
# Roll Development Basics

Rocks-A-Palooza III


# Available Rolls for Rocks 4.2.1

- Rolls we provide
  - Core: Base, Kernel, Web Server, OS, Service Pack
  - Area51 - security analysis tools
  - Bio - bioinformatics tools
  - Condor
  - HPC - MPICH and cluster tools
  - Ganglia - cluster monitoring
  - Grid - Globus
  - PVFS2 - parallel file system
  - SGE
  - Viz
  - Java

# Available Rolls for Rocks 4.2.1

- Rolls produced by academic community
  - PBS/Maui
    - HPC group at University of Tromso, Norway
  - APBS (Adaptive Poisson-Boltzmann Solver)
    - NBCR group, UCSD


# Available Rolls for Rocks 4.2.1

- Rolls produced by commercial entities
  - Voltaire, SilverStorm, Topspin
    - IB Rolls
  - Myricom
    - Myrinet Roll
  - Scalable Informatics
    - ScalableInformatics Roll (cluster tools)


# Roll Contents

- RPMS
  - Your software.
  - Tasks:
    - Package bits into RPM
- Kickstart Graph
  - Your configuration.
  - Tasks:
    - Verify correct files exist after installation
    - Verify correct operation on frontend and compute nodes
    - Test, Test, Test

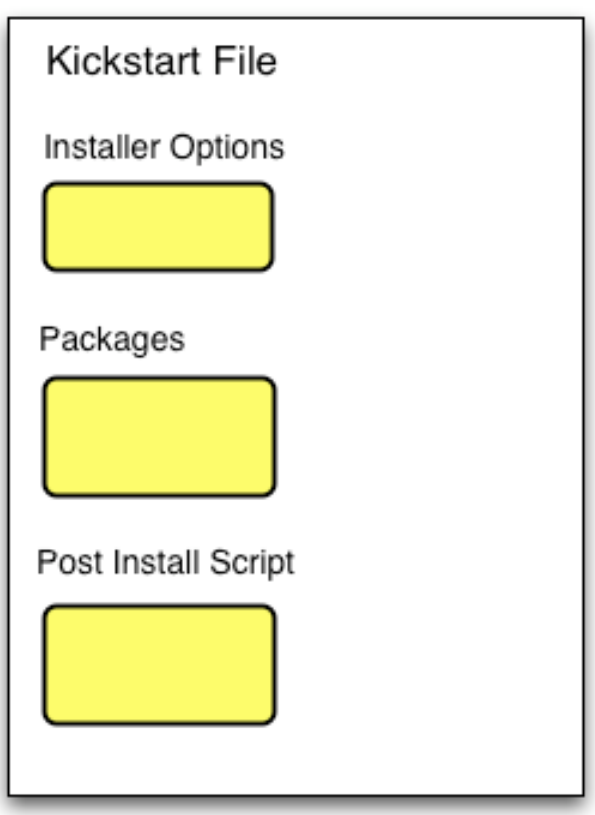
# Rolls Codify Configuration for Cluster Services

- How do you configure NTP on compute nodes?
  - ntp-client.xml:

