Creating and Testing Rolls

Rocks-A-Palooza III

SDSC

What We'll Be Doing

- Discuss methods on how to add new programs to compute nodes
- How to change configuration on compute node
- Build a new roll
- Testing the roll

Add A New Package

Steps to Add a New Package to the Cluster

- All packages are found under '/home/install'
- Put the new package in /home/install/contrib/4.2.1/<arch>/RPMS
  - Where <arch> is 'i386', 'x86_64' or 'ia64'
- "Extend" an XML configuration file
- Rebind the distro:

```
# cd /home/install
# rocks-dist dist
```

- Apply the changes by reinstalling the compute nodes:
  - "shoot-node compute-0-0"

Extend the “Compute” XML Configuration File

- To add the package named “strace”

```
$ cd /home/install/site-profiles/4.2.1/nodes
$ cp skeleton.xml extend-compute.xml
```

- In ‘extend-compute.xml’, change:

```
<!-- <package> insert your 1st package name here and uncomment the line</package> -->
```

- To:

```
<package>strace</package>
```

Extend the “Compute” XML Configuration File

- Rebind the distro
  - This copies ‘extend-compute.xml’ into /home/install/rocks-dist/…/build/nodes

```
# cd /home/install
# rocks-dist dist
```

- Test the changes
  - Generate a test kickstart file

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

  - You should see ‘strace’ under the ‘%packages’ section

Extend the "Compute" XML Configuration File

- When you are satisfied with the changes, reinstall a compute node

```
# shoot-node compute-0-0
```

  - Or:

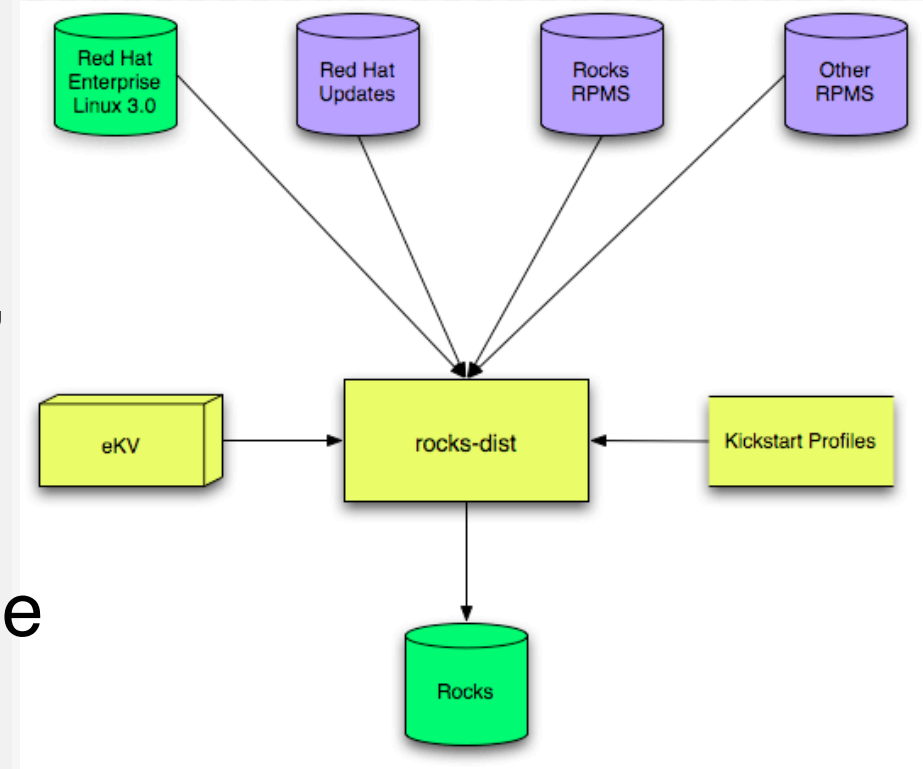
```
# ssh compute-0-0 /boot/kickstart/cluster-kickstart
```

