Rocks Cluster Administration

Learn how to manage your Rocks Cluster Effectively

Module 1: Customizing Your Cluster

Customizing Nodes

- Using built in node attributes and the Rocks Command line
- Using extend-node.xml files.
- Creating custom appliance types.

Attributes

- Attributes are settings within the rocks database to enable and disable features.
- Attributes are controlled by the rocks command line. You will need to become familiar with this utility before beginning any Rocks customization.

Example Attributes

```
File  Edit  View  Terminal  Help
[root@nic-d1 ~]# rocks list attr
ATTR                                VALUE
Kickstart_PublicHostname:           nic-d1.srv.mst.edu
Info_ClusterName:                   Rocks-Cluster
Info_CertificateOrganization:       MST
Info_CertificateLocality:           Rolla
Info_CertificateState:              Missouri
Info_CertificateCountry:            US
Info_ClusterContact:                hamiltonsl@mst.edu
Info_ClusterURL:                    http://www.mst.edu
Info_ClusterLatlong:                N32.87 W117.22
Kickstart_PrivateHostname:          nic-d1
Kickstart_PrivateKickstartCGI:      sbin/kickstart.cgi
Kickstart_DistroDir:                /export/rocks
Kickstart_PrivateKickstartBasedir:  install
Kickstart_PublicKickstartHost:      central.rocksclusters.org
Kickstart_PrivateDNSDomain:         local
Kickstart_PublicDNSDomain:          srv.mst.edu
Kickstart_Lang:                     en_US
Kickstart_Langsupport:              en_US
Kickstart_Keyboard:                 us
Kickstart_PrivateAddress:           10.2.7.1
Kickstart_PrivateNetmask:           255.255.255.0
Kickstart_PrivateNetwork:           10.2.7.0
Kickstart_PrivateBroadcast:         10.2.7.255
Kickstart_PrivateNetmaskCIDR:       24
Kickstart_PrivateKickstartHost:     10.2.7.1
Kickstart_PrivateNTPHost:           10.2.7.1
Kickstart_PrivateGateway:           10.2.7.1
Kickstart_PrivateDNSServers:        10.2.7.1
Kickstart_PrivateSyslogHost:        10.2.7.1
Kickstart_Multicast:                234.182.2.59
Kickstart_PublicAddress:            131.151.245.232
Kickstart_PublicNetmask:            255.255.255.224
Kickstart_PublicNetwork:            131.151.245.224
Kickstart_PublicBroadcast:          131.151.245.255
Kickstart_PublicNetmaskCIDR:        27
Kickstart_PublicGateway:            131.151.245.254
Kickstart_PublicDNSServers:         131.151.247.40,131.151.247.41
Kickstart_PrivateRootPassword:      $1$WOtgn/8W$h3hILogovuvnDvawi02Q2.
Kickstart_PrivateSHARootPassword:   7280c8cb0640263c9da3179ea684f96334769cb4
Kickstart_PrivatePortableRootPassword: $P$Bx32Dnoo8ihHblAmfnoUaKclG2VYFc.
Kickstart_Timezone:                 Africa/Abidjan
Kickstart_PublicNTPHost:            ntp-server-a.srv.mst.edu
Server_Partitioning:                force-default-root-disk-only
rocks_version:                      5.3
ssh_use_dns:                        true
tripwire_mail:                      root@nic-d1.srv.mst.edu
[root@nic-d1 ~]#
```

- "rocks list attr" returns system wide attributes.
  - Info_ClusterName
  - rocks_version
  - Kickstart_Timezone
  - Kickstart_Lang
  - ssh_use_dns
  - And many others.
- Changing these allows you to reconfigure a running cluster.

Appliance Attributes

- Appliance attributes are settings within the rocks database to enable and disable features on nodes based on the selected appliance type.
- You can create custom appliances that are as simple as compute nodes with modified default attributes.

