”*. We'll include some examples, too, such as the one in *Section 1.5.1, “Example: Koji for RPM Fusion”*.

1.5.1. Example: Koji for RPM Fusion

RPM Fusion is the primary third party add-on repository for users of Fedora. It is also one of the most used third party add-on repository for Red Hat Enterprise Linux, Scientific Linux and CentOS users.

Procedure 1.3. Setup the Koji Build System for RPM Fusion

1. First, create a **`-release`** and **`-updates`** tags for the **`free`** repository, and the corresponding **`-override`** tags:

```
koji add-tag rpmfusion-f11-free-release
koji add-tag --parent rpmfusion-f11-free-release \
    rpmfusion-f11-free-release-override

koji add-tag --parent rpmfusion-f11-free-release-override \
    rpmfusion-f11-free-updates

koji add-tag --parent rpmfusion-f11-free-updates \
    rpmfusion-f11-free-updates-override
```

2. Create the **`-build`** tag for the **`free`** repository, inheriting both **`-updates-override`** and **`-release-override`**, in that order:

```
koji add-tag --arches="i586 x86_64" --parent rpmfusion-f11-free-updates-override \
    rpmfusion-11-free-build

koji add-tag-inheritance --priority=1 \
    rpmfusion-f11-free-build dist-f11-build
```

3. Now, create the **`-updates-candidate`** and **`-updates-testing`** tags:

```
koji add-tag --parent rpmfusion-f11-free-updates \
    rpmfusion-f11-free-updates-candidate

koji add-tag --parent rpmfusion-f11-free-updates \
    rpmfusion-f11-free-updates-testing
```

4. We are going to repeat *Step 1*, *Step 2* and *Step 3* for the **`nonfree`** repository, but the **`nonfree`** repository requires a slightly different approach since it inherits the **`free`** repository:

```
koji add-tag rpmfusion-f11-nonfree-release
koji add-tag --parent rpmfusion-f11-nonfree-release \
    rpmfusion-f11-nonfree-release-override

koji add-tag --parent rpmfusion-f11-nonfree-release-override \
    rpmfusion-f11-nonfree-updates

koji add-tag --parent rpmfusion-f11-nonfree-updates \
    rpmfusion-f11-nonfree-updates-override
```

5. Since the software packages in **`nonfree`** can have build requirements that are in the **`free`** repository, we add the **`-free-updates-override`** tag as the parent with the highest priority to the **`-nonfree-build`** build tag:

```
koji add-tag --arches="i586 x86_64" --parent rpmfusion-f11-free-updates-override \
    rpmfusion-11-nonfree-build
```

6. The **`-nonfree-build`** build tag also inherits the updates built and released for the **`nonfree`** repository, through the **`-nonfree-updates-override`** tag:

```
koji add-tag-inheritance --priority=1 rpmfusion-f11-nonfree-build \
    rpmfusion-f11-nonfree-updates-override
```

7. The **`-nonfree-build`** build tag also inherits the **`dist-f11-build`** build tag, from where it gets the settings to populate the build root:

```
koji add-tag-inheritance --priority=2 rpmfusion-f11-nonfree-build \
    dist-f11-build
```

8. Finally, create the **`-updates-candidate`** and **`-updates-testing`** tags for the **`nonfree`** repository:

```
koji add-tag --parent rpmfusion-f11-nonfree-updates \
    rpmfusion-f11-nonfree-updates-candidate

koji add-tag --parent rpmfusion-f11-nonfree-updates \
    rpmfusion-f11-nonfree-updates-testing
```

9. Now, in order to be able to build anything, we need *build targets*:

```
koji add-target rpmfusion-f11-free-release \
    rpmfusion-f11-free-build rpmfusion-f11-free-release

koji add-target rpmfusion-f11-nonfree-release \
    rpmfusion-f11-nonfree-build rpmfusion-f11-nonfree-release

koji add-target rpmfusion-f11-free-updates rpmfusion-f11-free-build \
    rpmfusion-f11-free-updates-candidate

koji add-target rpmfusion-f11-nonfree-updates rpmfusion-f11-nonfree-build \
    rpmfusion-f11-nonfree-updates-candidate
```

1.5.2. Creating a Custom Repository

This is supposed to be a quick-and-fast repository that adds to the Fedora releases it is supposed to be configured on.

Procedure 1.4. A simple, fast custom repository

1.
```
koji add-tag custom-f11-ruby
koji add-tag --arches="i586 x86_64" --parent=custom-f11-ruby custom-f11-ruby-build
koji add-tag-inheritance --priority=1 custom-f11-ruby-build dist-f11-build
```

   We now have a simple build tag that inherits most of the settings from the base **`dist-f11-build`** tag.

