Roll Developer's Guide

7.0 Edition

Roll Developer's Guide:
7.0 Edition
Published Dec 01 2017
Copyright © 2017 University of California

This document is subject to the Rocks® License (see Rocks Copyright).

Table of Contents

List of Figures

Chapter 1. Overview

The Roll Developer's Guide is for people who wish to build their own rolls. In this document, we'll describe the internals of rolls, how to create a roll and how to test a roll.

Throughout the discussion, it's easy to lose sight of the basic simplicity of a roll. A roll consists of

- Software packages (RPMS)
- How to configure packages after installation (XML Node Files)
- On which types of appliances to install/configure packages (XML Graph)

Chapter 2. Roll Internals

This section describes in detail the two major components of a roll: packages and configuration files.

2.1. Roll Development Environment

In this section, we'll build an example roll (the Valgrind Roll) from the ground up.

First, we need to create a development environment that can be used to build the Valgrind Roll. On a Rocks frontend, execute:

```
# mkdir -p /export/src/roll
# cd /export/src/roll
# rocks create new roll name=valgrind
```

There are several more options you can supply to "`rocks create new roll`" (execute "`rocks create new roll help`" to see all the options).

The command "`rocks create new roll name=valgrind`" creates the following file system:

```
<9492><9472><9472> valgrind
    <9500><9472><9472> graphs
    <9474>Â Â  <9492><9472><9472> default
    <9500><9472><9472> nodes
    <9492><9472><9472> src
        <9500><9472><9472> usersguide
        <9474>Â Â  <9492><9472><9472> images
     Â Â  <9492><9472><9472> valgrind
     Â Â      <9492><9472><9472> Makefile
     Â Â      <9492><9472><9472> test-1.0.tgz
     Â Â      <9500><9472><9472> version.mk
```

At this point, the roll is "buildable". It doesn't actually do much, but the structure is functional. Type `make roll` to build the roll. When the build completes, a `valgrind-1.0-0.<arch>.disk1.iso` file should be in the build directory. This is the complete roll image.

Now we are ready to add packages to the Valgrind Roll.

2.2. Packages

A Rocks roll requires that all packages contained in the roll must be in the native format of the OS. For Redhat-based Rocks clusters, this means all packages must be RPMS.

This guide covers a few ways to deal with RPMs. Already built, hand-built in a directory, and compiled RPMs. The last (compiled RPMs) is by far the most common.

2.2.1. The Software is Already in an RPM

If the software you wish to install is already in RPM format, you can directly put it into the roll source tree. For example, to put the RPM `unzip-5.52-3.el5.x86_64.rpm` into the Valgrind Roll, execute:

```
# mkdir -p /export/src/roll/valgrind/RPMS/x86_64
# cp /tmp/unzip-5.52-3.el5.x86_64.rpm /export/src/roll/valgrind/RPMS/x86_64
```

2.2.2. Bundle an Existing Subdirectory into an RPM

If your application is already installed on your frontend and you would like to bundle its subdirectory into and RPM, you can create an RPM that contains all the files in a subdirectory. For example, let's say you want to create an RPM from all the files under /opt/stream. You can execute:

```
# rocks create package /opt/stream stream
```

This will create a package named `stream-1.0-1.x86_64.rpm` in the current working directory. To see the contents of the package, execute:

```
# rpm -qlp stream-1.0-1.x86_64.rpm
/
/opt
/opt/stream
/opt/stream/bin
/opt/stream/bin/stream
/opt/stream/bin/stream_f
/opt/stream/docs
/opt/stream/docs/HISTORY.txt
/opt/stream/docs/LICENSE.txt
/opt/stream/docs/Makefile
/opt/stream/docs/READ.ME
/opt/stream/docs/ROCKS.txt
/opt/stream/docs/linux.mk
/opt/stream/docs/mysecond.c
/opt/stream/docs/stream.c
/opt/stream/docs/stream.f
/opt/stream/docs/sunos.mk
/opt/stream/docs/version.mk
```

There are several more options you can supply to "`rocks create package`" (execute "`rocks create package help`" to see all the options).

Now place the RPM into the correct directory within the roll:

```
# mkdir -p /export/src/roll/valgrind/RPMS/x86_64
# cp stream-1.0-1.x86_64.rpm /export/src/roll/valgrind/RPMS/x86_64
```

2.2.3. Create an RPM from a Source Code Tarball (Compiled RPM)

The most common way we create RPMS is from source tarballs (the classic: untar, ./configure, make, make install). Rocks has a set of Makefile includes that greatly simplifies the creation of RPMs. Most software can be packaged as an RPM without ever seeing the internals of the package format.

First, we'll download the source tarball into the correct directory:

```
# cd /export/src/roll/valgrind/src/valgrind
# wget http://valgrind.org/downloads/valgrind-3.6.0.tar.bz2
```

Edit `version.mk` and change `NAME = test` to `NAME = valgrind`, change `TARBALL_POSTFIX = tgz` to `TARBALL_POSTFIX = tar.bz2`, change `PKGROOT = /opt/valgrind` to `PKGROOT = /opt`,and change `VERSION = 1.0` to `VERSION = 3.6.0`. Your version.mk should look like:

```
PKGROOT  = /opt
NAME     = valgrind
VERSION  = 3.6.0
RELEASE  = 1
TARBALL_POSTFIX = tar.bz2
```

Edit `Makefile` and change the line: `gunzip -c` to `bzcat` (since the tarball is a bz2).

Now build the RPM:

```
# make rpm
```

It will take several minutes to build the package. When it completes, you'll see the line:

```
Wrote: /export/src/roll/valgrind/RPMS/x86_64/valgrind-3.6.0-1.x86_64.rpm
```

You can inspect the contents of the `valgrind` RPM:

```
# rpm -qlp /export/src/roll/valgrind/RPMS/x86_64/valgrind-3.6.0-1.x86_64.rpm
/
/opt
/opt/valgrind
/opt/valgrind/bin
/opt/valgrind/bin/callgrind_annotate
/opt/valgrind/bin/callgrind_control
/opt/valgrind/bin/cg_annotate
/opt/valgrind/bin/cg_diff
/opt/valgrind/bin/cg_merge
/opt/valgrind/bin/ms_print
/opt/valgrind/bin/no_op_client_for_valgrind
/opt/valgrind/bin/valgrind
/opt/valgrind/bin/valgrind-listener
/opt/valgrind/include
/opt/valgrind/include/valgrind
/opt/valgrind/include/valgrind/callgrind.h
   .
   .
   .
```

Note that the valgrind-3.6.0-1.x86_64.rpm RPM was automatically placed into the correct directory (/export/src/roll/valgrind/RPMS/x86_64).

If you want to pass different `configure` options to for your build, edit your `Makefile`.

2.2.4. The RPM Makefile Process

The Rocks Makefile include structure is invoked from the sample Makefile in the lines.

```
-include $(ROCKSROOT)/etc/Rules.mk
include Rules.mk
```

The main purpose of this structure is to create the correct files so that the `rpmbuild` can work properly. The core of Rocks builds more than 300 different packages, and our goal was to eliminate the creation of custom RPM spec files. Spec files are the text files that drive the overall package creation.

Let's look at the other directories created on demand when you type `make rpm`. The roll directory structure now looks like before:

```
.
<9492><9472><9472> valgrind
    <9500><9472><9472> graphs
    <9474>Â Â  <9492><9472><9472> default
    <9500><9472><9472> nodes
    <9492><9472><9472> src
        <9500><9472><9472> usersguide
        <9474>Â Â  <9492><9472><9472> images
        <9492><9472><9472> valgrind
```

and AFTER `make rpm` is executed.

```
<9492><9472><9472> valgrind
    <9500><9472><9472> BUILD
    <9474>Â Â  <9492><9472><9472> valgrind-3.6.0
    <9500><9472><9472> graphs
    <9474>Â Â  <9492><9472><9472> default
    <9500><9472><9472> nodes
    <9500><9472><9472> RPMS
    <9474>Â Â  <9500><9472><9472> noarch
    <9474>Â Â  <9492><9472><9472> x86_64
    <9500><9472><9472> SOURCES
    <9500><9472><9472> SPECS
    <9500><9472><9472> src
    <9474>Â Â  <9500><9472><9472> usersguide
    <9474>Â Â  <9492><9472><9472> valgrind
    <9492><9472><9472> SRPMS
```

The important directories are `BUILD, SOURCES, SPECS, and RPMS`. This is the structure that `rpmbuild` is expecting. It is worth looking at the contents of the directories in some detail.

- SPECS/valgrind.spec - created automatically. Can create in the src/valgrid with `make valgrind.spec`
- SOURCES/valgrind-3.6.0.tar.gz - This is a tar of src/valgrind directory after the `make pretar` target (if present) has been evaluated. This becomes the "source" referenced in the spec file.
- BUILD/valgrind-3.6.0 - this is where the actual build/compile takes place.

Let's take a look at the contents of the BUILD directory

```
BUILD
<9492><9472><9472> valgrind-3.6.0
    <9500><9472><9472> _arch
    <9500><9472><9472> _distribution
```

```
<9500><9472><9472> Makefile
<9500><9472><9472> _os
<9500><9472><9472> python.mk
<9500><9472><9472> rocks-version-common.mk
<9500><9472><9472> rocks-version.mk
<9500><9472><9472> Rules-install.mk
<9500><9472><9472> Rules-linux-centos.mk
<9500><9472><9472> Rules-linux.mk
<9500><9472><9472> Rules.mk
<9500><9472><9472> Rules-rcfiles.mk
<9500><9472><9472> Rules-scripts.mk
<9500><9472><9472> test-1.0.tgz
<9500><9472><9472> valgrind-3.6.0
<9500><9472><9472> valgrind-3.6.0.tar.bz2
<9492><9472><9472> version.mk
```

The `Makefile` is the *same* file as the `src/valgrind` directory. The file `valgrind-3.6.0.tar.bz2` is also the same.