Ex: Appliance Attributes

```
File  Edit  View  Terminal  Help
[root@nic-d1 ~]# rocks list appliance
APPLIANCE   GRAPH   NODE
frontend:   default server
compute:    default compute
nas:        default nas
network:    ------- ----------
power:      ------- ----------
ipmi:       ------- ----------
tile:       default viz-tile
login:      default login
gao-login:  default gao-login
math-login: default math-login
[root@nic-d1 ~]# rocks list appliance attr compute
APPLIANCE ATTR    VALUE
compute:  managed true
compute:  pbs     true
[root@nic-d1 ~]# rocks list appliance attr tile
APPLIANCE ATTR                  VALUE
tile:     x11                   true
tile:     cuda                  true
tile:     viz_nvidia_driver     /opt/viz/drivers/nvidia.run
tile:     viz_nvidia_driver_options --no-network -s
tile:     viz_tile_resolution   1920x1200
tile:     viz_tile_left_bezel   100
tile:     viz_tile_right_bezel  100
tile:     viz_tile_top_bezel    80
tile:     viz_tile_bottom_bezel 80
[root@nic-d1 ~]# rocks list appliance attr login
APPLIANCE ATTR    VALUE
login:    managed true
login:    pbs     true
[root@nic-d1 ~]# rocks list appliance attr nas
APPLIANCE ATTR    VALUE
nas:      managed true
[root@nic-d1 ~]#
```

- "rocks list appliance" returns a list of available appliance types
  - frontend
  - compute
  - nas
  - network
  - power
  - ipmi
  - tile
  - login
- "rocks list appliance attr compute" returns a list of compute node attributes.
  - managed
  - pbs

Node Attributes

- Node attributes are settings for individual nodes of the cluster. They default to the appliance attributes, but can be modified node to node.

Ex: Node Attributes

File Edit View Terminal Help

```
[root@nic-dl ~]# rocks list host attr
HOST    ATTR                                VALUE                                SOURCE
nic-dl: HttpConf                            /etc/httpd/conf                      O
nic-dl: HttpConfigDirExt                    /etc/httpd/conf.d                    O
nic-dl: HttpRoot                            /var/www/html                        O
nic-dl: Info_CertificateCountry             US                                   G
nic-dl: Info_CertificateLocality            Rolla                                G
nic-dl: Info_CertificateOrganization        MST                                  G
nic-dl: Info_CertificateState               Missouri                             G
nic-dl: Info_ClusterContact                 hamiltonsl@mst.edu                   G
nic-dl: Info_ClusterLatlong                 N32.87 W117.22                       G
nic-dl: Info_ClusterName                    Rocks-Cluster                        G
nic-dl: Info_ClusterURL                     http://www.mst.edu                   G
nic-dl: Kickstart_DistroDir                 /export/rocks                        G
nic-dl: Kickstart_Keyboard                  us                                   G
nic-dl: Kickstart_Lang                      en_US                                G
nic-dl: Kickstart_Langsupport               en_US                                G
nic-dl: Kickstart_Multicast                 234.182.2.59                         G
nic-dl: Kickstart_PrivateAddress            10.2.7.1                             G
nic-dl: Kickstart_PrivateBroadcast          10.2.7.255                           G
nic-dl: Kickstart_PrivateDNSDomain          local                                G
nic-dl: Kickstart_PrivateDNSServers         10.2.7.1                             G
nic-dl: Kickstart_PrivateGateway            10.2.7.1                             G
nic-dl: Kickstart_PrivateHostname           nic-dl                               G
nic-dl: Kickstart_PrivateKickstartBasedir   install                              G
nic-dl: Kickstart_PrivateKickstartCGI       sbin/kickstart.cgi                   G
nic-dl: Kickstart_PrivateKickstartHost      10.2.7.1                             G
nic-dl: Kickstart_PrivateNTPHost            10.2.7.1                             G
nic-dl: Kickstart_PrivateNetmask            255.255.255.0                        G
nic-dl: Kickstart_PrivateNetmaskCIDR        24                                   G
nic-dl: Kickstart_PrivateNetwork            10.2.7.0                             G
nic-dl: Kickstart_PrivatePortableRootPassword $P$Bx32Dnoo8ihHblAmfnoUaKclG2VYFc.  G
nic-dl: Kickstart_PrivateRootPassword       $1$WOtgn/8W$h3hILogovuvnDvawi02Q2.   G
nic-dl: Kickstart_PrivateSHARootPassword    7280c8cb0640263c9da3179ea684f96334769cb4 G
nic-dl: Kickstart_PrivateSyslogHost         10.2.7.1                             G
nic-dl: Kickstart_PublicAddress             131.151.245.232                      G
nic-dl: Kickstart_PublicBroadcast           131.151.245.255                      G
nic-dl: Kickstart_PublicDNSDomain           srv.mst.edu                          G
nic-dl: Kickstart_PublicDNSServers          131.151.247.40,131.151.247.41        G
nic-dl: Kickstart_PublicGateway             131.151.245.254                      G
nic-dl: Kickstart_PublicHostname            nic-dl.srv.mst.edu                   G
nic-dl: Kickstart_PublicKickstartHost       central.rocksclusters.org            G
nic-dl: Kickstart_PublicNTPHost             ntp-server-a.srv.mst.edu             G
nic-dl: Kickstart_PublicNetmask             255.255.255.224                      G
nic-dl: Kickstart_PublicNetmaskCIDR         27                                   G
nic-dl: Kickstart_PublicNetwork             131.151.245.224                      G
nic-dl: Kickstart_Timezone                  Africa/Abidjan                       G
nic-dl: RootDir                             /root                                O
```

- “rocks list host attr” returns a list of all node attributes from all nodes.
- “rocks list host attr *hostname*” returns a list of all attributes for the requested host.