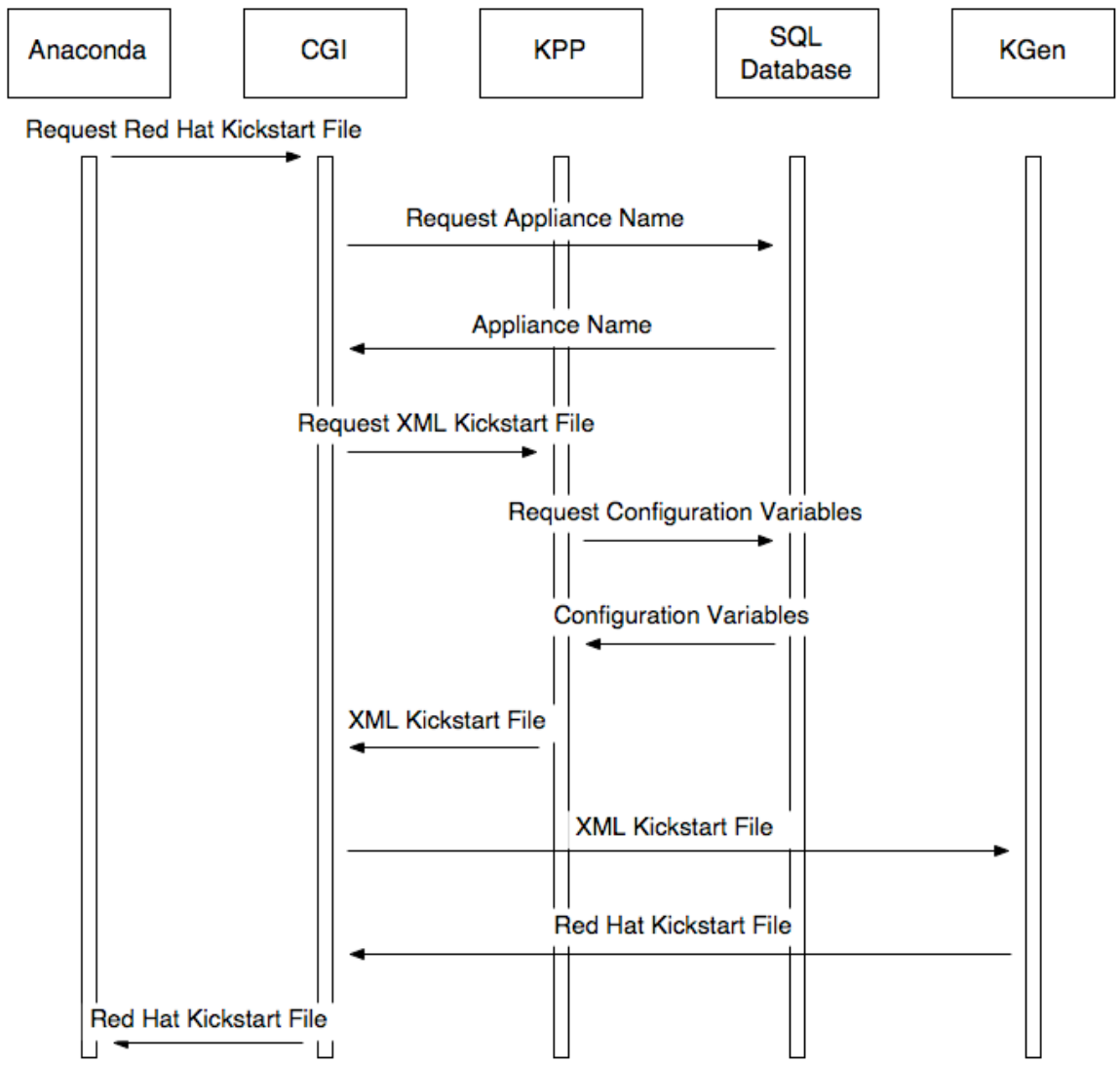
```
<post>

<!-- Configure NTP to use an external server -->

<file name="/etc/ntp.conf">
server <var name="Kickstart_PrivateNTPHost"/>
authenticate no
driftfile /var/lib/ntp/drift
</file>

<!-- Force the clock to be set to the server upon reboot -->

/bin/mkdir -p /etc/ntp

<file name="/etc/ntp/step-tickers">
<var name="Kickstart_PrivateNTPHost"/>
</file>

<!-- Force the clock to be set to the server right now -->

/usr/sbin/ntpdate <var name="Kickstart_PrivateNTPHost"/>
/sbin/hwclock --systohc
</post>
```

# Kickstart File

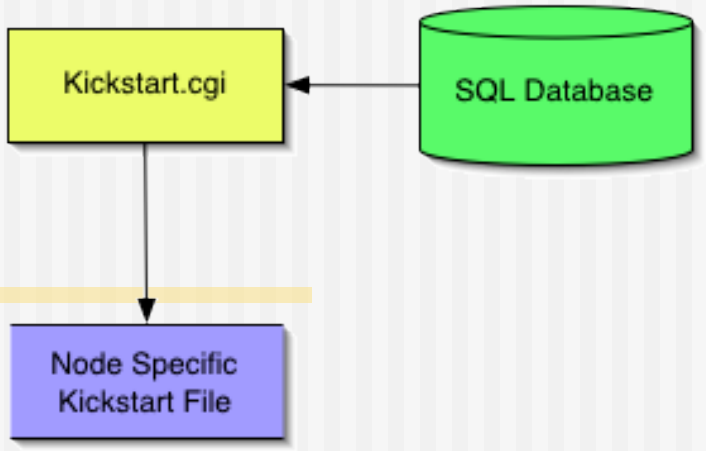
- RedHat's Kickstart: DNA of a node
  - Monolithic flat ASCII file
    - "Main": disk partitioning, timezone
    - "Packages": list of RPM names
    - "Post": shell scripts for config
  - No macro language
  - Requires forking based on site information and node type.



