Roll Development Basics

Practical Assistance for Building Rocks Rolls

Rocks-A-Palooza I

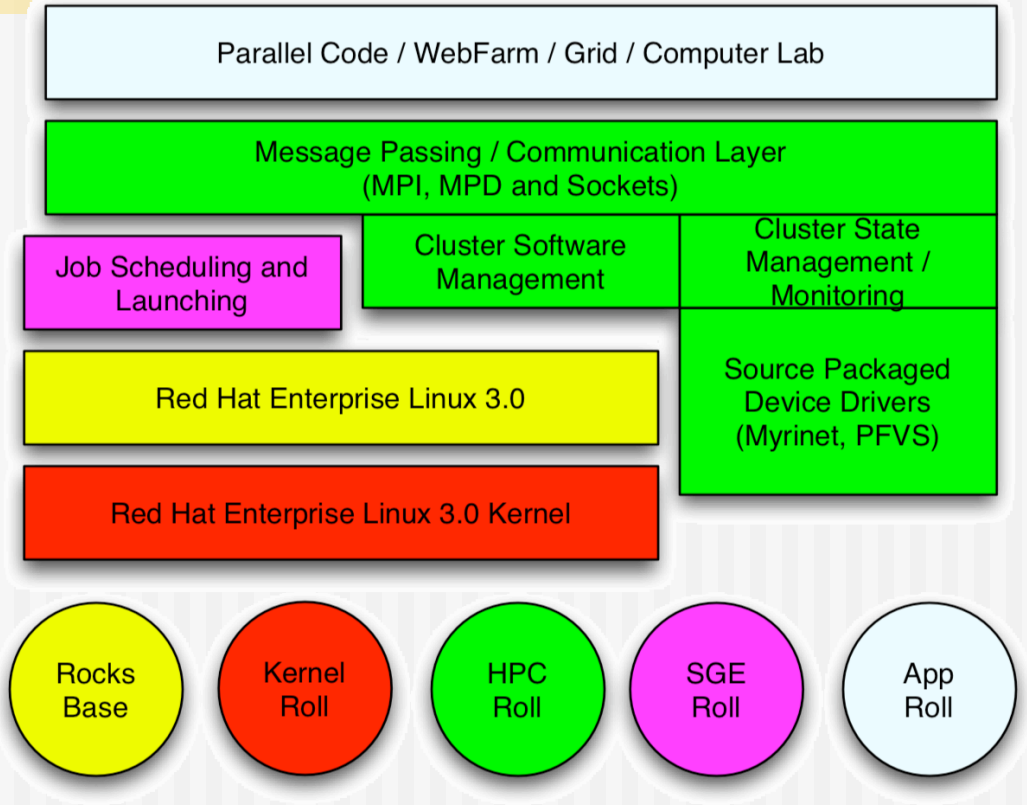
Track 2, Session 2

Federico Sacerdoti 2005

SDSC

Roll Overview

- Software Modules for Cluster
- Rolls auto install and configure a set of services
  - Injected into Rocks installer
  - No interaction during install
  - Fully tested before release



Available Rolls (circa 3.3.0)

- **Area51**
  - Tripwire and rootkit
- **Condor**
  - High-throughput computing grid package
- **IB**
  - Infiniband drivers and MPI from Infinicon
- **Intel**
  - Compiler and libraries for Intel-based clusters (Scalable Systems)
- **Grid**
  - NMI packaging of Globus
- **PBS/Maui**
  - Job scheduling
- **SCE**
  - Scalable cluster environment (Thailand)
- **SGE**
  - Job scheduling
- **Viz**
  - Easily set up nVidia-based viz clusters
- **Java**
  - Java environment
- **RxC**
  - Graphical cluster management tool (Scalable Systems)
- **Lava**
  - Workload management (Platform Computing)
- **IB-Voltaire**
  - Infiniband drivers and MPI from Voltaire

Roll Contents

- RPMS
  - Your software. Devel time: short (hopefully)
  - Tasks:
    - Package bits into RPM
- Kickstart Graph
  - Your configuration. Devel time: long
  - Tasks:
    - Verify correct files exist after installation
    - Verify correct operation on frontend, computes.
    - Test, Test, Test