More on the Distro

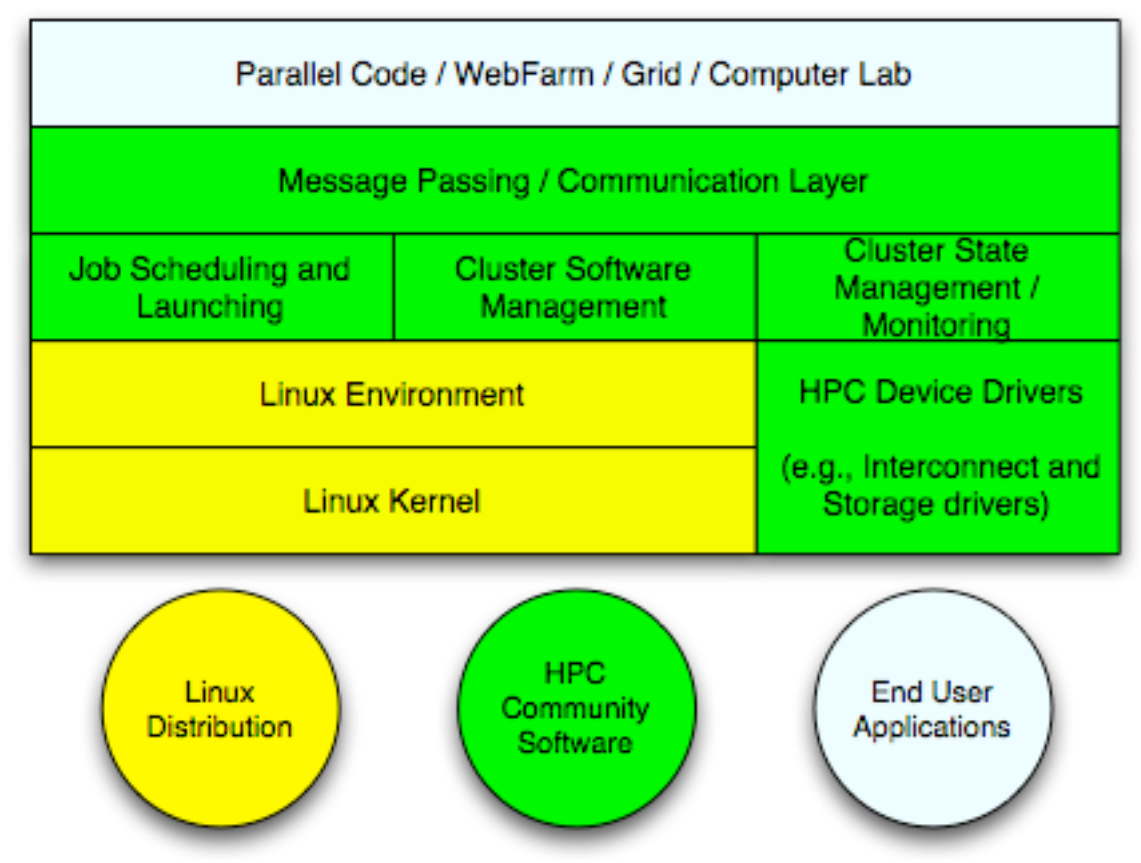
- Rocks-dist looks for packages in:
  - "/home/install/rolls"
    - RedHat and Rocks packages
  - "/home/install/contrib"
    - Pre-built 3rd party packages
  - "/usr/src/redhat/RPMS"
    - RedHat default location for 'built' packages
    - But, when building packages in Rocks source tree, packages are not placed here
      - The packages are placed local to the roll source code

More on the Distro

- Any time you add a package to the distro, you must re-run "rocks-dist dist"
  - Rocks-dist binds all the discovered packages into a RedHat-compliant distribution