# Getting A Kickstart File

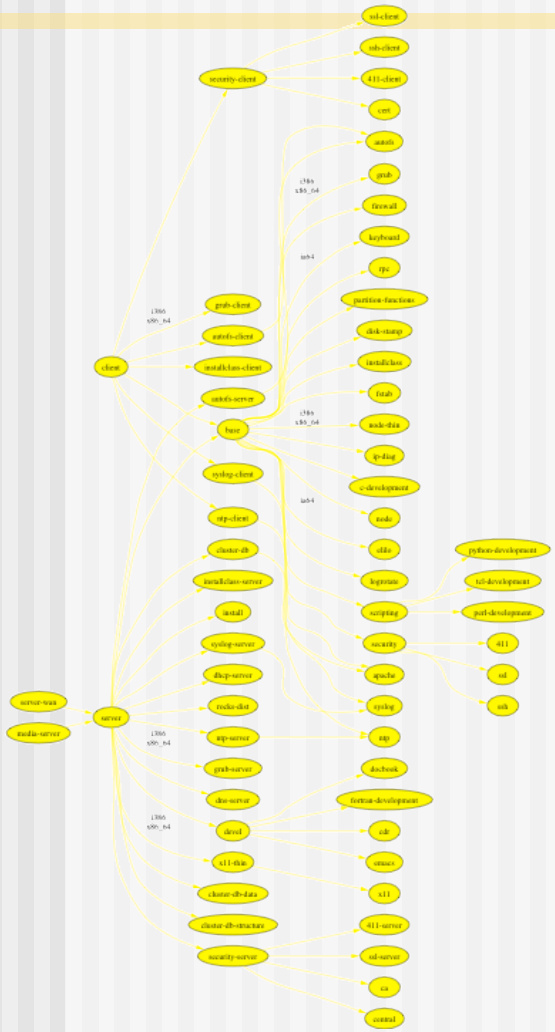


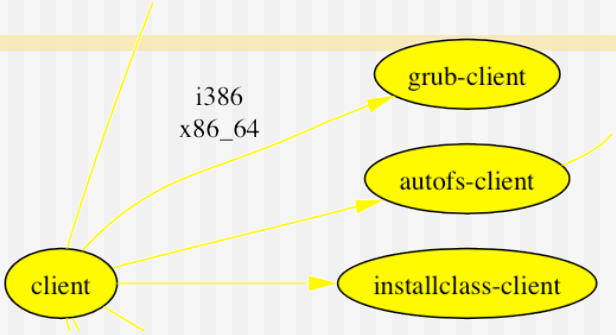
# Kickstart File



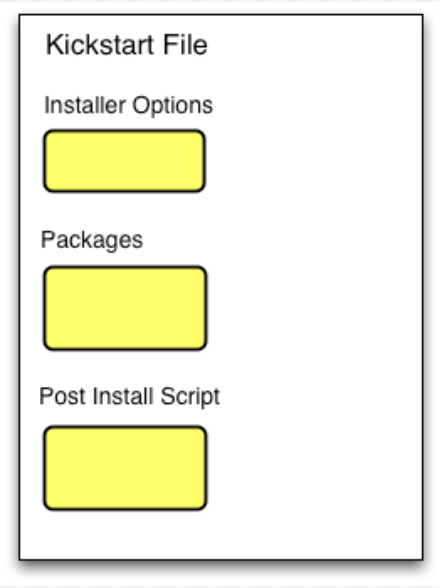

- Rocks XML Kickstart
  - Decompose a kickstart file into nodes and a graph
    - Graph specifies OO framework
    - Each node specifies a service and its configuration
  - SQL Database to help site configuration
  - “Compile” flat kickstart file from a web cgi script

# Kickstart Graph for Kgen

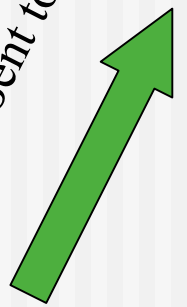




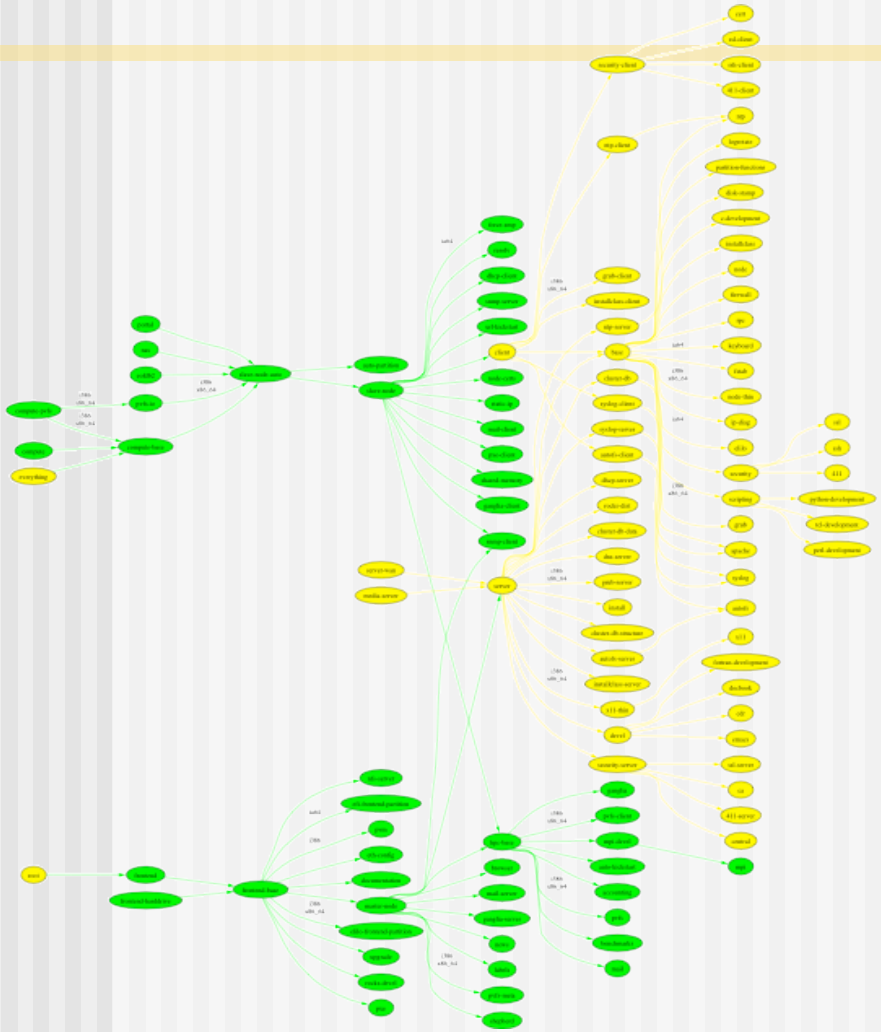

Preprocess (kpp)