2.2.4.1. How the SPEC file does it's building/installing

Ultimately, the program `rpmbuild` must be called to do its work. The generated spec file (valgrind.spec) is used to drive the rpmbuild process. RPM spec files can be very complicated, Rocks takes a very simple approach, essentially ignorning many of the advanced capabilities of RPM specification.

What follows here is an abbreviated spec file. It's beyond this guide to describe in detail options in a spec file. A number of web resources are available if more in-depth information is desired.

```
Summary: valgrind
Name: valgrind
Version: 3.6.0
Release: 1
Source: valgrind-3.6.0.tar.gz
Buildroot: /export/src/roll/valgrind/src/valgrind/valgrind.buildroot
%description
valgrind
%prep
%setup
%build
printf "\n\n\n### build ###\n\n\n"
BUILDROOT=/export/src/roll/valgrind/src/valgrind/valgrind.buildroot make -f /export/src/roll/valg
%install
printf "\n\n\n### install ###\n\n\n"
BUILDROOT=/export/src/roll/valgrind/src/valgrind/valgrind.buildroot make -f /export/src/roll/valg
%files
/
```

The generated spec file must have `%build` and `%install` sections to, respectively, build and install software. The concept is very simple, the `%build` section of the spec simply directs `rpmbuild` to call `make build`.

The last section is `%files` which takes "/" as the default. This will package ALL files that have installed into the BUILDROOT directory.

2.3. Configuring Software with XML Files

Rocks generates kickstart files for compute nodes dynamically using a structure called the "kickstart graph". This graph is made from graph XML files and node XML files. In general, a node XML file contains a list of packages that should be installed on a host and commands to configure those packages. Graph XML files contain a description of "edges" that tie the node XML files together.

When a host asks for its configuration file from the frontend, a process on the frontend traverses the node XML files based on the definitions within the graph XML files. In addition, the graph XML files can enforce a relative order between the node XML files.

2.3.1. Node XML Files

All software configuration commands are contained within node XML files. Let's look at a real node XML (`grub.xml` from the Base Roll):

```
<?xml version="1.0" standalone="no"?>

<kickstart>

  <description>

  Boot loader support (GRand Unified Bootloader)

  </description>


<package>grub</package>


<post>

<!-- take out the splashscreen -->
<file name="/tmp/grub.conf" expr="grep -v splashimage /boot/grub/grub.conf"/>
mv /tmp/grub.conf /boot/grub/grub.conf

<!-- Preserve the original grub.conf -->
cp /boot/grub/grub.conf /boot/grub/grub-orig.conf

</post>


</kickstart>
```

We see that the above node XML file will install the "grub" package (`<package>grub</package>`), and in the "post" configuration step during a kickstart installation, it will modify the configuration file (`/boot/grub/grub.conf`).

During a kickstart installation, all the packages from all the node XML files are installed first, then all the "post" sections are executed. That is, when you write a post section, you are guaranteed that all the software packages have been installed.

For a complete definition of the node XML file syntax, see Node XML Tags.

For our Valgrind example, we will modify the node XML file that was already created for us:

```
# cd /export/src/roll/valgrind/nodes
# mv valgrind.xml valgrind-base.xml
```

A node XML file naming convention in Rocks is to begin the name all the node XML files with the name of the roll they are associated with. For example, all node XML files in the SGE roll begin with "sge-".

Another naming convention is to add the name "client", "server" or "base" to the name of the node XML file based on which appliance type the node XML file will be applied to. For example, if the node XML file is intended to configure only backend appliances (e.g., compute nodes or tile nodes), then we'd name our node XML file "valgrind-client.xml". If the node XML file is intended to configure only frontend appliances, then we'd name our node XML file "valgrind-server.xml". And if the node XML file is for all appliance types, we'd name it "valgrind-base.xml".

When you edit "valgrind-base.xml", you'll see:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

        <description>
        Your valgrind roll description here
        </description>

        <copyright>

            Rocks(r)
                   www.rocksclusters.org
                   version 6.2 (SideWinder)
                   version 7.0 (Manzanita)

        Copyright (c) 2000 - 2017 The Regents of the University of California.
        All rights reserved.

        Redistribution and use in source and binary forms, with or without
        modification, are permitted provided that the following conditions are
        met:

        1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

        2. Redistributions in binary form must reproduce the above copyright
        notice unmodified and in its entirety, this list of conditions and the
        following disclaimer in the documentation and/or other materials provided
        with the distribution.

        3. All advertising and press materials, printed or electronic, mentioning
        features or use of this software must display the following acknowledgement:

         "This product includes software developed by the Rocks(r)
         Cluster Group at the San Diego Supercomputer Center at the
         University of California, San Diego and its contributors."
```

```
4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission.  The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

</copyright>

<changelog>
$Log: internals.sgml,v $
Revision 1.11  2013/01/16 04:09:36  phil
Updates/more detail of internals

Revision 1.10  2012/11/27 00:48:34  phil
Copyright Storm for Emerald Boa

Revision 1.9  2012/05/06 05:48:40  phil
Copyright Storm for Mamba

Revision 1.8  2011/07/23 02:30:44  phil
Viper Copyright

Revision 1.7  2011/02/08 21:59:41  bruno
edits

Revision 1.6  2011/02/07 23:30:26  bruno
first pass at build section

Revision 1.5  2011/02/07 20:48:46  bruno
the first draft of 'roll internals' is done.

Revision 1.4  2011/02/05 01:04:49  bruno
checkpoint

</changelog>

<package>valgrind</package>
<package>roll-valgrind-usersguide</package>
```

```
</kickstart>
```

Notice above that the "valgrind" and "roll-valgrind-usersguide" RPMs are specified in `<package>` tags which means those RPMs will be installed by the Red Hat installer.

Now let's add a `<post>` section to it:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

        <description>
        Your valgrind roll description here
        </description>

        <copyright>

            Rocks(r)
                   www.rocksclusters.org
                   version 6.2 (SideWinder)
                   version 7.0 (Manzanita)

        Copyright (c) 2000 - 2017 The Regents of the University of California.
        All rights reserved.

        Redistribution and use in source and binary forms, with or without
        modification, are permitted provided that the following conditions are
        met:

        1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

        2. Redistributions in binary form must reproduce the above copyright
        notice unmodified and in its entirety, this list of conditions and the
        following disclaimer in the documentation and/or other materials provided
        with the distribution.

        3. All advertising and press materials, printed or electronic, mentioning
        features or use of this software must display the following acknowledgement:

         "This product includes software developed by the Rocks(r)
         Cluster Group at the San Diego Supercomputer Center at the
         University of California, San Diego and its contributors."

        4. Except as permitted for the purposes of acknowledgment in paragraph 3,
        neither the name or logo of this software nor the names of its
        authors may be used to endorse or promote products derived from this
        software without specific prior written permission.  The name of the
        software includes the following terms, and any derivatives thereof:
        "Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
        the associated name, interested parties should contact Technology
        Transfer & Intellectual Property Services, University of California,
        San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
        Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

        THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS"
        AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
        THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
```

```
        PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
        BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
        CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
        SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
        BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
        WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
        OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
        IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

        </copyright>

        <changelog>
        $Log: internals.sgml,v $
        Revision 1.11  2013/01/16 04:09:36  phil
        Updates/more detail of internals

        Revision 1.10  2012/11/27 00:48:34  phil
        Copyright Storm for Emerald Boa

        Revision 1.9  2012/05/06 05:48:40  phil
        Copyright Storm for Mamba

        Revision 1.8  2011/07/23 02:30:44  phil
        Viper Copyright

        Revision 1.7  2011/02/08 21:59:41  bruno
        edits

        Revision 1.6  2011/02/07 23:30:26  bruno
        first pass at build section

        Revision 1.5  2011/02/07 20:48:46  bruno
        the first draft of 'roll internals' is done.

        Revision 1.4  2011/02/05 01:04:49  bruno
        checkpoint

        </changelog>

        <package>valgrind</package>
        <package>roll-valgrind-usersguide</package>

<post>

<file name="/etc/motd" mode="append">

This node has "valgrind" configured for it.

</file>

</post>

</kickstart>
```

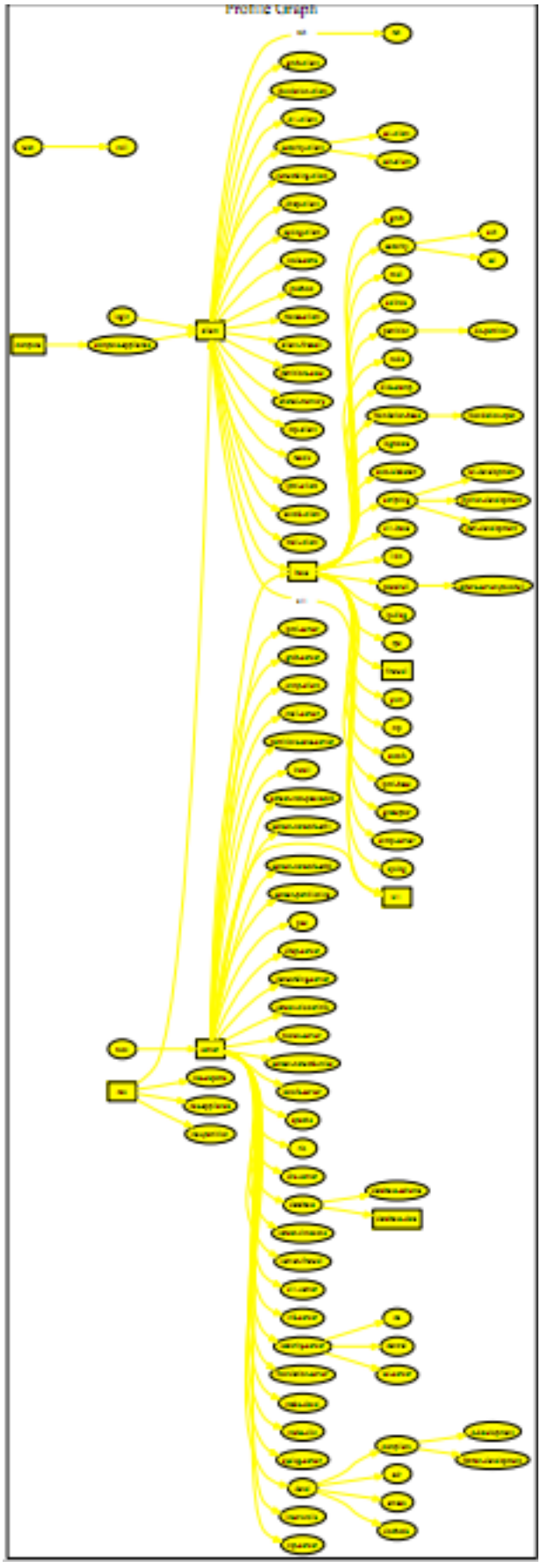
This post section will append a small note to an installing host's `/etc/motd`.