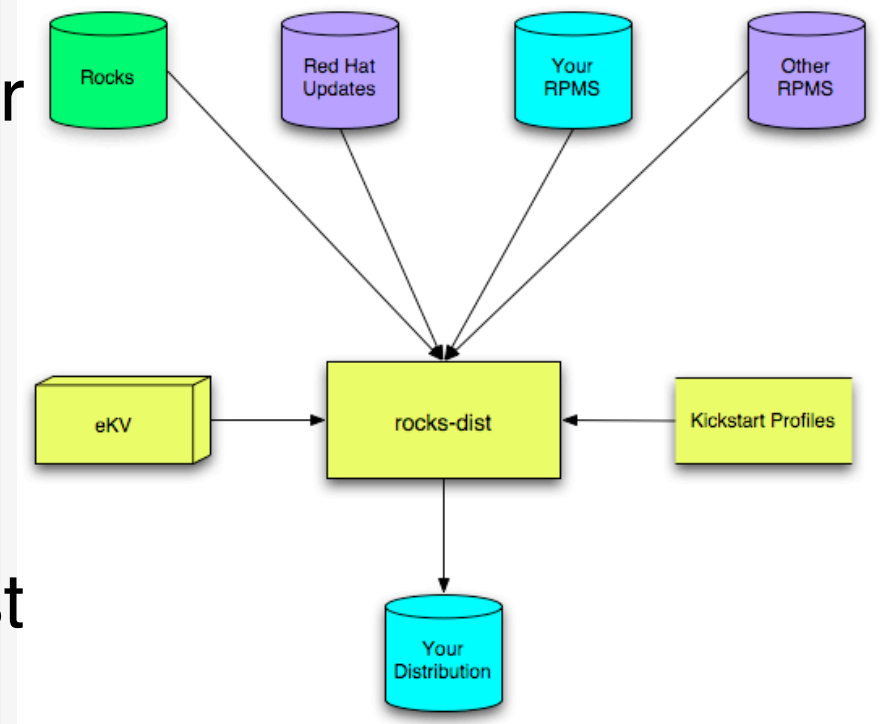
More on the Distro



- Rocks-dist assembles a RedHat compliant distribution

Your Distro - Extending Rocks

- You can use “rocks-dist” to build and distribute your own distribution
  - Merges RPMS
    - When two RPMS have the same basename, rocks-dist selects the one with the newest timestamp
- Final distribution looks just like Rocks
  - And, Rocks looks just like RedHat



Add an Application to Compute Nodes

Default NFS Share

- By default, each node has access to an NFS shared directory named ‘/share/apps’
- The actual location is on the frontend
  - ‘/export/apps’ on the frontend is mounted on all nodes (including the frontend) as ‘/share/apps’
- Simply add directories and files to /export/apps on frontend

Default NFS Share - Example

- On frontend:

```
# cd /export/apps
# touch myapp
```

- On compute node:

```
# ssh compute-0-0
# cd /share/apps
# ls
myapp
```

Default NFS Share Adding ‘bonnie’

- Bonnie is a file system benchmark
- We’ll download the source and build it
  - On frontend:

```
# cd /share/apps
# mkdir -p benchmarks/bonnie++/src
# cd benchmarks/bonnie++/src
# wget http://www.coker.com.au/bonnie++/bonnie++-1.03a.tgz
```

Adding bonnie

- Build and install it:

```
# tar -zxvf bonnie++-1.03a.tgz
# cd bonnie++-1.03a
# ./configure
# make
# make prefix=/share/apps/benchmarks/bonnie++ install
```

- You can now run it on a compute node:

```
# ssh compute-0-0
# mkdir /state/partition1/output_files
# cd /share/apps/benchmarks/bonnie++/sbin/
# ./bonnie++ -u root -s 4096 -n 0 -f -d /state/partition1/output_files
```

Bonnie

- Execute bonnie

```
bonnie++ -u root -s 4096 -n 0 -f -d /state/partition1/output_files
```

- Flags
  - '-u root' - execute as root user
  - '-s 4096' - write a 4 GB file
  - '-n 0' - skip the 'file creation' test
  - '-f' - fast mode, don't do character (one byte) tests
  - '-d /state/partition1/output_files' - put all temporary files in /state/partition1/output_files
    - If argument to -d flag is a directory that is mounted on NFS, then this is a NFS benchmark