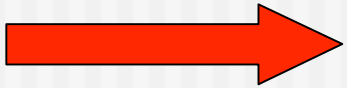


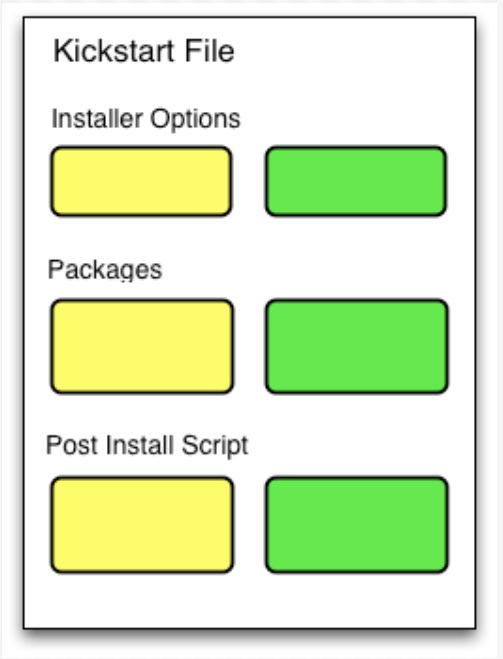

Sent to node (http)



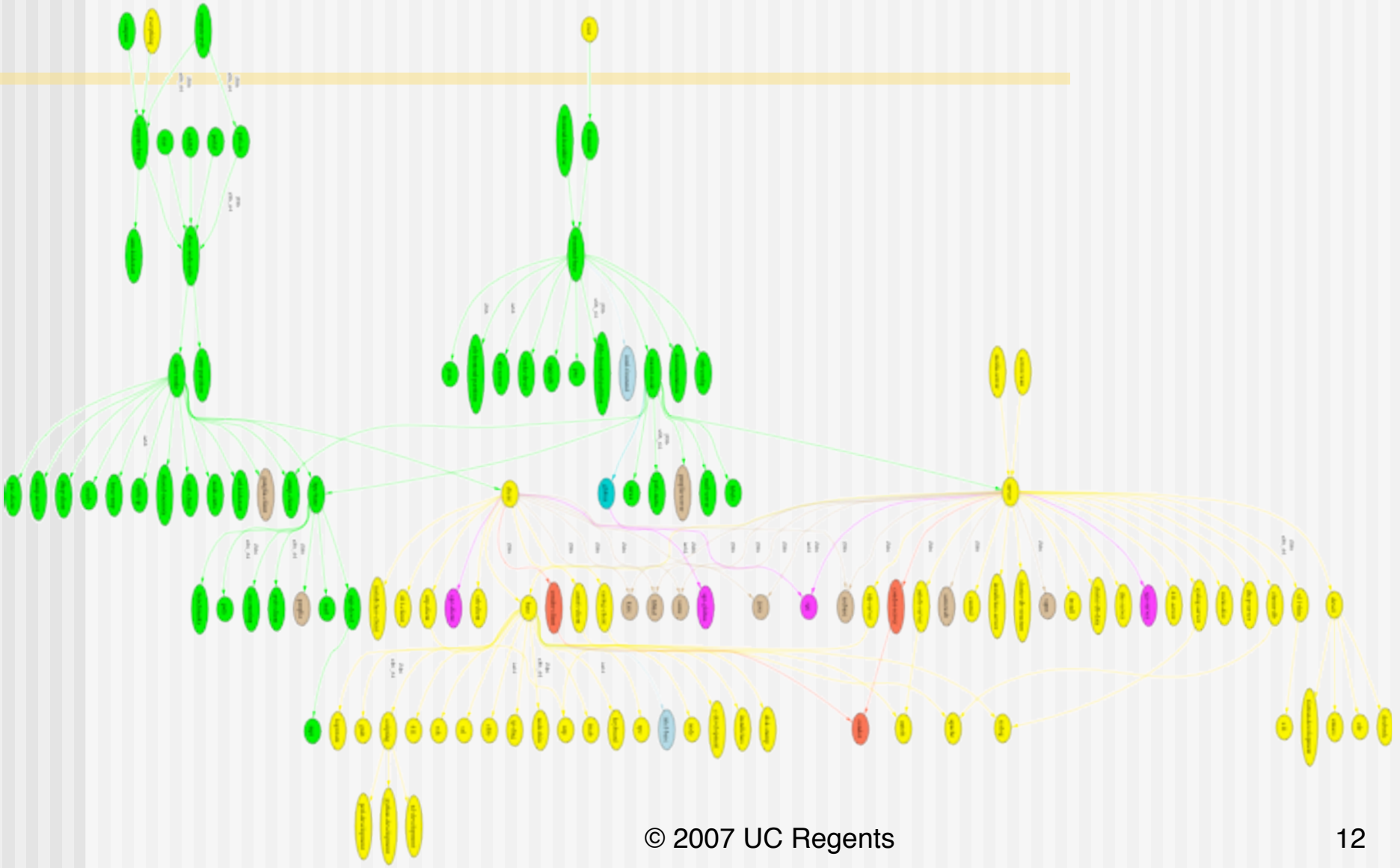
# Kickstart Graph with Roll







# Full Kickstart Graph



# Kickstart XML Language

- Graph contains
  - Nodes
    - Rich language to help with configuration tasks
  - Edges
    - Simple. Defines node *MEMBERSHIP* in compiled kickstart files
  - Order
    - Simple syntax. Defines *POST SECTION ORDER* among nodes.