Now that we have our first node XML file, we need to "splice" it into the Rocks kickstart graph -- which is the subject of the next section.

2.3.2. Graph XML Files

Every roll has node XML files and a graph XML file. Node XML files describe what packages should be installed and how those packages should be configured. A graph XML file describes how all the node XML files are "connected", that is, a graph XML file describes the "edges" between nodes.

Below is a picture of how the Base Roll’s nodes are connected together via its graph XML file:


Profile Graph

Let's "splice" our "valgrind-base.xml" node XML file into the Rocks kickstart graph. We'll look at the default graph file that was automatically created for us:

```
# cd /export/src/roll/valgrind/graphs/default
```

Now edit the file "valgrind.xml":

```
<?xml version="1.0" standalone="no"?>

<graph>

        <description>

        The valgrind Roll

        </description>

        <copyright>

            Rocks(r)
                   www.rocksclusters.org
                   version 6.2 (SideWinder)
                   version 7.0 (Manzanita)

        Copyright (c) 2000 - 2017 The Regents of the University of California.
        All rights reserved.

        Redistribution and use in source and binary forms, with or without
        modification, are permitted provided that the following conditions are
        met:

        1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

        2. Redistributions in binary form must reproduce the above copyright
        notice unmodified and in its entirety, this list of conditions and the
        following disclaimer in the documentation and/or other materials provided
        with the distribution.

        3. All advertising and press materials, printed or electronic, mentioning
        features or use of this software must display the following acknowledgement:

         "This product includes software developed by the Rocks(r)
         Cluster Group at the San Diego Supercomputer Center at the
         University of California, San Diego and its contributors."

        4. Except as permitted for the purposes of acknowledgment in paragraph 3,
        neither the name or logo of this software nor the names of its
        authors may be used to endorse or promote products derived from this
        software without specific prior written permission.  The name of the
        software includes the following terms, and any derivatives thereof:
        "Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
        the associated name, interested parties should contact Technology
        Transfer & Intellectual Property Services, University of California,
        San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
        Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu
```

```
        THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS"
        AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
        THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
        PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
        BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
        CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
        SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
        BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
        WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
        OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
        IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

        </copyright>

        <changelog>
        $Log: internals.sgml,v $
        Revision 1.11  2013/01/16 04:09:36  phil
        Updates/more detail of internals

        Revision 1.10  2012/11/27 00:48:34  phil
        Copyright Storm for Emerald Boa

        Revision 1.9  2012/05/06 05:48:40  phil
        Copyright Storm for Mamba

        Revision 1.8  2011/07/23 02:30:44  phil
        Viper Copyright

        Revision 1.7  2011/02/08 21:59:41  bruno
        edits

        Revision 1.6  2011/02/07 23:30:26  bruno
        first pass at build section

        Revision 1.5  2011/02/07 20:48:46  bruno
        the first draft of 'roll internals' is done.

        Revision 1.4  2011/02/05 01:04:49  bruno
        checkpoint

        </changelog>

        <!-- add edges here -->

</graph>
```

We will add an "edge". Since we want our "valgrind-base.xml" node XML file to be installed on all nodes, we'll make an edge from the "base" node XML file to "valgrind-base":

```
<?xml version="1.0" standalone="no"?>

<graph>

        <description>

        The valgrind Roll
```

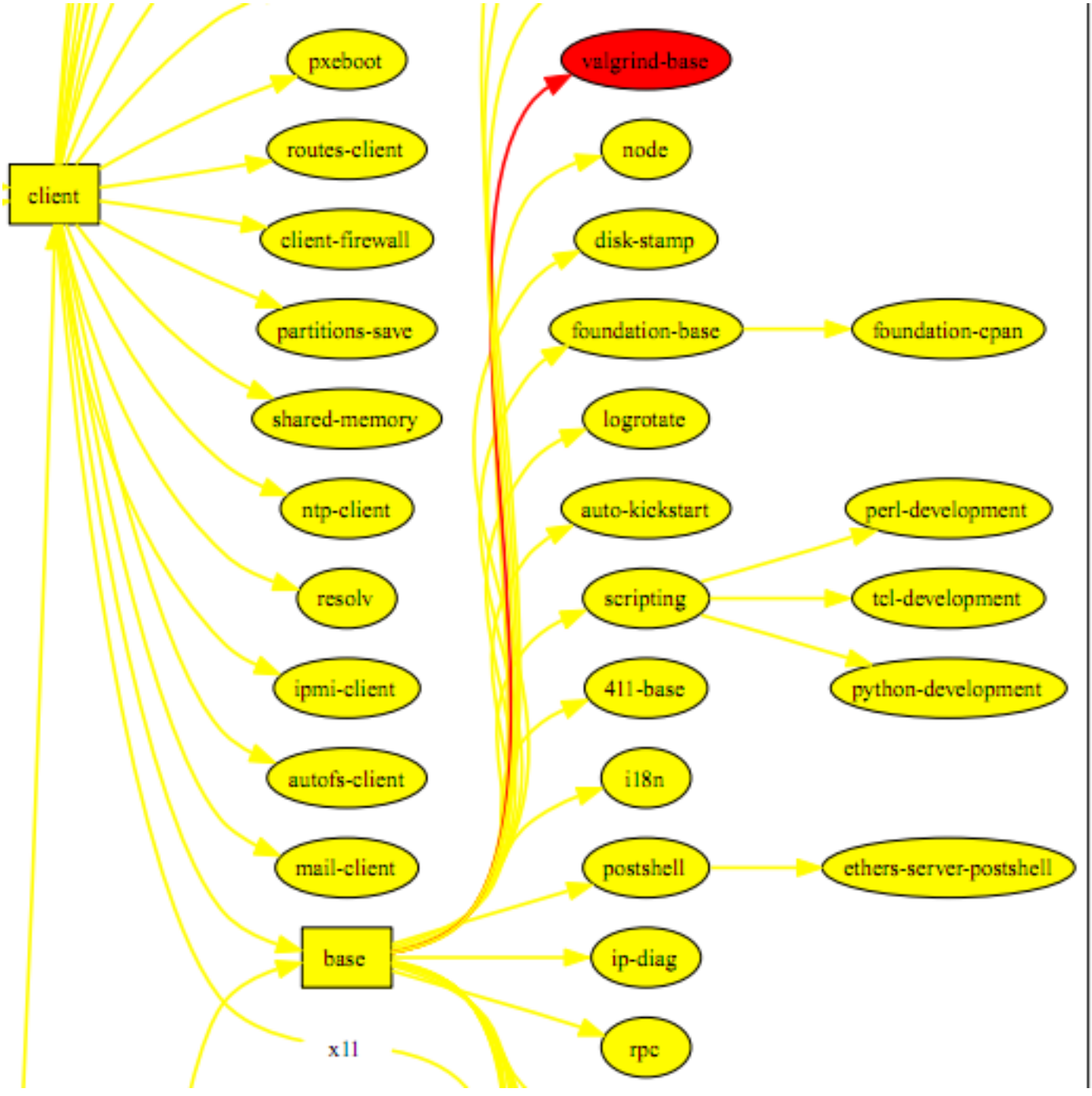
```
</description>

<copyright>

    Rocks(r)
           www.rocksclusters.org
           version 6.2 (SideWinder)
           version 7.0 (Manzanita)

Copyright (c) 2000 - 2017 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice unmodified and in its entirety, this list of conditions and the
following disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising and press materials, printed or electronic, mentioning
features or use of this software must display the following acknowledgement:

 "This product includes software developed by the Rocks(r)
 Cluster Group at the San Diego Supercomputer Center at the
 University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission.  The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

</copyright>

<changelog>
```

```
        $Log: internals.sgml,v $
        Revision 1.11  2013/01/16 04:09:36  phil
        Updates/more detail of internals

        Revision 1.10  2012/11/27 00:48:34  phil
        Copyright Storm for Emerald Boa

        Revision 1.9  2012/05/06 05:48:40  phil
        Copyright Storm for Mamba

        Revision 1.8  2011/07/23 02:30:44  phil
        Viper Copyright

        Revision 1.7  2011/02/08 21:59:41  bruno
        edits

        Revision 1.6  2011/02/07 23:30:26  bruno
        first pass at build section

        Revision 1.5  2011/02/07 20:48:46  bruno
        the first draft of 'roll internals' is done.

        Revision 1.4  2011/02/05 01:04:49  bruno
        checkpoint

        </changelog>

        <!-- add edges here -->

        <edge from="base">
                <to>valgrind-base</to>
        </edge>

</graph>
```

Note that we don't add the ".xml" file extension in edge descriptions.

Our node XML file has now been spliced into the Rocks kickstart graph. Note: the node XML file "base" is from the Base Roll.