Bonnie Output

```
Writing intelligently...done
Rewriting...done
Reading intelligently...done
start 'em...done...done...done...
Version  1.03       ------Sequential Output------ --Sequential Input- --Random-
                    -Per Chr- --Block-- -Rewrite- -Per Chr- --Block-- --Seeks--
Machine        Size K/sec %CP K/sec %CP K/sec %CP K/sec %CP K/sec %CP  /sec %CP
rocks-45.sdsc.ed 4G           36597  15 17056   5           38552   6 156.6   0
rocks-45.sdsc.edu,4G,,,36597,15,17056,5,,,38552,6,156.6,0,,,,,,,,,,,,
```

- Measurements for sequential output/input
- Last line is comma-separated values
  - Can be used import values into analysis program

Creating RPMS

Package bonnie as an RPM

- Go to the Rocks roll development directory

```
# cd /export/site-roll/rocks/src/roll
```

- Side note: this is where the Restore Roll lives

```
# ls
bin  etc  restore  template
```

Create a Benchmark Roll

- Use the ‘template’ roll to populate a skeleton ‘benchmark’ roll

```
# cd /export/site-roll/rocks/src/roll/
# bin/make-roll-dir.py -n benchmark
```

- Create directory for bonnie

```
# cd benchmark/src
# mkdir bonnie++
```

Create a Bonnie RPM

- Get the source

```
# cd bonnie++
# wget http://www.coker.com.au/bonnie++/bonnie++-1.03a.tgz
```

Create a Bonnie RPM

- Create a version.mk file:

vi version.mk

```
NAME    = bonnie++
VERSION = 1.03a
RELEASE = 1
PKGROOT = /opt/$(NAME)
```

Create a Bonnie RPM

- Create a Makefile:

  # vi Makefile


```
REDHAT.ROOT     = $(CURDIR)/../../
ROCKSROOT       = ../../../../..
-include $(ROCKSROOT)/etc/Rules.mk
include Rules.mk

build:
        tar -zxvf $(NAME)-$(VERSION).tgz
        (                                       \
                cd $(NAME)-$(VERSION) ; \
                ./configure ;                   \
                make                            \
        )


install::
        mkdir -p $(ROOT)/$(PKGROOT)
        (                                                       \
                cd $(NAME)-$(VERSION) ;                         \
                make prefix=$(ROOT)/$(PKGROOT) install \
        )

clean::
        rm -f $(NAME).spec.in
```

Create a Bonnie RPM

- Build the RPM

```
# make rpm
```

- You see lots of output
  - The last line shows you where the resulting binary RPM is:

```
Wrote: /state/partition1/site-roll/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm
```

Create a Bonnie RPM

- View the RPM contents

```
# rpm -qlp /state/partition1/site-roll/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm
```

- Which outputs:

```
/
/opt
/opt/benchmark
/opt/benchmark/bonnie++
/opt/benchmark/bonnie++/bin
/opt/benchmark/bonnie++/bin/bon_csv2html
/opt/benchmark/bonnie++/bin/bon_csv2txt
/opt/benchmark/bonnie++/man
/opt/benchmark/bonnie++/man/man1
/opt/benchmark/bonnie++/man/man1/bon_csv2html.1
/opt/benchmark/bonnie++/man/man1/bon_csv2txt.1
/opt/benchmark/bonnie++/man/man8
/opt/benchmark/bonnie++/man/man8/bonnie++.8
/opt/benchmark/bonnie++/man/man8/zcav.8
/opt/benchmark/bonnie++/sbin
/opt/benchmark/bonnie++/sbin/bonnie++
/opt/benchmark/bonnie++/sbin/zcav
```

Copy the bonnie++ RPM so rocks-dist Can Find It

- All packages are found under '/home/install'
- Put bonnie++ RPM package in /home/install/contrib/4.2.1/<arch>/RPMS
  - Where <arch> is 'i386', 'x86_64' or 'ia64'

```
# cd /home/install/contrib/4.2.1/i386/RPMS
# cp /state/partition1/site-roll/rocks/src/roll/benchmark/RPMS/i386/bonnie++-1.03a-1.i386.rpm .
```