Rocks command line

- The rocks command line has limited documentation, but decent on-line help. To get help with the rocks command line simply type “rocks” and hit enter. You will get a full list of available rocks commands.
- The most widely used are:
  - rocks list
  - rocks set
  - rocks sync
  - rocks report

LAB 1:

Customizing with the Rocks Command line.

In our example we will create a custom appliance that adds X11 to the node and makes it a submit host.

This would be a good fit for installing Rocks on lab PCs to use them as submission and job hosts.

First you will need to create the node profile. This is done using a new node XML file.

Start by copying

/export/rocks/install/site-profiles/5.4.3/nodes/skeleton.xml

To

/export/rocks/install/site-profiles/5.4.3/nodes/yourname.xml

Note: Substitute 5.4.3 for your rocks version and your first name for yourname.xml.

```
$ cd /export/rocks/install/site-profiles/5.4.3/nodes
$ cp skeleton.xml yourname.xml
```

For our simple example we do not need to edit the file as we are not adding anything extra to the node. We will use the rocks attributes for that. However, let's take a look inside the file anyway.

The node XML file contains the following sections:

- <description>
- <changelog>
- <main>
- <pre>
- <package>
- <post>

Description

- The description should contain a short description of the purpose behind your node definition.
- Ours should probably read something like this:

  The lab node provides a method of installing Linux based HPC nodes as access points to the cluster in a lab environment. This allows the lab to be used as computational resources outside of normal operating hours.

Main

- This section will rarely be used in custom appliances. It is used to control the main commands within the kickstart system.
- For example setting a root password.
  - <rootpw>--iscrypted encryptedpassword</rootpw>

Pre section

- This section is mainly used for custom partitioning of nodes.
- It uses standard shell scripting directives as well as standard kickstart variables.
- We will not cover custom partitioning in this class unless someone requests it.

Package section

- This is not so much a section as a series of tags. These tags tell a package to install based on the internal package name.
- For example you can have an rpm file named john.rpm but it contains a package definition for vim. You would install it using <package>vim</package> not <package>john</package>
- One strange thing to keep in mind is that this will install the newest package of a given name based on the time stamp of the rpm file name, not the internal version number.

Post section

- This section contains any post installation scripting. These scripts run after all RPMs have been installed, but before the system boots for the first time.
- It is possible to write a post install script that works correctly on an already running system, but fails when installed during the initial cluster installation. If you plan on sharing your roll be sure to test it in both scenarios.
- Wrapping <eval> tags around a section of the post install causes it to execute on the head node after a node installs.

Next you will need to link your new appliance into the Rocks appliance graph. Since we are simply extending a standard compute appliance this is fairly easy.

You will need to create a graph XML file in the /export/rocks/install/site-profiles/5.4.3/graphs/default directory.

Sample content of graph XML files.

```
$ cd /export/rocks/install/site-profiles/5.4.3/graphs/default
$ edit yourname-app.xml
<?xml version="1.0" standalone="no"?>
<graph>
<description>
</description>
<changelog>
</changelog>
<edge from="yourname">
        <to>compute</to>
</edge>
<order gen="kgen" head="TAIL">
        <tail>yourname</tail>
</order>
</graph>
```

Now we have to rebuild the cluster distribution to include our new appliance files.

```
# cd /export/rocks/install
# rocks create distro
```

Only one student should do this step after everyone has completed their custom appliance configuration.

Let's take a quick break while the system builds the distribution.

Now we have to add the custom appliance to the database.

```
# rocks add appliance yourname membership='yourname'  \
shortname='yourinitials' node='yourname'
```

Now we need to tell the cluster that your node gets X11 installed and is a submit and execute host.

```
# rocks set appliance attr yourname x11 true
# rocks set appliance attr yourname exec_host true
# rocks set appliance attr yourname submit_host true
```

Module 2: Adding Custom Applications

There are two basic ways to add applications to a Rocks Cluster.