In the picture below, we see how "valgrind-base.xml" has been spliced into the Rocks kickstart graph:



2.3.2.1. Controlling the Order of Post Section Execution

There are instances when we need to ensure that a post section in one node XML file executes before (or after) a post section in another node XML file. We can accomplish this by specifying an `<order>` tag in a graph XML file. To show how this is done, we'll create two new node XML files and then we'll edit the graph XML file.

We will create two new node XML files where one file should be applied to a frontend (named "valgrind-server.xml") and the other should be applied to the backend appliances (named "valgrind-client.xml). We'll add `<package>` and `<post>` tags to both.

Here's the contents of "valgrind-client.xml":

```
<?xml version="1.0" standalone="no"?>

<kickstart>

 <description>
 Valgrind client node XML file. This file should be applied to
 backend appliances (e.g., compute nodes and tile nodes).
 </description>

 <copyright>

     Rocks(r)
            www.rocksclusters.org
            version 6.2 (SideWinder)
            version 7.0 (Manzanita)

 Copyright (c) 2000 - 2017 The Regents of the University of California.
 All rights reserved.

 Redistribution and use in source and binary forms, with or without
 modification, are permitted provided that the following conditions are
 met:

 1. Redistributions of source code must retain the above copyright
 notice, this list of conditions and the following disclaimer.

 2. Redistributions in binary form must reproduce the above copyright
 notice unmodified and in its entirety, this list of conditions and the
 following disclaimer in the documentation and/or other materials provided
 with the distribution.

 3. All advertising and press materials, printed or electronic, mentioning
 features or use of this software must display the following acknowledgement:

  "This product includes software developed by the Rocks(r)
  Cluster Group at the San Diego Supercomputer Center at the
  University of California, San Diego and its contributors."

 4. Except as permitted for the purposes of acknowledgment in paragraph 3,
 neither the name or logo of this software nor the names of its
 authors may be used to endorse or promote products derived from this
 software without specific prior written permission.  The name of the
 software includes the following terms, and any derivatives thereof:
 "Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
 the associated name, interested parties should contact Technology
 Transfer & Intellectual Property Services, University of California,
 San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
 Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

 THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
 AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
 BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
 BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
```

```
 WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
 OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
 IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

 </copyright>

 <changelog>
 $Log: internals.sgml,v $
 Revision 1.11  2013/01/16 04:09:36  phil
 Updates/more detail of internals

 Revision 1.10  2012/11/27 00:48:34  phil
 Copyright Storm for Emerald Boa

 Revision 1.9  2012/05/06 05:48:40  phil
 Copyright Storm for Mamba

 Revision 1.8  2011/07/23 02:30:44  phil
 Viper Copyright

 Revision 1.7  2011/02/08 21:59:41  bruno
 edits

 Revision 1.6  2011/02/07 23:30:26  bruno
 first pass at build section

 Revision 1.5  2011/02/07 20:48:46  bruno
 the first draft of 'roll internals' is done.

 </changelog>

<post>

<file name="/etc/motd" mode="append">

Valgrind on a "client".

</file>

</post>

</kickstart>
```

And here is the contents of "valgrind-server.xml":

```
<?xml version="1.0" standalone="no"?>

<kickstart>

 <description>
 Valgrind server node XML file. This file should be applied to
 frontends.
 </description>

 <copyright>

     Rocks(r)
```

```
           www.rocksclusters.org
           version 6.2 (SideWinder)
           version 7.0 (Manzanita)

Copyright (c) 2000 - 2017 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice unmodified and in its entirety, this list of conditions and the
following disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising and press materials, printed or electronic, mentioning
features or use of this software must display the following acknowledgement:

 "This product includes software developed by the Rocks(r)
 Cluster Group at the San Diego Supercomputer Center at the
 University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission.  The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

</copyright>

<changelog>
$Log: internals.sgml,v $
Revision 1.11  2013/01/16 04:09:36  phil
Updates/more detail of internals

Revision 1.10  2012/11/27 00:48:34  phil
```

```
 Copyright Storm for Emerald Boa

 Revision 1.9  2012/05/06 05:48:40  phil
 Copyright Storm for Mamba

 Revision 1.8  2011/07/23 02:30:44  phil
 Viper Copyright

 Revision 1.7  2011/02/08 21:59:41  bruno
 edits

 Revision 1.6  2011/02/07 23:30:26  bruno
 first pass at build section

 Revision 1.5  2011/02/07 20:48:46  bruno
 the first draft of 'roll internals' is done.

 </changelog>

 <package>roll-valgrind-usersguide</package>

<post>

<file name="/etc/motd" mode="append">

Valgrind on a "server".

</file>

</post>

</kickstart>
```

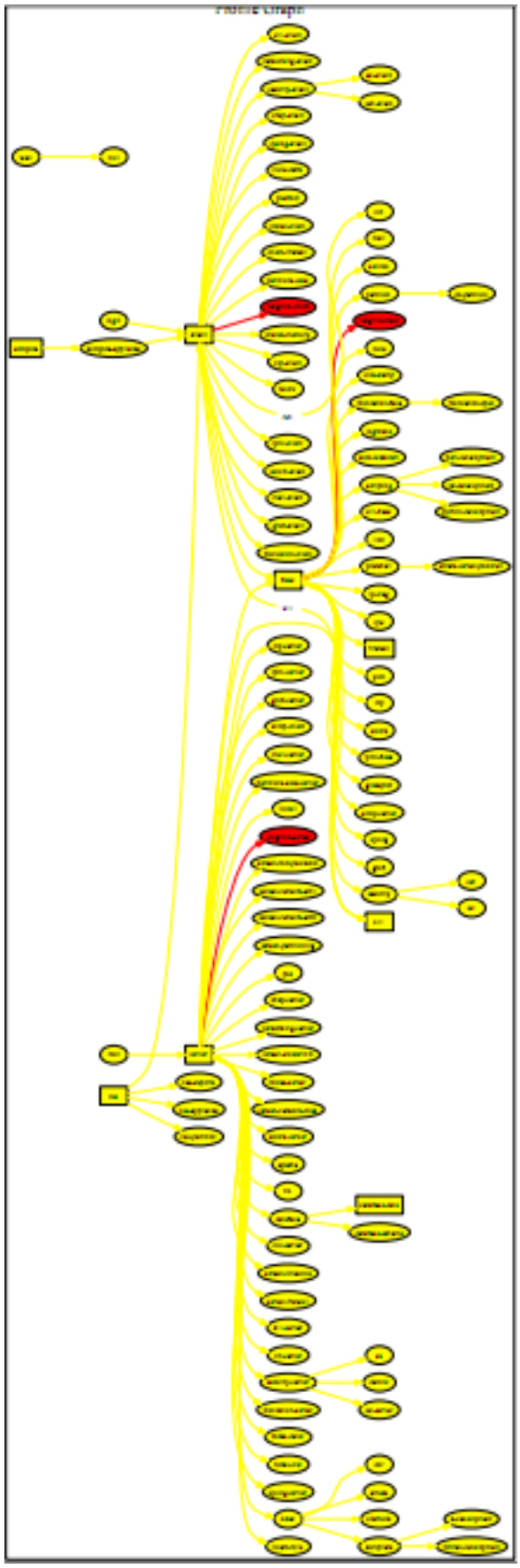
Then, to splice them in to the Rocks kickstart graph, we'll modify our graph XML file "valgrind.xml" to look like:

```
<?xml version="1.0" standalone="no"?>

<graph>

        <description>

        The valgrind Roll

        </description>

        <copyright>

            Rocks(r)
                   www.rocksclusters.org
                   version 6.2 (SideWinder)
                   version 7.0 (Manzanita)

        Copyright (c) 2000 - 2017 The Regents of the University of California.
        All rights reserved.

        Redistribution and use in source and binary forms, with or without
```