Extend the “Compute” XML Configuration File

- To add the package named “bonnie++”

```
$ cd /home/install/site-profiles/4.2.1/nodes
$ vi extend-compute.xml
```

- In ‘extend-compute.xml’, change the section:

```
<package>strace</package>
```

- To:

```
<package>strace</package>
<package>bonnie++</package>
```

Extend the “Compute” XML Configuration File

- Rebind the distro
  - This copies ‘extend-compute.xml’ into /home/install/rocks-dist/…/build/nodes

```
# cd /home/install
# rocks-dist dist
```

- Test the changes
  - Generate a test kickstart file

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

  - You should see ‘bonnie++’ under the ‘%packages’ section

Extend the “Compute” XML Configuration File

- When you are satisfied with the changes, reinstall a compute node

```
# shoot-node compute-0-0
```

  - Or:

```
# ssh compute-0-0 /boot/kickstart/cluster-kickstart
```

- If you are satisfied with the compute node, shoot ‘em all:

```
# tentakel /boot/kickstart/cluster-kickstart
```

Make a Roll

Create a Node XML file

- This file will contain packages and configuration for the roll
  - We'll create a benchmark roll
  - Use 'extend-compute.xml' as a guide

Create a Node XML file

- Node XML files are in:

  /export/site-roll/rocks/src/roll/benchmark/nodes

- 'make-roll-dir.py' created a stub node XML file named 'benchmark.xml'
  - Let's rename benchmark.xml to benchmark-client.xml
  - Then, take contents from extend-compute.xml and put them in benchmark-client.xml

Create a Node XML file

- In benchmark-client.xml, remove:

  <package>benchmark</package>
  <package>roll-benchmark-usersguide</package>

- Add:

  <package>bonnie++</package>

Create a Node XML file

- Also can add package/node configuration scripts in ‘post’ section:

```
<post>

<file name=“/etc/motd” mode=“append”>
Benchmark Roll is installed
</file>

<!-- update permissions -->
chmod -R 755 /opt/bonnie++

</post>
```

Linking Node XML Files into the Graph

Create a Graph XML file

- The graph file describes how the node files in your roll are linked into the node files supplied by the Rocks core
- Can link into any node file from the Rocks core

Create a Graph XML file

- But, there are three common ‘link points’
  - ‘client’
    - When the node XML file applies only to client nodes (compute nodes, tile nodes)
  - ‘server’
    - When the node XML file applies only to the frontend
  - ‘base’
    - When the node XML file applies to all cluster nodes

Create a Graph XML file

- We'll link the 'benchmark-client.xml' node file to the 'client.xml' node file from the Rocks core

- The roll's graph file is named 'benchmark.xml' and is in:

/export/site-roll/rocks/src/roll/benchmark/graphs/default

Create a Graph XML file

- Edit ‘benchmark.xml
- Link benchmark-client.xml and client.xml:

```
<!-- add edges here -->

<edge from="client">
    <to>benchmark-client</to>
</edge>
```

Build the Roll

```
# cd /export/site-roll/rocks/src/roll/benchmark
# make roll
```

- We see lots of output
- All built RPMS are under ‘RPMS’:

```
# find RPMS -type f
RPMS/i386/bonnie++-1.03a-1.i386.rpm
RPMS/noarch/roll-benchmark-kickstart-4.2.1-0.noarch.rpm
RPMS/noarch/roll-benchmark-usersguide-4.2.1-0.noarch.rpm
```

Build the Roll

- Node and graph XML files are packaged in ‘roll-benchmark-kickstart-4.2.1-0.noarch.rpm’

```
# rpm -qlp RPMS/noarch/roll-benchmark-kickstart-4.2.1-0.noarch.rpm
/
/export
/export/profiles
/export/profiles/graphs
/export/profiles/graphs/default
/export/profiles/graphs/default/benchmark.xml
/export/profiles/nodes
/export/profiles/nodes/benchmark-client.xml
/export/profiles/roll-benchmark.xml
```