# Example Roll: Sweetroll

- Will use a fictitious roll named “Sweetroll”

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

<kickstart>
  <description>
The sweet roll.
  <description>
</kickstart>
```

# Kickstart Nodes

- Altering Default Nodes
  - Can *replace* or *extend* default nodes in Roll
    - Extend: concatenated to the end of a default node
    - Replace: overwrite default node
  - *Discouraged use: Reserved for end users*
  - Extend by name: extend-[node].xml
    - sweetroll/nodes/extend-compute.xml
  - Replace by name: replace-[node].xml
    - sweetroll/nodes/replace-compute.xml

# Kickstart Nodes

- Graph
  - Nodes
    - Rich language to help with configuration tasks
    - “Main” section
    - “Package” section
    - “Post” section
    - “Installclass” section
      - Used to modify Anaconda

# Nodes XML Tools: <var>

- Get Variables from Database
  - `<var name="Kickstart_PrivateAddress"/>`
  - `<var name="Node_Hostname"/>`

```
10.1.1.1
compute-0-0
```

  - Can grab any value from the *app_globals* database table

# <var> values from app_globals

| ←T→ | | ID | Membership | Service | Component | Value |
|---|---|---|---|---|---|---|
| Edit | Delete | 6 | 0 | Info | ClusterLatlong | N32.87 W117.22 |
| Edit | Delete | 16 | 0 | Info | ClusterName | Onyx |
| Edit | Delete | 30 | 0 | Info | CertificateState | California |
| Edit | Delete | 34 | 0 | Info | CertificateOrganization | Rocksclusters |
| Edit | Delete | 37 | 0 | Info | CertificateLocality | San Diego |
| Edit | Delete | 44 | 0 | Info | CertificateCountry | US |
| Edit | Delete | 45 | 0 | Info | ClusterURL | http://onyx.rocksclusters.org/ |
| Edit | Delete | 50 | 0 | Info | RocksRelease | Makalu |
| Edit | Delete | 52 | 0 | Info | RocksVersion | 3.3.0 |
| Edit | Delete | 54 | 0 | Info | ClusterContact | admin@onyx.rocksclusters.org |
| Edit | Delete | 58 | 0 | Info | Born | 2005-02-23 14:30:13 |
| Edit | Delete | 1 | 0 | Kickstart | PrivateKickstartBasedir | install |
| Edit | Delete | 2 | 0 | Kickstart | PartsizeRoot | 6000 |
| Edit | Delete | 3 | 0 | Kickstart | PublicAddress | 198.202.88.74 |
| Edit | Delete | 4 | 0 | Kickstart | PublicHostname | onyx.rocksclusters.org |

- Combine “Service” and “Component”
  - For example, Kickstart_PublicAddress


# Nodes XML Tools: <var>

- <var> attributes
  - name
    - Required. Format is “Service_Component”
    - Service and Component relate to column names in the app_global database table.
  - val
    - Optional. Sets the value of this variable
      - <var name=“Info_ClusterName” val=“Seinfeld”/>
  - ref
    - Optional. Set this variable equal to another
      - <var name=“Info_Weather” ref=“Info_Forecast”/>

# Nodes XML Tools: <eval>

- Do processing on the frontend when the kickstart file is generated (by the CGI script):
  - **<eval shell="bash">**
- To insert the Rocks release info in the kickstart file:

```
<eval shell="bash">
cat /etc/rocks-release
</eval>
```

```
Rocks release 4.2.1 (Cydonia)
```

# Nodes XML Tools: <eval>

- <eval> attributes
  - shell
    - Optional. The interpreter to use. Default “sh”
  - mode
    - Optional. Value is quote or xml. Default of quote specifies for kpp to escape any XML characters in output.
    - XML mode allows you to generate other tags:
      - <eval shell=“python” mode=“xml”>
        - import time
        - now = time.time()
        - print “<var name=‘Info_Now’ val=‘%s’/>” % now
      - </eval>

# Nodes XML Tools: <eval>

- Inside <eval> variables are not accessed with <var>: use the environment instead.