- NFS Mounted
- Locally Installed

There are also three different types of custom applications you can install.

- From Source Code
- From Commercial Media
- From RPM Packages

NFS Mounted

The simplest method of adding software to the cluster is by using the already existing NFS mounted applications folder “/share/apps”

1) Install the software on the head node following manufacturer instructions. Pick /share/apps/application_name as the installation location.
2) Add any custom environment settings to /etc/profile.d/appname.sh
3) Add /etc/profile.d/appname.sh to the 411 Service configuration
    1) Add the file name to the list in /var/411/Files.mk
    2)Execute make restart in /var/411
4)If there are no custom environment settings needed you do not have to update the 411 service definitions.

The 411 Service

The 411 Service is an important part of the Rocks Cluster management suite and it is important that we cover a little of how it works here.

1)411 keeps configuration files consistent across the cluster.

2)The list of files that are the same everywhere is contained in /var/411/Files.mk

3)The 411 service makes provisions for various configurations by node type.

- 1)These node type configurations are stored in the /var/411/Group.mk file.
- 2)Each group has its own subfolder that contains common configuration files under /var/411/groups/groupname

4) Modifications to any file managed by the 411 service automatically get pushed to the nodes in the cluster. It is important to look at this configuration when doing customization.

Let's log into the cluster and look around the 411 configuration directory now.

Locally Installed

Installing applications locally on the nodes is a lot harder, but has the advantage of lower network impact.

1) Install the software on the head node following manufacturer instructions. This can be installed anywhere. On our campus we choose to use /opt/appname for our custom packages.

2) Add any custom environment settings to /etc/profile.d/appname.sh

3) No comes the tricky part, you have to figure out a method of mirroring this application to the rest of the cluster nodes.

   1)A simple way is to use the tentakel command along with an rsync command to replicate the software to the other nodes. (It works, but does not guarantee that all the nodes remain in sync over time)

   2)The best method is to create an RPM of the installed files and add your custom RPM to the configuration scripts.

Deploying RPM Software

- This is the easiest to deploy, mainly because it already has a mostly automated installation.
- To deploy RPM software you simply copy the RPM into the contrib folder of the Rocks install directory and add it to a node.xml file.

Ex: Adding RPM

 # cp myfile.rpm /export/rocks/install/contrib/5.4.3/x86_64/RPMS

The you will need to edit or create an extend-compute.xml file.

 # cd /export/rocks/install/site-profiles/5.4.3/nodes

 # cp skeleton.xml extend-compute.xml (unless it already exists).

(In our lab we will use extend-yourname.xml)

You will need to add package tags to the xml file like so:

Just above the <post> tag in the file you will add a line like this:

<package>packagename</package>

Then exactly like the custom appliance you have to rebuild the distribution to make the package available.

 # cd /export/rocks/install

 # rocks create distro

(Only one student should do this step)

This will make the rpm available to the nodes. The way Rocks recommends adding the package to all the nodes is to reinstall all the nodes. I feel that this is not necessary in most cases you can do the following:

 # tentakel yum install packagename

Installing Commercial Software

- This is best done either as an NFS deployment, or as a custom roll.
- In our final lab we will build a custom roll.

Installing Software from Source

- This is best done either as an NFS deployment, or as a custom roll.
- If the source code has options for building RPM files you can follow the procedures for installing RPM based software.
- In our final lab we will build a custom roll.

Lab 2: Installing a Custom Application

Step one: Get the RPM rocks 5.4.3 is based on CentOS 5.6 so you can use RPMS designed for CentOS 5.6

For the lab pick one from here:

http://mirror.centos.org/centos/5.6/extras/x86_64/RPMS/

Step two: Copy your RPM to right place:

```
# cp yourrpm.rpm /export/rocks/install/contrib/5.4.3/x86_64/RPMS
```

Step three: Update node.xml

Edit
/export/rocks/install/site-profiles/nodes/yourname.xml

File Edit View Terminal Help

```
        scripts for your cluster.

        XML files in the site-nodes/ directory should be named either
        "extend-[name].xml" or "replace-[name].xml", where [name] is
        the name of an existing xml node.

        If your node is prefixed with replace, its instructions will be used
        instead of the official node's. If it is named extend, its directives
        will be concatenated to the end of the official node.

</description>

<changelog>
</changelog>

<main>
        <!-- kickstart 'main' commands go here -->
</main>

<pre>
        <!-- partitioning commands go here -->
</pre>

<!-- There may be as many packages as needed here. Just make sure you only
     uncomment as many package lines as you need. Any empty <package></package>
     tags are going to confuse rocks and kill the installation procedure
-->
<!-- <package> insert 1st package name here and uncomment the line</package> -->
<!-- <package> insert 2nd package name here and uncomment the line</package> -->
<!-- <package> insert 3rd package name here and uncomment the line</package> -->
<package>yourpackagename</package>

<post>
        <!-- Insert your post installation script here. This
        code will be executed on the destination node after the
        packages have been installed. Typically configuration files
        are built and services setup in this section. -->

        <!-- WARNING: Watch out for special XML chars like ampersand,
        greater/less than and quotes. A stray ampersand will cause the
```