Build the Roll

- When the roll is installed on the frontend, ‘rocks-dist’ extracts the node and graph files from ‘roll-benchmark-kickstart-4.2.1-0.noarch.rpm’

Install the Roll

- Use ‘rocks-dist’ to install the roll:

```
# mount -o loop benchmark-4.2.1-0.i386.disk1.iso /mnt/cdrom

# rocks-dist --install copyroll
Copying roll from media (directory "/mnt/cdrom") into mirror
Copying "benchmark" (4.2.1,i386) roll...
219 blocks

# umount /mnt/cdrom
```

Install the Roll

- The '--install' flag to 'rocks-dist' enables the roll:

```
# dbreport rolls
base 4.2.1 i386 enabled
ganglia 4.2.1 i386 enabled
hpc 4.2.1 i386 enabled
kernel 4.2.1 i386 enabled
os 4.2.1 i386 enabled
service-pack 4.2.1.2 i386 enabled
sge 4.2.1 i386 enabled
web-server 4.2.1 i386 enabled
benchmark 4.2.1 i386 enabled
```

Install the Roll

- When building distributions with ‘rocks-dist’, it will ignore ‘disabled’ rolls
  - That is, the node and graph files from roll-<rollname>-kickstart*rpm will not be included in the distribution

Rebuild the Distribution

```
# cd /home/install
# rm -rf rocks-dist ; rocks-dist dist
    .
    .
     including "kernel" (4.2.1,i386) roll...
     including "sge" (4.2.1,i386) roll...
     including "benchmark" (4.2.1,i386) roll...
     including "service-pack" (4.2.1.2,i386) roll...
    .
    .
```

Generate a Test Kickstart File

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

- Make sure the packages and configuration from the benchmark roll are present:

```
# grep bonnie /tmp/ks.cfg
bonnie++
chmod -R 755 /opt/bonnie++
```

Full Test - Reinstall a Node

```
# shoot-node compute-0-0
```

Ordering Roll Node File Relative to Other Node Files

Ordering Node XML files

- Used when a node file must come before/after another node file
- Simple example
  - One node file writes a file
  - Another node appends to the file

Ordering Node XML files

- At the top of each ‘processed’ kickstart file, there is the list of nodes and the order in which they are included
- To create a ‘processed’ kickstart file:

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


```
#
# Kickstart Generator version 4.2.1
#
#
# Node Traversal Order
#
# ./nodes/replace-installclass.xml (service-pack)
# ./nodes/installclass-client.xml (base)
# ./nodes/python-development.xml (base)
# ./nodes/base.xml (base)
    .
    .
    .
# ./nodes/routes-client.xml (base)
# ./nodes/ganglia-client.xml (ganglia)
# ./nodes/benchmark-client.xml (benchmark)
# ./nodes/x11.xml (base)
# ./nodes/pxeboot.xml (base)
# ./nodes/postshell.xml (base)
# ./nodes/ethers-server-postshell.xml (base)
# ./nodes/partition-functions.xml (base)
```

Ordering a Node File After Another

- In graph XML file in your roll, specify an '<order>' paragraph:

```
# cd /export/site-roll/rocks/src/roll/benchmark/graphs/default
# vi benchmark.xml
```

```
<order head="partition-functions">
        <tail>benchmark-client</tail>
</order>

<!-- add edges here -->

<edge from="client">
        <to>benchmark-client</to>
</edge>
```

Rebuild and Reinstall the Roll

- ‘make roll’
- ‘rocks-dist copyroll’
- ‘rocks-dist dist’
- Look at the processed kickstart file


```
#
# Kickstart Generator version 4.2.1
#
#
# Node Traversal Order
#
# ./nodes/replace-installclass.xml (service-pack)
# ./nodes/installclass-client.xml (base)
# ./nodes/python-development.xml (base)
# ./nodes/base.xml (base)
# ./nodes/apache.xml (base)

    .
    .
    .

# ./nodes/x11.xml (base)
# ./nodes/pxeboot.xml (base)
# ./nodes/postshell.xml (base)
# ./nodes/ethers-server-postshell.xml (base)
# ./nodes/partition-functions.xml (base)
# ./nodes/benchmark-client.xml (benchmark)
```