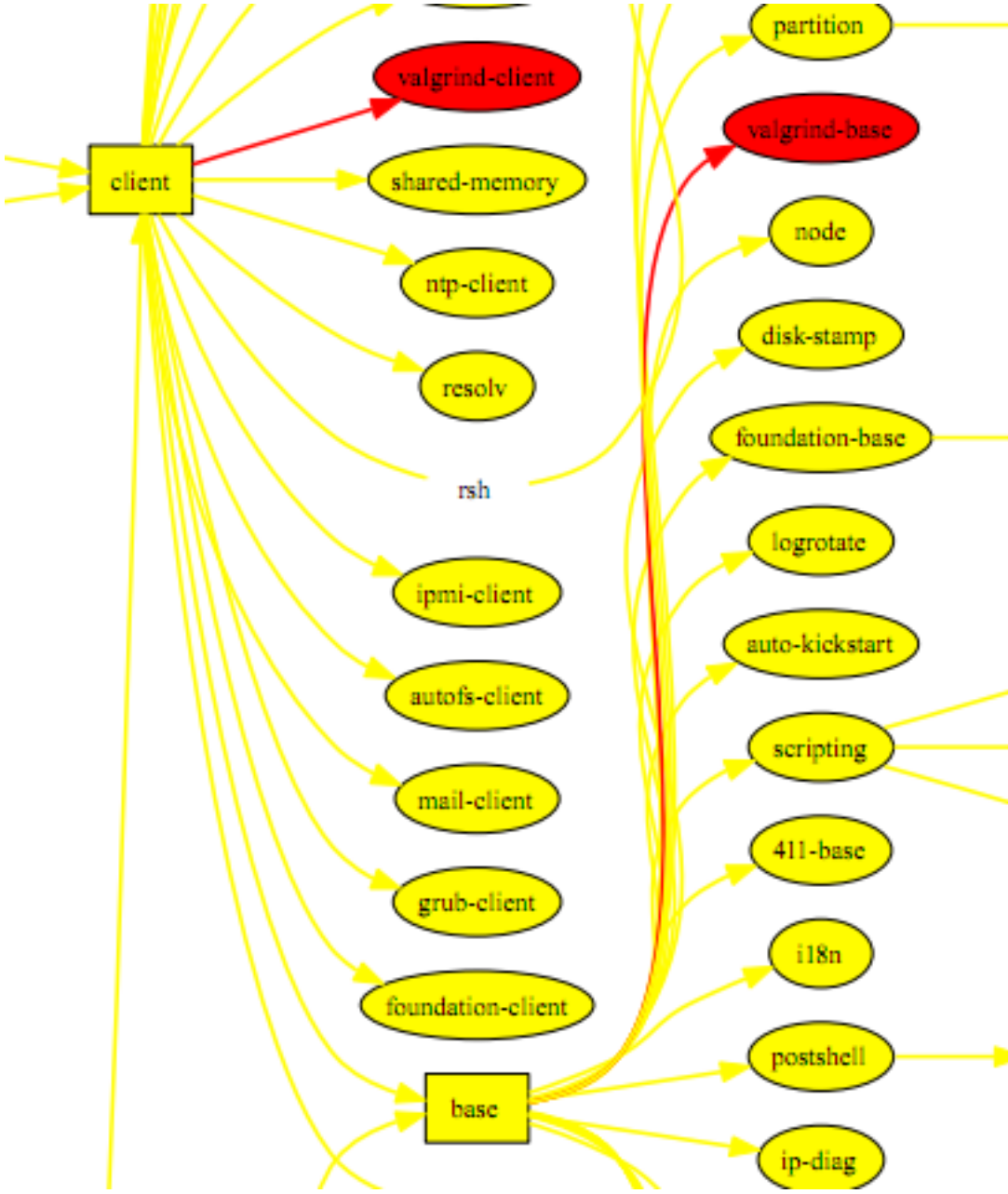
```
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice unmodified and in its entirety, this list of conditions and the
following disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising and press materials, printed or electronic, mentioning
features or use of this software must display the following acknowledgement:

 "This product includes software developed by the Rocks(r)
 Cluster Group at the San Diego Supercomputer Center at the
 University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission.  The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

</copyright>

<changelog>
$Log: internals.sgml,v $
Revision 1.11  2013/01/16 04:09:36  phil
Updates/more detail of internals

Revision 1.10  2012/11/27 00:48:34  phil
Copyright Storm for Emerald Boa

Revision 1.9  2012/05/06 05:48:40  phil
Copyright Storm for Mamba

Revision 1.8  2011/07/23 02:30:44  phil
Viper Copyright
```

```
        Revision 1.7  2011/02/08 21:59:41  bruno
        edits

        Revision 1.6  2011/02/07 23:30:26  bruno
        first pass at build section

        Revision 1.5  2011/02/07 20:48:46  bruno
        the first draft of 'roll internals' is done.

        Revision 1.4  2011/02/05 01:04:49  bruno
        checkpoint

        </changelog>

        <!-- add edges here -->

        <edge from="base">
                <to>valgrind-base</to>
        </edge>

        <edge from="server">
                <to>valgrind-server</to>
        </edge>

        <edge from="client">
                <to>valgrind-client</to>
        </edge>

</graph>
```

With the above graph XML file, we can see (at a high level) how the Valgrind Roll is spliced in with the Base Roll. the Base Roll node XML files are yellow and the Valgrind Roll node XML files are red.



If we zoom in, we can see how two of the Valgrind node XML files have been spliced in:



Now suppose we want the post section of "valgrind-server.xml" to execute *before* the post section of "valgrind-base.xml" and we want the post section of "valgrind-client.xml" to execute *after* "valgrind-base.xml". We can enforce this ordering by adding two `<order>` tags:

```
<?xml version="1.0" standalone="no"?>

<graph roll="valgrind">

 <description>

 The valgrind Roll

 </description>
```

```
<copyright>

    Rocks(r)
           www.rocksclusters.org
           version 6.2 (SideWinder)
           version 7.0 (Manzanita)

Copyright (c) 2000 - 2017 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice unmodified and in its entirety, this list of conditions and the
following disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising and press materials, printed or electronic, mentioning
features or use of this software must display the following acknowledgement:

 "This product includes software developed by the Rocks(r)
 Cluster Group at the San Diego Supercomputer Center at the
 University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission.  The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer".  For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS”
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

</copyright>

<changelog>
$Log: internals.sgml,v $
```

```
 Revision 1.11  2013/01/16 04:09:36  phil
 Updates/more detail of internals

 Revision 1.10  2012/11/27 00:48:34  phil
 Copyright Storm for Emerald Boa

 Revision 1.9  2012/05/06 05:48:40  phil
 Copyright Storm for Mamba

 Revision 1.8  2011/07/23 02:30:44  phil
 Viper Copyright

 Revision 1.7  2011/02/08 21:59:41  bruno
 edits

 Revision 1.6  2011/02/07 23:30:26  bruno
 first pass at build section

 Revision 1.5  2011/02/07 20:48:46  bruno
 the first draft of 'roll internals' is done.

 </changelog>

 <!-- add edges here -->

 <edge from="base">
  <to>valgrind-base</to>
 </edge>

 <edge from="server">
  <to>valgrind-server</to>
 </edge>

 <edge from="client">
  <to>valgrind-client</to>
 </edge>


 <!-- enforce post section ordering -->

 <!--
  valgrind-server post sections execute before valgrind-base
  post sections
 -->
 <order head="valgrind-server">
  <tail>valgrind-base</tail>
 </order>

 <!--
  valgrind-client post sections execute after valgrind-base
  post sections
 -->
 <order head="valgrind-base">
  <tail>valgrind-client</tail>
 </order>

</graph>
```

2.3.3. Attributes

In Rocks, "attributes" are a way to have variables inside node and graph XML files. Attributes are evaluated during kickstart file creation (one of the first actions done when a host is installed). The following is an example of how the "hostname" attribute is used in a post section of a node XML file:

```
<post>

<!--
        set the hostname to the private name.
-->
/bin/hostname &hostname;

</post>
```

In node XML files, if you want to use an attribute in a post section, you need to encode it as an XML entity (thus the '&' and ';' characters that bracket the "hostname" attribute).

There are four levels of attributes: global, OS, appliance and host. Global attributes apply to all hosts in a cluster, OS attributes apply to hosts of a specific OS type (currently "linux" or "sunos"), appliance attributes apply to hosts that have the same appliance type (e.g., compute, tile, etc.), and host attributes apply to only one host. You can add an attribute with one of the following commands: `rocks add attr` (global), `rocks add os attr` (OS), `rocks add appliance attr` (appliance global), `rocks add host attr` (host).

To examine which attributes are currently set for a host, execute `rocks list host attr "hostname"`. For example:

```
# rocks list host attr compute-0-0
HOST         ATTR                          VALUE                                SOURC
compute-0-0: Condor_Client                 true                                 A
compute-0-0: Condor_Daemons                MASTER, STARTD                       G
compute-0-0: Condor_EnableMPI              no                                   G
compute-0-0: Condor_HostAllow              +                                    G
compute-0-0: Condor_Master                 brunoland.rocksclusters.org          G
compute-0-0: Condor_Network                private                              G
compute-0-0: Condor_PasswordAuth           no                                   G
compute-0-0: Condor_PortHigh               50000                                G
compute-0-0: Condor_PortLow                40000                                G
compute-0-0: HttpConf                      /etc/httpd/conf                      O
compute-0-0: HttpConfigDirExt              /etc/httpd/conf.d                    O
compute-0-0: HttpRoot                      /var/www/html                        O
compute-0-0: Info_CertificateCountry       US                                   G
compute-0-0: Info_CertificateLocality      San Diego                            G
compute-0-0: Info_CertificateOrganization  SDSC                                 G
compute-0-0: Info_CertificateState         California                           G
compute-0-0: Info_ClusterContact           admin@place.org                      G
compute-0-0: Info_ClusterLatlong           N32.87 W117.22                       G
compute-0-0: Info_ClusterName              Brunoland                            G
compute-0-0: Info_ClusterURL               http://www.place.org/                G
compute-0-0: Kickstart_DistroDir           /export/rocks                        G
compute-0-0: Kickstart_Keyboard            us                                   G
compute-0-0: Kickstart_Lang                en_US                                G
compute-0-0: Kickstart_Langsupport         en_US                                G
compute-0-0: Kickstart_Multicast           231.253.121.191                      G
compute-0-0: Kickstart_PrivateAddress      10.1.1.1                             G
compute-0-0: Kickstart_PrivateBroadcast    10.1.255.255                         G
compute-0-0: Kickstart_PrivateDNSDomain    local                                G
compute-0-0: Kickstart_PrivateDNSServers   10.1.1.1                             G
```

```
compute-0-0: Kickstart_PrivateGateway              10.1.1.1                          G
compute-0-0: Kickstart_PrivateHostname             brunoland                         G
compute-0-0: Kickstart_PrivateKickstartBasedir     install                           G
compute-0-0: Kickstart_PrivateKickstartCGI         sbin/kickstart.cgi                G
compute-0-0: Kickstart_PrivateKickstartHost        10.1.1.1                          G
compute-0-0: Kickstart_PrivateNTPHost              10.1.1.1                          G
compute-0-0: Kickstart_PrivateNetmask              255.255.0.0                       G
compute-0-0: Kickstart_PrivateNetmaskCIDR          16                                G
compute-0-0: Kickstart_PrivateNetwork              10.1.0.0                          G
compute-0-0: Kickstart_PrivateSyslogHost           10.1.1.1                          G
compute-0-0: Kickstart_PublicAddress               198.202.88.152                    G
compute-0-0: Kickstart_PublicBroadcast             198.202.88.255                    G
compute-0-0: Kickstart_PublicDNSDomain             rocksclusters.org                 G
compute-0-0: Kickstart_PublicDNSServers            198.202.75.26                     G
compute-0-0: Kickstart_PublicGateway               198.202.88.20                     G
compute-0-0: Kickstart_PublicHostname              brunoland.rocksclusters.org       G
compute-0-0: Kickstart_PublicKickstartHost         central.rocksclusters.org         G
compute-0-0: Kickstart_PublicNTPHost               pool.ntp.org                      G
compute-0-0: Kickstart_PublicNetmask               255.255.255.0                     G
compute-0-0: Kickstart_PublicNetmaskCIDR           24                                G
compute-0-0: Kickstart_PublicNetwork               198.202.88.0                      G
compute-0-0: Kickstart_Timezone                    America/Los_Angeles               G
compute-0-0: RootDir                               /root                             O
compute-0-0: Server_Partitioning                   manual                            G
compute-0-0: Xen_Dom0MinMem                        768                               G
compute-0-0: arch                                  x86_64                            H
compute-0-0: bio                                   true                              A
compute-0-0: dhcp_filename                         pxelinux.0                        A
compute-0-0: dhcp_nextserver                       10.1.1.1                          A
compute-0-0: exec_host                             true                              A
compute-0-0: ganglia_address                       224.0.0.3                         G
compute-0-0: hostname                              compute-0-0                       I
compute-0-0: kickstartable                         yes                               A
compute-0-0: managed                               true                              A
compute-0-0: os                                    linux                             H
compute-0-0: rack                                  0                                 I
compute-0-0: rank                                  0                                 I
compute-0-0: rocks_version                         5.4                               G
compute-0-0: sge                                   true                              A
compute-0-0: ssh_use_dns                           true                              G
compute-0-0: submit_host                           false                             A
compute-0-0: tripwire_mail                         root@brunoland.rocksclusters.org  G
compute-0-0: vm_mac_base_addr                      9a:58:ca:0:00:00                  G
compute-0-0: vm_mac_base_addr_mask                 ff:ff:ff:c0:00:00                 G
```