Now we have to rebuild the cluster distribution to include our new appliance files.

```
# cd /export/rocks/install
# rocks create distro
```

Only one student should do this step after everyone has completed their custom appliance configuration.

Let's take a quick break while the system builds the distribution.

Module 3: Rocks Rolls

What exactly is a Rocks Roll?

It is a set of xml files and software distribution packages designed for easy deployment to Rocks clusters.

What is the advantage of a Roll?

The main advantage is portability to multiple clusters. If you only manage one cluster the processes we discussed so far will work fine.

How does a roll Work?

The contents of a roll are basically the same set of XML files we have already discussed. A roll consists of node.xml files, graph.xml files, and software packages, organized in a manner that allows them to be automatically installed to cluster nodes.

In newer version of Rocks the developers have made it very easy to create custom Rolls, but it still requires that you have an understanding of how they work, and a basic knowledge of xml. We will begin by examining the contents of a Rocks Roll source code directory.

Structure of a Roll

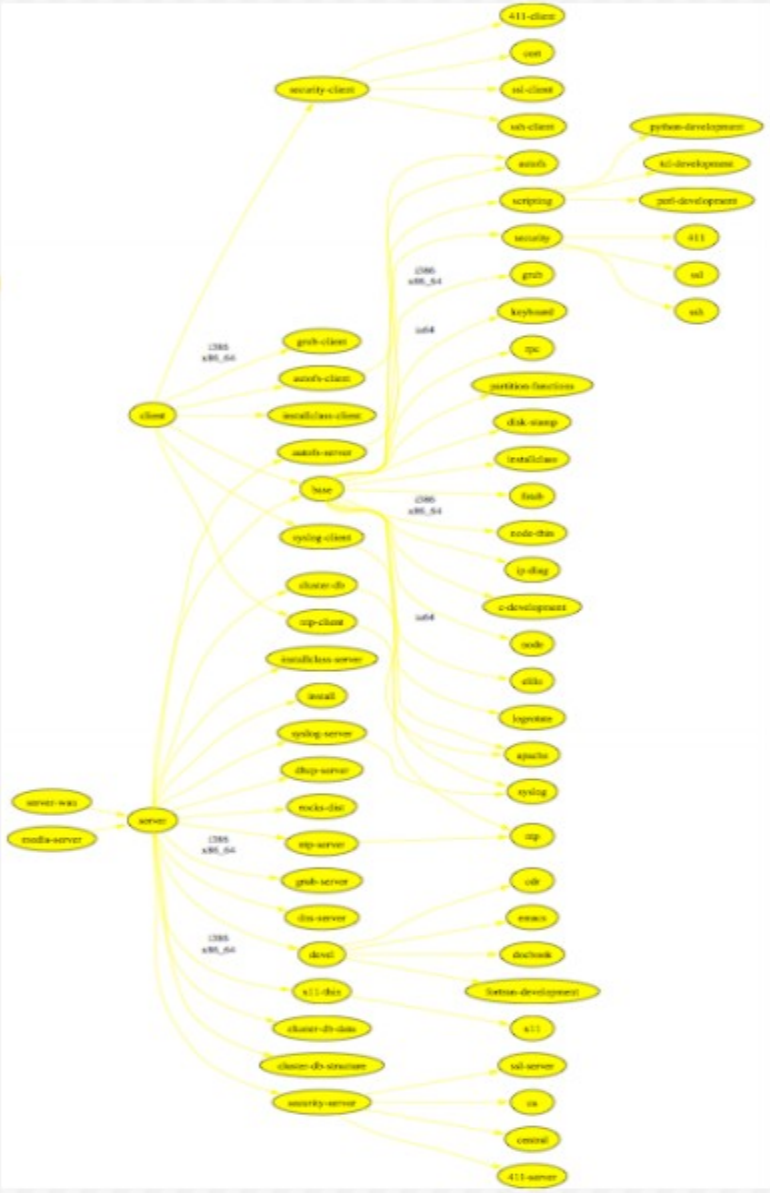
- Makefile
- graphs
  - default
    - rollname.xml
- nodes
  - rollname.xml
- RPMS
- src
  - Makefile
  - linux.mk
  - rollname
    - Makefile
    - version.mk
  - sunos.mk
  - Usersguide
    - Makefile
    - copyrights.sgml
    - Images
      - i-01.png
    - index.sgml
    - installing.sgml
    - preface.sgml
    - using.sgml
    - version.mk
- version.mk

- The only files you need to worry about for simple rolls are the rollname.xml files under graphs and nodes.
- More complex rolls will require adding contents to the src folder.