Ordering a Node File Before Another

- In graph XML file in your roll, specify an ‘<order>’ paragraph:

```
# cd /export/site-roll/rocks/src/roll/benchmark/graphs/default
# vi benchmark.xml
```

```
<order head=”benchmark-client">
        <tail>base</tail>
</order>

<!-- add edges here -->

<edge from="client">
        <to>benchmark-client</to>
</edge>
```


```
#
# Kickstart Generator version 4.2.1
#
#
# Node Traversal Order
#
# ./nodes/replace-installclass.xml (service-pack)
# ./nodes/installclass-client.xml (base)
# ./nodes/python-development.xml (base)
# ./nodes/benchmark-client.xml (benchmark)
# ./nodes/base.xml (base)
# ./nodes/apache.xml (base)

    .
    .
    .
```

Rolls for the Frontend

Attaching Node XML Files to the Frontend Graph

- For node XML files that should just be applied to the frontend, use the ‘server’ link point
  - Similar to previous section where attached a node file to ‘client’ link point

```
<!-- add edges here -->

<edge from=”server">
        <to>benchmark-server</to>
</edge>
```

- Use ‘base’ link point for common node XML files (e.g., node XML files that should be applied to all nodes in the cluster)

Install the Roll On-The-Fly on the Frontend

- Build your roll
  - ‘make roll’
- Put it in the distro:
  - ‘rocks-dist copyroll’
  - ‘rocks-dist dist’
- To install it on the frontend:
  - ‘kroll rollname > /tmp/install-rollname.sh
  - ‘sh /tmp/install-rollname.sh’

Install the Roll On-The-Fly on the Frontend

- Warning: Not all rolls can be installed on the fly
  - Examples:
    - Grid Roll
    - Web Server Roll

Using a Central Server to Test Your Roll

- Rocks frontend machines can retrieve their Rolls over the network
  - Only the Kernel/Boot Roll required
    - Need to boot the machine into installation mode
- We call this a ‘central’ install
  - The frontend gets the Rolls from a central server

Using a Central Server to Test Your Roll

- All Rocks frontends are central servers
  - Just need to open up ‘http’ access in iptables

```
# Uncomment the lines below to activate web access to the cluster.
#-A INPUT -m state --state NEW -p tcp --dport https -j ACCEPT
#-A INPUT -m state --state NEW -p tcp --dport www -j ACCEPT
```