In the output above, the letters in the "SOURCE" column indicate the level at which this host got the attribute. 'G' means the attribute is global, 'O' is an OS attribute, 'A' is an appliance attribute, 'H' is a host attribute and 'I' in an intrinsic attribute (these attributes cannot be removed or modified).

If a host has an attribute assigned to it, you can use it in a post section by referring to it in its entity form ("&attribute_name;"). For example, if you'd like to dynamically get the IP address of the private network for the frontend, you'd use "&Kickstart_PrivateAddress;" in a post section.

Attributes can also be used as a "edge conditional" in graph XML files or as a "post section conditional" in node XML files. An edge conditional is used to conditionally traverse an edge in a graph XML file. The following is an excerpt from the Base Roll graph XML file:

```
<edge from="client" to="x11" cond="x11"/>
```

If the "x11" attribute is set to "true", then when a kickstart file is built for a "client" host (e.g., a compute node or tile node), then the kickstart file generation code will traverse the edge from "client" to "x11", that is, the "x11" node XML file will be included in the kickstart file. Otherwise, if the "x11" attribute is set to false, then the "x11" node XML file will not be included in the kickstart file.

Post section conditions are used to conditionally execute post sections. The following is an excerpt from a node XML file from the SGE Roll:

```
<post os="linux" cond="exec_host">

<file name="/etc/rc.d/rocksconfig.d/post-91-sge" mode="append">
SET_HOST_TYPE=" -x "
</file>

</post>
```

If the attribute "exec_host" is true (and if this is a "linux" host), then the post section will be executed, otherwise, this post section will be skipped.

Chapter 3. Building Your Roll

In this section, we'll show you how to build your roll and we'll examine some of the files that are created when you build your roll.

Building your roll is rather simple. To build the example Valgrind Roll that we developed in the previous section, execute:

```
# cd /export/src/roll/valgrind
# make roll
```

This will take several minutes to build. When it completes, the most important file is `valgrind-1.0-0.x86_64.disk1.iso` -- this is the Valgrind Roll in an ISO image. You can burn this ISO to a CD/DVD or you can transport it to a frontend and install it like you would any other roll.

When you execute "make roll", it will descend into the `src` directory and try to build every package. In the case of the Valgrind Roll, there are two subdirectories under `src`: `usersguide` and `valgrind`. The "make roll" process will go into each of those directories and execute "make rpm", and as decribed in the section Create an RPM from a Source Code Tarball, the resulting RPMs will be placed in `/export/src/roll/valgrind/RPMS/x86_64`.

Also, all the node XML files and the graph XML file will be bundled into a package called `roll-valgrind-kickstart-1.0-0.noarch.rpm`. To see this, execute:

```
# cd /export/src/roll/valgrind
# rpm -qlp RPMS/noarch/roll-valgrind-kickstart-1.0-0.noarch.rpm
/export/profile
/export/profile/graphs
/export/profile/graphs/default
/export/profile/graphs/default/valgrind.xml
/export/profile/nodes
/export/profile/nodes/valgrind-base.xml
/export/profile/nodes/valgrind-client.xml
/export/profile/nodes/valgrind-server.xml
/export/profile/roll-valgrind.xml
```

Above we see the graph XML file (`/export/profile/graphs/default/valgrind.xml`) and the node XML files: (`/export/profile/nodes/valgrind-base.xml`, `/export/profile/nodes/valgrind-client.xml`, `/export/profile/nodes/valgrind-server.xml`). The file `/export/profile/roll-valgrind.xml` contains info that describes how the roll was built.

After the Roll is built, you can add a file to the Valgrind Roll that will help you determine if all the expected packages were built. We do this with the "manifest-check.py" program.

The "manifest-check.py" program looks at file called "manifest" in the base directory of the Roll and then looks at the RPMs under the directory "disk1". The "manifest-check.py" program expects that only the RPMs listed in "manifest" will be present. If there is an entry in "manifest" and if the RPM doesn't exist under "disk1", you will see an error message like:

```
ERROR - the following packages were not built:
        roll-condor-usersguide
```

And if there is package that was built, but there is no entry in "manifest" for it, you'll see an error message like:

```
ERROR - the following packages were built but not in manifest:
        foundation-perl-Statistics-Descriptive
```

```
        foundation-perl-Spreadsheet-ParseExcel
```

For the Valgrind Roll, create a file named "manifest":

```
# cd /export/src/roll/valgrind
# vi manifest
```

Then put the following package names in it:

```
roll-valgrind-kickstart
roll-valgrind-usersguide
valgrind
```

Now run "manifest-check.py" to verify that all the expected packages have been built:

```
# /opt/rocks/share/devel/src/roll/bin/manifest-check.py
```

Everytime you add a new package to your Roll, make sure to add the package's base name to the "manifest" file.

Chapter 4. Testing Your Roll

4.1. Post Installation Script Debugging

This section will discuss some common mistakes made in post installation scripts and provide some techniques on how to help you debug them.

Below is an example node XML file that has a XML syntax error:

```
<?xml version="1.0" standalone="no"?>

<kickstart>

<post>
ech "yo" 2&>1 > /tmp/fun
</post>

</kickstart>
```

When the above node XML file is included in a roll and when a compute node asks for its kickstart file, no file will be returned and the compute node will not install. The compute node will display a screen asking for the user to input a "language" (compute node installations are 100% automated, so a compute node should never ask for user input).

The command `rocks list host profile <hostname>` executes the same functions as the code which automatically generates kickstart files. So, if we execute `rocks list host profile compute-0-0`, we'll see what error occurs:

```
# rocks list host profile compute-0-0 > /tmp/ks.cfg
Traceback (most recent call last):
  File "/opt/rocks/bin/rocks", line 267, in ?
    command.runWrapper(name, args[i:])
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1981, in runWrap
    self.run(self._params, self._args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/list/host/profile/__init__.py", lin
    [
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1667, in command
    o.runWrapper(name, args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1981, in runWrap
    self.run(self._params, self._args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/list/host/xml/__init__.py", line 19
    xml = self.command('list.node.xml', args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1667, in command
    o.runWrapper(name, args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1981, in runWrap
    self.run(self._params, self._args)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/list/node/xml/__init__.py", line 51
    handler.parseNode(node, doEval)
  File "/opt/rocks/lib/python2.4/site-packages/rocks/profile.py", line 409, in parseNode
    parser.feed(line)
  File "/opt/rocks/lib/python2.4/site-packages/_xmlplus/sax/expatreader.py", line 220, in feed
    self._err_handler.fatalError(exc)
  File "/opt/rocks/lib/python2.4/site-packages/_xmlplus/sax/handler.py", line 38, in fatalError
    raise exception
xml.sax._exceptions.SAXParseException: <unknown>:91:11: not well-formed (invalid token)
```

The last line of the output above let's us know there is a problem, but it is not clear in which file and which line number. One way to debug the problem is to add `ROCKSDEBUG=y` to the `rocks list host profile` command:

```
# ROCKSDEBUG=y rocks list host profile compute-0-0 > /tmp/ks.cfg
  .
  .
  .
</kickstart>[parse1]
[parse1]<kickstart roll="valgrind">
[parse1]
[parse1]<post>
[parse1]ech "yo" 2&>1 > /tmp/fun
Traceback (most recent call last):
  File "/opt/rocks/bin/rocks", line 267, in ?
    command.runWrapper(name, args[i:])
  File "/opt/rocks/lib/python2.4/site-packages/rocks/commands/__init__.py", line 1981, in runWrap
    self.run(self._params, self._args)
```

The command produces a lot of output, but it does stop on the line with the bug. Now we have something to `grep` for. On a stock Rocks frontend, all the node XML files for a distribution can be found in `/export/rocks/install/rocks-dist/*/build/nodes`. So to find the file with the syntax error, execute:

```
# cd /export/rocks/install/rocks-dist/x86_64/build/nodes
# grep 'ech "yo" 2&>1 > /tmp/fun' *
valgrind-bug.xml:ech "yo" 2&>1 > /tmp/fun
```

The bug is in `valgrind-bug.xml`, so we can go to the source code for the Valgrind Roll and fix the node XML file.