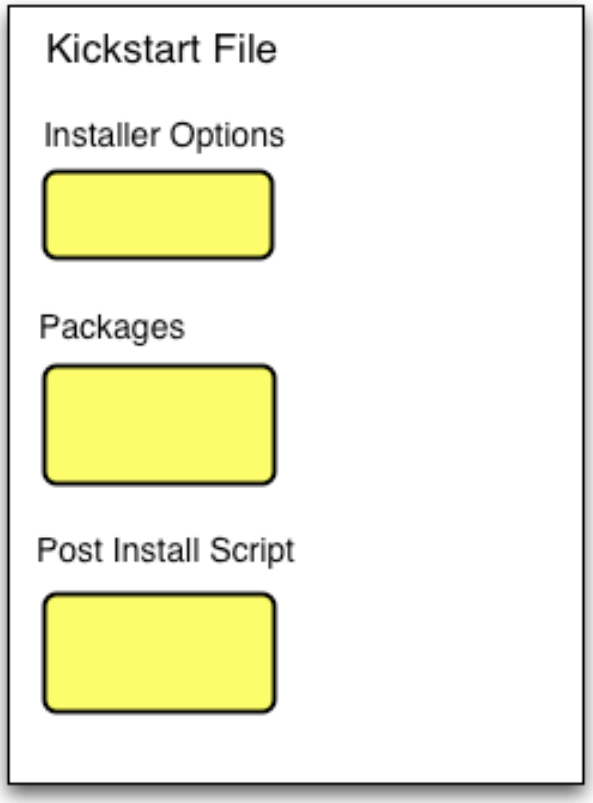
Rolls Codify Configuration for Cluster Services