- Restart iptables

```
# service iptables restart
```

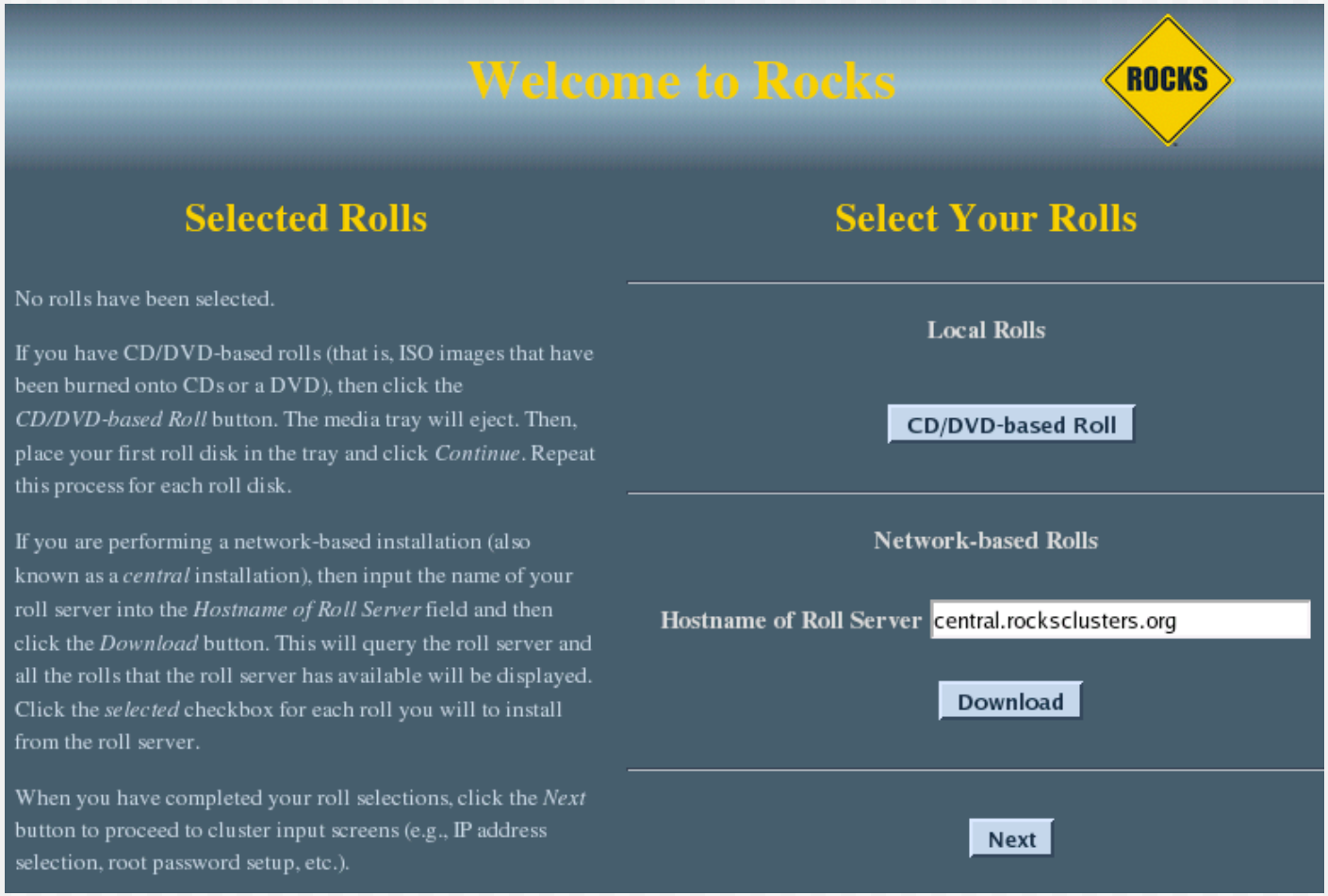
Using a Central Server to Test Your Roll

- Use ‘rocks-dist copyroll’ to copy the roll under test onto your central server
  - mount -o loop roll*iso /mnt/cdrom
  - rocks-dist copyroll
    - No need to throw ‘--install’ flag
    - Only serving the roll bits, not installing the roll on the central

Using a Central Server to Test Your Roll

- Boot the frontend test machine with the Kernel/Boot Roll
- At first configuration screen, input name of your central server in the ‘Hostname of Roll Server’ field and click ‘Download’ button

Using a Central Server to Test Your Roll



Final Test - Test with CD

- Burn ISO onto CD and install frontend in ‘traditional’ method

Other Tricks

Other Tricks

- Use tags ‘<[[CDATA’ and ‘]]>’ when your node XML code contains several XML ‘escape’ characters
  - For example: &, <, >
- Any code within <[[CDATA and]]> is ignored by the XML parser

Futures

Future Features

- Rolls for Solaris
- Pre-packaged appliances
  - When you select ‘Compute Cluster’, all the rolls to build a compute cluster will be enabled
    - Base, Kernel, OS, HPC, Web Server
- Lights-out cluster reinstallation
  - One command will reinstall the frontend with zero user interaction
    - Then the frontend will reinstall the compute nodes
    - Utilizes the Restore Roll