But let's say we just fix the XML syntax error:

```
<?xml version="1.0" standalone="no"?>

<kickstart roll="valgrind">

<post>
ech "yo" > /tmp/fun
</post>

</kickstart>
```

There is still a bug (the command `ech` should be `echo`). After a host installs, there are several log files saved on a host. One that we'll look at is: `/var/log/rocks-install.log`:

```
./nodes/valgrind-bug.xml: begin post section
/tmp/ks-script-MP2uTd: line 2: ech: command not found
./nodes/valgrind-bug.xml: end post section
```

This may be enough information to help determine the root cause of the problem, but we have found that sometimes we need to "stall" an installation at a specific point. We'll modify the post section in `valgrind-bug.xml` to loop indefinitely:

```
<post>
ech "yo" > /tmp/fun

touch /tmp/stall
while [ -f /tmp/stall ]
```

```
do
 sleep 1
done

</post>
```

When a compute node installs, you wll see a screen that says "Running post-install scripts" -- the installation will not progress beyond this point due to the loop above. From the frontend, we can access the installing node by executing:

```
# ssh compute-0-0 -p 2200
```

We'll see a bash prompt -- we are now in the installation environment for compute-0-0. This environment is running out of a ramdisk, so the partitions on the hard disk are all mounted with the prefix `/mnt/sysimage`. For example, the root partition is `/mnt/sysimage`, the var partition is `/mnt/sysimage/var`, etc. Before a post installation script is run, the installer executes a `chroot` to `/mnt/sysimage` so the script has the illusion it is running on the hard disk environment (not the ramdisk environment).

After we debugged the issue, we can instruct the installaation to continue by executing:

```
# rm -f /mnt/sysimage/tmp/stall
```

4.2. Installation Log Files

After a node installs, there are several log files you can examine to help you debug installation errors:

- `/root/install.log`

  A list of all the RPMs that were installed. This is the order in which the RPMs were installed and the architecture of each package that was installed.

  Example output:

  ```
  Installing words-3.0-9.1.noarch
  Installing libgcc-4.1.2-48.el5.i386
  Installing libgcc-4.1.2-48.el5.x86_64
  Installing glibc-2.5-49.el5_5.7.x86_64
  Installing glibc-2.5-49.el5_5.7.i686
  Installing chkconfig-1.3.30.2-2.el5.x86_64
  Installing zlib-1.2.3-3.x86_64
  ```

- `/root/anaconda.log`

  The output of the ananconda installer. It shows the steps the anaconda installer executed as well as the output of the sub-commands anaconda executed.

  Example output:

  ```
  18:58:53 INFO    : moving (1) to step partitionobjinit
  18:58:53 INFO    : ISCSID is /usr/sbin/iscsid
  18:58:53 INFO    : no initiator set18:58:53 INFO    : no /tmp/fcpconfig; not configuring zfcp
  18:58:53 DEBUG   : starting mpaths18:58:53 DEBUG   : self.driveList(): ['sda', 'sdb']
  18:58:53 DEBUG   : DiskSet.skippedDisks: []
  18:58:53 DEBUG   : DiskSet.skippedDisks: []
  18:58:53 DEBUG   : done starting mpaths.  Drivelist: ['sda', 'sdb']
  ```

- `/var/log/rocks-install.log`

Error messages from all the post sections will be found in this log file. In general, all you will see are "begin post section" and "end post section" messages.

Example output:

```
./nodes/411-client.xml: begin post section
./nodes/411-client.xml: end post section
./nodes/411-client.xml: begin post section
./nodes/411-client.xml: end post section
./nodes/411-client.xml: begin post section
./nodes/411-client.xml: end post section
```

- `/root/syslog`

  This is the syslog from the host during its installation phase.

  Example output:

```
<6>Checking 'hlt' instruction... OK.
<6>SMP alternatives: switching to UP code
<6>ACPI: Core revision 20060707
<4>CPU0: Intel(R) Xeon(R) CPU           E5320  @ 1.86GHz stepping 07
<6>SMP alternatives: switching to SMP code
<4>Booting processor 1/1 eip 11000
<4>CPU 1 irqstacks, hard=c0769000 soft=c0749000
<6>Initializing CPU#1
```

- `/root/httpd.log`

  A copy of the log messages from lighttpd. Lighttpd is used on an installing node to execute key parts of the Avalanche Installer. You will see messages (in apache log format) that log the http requests to lighttpd.

  Example output:

```
127.0.0.1 127.0.0.1 - [28/May/2010:10:45:11 -0700]
"GET /install/rocks-dist/x86_64/RedHat/RPMS/dmraid-events-1.0.0.rc13-53.el5.x86_64.rpm HTTP/1.1
200 18083 "-" "urlgrabber/3.1.0 yum/3.2.22"

127.0.0.1 127.0.0.1 - [28/May/2010:10:45:13 -0700]
"GET /install/rocks-dist/x86_64/RedHat/RPMS/compat-libf2c-34-3.4.6-4.i386.rpm HTTP/1.1"
200 2674 "-" "urlgrabber/3.1.0 yum/3.2.22"
```

Appendix A. XML File Syntax

Rocks generates kickstart files for compute nodes dynamically using a structure called the "kickstart graph". This graph is made from edges and nodes, both described in an XML language. This section serves as a reference for the XML tags and attributes.

A traversal of the kickstart nodes and edges makes a full kickstart file used to specify the software and configuration for a node. Edges are described in "graph" files, while each node is specified in a "node" file.

When the order of nodes' contribution to the kickstart file matters, we use special edges made with `<order>` tags. We describe these tags in this section as well.

A.1. Node XML Tags

- **<kickstart>**

  Wraps an XML node file.

- **<description>**

  Text description of what this node does.

- **<copyright>**

  Text description of the copyright associated with this node file.

- **<changelog>**

  Text description of the changes made to this node file. Generally a "Log" CVS directive.

- **<package>**

  Specifies a single RPM package. Includes only the RPM name, not its version or release number.

  type

  Optional. Value is "meta". Used to describe RedHat meta packages.

  Example: `<package type="meta">gnome-desktop</package>`.

  disable

  Optional. If this value is non-zero, the RPM will not be installed.

  Example: `<package disable="1">emacs</package>`.

  arch

  Optional. This value can be "i386" or "x86_64". If "arch" matches the architecture of the host that is being installed, then the package will be installed, otherwise the package will be skipped. For example, if the value is "i386" and if the x86_64 version of Rocks is being installed on a host, then the package will be skipped. Or said another way, the package will only be installed on i386 hosts.

  Example: `<package arch="i386">nasm</package>`.

  os

  Optional. Value is "linux" or "sunos". If this is specified, then the package will only be installed on the specified OS. For example, if "linux" is specified, then the package will only be installed on Linux hosts (and the package will not be installed on Solaris hosts).

Example: `<package os="linux">`.

cond

Optional. If the value of the "cond" is true, then this package will be installed. Values can be Rocks attributes or generic python expressions.

Example: `<post cond="rocks_version_major >= 6">`.

If the installed Rocks version is greater or equal than 6 then the package will be installed otherwise not. See <post> section below for more example.

- **<eval>**

Replaced with the output of the script specified between these tags. The script is run on the host generating the kickstart file (generally the frontend).

shell

Optional. Specifies the script interpreter to use. Default "sh".

Example: `<eval shell="/bin/bash">`.

Example: `<eval shell="/opt/rocks/bin/python">`.

mode

Optional. Value is "quote" or "xml". If value is "quote", XML characters are escaped in the shell output. Default is "quote".

Example: `<eval mode="xml">`.

Example: `<eval shell="/bin/bash" mode='xml'>`.

- **<post>**

Wraps a post section. Configuration is generally carried out in post sections, making this a popular tag. The commands specified here correspond to an RPM post section, and they are executed on the client machine (not on the frontend, in contrast to the `<eval>` tag).

arch

Optional. This value can be "i386" or "x86_64". If "arch" matches the architecture of the host that is being installed, then this post section will be run, otherwise the post section will be skipped.

Example: `<post arch="i386">`.

os

Optional. Value is "linux" or "sunos". If this is specified, then the post section will only be run during the installation of the specified OS. For example, if "linux" is specified, then the post section will be run when a Linux host is installed, but the post section will not be run when a Solaris host is installed.

Example: `<post os="linux">`.

cond

Optional. If the value of the "cond" is true, then this post section will be executed. Values can be Rocks attributes or generic python expressions.

Example: `<post cond="submit_host">`.

If the "submit_host" Rocks attribute is set to "true" for this host, then the post section will be executed, otherwise, this post section will be skipped.

Example: `<post cond="not ssh_use_dns">`.

If the "ssh_use_dns" Rocks attribute is set to "false" for this host, then the post section will be executed (because of the python "not" operator).

interpreter

Optional. Specifies the script interpreter to use. Default "/bin/bash".

Example: `<post interpreter="/opt/rocks/bin/python">`.

arg

Optional. Value can be "--nochroot". The "--nochroot" value means run this post section in the context of the installer, not in the context of the installing machine. The installer runs on a ramdisk on an installing machine, so if this argument is supplied, then the post section has access to the ramdisk file system as well as the mounted file systems for the installing node.

Example: `<post arg="--nochroot">`.

- **<pre>**

  Wraps the pre section commands. All pre sections run before package installation, in contrast to commands from the post section. All pre sections are run in the context of the installation environment, that is, the root file system is a ramdisk.

  The <pre> section supports the same attributes as the <post> section, please refer to the documentation above for a detailed description.

- **<configure>**

  The configure tag (introduced in Rocks 6.2) contains a snipped of code which should be executed to configure a particular software, exactly like the <pre> and <post> tag. Although configure must specify when the code snipped should be executed using the attribute phase.

  This tag support the same attributes as the <post> section, please refer to the documentation above for a detailed description.

  phase

  Required. It is a comma separated list of phases in which the code should be executed. Supported phases are: pre (which behaves exactly as for the <pre> tag), post (it behaves exactly as for the <post> tag) and reconfigure which is executed only during a reconfiguration (see the userguide).