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

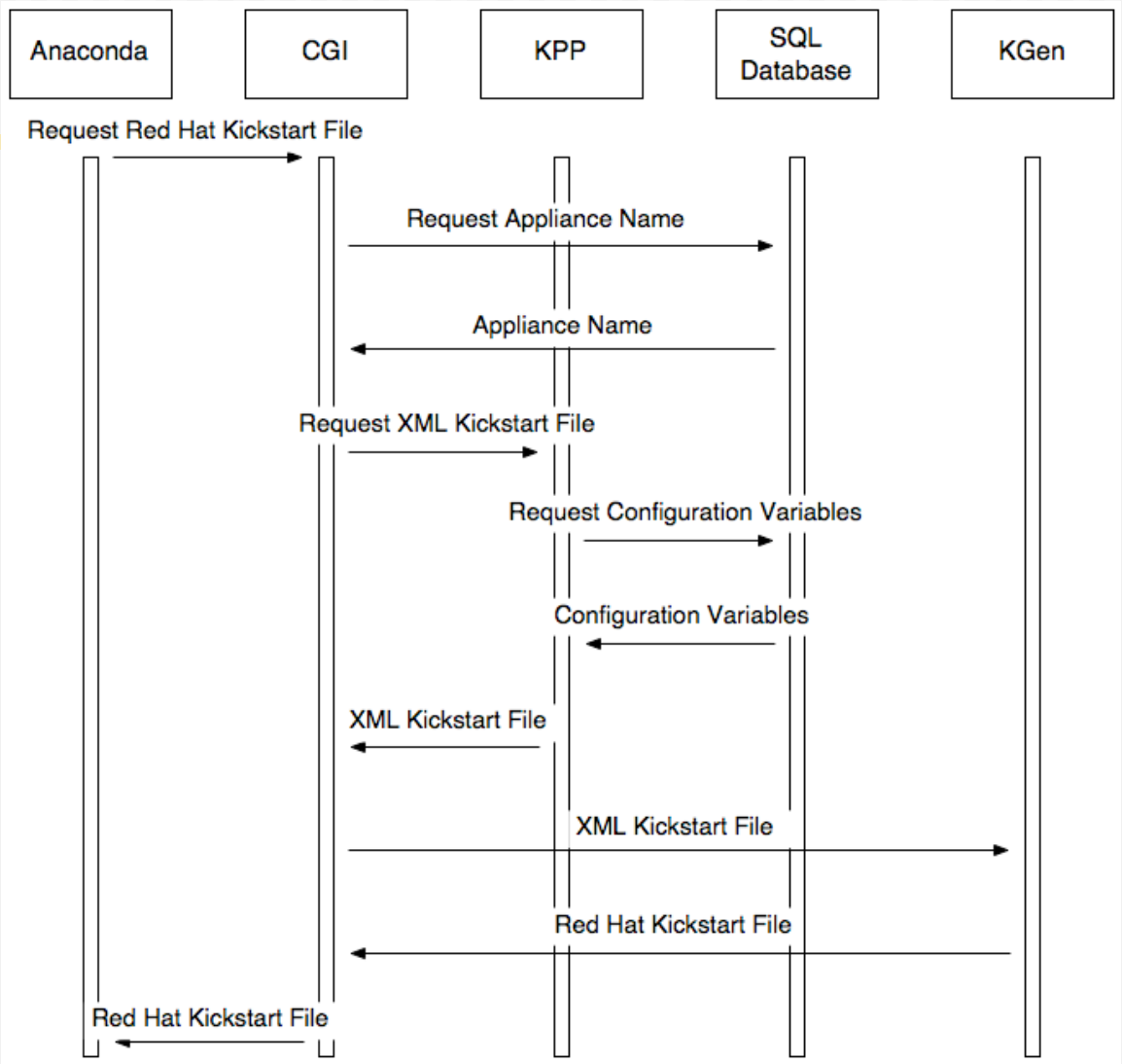
```
<post>

<!-- 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

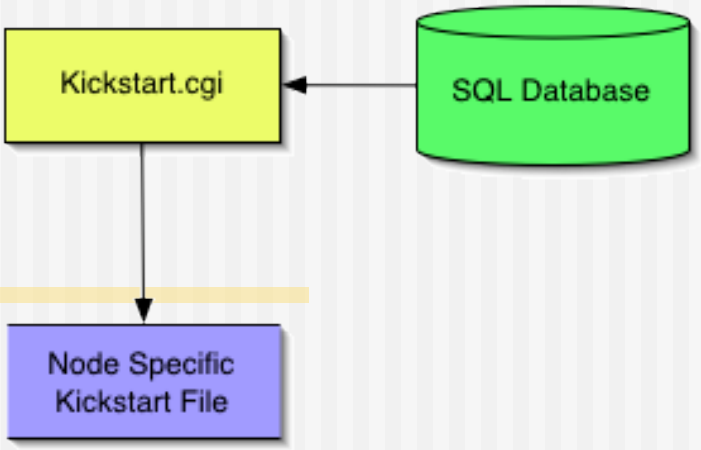
- Red Hat's Kickstart: DNA of a node
  - Monolithic flat ASCII file
    - "Main": disk partitioning, timezone
    - "Packages": list of RPM names
    - "Post": shell scripts for config
  - No macro language
  - Requires forking based on site information and node type.



Getting A Kickstart File

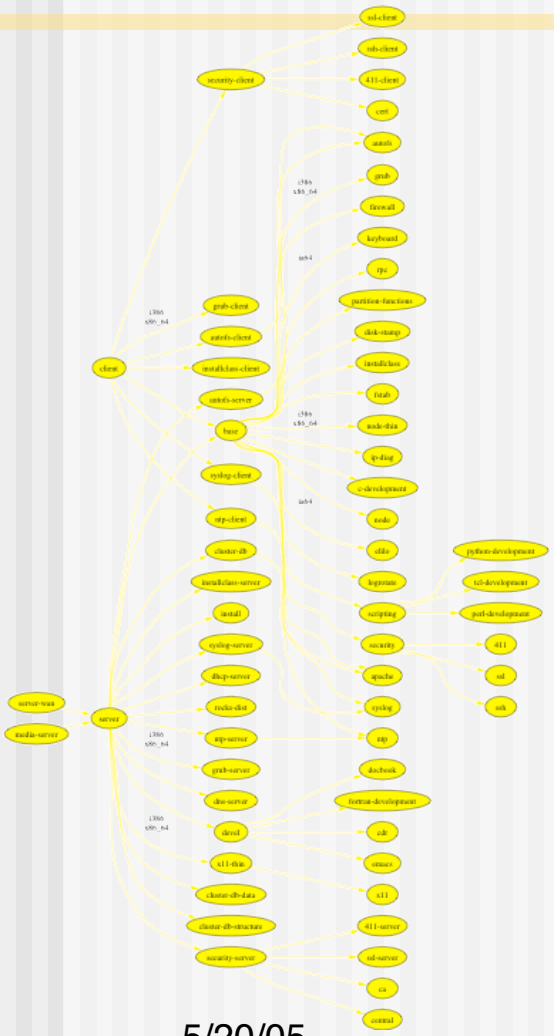