```
<eval shell="python">
import os
print "My NTP time server is",
 os.environ['Kickstart_PublicNTPHost']
print "Got it?"
</eval>
```

**My NTP time server is time.apple.com**
**Got it?**

# Nodes XML Tools <include>

- Auto-quote XML characters in a file
  - **`<include file="foo.py"/>`**
- Quotes and includes file

  `sweetroll/include/foo.py`
- foo.py (native) → foo.py (quoted xml):

```
#!/usr/bin/python

import sys

def hi(s):
        print >> sys.stderr, s
```

```
#!/usr/bin/python

import sys

def hi(s):
        print &gt;&gt; sys.stderr, s
```

# Nodes XML Tools: <include>

- <include> attributes
  - file
    - Required. The file to include (relative to “include/”) dir in roll src.
  - mode
    - Optional. Value is quote or xml. Default of quote specifies for kpp to escape any XML characters in file.
      - <include file=“my-favorite-things” mode=“quote”/>

# Nodes XML Tools <file>

- Create a file on the system:
  - **<file name="/etc/hi-mom" mode="append">**
    - **How are you today?**
  - **</file>**
- Used extensively throughout Rocks post sections
  - Keeps track of alterations automatically via RCS.

```
<file name="/etc/hi" perms="444">
How are you today?
I am fine.
</file>
```

```
…RCS checkin commands...
cat > /etc/hi << 'EOF'
How are you today?
I am fine.
EOF
chmod 444 /etc/hi-mom
…RCS cleanup commands…
```

# Nodes XML Tools: <file>

- <file> attributes
  - name
    - Required. The full path of the file to write.
  - mode
    - Optional. Value is “create” or “append”. Default is create.
  - owner
    - Optional. Value is “user.group”, can be numbers or names.
      - <file name=“/etc/hi” owner=“daemon.root”>
  - perms
    - Optional. The permissions of the file. Can be any valid “chmod” string.
      - <file name=“/etc/hi” perms=“a+x”>

# Nodes XML Tools: <file>

- <file> attributes (continued)
  - vars
    - Optional. Value is “literal” or “expanded”. In literal (default), no variables or backticks in file contents are processed. In expanded, they work normally.
      - <file name=“/etc/hi” vars=“expanded”>
        - The current date is `date`
      - </file>
  - expr
    - Optional. Specifies a command (run on the frontend) whose output is placed in the file.
      - <file name=“/etc/hi” expr=“/opt/rocks/dbreport hi”/>

# Fancy <file>: nested tags

```
<file name="/etc/hi">

Rocks release:
<eval>
date +"%d-%b-%Y"
echo ""
cat /etc/rocks-release
</eval>

</file>
```

…RCS checkin commands...
**cat > /etc/hi << 'EOF'**

**Rocks release:**
**13-May-2005**

**Rocks release 4.2.1 (Cydonia)**

**EOF**
…RCS cleanup commands…

# Nodes Main

- Used to specify basic configuration:
  - timezone
  - mouse, keyboard types
  - install language
- Used more rarely than other tags
- Rocks main tags are usually a straight translation:

```
<main>

  <timezone>America/Mission_Beach
  </timezone>

</main>
```

```
…
timezone America/Mission_Beach
…
rootpw --iscrypted sndk48shdlwis
mouse genericps/2
url --url http://10.1.1.1/install/rocks-dist/..
```

# Nodes Main: Partitioning

- `<main>`
  - `<part> / --size 8000 --ondisk hda </part>`
  - `<part> swap --size 1000 --ondisk hda </part>`
  - `<part> /mydata --size 1 --grow --ondisk hda </part>`
- `</main>`

```
part / --size 8000 --ondisk hda
part swap --size 1000 --ondisk hda
part /mydata --size 1 --grow --ondisk hda
```

# Nodes Packages

- **`<package>java</package>`**
  - Specifies an RPM package. Version is automatically determined: take the *newest* rpm on the system with the name 'java'.
- **`<package arch="x86_64">java</package>`**
  - Only install this package on x86_64 architectures
- **`<package arch="i386,x86_64">java</package>`**

```
<package>newcastle</package>
<package>stone-pale</package>
<package>guinness</package>
```

```
%packages
newcastle
stone-pale
guinness
```

# Nodes Packages

- RPMS are installed brute-force: no dependancy checking, always --force

# Nodes Packages

- RPM name is a basename (not fullname of RPM)
  - For example, RPM name of package below is ‘kernel’

```
# rpm -qip /home/install/rocks-dist/lan/i386/RedHat/RPMS/kernel-2.6.9-22.EL.i686.rpm
Name        : kernel                       Relocations: (not relocatable)
Version     : 2.6.9                           Vendor: CentOS
Release     : 22.EL                        Build Date: Sun 09 Oct 2005 03:01:51 AM WET
Install Date: (not installed)              Build Host: louisa.home.local
Group       : System Environment/Kernel    Source RPM: kernel-2.6.9-22.EL.src.rpm
Size        : 25589794                        License: GPLv2
Signature   : DSA/SHA1, Sun 09 Oct 2005 10:44:40 AM WET, Key ID a53d0bab443e1821
Packager    : Johnny Hughes <johnny@centos.org>
Summary     : the linux kernel (the core of the linux operating system)
Description :
The kernel package contains the Linux kernel (vmlinuz), the core of any
Linux operating system
```


# Nodes Post

- `<post>` for *Post-Install* configuration scripts
- Configuration scripts in <post> section run after *all* RPMs have been installed.
  - Useful: you have all your software available
  - Scripts run in “target” environment: /etc in <post> will be /etc on the final installed system
- Scripts are always non-interactive
  - No Human is driving

# Nodes Post

ntp-client.xml