2. However, we do not have a *dist-tag* yet. Therefor, we add the **`fedora-release`** package to the **`build`** and **`srpm-build`** groups. In this particular instance (**`custom-f11-build`**) we require no distinguishing dist-tag from other packages.

```
koji add-group-pkg custom-f11-ruby-build build fedora-release
koji add-group-pkg custom-f11-ruby-build srpm-build fedora-release
```

   Note how it is in the downstream repository build tag we add the **`fedora-release`** package to groups **`build`** and **`srpm-build`**.

3. Add the build target:

```
koji add-target custom-f11-ruby custom-f11-ruby-build custom-f11-ruby
```

4.
```
koji add-tag --parent custom-f11-ruby custom-f12-ruby
koji add-tag --arches="i686 x86_64" --parent=custom-f12-ruby custom-f12-ruby-build
koji add-tag-inheritance --priority=1 custom-f12-ruby-build dist-f12-build
koji add-tag-inheritance --priority=2 --maxdepth=1 custom-f12-ruby-build custom-f11-ruby-
build
```

5.
```
koji add-target custom-f12-ruby custom-f12-ruby-build custom-f12-ruby
```

6.
```
koji add-pkg --owner=jmeeuwen custom-f11-ruby rubygem-passenger
```

1.5.3. A more sustainable custom repository

This repository has to last for a little while, and so we make it just a little more sustainable

```
koji add-tag custom-el5-ruby
koji add-tag --parent custom-el5-ruby custom-el5-ruby-updates
koji add-tag --parent custom-el5-ruby-updates custom-el5-ruby-updates-candidate
koji add-tag --parent custom-el5-ruby-updates custom-el5-ruby-updates-testing

koji add-tag --arches="i386 x86_64" --parent=custom-el5-ruby-updates custom-el5-ruby-build
koji add-tag-inheritance --priority=1 custom-el5-ruby-build dist-el5-build

koji add-group-pkg custom-el5-ruby-build build epel-release
koji add-group-pkg custom-el5-ruby-build srpm-build epel-release
```

```

koji add-target custom-el5-ruby-updates-candidate custom-el5-ruby-build custom-el5-ruby-
updates-candidate
```

```
koji add-tag custom-el5-buildsys
koji add-tag --parent custom-el5-buildsys custom-el5-buildsys-updates
koji add-tag --parent custom-el5-buildsys-updates custom-el5-buildsys-updates-candidate
koji add-tag --parent custom-el5-buildsys-updates custom-el5-buildsys-updates-testing

koji add-tag --arches="i386 x86_64" --parent=custom-el5-buildsys-updates custom-el5-buildsys-
build
koji add-tag-inheritance --priority=1 custom-el5-buildsys-build dist-el5-build

koji add-group-pkg custom-el5-buildsys-build build epel-release
koji add-group-pkg custom-el5-buildsys-build srpm-build epel-release
```

```

koji add-target custom-el5-ruby-updates-candidate custom-el5-ruby-build custom-el5-ruby-
updates-candidate
```

1.5.4. A buildsystem repository for Fedora 12

```
[jmeeuwen@ghandalf SPECS]$ koji add-tag custom-f12-buildsys
[jmeeuwen@ghandalf SPECS]$ koji add-tag --arches="i686 x86_64" --parent custom-f12-buildsys
 custom-f12-buildsys-build
[jmeeuwen@ghandalf SPECS]$ koji add-tag-inheritance --priority=1 custom-f12-buildsys-build
 dist-f12-build
[jmeeuwen@ghandalf SPECS]$ koji add-group-pkg custom-f12-buildsys-build build buildsys-macros
[jmeeuwen@ghandalf SPECS]$ koji add-group-pkg custom-f12-buildsys-build srpm-build buildsys-
macros
[jmeeuwen@ghandalf SPECS]$ koji add-pkg --owner=jmeeuwen custom-f12-buildsys buildsys-macros
[jmeeuwen@ghandalf SPECS]$ koji add-target custom-f12-buildsys custom-f12-buildsys-build
 custom-f12-buildsys
```

The Busby Build System

Upstream software projects answer to a greater audience of which you are just a small part.

Appendix A. Revision History

Revision 0	**Mon Mar 22 2010**	**Dude McPants** *`Dude.McPants@example.com`*
Initial creation of book by publican		

Index

F

feedback
    contact information for this manual, viii