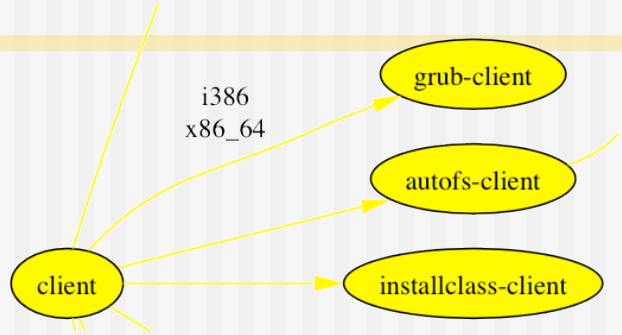
Kickstart File



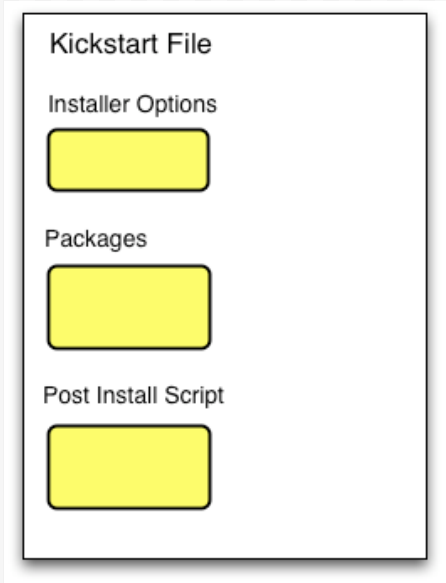
- Rocks XML Kickstart
  - Decompose a kickstart file into nodes and a graph
    - Graph specifies OO framework
    - Each node specifies a service and its configuration
  - SQL Database to help site configuration
  - “Compile” flat kickstart file from a web cgi script

Kickstart Graph for Kgen



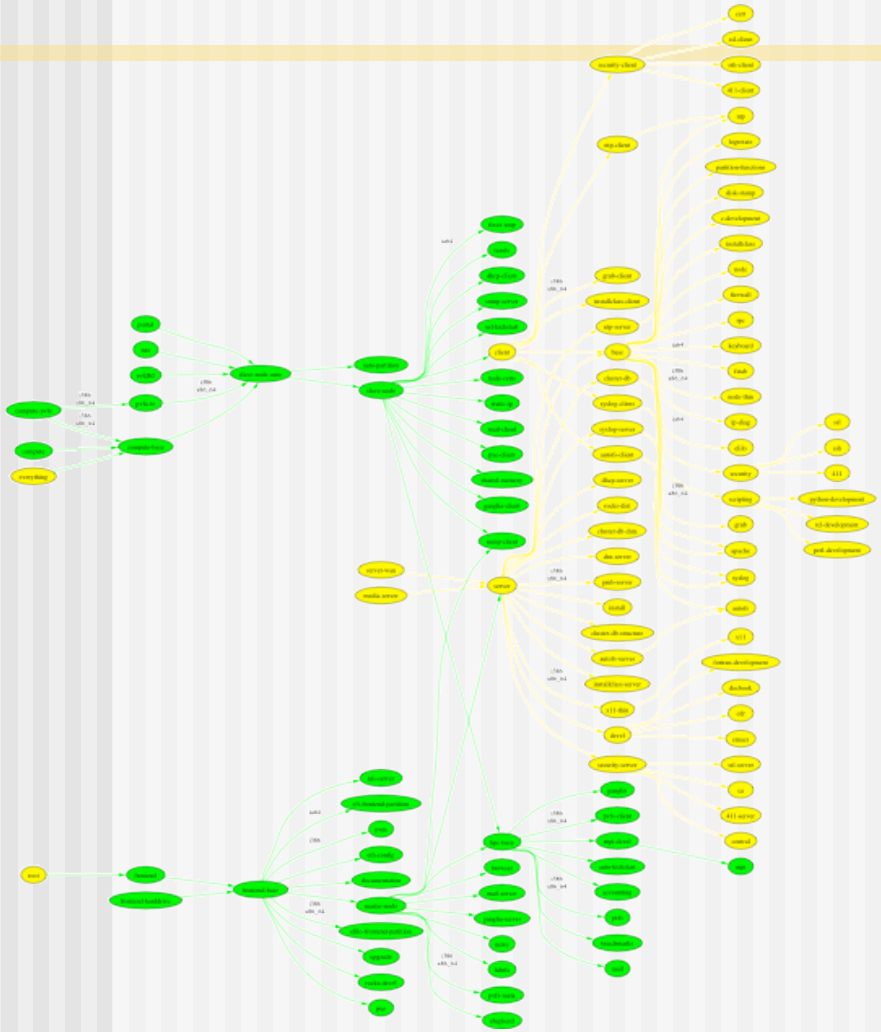


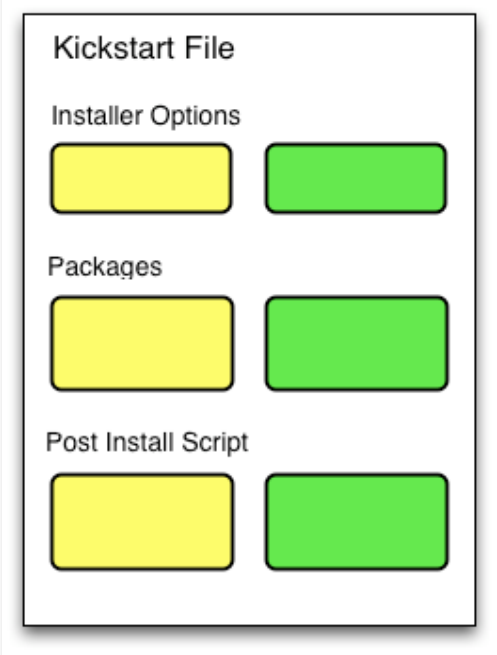
Compile (kgen)



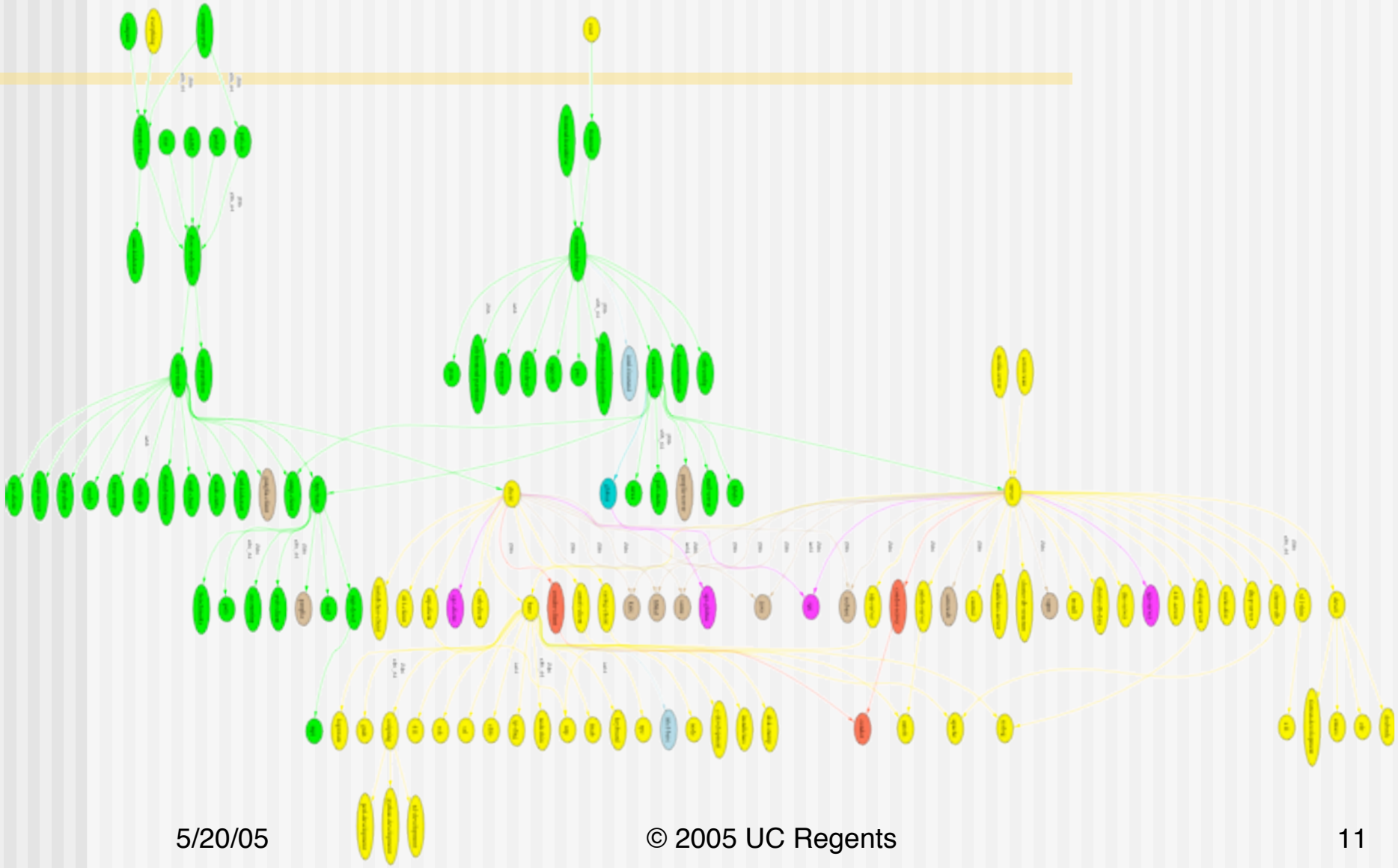
Sent to node (http)

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

Kickstart Nodes

- Altering Default Nodes
  - Can *replace* or *extend* default nodes in Roll
    - Extend: concatenate extend and default nodes
    - 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
    - “Installclass” section
    - “Post” section

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

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_Forcast”/>

Nodes XML Tools: <eval>

- Do processing on the frontend:
  - **<eval shell="bash">**
- To insert a fortune in the kickstart file:

```
<eval shell="bash">
/usr/games/fortune
</eval>
```

"Been through Hell?
Whaddya bring back for
me?"
-- A. Brilliant

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="sh">
echo "My NTP time server is
$Kickstart_PublicNTPHost"
echo "Got it?"
</eval>
```

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

```
<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 | 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 I 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">

Here is your fortune for today:
<eval>
date +"%d-%b-%Y"
echo ""
/usr/games/fortune
</eval>

</file>
```

…RCS checkin commands...

cat > /etc/hi << 'EOF'

Here is your fortune for today:
13-May-2005

"Been through Hell? Whaddya bring back for me?"
-- A. Brilliant

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 ususally 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 4096 --ondisk hda </part>`
  - `<part> swap --size 1000 --ondisk hda </part>`
  - `<part> /mydata --size 1 --grow --ondisk hda </part>`
- `</main>`

```
part / --size 4096 --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
- RPM name is not just filename, but also contained in a header of the rpm file itself.
  - Important for kernel rpm especially

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

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

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/…/pxelinux.cfg/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

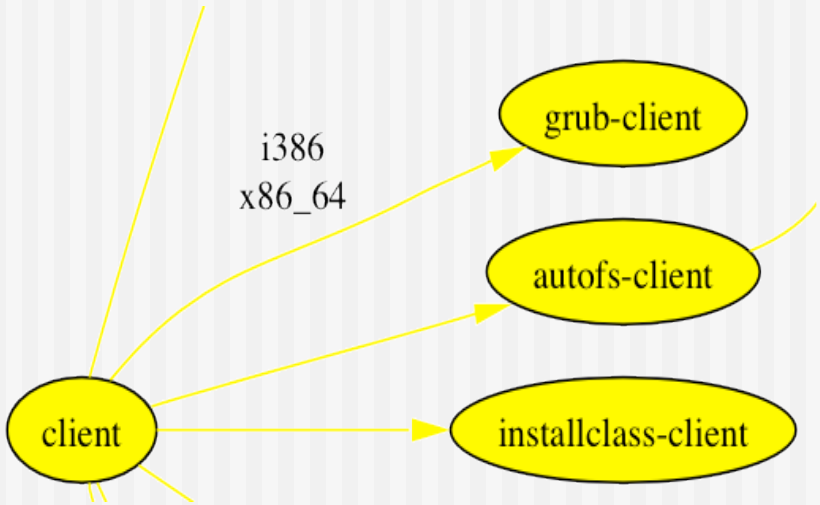
- <edge>
- Specifies *membership* in a kickstart file
  - To make a kickstart file for a compute node type:
    1. Take contents of “compute” xml node
    2. Follow all outgoing edges from “compute”
    3. take all contents of child node
    4. follow all its outgoing edges, etc, etc, etc
  - Edges between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

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>
  - <to>411-client</to>
- </edge>



Graph Ordering

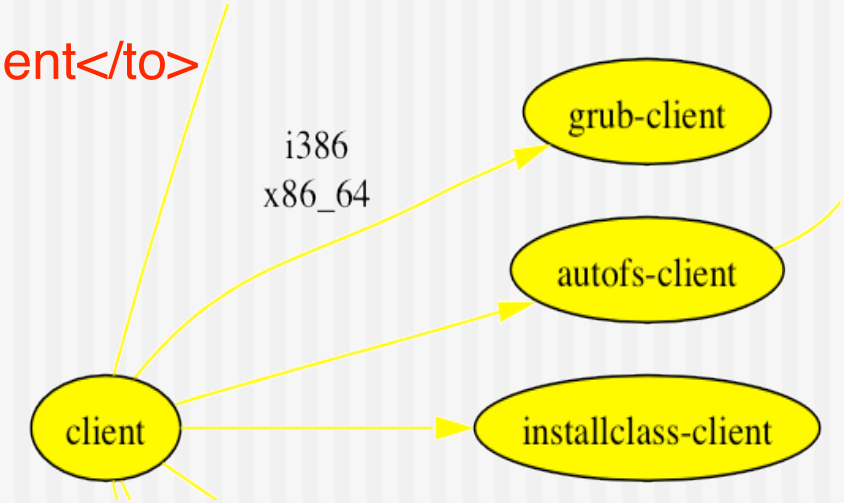
- Added recently to give us control over when node <post> sections are run
  - <order head="database">
    - <tail>database-schema</tail>
  - </order>
- *database* node appears before *database-schema* in all kickstart files.
- Special HEAD and TAIL nodes represent “first” and “last”
  - <order head=“installclass”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

Graph Ordering

- Added recently to give us control over when node POST sections are run
- <edge from="client">
  - <to arch="i386,x86_64">grub-client</to>
  - <to>autofs-client</to>
  - <to>installclass-client</to>
  - <to>411-client</to>
- </edge>



When Things Go Wrong

- Test your Kickstart Graph
  - Check XML syntax: xmllint
  - Make a kickstart file: kickstart.cgi
    - Make kickstart file as a node will see it
  - Low level functionality test: kpp
    - Run the kickstart compilers by hand

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: kickstart.cgi
    - Checks variable substitution, etc.
  - First: install Sweetroll's kickstart graph
    - **Full roll build/install:**
      - # make roll; mount -o loop sweetroll-*.iso /mnt/cdrom
      - # rocks-dist copyroll; umount /mnt/cdrom
      - # cd /home/install; rocks-dist dist
    - **Install only roll-sweetroll-kickstart-*.rpm (nodes, graph)**
      - # make profile; make proof
    - **Install only one xml node**
      - # cp nodes/sweetroll.xml /home/install/rocks-dist/lan/enterprise/4/en/os/x86_64/build/nodes/

When Things Go Wrong

- Test your Kickstart Graph
  - With Sweetroll XML in place:
    - # cd /home/install/sbin
    - # ./kickstart --client=compute-0-0 > ks.cfg
    - # vi ks.cfg