  Example: `<configure phase="post">` is the same as using the tag `<post>`

- **<file>**

  Wraps the contents of a file.

  name

  Required. Specifies the name of this file, a full path.

  Example: `<file name="/etc/auto.master">`.

mode

Optional. Value is "append". If "append" is specified, the contents are appended to the end of an existing file. A record of the change is kept in a RCS repository in the same directory as the file.

Example: `<file name="/etc/man.config" mode="append">`.

rcs

Optional. This flag can disable tracing the file with Revision Control System. By the default all files are under RCS but if rcs=false it will disable revisioning.

Example: `<file name="/etc/modprobe.d/netloop.conf" rcs="false">`.

include

Optional. This attribute can specify a file name which will be sourced as the content of the file. The included file will be read on the frontend (and not on the installing host). All the text present between the <file> tags will be discharged when the include attribute is present. If the mode is append the included content will be appended to the destination file.

Example: `<file name="/etc/resolve.conf" include="/etc/resolve.conf"/>`.

owner

Optional. The "user.group" that owns this file. Can be specified either as names "root.root" or numbers (guids) "0.0".

Example: `<file name="/var/www/html/index.html" owner="root.apache">`.

perms

Optional. The permissions of this file. The value of this argument is passed to the "chmod" command, and accepts the same format.

Example: `<file name="/etc/ssh/ssh_host_key.pub" perms="0444">`.

vars

Optional. Value is "literal" or "expanded". If "literal" no variable or backtick expansion is done on the contents of the file. If value is "expanded", standard shell variable expansion is performed, as well as running commands quoted with backticks. Default is "literal".

Example: `<file name="/etc/motd" mode="append" vars="expanded">`.

expr

Optional. Specifies a command whose output is placed in the file.

Example: `<file name="/etc/dhcpd.conf" expr="/opt/rocks/bin/rocks report host dhcpd"/>`.

os

Optional. Value is "linux" or "sunos". If this is specified, then the file will only be created (or appended to) run during the installation of the specified OS.

Example: `<file name="/etc/postfix/main.cf" mode="append" os="sunos">`.

A.1.1. Kickstart Main Section

These tags specify commands in the "main" section of a kickstart file. Each of these tags are wrapped in `<main>` tags. They appear in node XML files. Only the tags normally used in a cluster appliance kickstart file are presented here; for a full reference of all the kickstart options and their respective arguments, see RHEL5 Documentation - Kickstart Installations[1].

<main>

- **<bootloader>**

  Specifies the bootloader arguments. Default "--location=mbr"

- **<interactive>**

  Optional. Allows for inspection and modification of the kickstart values given, via the snack screen interface. Default: present.

- **<url>**

  Specifies the installation method with the `--url` argument. Default is `http://&Kickstart_PrivateKickstartHost;/&Kickstart_PrivateKickstartBasedir;/&distribution;/&arc` and when the kickstart file is generated, all the attributes are substituted with their respective values.

- **<lang>**

  The installation language to use. Default "en_US".

- **<keyboard>**

  Sets the system keyboard type. Default "us".

- **<mouse>**

  Specifies the system mouse type. Default "none".

- **<timezone>**

  Required. Sets the system timezone. The default is the timezone selected when a frontend is installed.

- **<install>**

  If present, perform a fresh install (not an upgrade). Default: present.

- **<rootpw>**

  Optional. Set the root password for the installing system. Default is `<rootpw>--iscrypted &Kickstart_PrivateRootPassword;</rootpw>`, and when the kickstart file is generated, the attribute is substituted with its respective value.

A.2. Graph XML Tags

Edges in the kickstart graph are specified with the XML tags below. Order tags give control of the graph traversal (the order nodes appear in the final kickstart file).

Both the edge and order tags appear in kickstart graph files.

- **<graph>**

  Wraps an XML graph file.

- **<description>**

Text description of what this part of the graph does.

- **<copyright>**

  Text description of the copyright associated with this graph file.

- **<changelog>**

  Text description of the changes made to this node file. Generally a "Log" CVS directive.

- **<edge>**

  Specifies an edge in the kickstart graph, links graph nodes together. Can be specified as a singleton tag:

  `<edge from="base" to="ganglia"/>`, or a standard tag that wraps `<to>` or `<from>` tags.

  from

  Optional. Specifies the beginning of an edge, a node name.

  Example: `<edge from="server">`.

  to

  Optional. Specifies the end of an edge, a node name.

  Example: `<edge to="client">`.

  cond

  Optional. If the value of the "cond" is true, then this edge will be traversed. Values can be Rocks attributes or generic python expressions.

  Example: `<edge from="client" to="x11" cond="x11"/>`.

  If the "x11" Rocks attribute is set to "true", then this edge will be traversed.

  arch

  Optional. Specifies which architectures should follow this edge. Same format as arch attribute in node files. The edge is ignored if the client's architecture does not match this list.

  Example: `<edge from="client" to="x11" arch="x86_64"/>`.

- **<to>**

  Wraps a node name. Specifies the end of a directed edge in the graph. Used inside an edge tag with the "from" attribute:

  ```
  <edge from="compute">
   <to>compute-appliance</to>
  </edge>
  ```

  arch

  Optional. Specifies which architectures should follow this edge. The entire edge is ignored if the client's architecture does not match this list.

  Example: `<to arch="i386">pvm</to>`.

- **<from>**

  Wraps a node name. Specifies the beginning of a directed edge. Used like "to" tag.

```
<edge to="client">
        <from>compute-appliance</from>
</edge>
```

arch

Optional. Specifies which architectures should follow this edge. The entire edge is ignored if the client's architecture does not match this list.

- **<order>**

Specifies a ordering between nodes in the graph. While the `<edge>` tags specify a "membership" in the kickstart file, the `<order>` tags give a "relative ordering" between nodes.

The ordering is affected by a topological sort of nodes using order edges. While the kickstart graph allows cycles, the set of order tags must specify a directed-acyclic graph (DAG). Any nodes not touched by an order edge have a random order in the resultant kickstart file.

Can be used to wrap `<head>` and `<tail>` tags in the same fashion as the `<to>` and `<from>` tags for "edge".

arch

Optional. Specifies which architectures should follow this edge. Same format as arch attribute in node files. The edge is ignored if the client's architecture does not match this list.

head

Optional. Specifies the beginning of this edge, a node name. The node specified by the `<order head="node">` tag should be ordered before all child `<tail>` tags.

```
<order head="grub">
 <tail>grub-client</tail>
 <tail>grub-server</tail>
</order>
```

Special name "TAIL" is allowed, which specifies the node be placed last. Ordering among nodes with TAIL ordering is undefined.

```
<order head="TAIL">
 <tail>install</tail>
 <tail>pxeboot</tail>
 <tail>compute</tail>
</order>
```

In the above example, the node files are "install", "pxeboot" and "compute" will all be at the end of the kickstart file.

tail

Optional. Specifies the end of this edge, a node name. The node specified by the `<order tail="node">` tag should be ordered after all child `<head>` tags.

```
<order tail="server-firewall">
 <head>networking-server</head>
 <head>database-data</head>
</order>
```

Special node name "HEAD" is allowed, which specifies the node is placed at the beginning of the kickstart file. Ordering among nodes with HEAD ordering is undefined.

```
<order tail="HEAD">
 <head>python-development</head>
 <head>rlo</head>
</order>
```

- **<head>**

  Wraps a node name. This tag is a child of an `<order tail="some-node-name">` tag. All nodes wrapped by `<head>` tags are guaranteed to execute prior to the node specified in the `<order tail="some-node-name">` tag.

  ```
  <order tail="server-firewall">
   <head>networking-server</head>
   <head>database-data</head>
  </order>
  ```

  In the example above, "networking-server" and "database-data" are guaranteed to execute before "server-firewall".

- **<tail>**

  Wraps a node name. This tag is a child of an `<order head="some-node-name">` tag. All nodes wrapped by `<tail>` tags are guaranteed to execute after to the node specified in the `<order head="some-node-name">` tag.

  ```
  <order head="grub">
   <tail>grub-client</tail>
   <tail>grub-server</tail>
  </order>
  ```

  In the example above, "grub-client" and "grub-server" are guaranteed to execute after "grub".

Notes

1. http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5/html-single/Installation_Guide/index.html#ch-kickstart2

Appendix B. Rocks® Copyright and Trademark

B.1. Copyright Statement

Rocks(r)
www.rocksclusters.org
version 6.2 (SideWinder)
version 7.0 (Manzanita)

Copyright (c) 2000 - 2017 The Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
notice unmodified and in its entirety, this list of conditions and the
following disclaimer in the documentation and/or other materials provided
with the distribution.

3. All advertising and press materials, printed or electronic, mentioning
features or use of this software must display the following acknowledgement:

        "This product includes software developed by the Rocks(r)
        Cluster Group at the San Diego Supercomputer Center at the
        University of California, San Diego and its contributors."

4. Except as permitted for the purposes of acknowledgment in paragraph 3,
neither the name or logo of this software nor the names of its
authors may be used to endorse or promote products derived from this
software without specific prior written permission. The name of the
software includes the following terms, and any derivatives thereof:
"Rocks", "Rocks Clusters", and "Avalanche Installer". For licensing of
the associated name, interested parties should contact Technology
Transfer & Intellectual Property Services, University of California,
San Diego, 9500 Gilman Drive, Mail Code 0910, La Jolla, CA 92093-0910,
Ph: (858) 534-5815, FAX: (858) 534-7345, E-MAIL:invent@ucsd.edu

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS “AS IS
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS
BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE

```
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

B.2. Trademark Licensing

The Rocks® logo (Figure B-1) and the names Rocks®, Rocks Clusters™, and Avalanche Installer™ have been submitted to the United States Patent and Trademark Office to become registered trademarks.

Commercial entities that wish to use any of the above names or logo in a product name or marketing material, are required to get written permission from the Technology Transfer and Intellectual Property Services Office at UCSD <`invent@ucsd.edu`>.

Figure B-1. Rocks® logo