```
<post>

/bin/mkdir -p /etc/ntp
/usr/sbin/ntpdate <var name="Kickstart_PrivateNTPHost"/>
/sbin/hwclock --systohc

</post>
```

```
%post

/bin/mkdir -p /etc/ntp
/usr/sbin/ntpdate 10.1.1.1
/sbin/hwclock --systohc
```


# Nodes Post Section

- Scripts have minimal $PATH (/bin, /usr/bin)
- Error reporting is minimal
  - Write to personal log file if you need debugging
- Not all services are up. Network is however.
  - Order tag is useful to place yourself favorably relative to other services
- Can have multiple <post> sections in a single node

# Nodes XML Tools: <post>

- <post> attributes
  - arch
    - Optional. Specifies which architectures to apply package.
  - arg
    - Optional. Anaconda arguments to *%post*
      - --nochroot (rare): operate script in install environment, not target disk.
      - --interpreter: specifies script language
      - <post arg="--nochroot --interpreter /usr/bin/python">

# Post Example: PXE config

```
<post arch="x86_64,i386">
mkdir -p /tftpboot/pxelinux/pxelinux.cfg

<file name="/tftpboot/pxe../default">
default ks
prompt 0
label ks
        kernel vmlinuz
        append ks inird=initrd.img……
</file>
</post>

<post arch="ia64">

<!-- Itaniums do PXE differently -->
…

</post>
```

for an x86_64 machine:

```
cat >> /root/install.log << 'EOF'
./nodes/pxe.xml: begin post section
EOF
mkdir -p /tftpboot/pxelinux/pxelinux.cfg

…RCS…
cat > /tftpboot/pxe../default << EOF
default ks
prompt 0
…
EOF
..RCS…
```

# A Real Node file: ssh

```
<kickstart>
          <description>
          Enable SSH
          </description>

          <package>openssh/package>
          <package>openssh-clients</package>
          <package>openssh-server</package>
          <package>openssh-askpass</package>
<post>

<file name="/etc/ssh/ssh_config">
Host *
         CheckHostIP                 no
         ForwardX11                  yes
         ForwardAgent                yes
         StrictHostKeyChecking       no
         UsePrivilegedPort           no
         FallBackToRsh               no
         Protocol                    1,2
</file>

chmod o+rx /root
mkdir /root/.ssh
chmod o+rx /root/.ssh

</post>
</kickstart>
```

# Graph Edges

- <edge>
- Specifies *membership* in a kickstart file
  - To make a kickstart file for a compute node:
    1. Take contents of “compute” xml node
    2. Follow all outgoing edges from “compute”
    3. Take all contents of child node
    4. Follow all its outgoing edges, etc, etc, etc
  - Edges between nodes listed in a “graph” file
    - sweetroll/graphs/default/sweetroll.xml
  - All graph files concatenated together
    - E.g., base.xml, hpc.xml, sweetroll.xml, etc. all concatenated

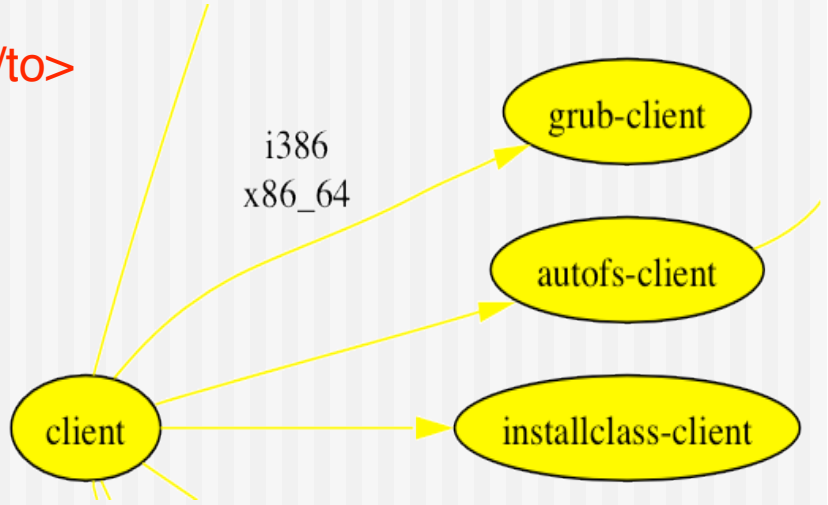
# Graph Edges: <edge>

- <edge> attributes
  - from
    - Required. The name of a node at end of the edge
      - <edge from="base" to="autofs"/>
  - to
    - Required. The name of a node at the head of an edge
  - arch
    - Optional. Which architecture should follow this edge. Default is all.
  - gen
    - Optional. Which generator should follow this edge. Default is "kgen"

# Graph Edges

<edge from="security-server" to="central"/>

<edge from="client">
    <to arch="i386,x86_64">grub-client</to>
    <to>autofs-client</to>
    <to>installclass-client</to>
</edge>



# Graph Ordering

- Added recently to give us control over when node <post> sections are run
  - <order head="database">
    - <tail>database-schema</tail>
  - </order>
- *database* node appears before *database-schema* in all kickstart files.
- Special HEAD and TAIL nodes represent “first” and “last” (post sections that you want to run first/last)
  - <order head=“installclass” tail=“HEAD”/> BEFORE HEAD
  - <order head=“TAIL” tail=“postshell”/> AFTER TAIL

# Graph Ordering: <order>

- <order> attributes
  - head
    - Required. The name of a node whose <post> section will appear BEFORE in the kickstart file.
  - tail
    - Required. The name of a node whose <post> section will appear AFTER in the kickstart file.
      - <order head="grub" tail="grub-server"/>
  - arch
    - Optional. Which architecture should follow this edge. Default is all.
  - gen
    - Optional. Which generator should follow this edge. Default is "kgen"


# When Things Go Wrong

- Test your Kickstart Graph
  - Check XML syntax: xmllint
  - Make a kickstart file
    - Make kickstart file as a node will see it