```
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*
    - Apache user vs root user makes ks file in practice

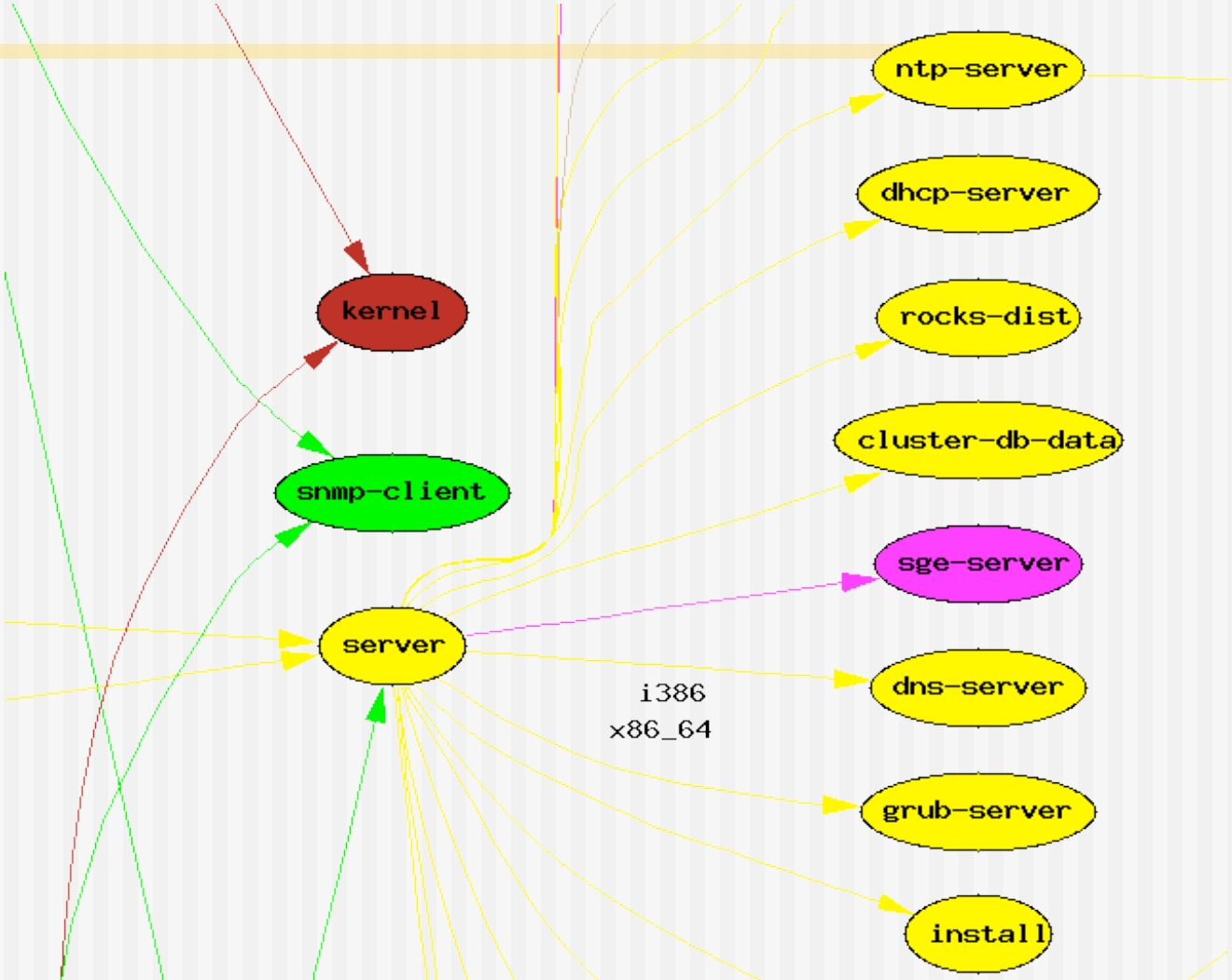
When Things Go Wrong

- 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/enterprise/4/en/os/x86_64/build
    - # kpp sweetroll
    - # kpp sweetroll I kgen