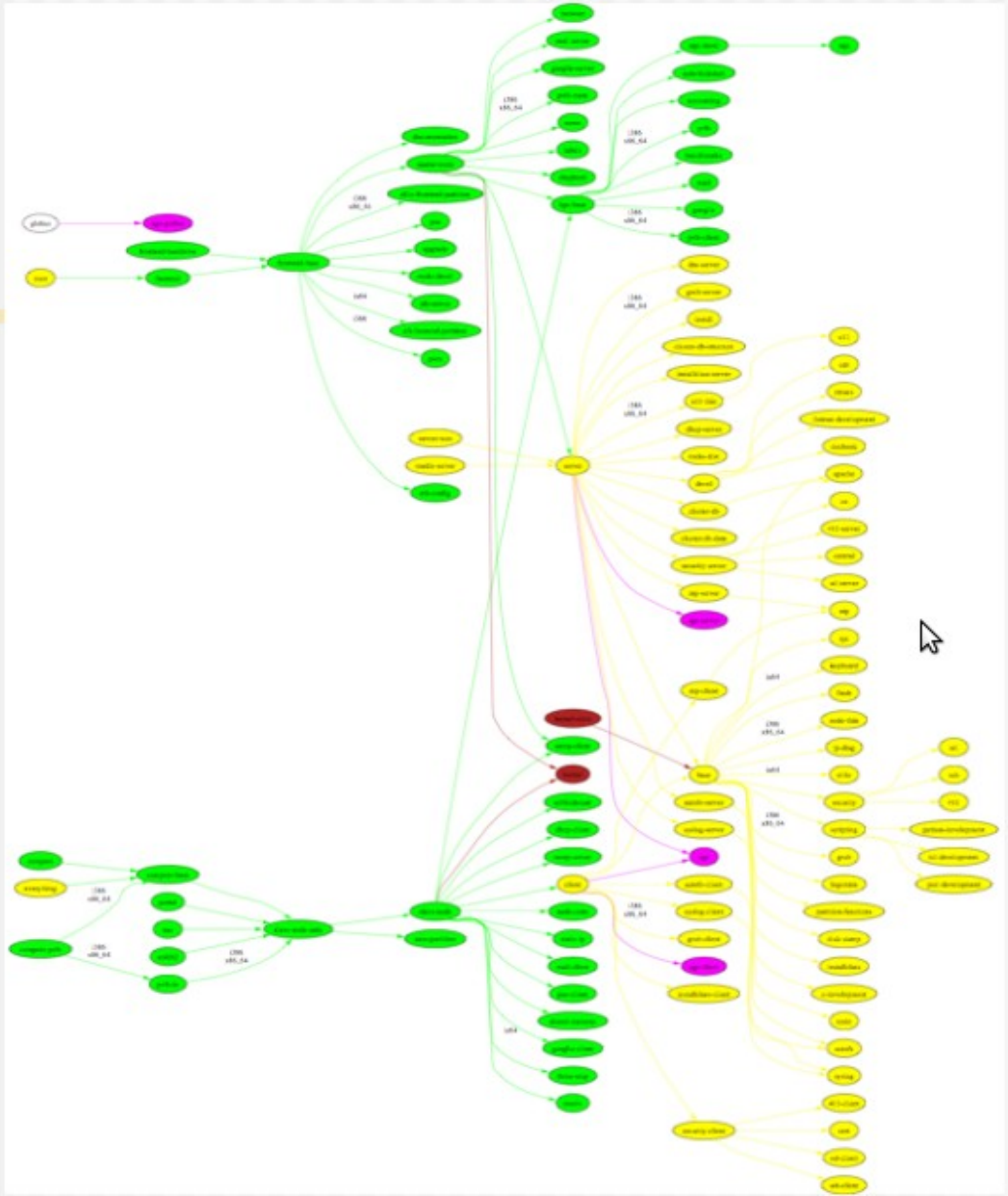
Install Rocks Base Graph

Basic Instructions that define all Rocks Appliances

Rolls have packages and graphs



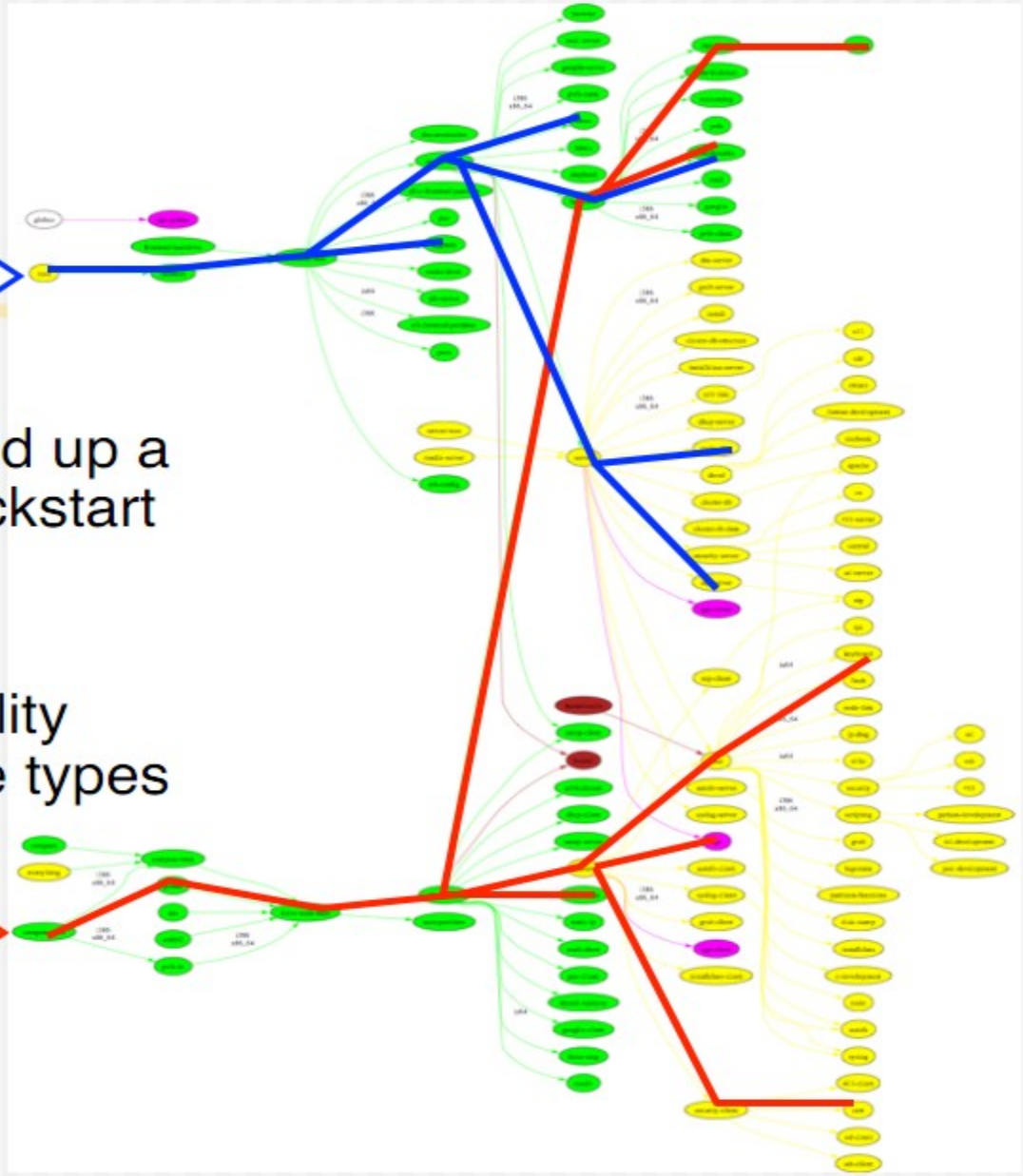
Base + Rolls



Frontend Root

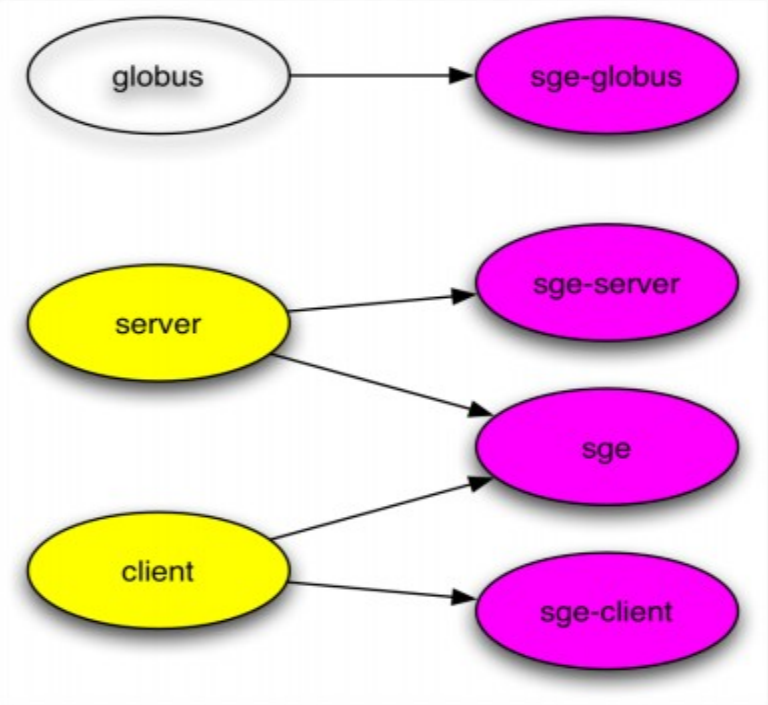
- Traverse a graph to build up a kickstart file (done at kickstart time)
- Flexible
- Easy to share functionality between disparate node types

Compute Root



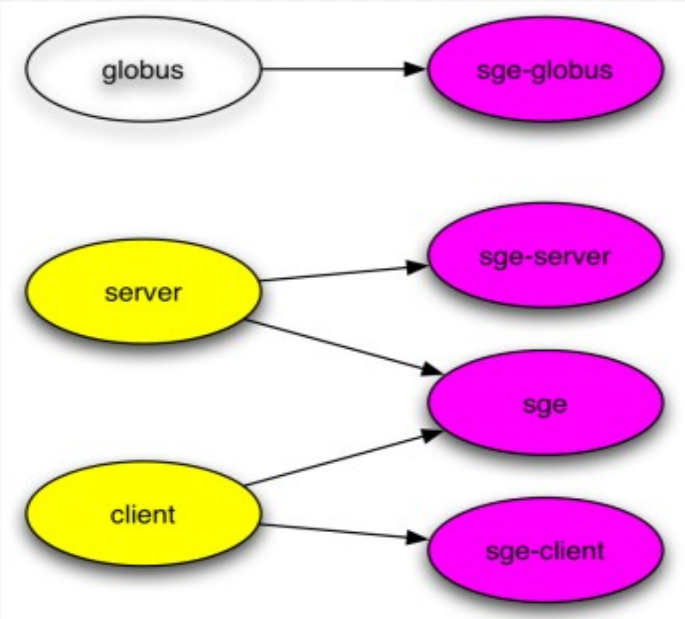
Use Graph Structure to Dissect Distribution

- Use ‘nodes’ and ‘edges’ to build a customized kickstart file
- Nodes contain portion of kickstart file
  - Can have a ‘main’, ‘package’ and ‘post’ section in node file
- Edges used to coalesce node files into one kickstart file



Why We Use A Graph

- A graph makes it easy to 'splice' in new nodes
- Each Roll contains its own nodes and splices them into the system graph file



Lab 3: Build a Custom Roll

Create a Roll Directory

- On Rocks 5.3 System or newer

```
# cd /export/site-roll/rocks/src/roll
# rocks create new roll yourname
```

This will result in a roll called yourname. You can also specify a version number to match the version of a particular software package you are installing.

Part I: Packages

- Rolls require packages to be in native OS format. (RPM for Linux PKG for Solaris)
- Advantages to using packages:
  - You can inspect software with native tools.
  - Can install by hand using OS tools.
  - Easy to track because OS knows about the package.
- Disadvantages
  - You have to create the package
  - Mechanisms can cause odd behavior.

Package build requirements:

- All requirements are met by using a frontend node for the build process.
- Faith
  - There is a large set of included make rules that allow us to quickly package software.
  - You must trust what the system is doing.

Different Ways of Packaging Software:

- Build or install the software by hand, then point "#rocks create package" at the installation directory.
- Build an RPM spec file.
- Use the Rocks-supplied Make infrastructure.
- We will build a package using the first and last methods. Building packages using RPM spec files is beyond the scope of this course.

The remainder of the lab instructions will be in the form of a handout.