```
# dbreport kickstart compute-0-0 > /tmp/ks.cfg
```

# When Things Go Wrong

- Test your Kickstart Graph
  - Check XML syntax: xmllint
    - # cd sweetroll/nodes
    - **# xmllint --noout sweetroll.xml**

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

<kickstart>
  <description>
The sweet roll. This roll is just sweet!
  <description>
</kickstart>
```

```
# xmllint --noout sweetroll.xml

sweetroll.xml:7: parser error : Opening and
ending tag mismatch: description line 6 and
kickstart
</kickstart>
                ^
```

# When Things Go Wrong

- Test your Kickstart Graph
  - Make a kickstart file
  - First, install Sweetroll on the frontend “on-the-fly”:
    - # make roll; mount -o loop sweetroll-*.iso /mnt/cdrom
    - # rocks-dist copyroll; umount /mnt/cdrom
    - # cd /home/install; rocks-dist dist
    - # kroll sweetroll > /tmp/install-sweetroll.sh
    - # sh /tmp/install-sweetroll.sh

# When Things Go Wrong

- Test your Kickstart Graph
  - With Sweetroll XML in place:

```
# dbreport kickstart compute-0-0 > /tmp/ks.cfg
```

  - Open /tmp/ks.cfg and look for the section:

```
cat >> /root/install.log << 'EOF'
./nodes/sweetroll.xml: begin post section
```

  - (We do this 10 times a day during release phase)
  - *Exactly the same as what a compute node actually sees during installation*


# When Things Go Wrong

- Test your Kickstart Graph
  - Low level functionality test: kpp
    - Run the kickstart compilers by hand
      - For more difficult to diagnose problems
  - KPP is Kickstart Pre Processor: runs <eval>, <var>
  - KGEN is generator: turns XML into kickstart
    - # cd /home/install/rocks-dist/lan/x86_64/build
    - # kpp sweetroll
    - # kpp sweetroll | kgen


# RPM Building


# Building an RPM

- Generic RPMs are built with ‘spec’ file and ‘rpmbuild’
- It takes time to learn how to write a spec file
- Can use Rocks development source tree to create RPMs without having to make a spec file

# Building an RPM

- We'll do the full procedure in the 'Building Your Own Roll Lab'
- Short story
  - Checkout rocks development source tree
  - Make a new roll from a 'template' roll
  - Download the source tarball
  - Update a description file (version.mk)
  - Execute: make rpm
    - Assumes tarball adheres to 'configure, make, make install'


# Loader Modifications


# Loader Modifications

- The first program that runs during a RedHat install is a C program called “loader”
- Performs low-level setup
  - Loads drivers
  - Configures network
  - Downloads anaconda
  - Gets kickstart file

# Loader Modifications

- Make HTTP the default install method
  - RedHat uses NFS as default
- Rationale
  - Installation is read-only, don't need a file system
  - HTTP traffic can be easily load balanced
  - Peer-to-peer networks use HTTP

# Loader Modifications

- Robust kickstart file acquistion
  - 10 retries to get kickstart file
    - RedHat has only 1
  - NACK to throttle kickstart file acquistion
    - When load on frontend is high, the compute node is told to wait before next retry
- Rationale
  - The kickstart file is everything -- without it, a node is just a $2,000 paperweight
  - NACK feature is for supporting large cluster reinstallations

# Loader Modifications

- Watchdog
  - If can't get kickstart file or if there is an error during the installation, reboot
    - This will restart the installation
    - RedHat just halts
- Rationale
  - Again, the kickstart file is everything

# Loader Modifications

- Network-based frontend installations
  - In Rocks lingo: a “central” install
- Rationale
  - The “CD dance” during installation is not optimal
  - Needed to support grids of clusters from a central place
  - Huge benefit for development
    - Don’t have to burn CDs just to test code changes

# Loader Modifications

- Secure kickstart
  - Added HTTPS support
- Rationale
  - Needed for support of network-based frontend installations (“central” installs)
    - Don’t want the root password for the frontend sent over the network in the clear!
  - Useful for compute nodes that are installed over a public network

# Loader Modifications

- Support adding compute node to any ethernet interface
  - The first interface that receives a kickstart file, is anointed ‘eth0’
- Rationale
  - Email reduction
    - We got lots of email from people who plugged their ethernet cable into the “wrong” port


# Loader Modifications

- Bug Fixes
  - Added support for multiple CD drives
  - A couple stack overflow problems
- Rationale
  - Without the fixes, the installer halts