Inspecting your Work

- See the roll graph immediately with graphviz
  - Install the roll locally
    - make roll; ../bin/install-roll.sh
      - mount -o loop sweetroll-*.iso /mnt/cdrom
      - rocks-dist copyroll; umount /mnt/cdrom
    - cd /home/install; rocks-dist dist
  - Look at “kickstart graph” link on cluster homepage
    - http://cluster.org/homepage/dot-graph.php
      - cluster.org is the build host for your roll (a Rocks frontend)
  - Image is dynamically generated, your roll nodes are part of the graph!

Inspecting your Work: Graphviz



RPM Building

RPM Building

- RPM: Redhat Package Management
  - How we learned to build RPMS:
    - http://www.rpm.org/max-rpm/
- Rocks preferred build method: *minimal spec file*
  - Use Makefile for all install logic
  - Make builds file tree in $RPM_BUILD_ROOT
  - RPM Spec includes ALL files in made tree:
    - %files
    - /

RPM Build trick: minimal spec

rocks-sql.spec.in

```
Summary: Cluster SQL Tools
Name: @NAME@
Version: @VERSION@
Release: @RELEASE@
Copyright: @COPYRIGHT@
Vendor: @VENDOR@
Group: System Environment/Base
Source: %{name}-%{version}.tar.gz
Buildroot: @VAR@/tmp/%{name}-buildroot
BuildArchitectures: noarch
Prefix: /opt/rocks

%description
Cluster SQL Tools

%prep
%setup
%build
%install
make ROOT=$RPM_BUILD_ROOT install

%files
/

%post
%clean
/bin/rm -rf $RPM_BUILD_ROOT
```

sql/Makefile

```
PKGROOT   = /opt/rocks
SCRIPTS   = insert-ethers add-extra-nic
PLUGINS = $(wildcard plugins/*py)
PLUGINDIR = $(PKGROOT)/var/plugins/insertethers
REDHAT.ROOT     = $(PWD)/../../
ROCKSROOT       = ../../../../..
-include $(ROCKSROOT)/etc/Rules.mk
include Rules.mk

build: $(SCRIPTS) $(RCFILES)

install:: build
    mkdir -p $(ROOT)/$(PKGROOT)/sbin/
    mkdir -p $(ROOT)/$(PKGROOT)/etc/
    mkdir -p $(ROOT)/$(PLUGINDIR)
    mkdir -p $(ROOT)/etc/cron.daily
    install -ma+rx $(SCRIPTS) \
                $(ROOT)/$(PKGROOT)/sbin/
    install -m0644 $(PLUGINS) \
                $(ROOT)/$(PLUGINDIR)
```

RPM build trick: overlay mounts

- We tell packages to install themselves in a build directory:
  - ./configure --prefix=$RPM_BUILD_ROOT
- Problem: some packages HAVE to be built into their final location
  - MPICH: ./configure --prefix=/opt/mpich
- Trick: mount a tmpfs filesystem on /opt/mpich before build
  - Overlay mount (Ugly, statefull, but effective)
  - Does not confuse build with existing installation.

RPM build trick: overlay mounts

- In Makefile:
  - INSTALL_DIR=%{mpich_dir}/%{interconnect}/%{compiler}
  - mkdir -p $INSTALL_DIR
  - mount -t tmpfs tmpfs $INSTALL_DIR

  - sh ./configure --prefix=$INSTALL_DIR %{mpich_configure}
  - make
  - make install
  - … umount $INSTALL_DIR …

    - Parameterize vars with ‘rpm --define “name val” ‘

RPM build trick: patch-files

- We may want to patch a few files of a "pristine" tarball
  1. Create a patch-files tree for new files that mimics the tarbal's file tree.
     - patch-files/webapps/axis/WEB-INF/lib/jaxp.jar
     - patch-files/webapps/axis/WEB-INF/jwsClasses/Echo…
  2. Just after untarring software, overwrite relevant files with their 'patch-files' counterpart.
     - cd patch-files && find . -type f | grep -v CVS | \
     - cpio -pduv ../$(BASENAME)
  3. Build normally

Roll Development Basics

- Good Luck building and testing your Rolls!

- Coming Up:
  - The Roll Template in CVS
  - More on Roll Testing