



# Table of Contents

<b>1 Add Compute Nodes</b>	<b>1</b>
1.1 Compute Node	1
1.1.1 Overview	1
1.1.2 Tools for Installing Compute Nodes	1
1.1.3 Installing Compute Nodes	1
1.2 Kits	1
<b>2 AdminGuide Adding Custom Kernels &amp; initrds</b>	<b>2</b>
2.1 Adding Custom Kernels & initrds	2
<b>3 AdminGuide Adding Hosts to the Cluster</b>	<b>3</b>
3.1 Add a Node	3
3.1.1 Add an Unmanaged Node	3
3.1.2 Update a Node	3
3.1.3 Remove a Node	3
3.1.4 List Nodes	3
3.1.5 Replace an Existing Host	3
3.1.6 Pre-populate a Compute Node	3
3.1.7 Addhost Plugins	3
<b>4 AdminGuide Adding a Network Filesystem to the Cluster</b>	<b>5</b>
4.1 Introduction	5
4.2 Configuration Steps	5
4.2.1 Add the NAS box to the Cluster	5
4.2.2 Creating the fstab.append entry	5
<b>5 AdminGuide Cluster monitoring</b>	<b>6</b>
5.1 Introduction	6
5.1.1 Ganglia	6
5.1.2 Cacti	6
5.1.3 Nagios	6
<b>6 AdminGuide Configuring BMC on Nodes</b>	<b>7</b>
6.1 Initial Configuration	7
6.2 After LAN Access Is Enabled	7
6.3 Post Configuration	7
<b>7 AdminGuide Configuring IPMI to Nodes</b>	<b>8</b>
7.1 Overview	8
<b>8 AdminGuide Creating your own Kits</b>	<b>9</b>
8.1 Overview	9
8.2 An Example	9
8.3 Building OS Kits	10
8.3.1 Overview	10
8.3.2 OS Kits	10
8.3.3 Supporting New OSs	10
8.4 Building Feature Kits	10
8.4.1 Overview	10
8.4.2 The Automated Way	11
<b>9 AdminGuide DNS Configuration</b>	<b>13</b>
9.1 DNS in UniCluster 4.0	13
9.2 General Configuration Philosophy	13
9.3 Configuration Implementation	13
9.4 Integrating with an External DNS Server	13
<b>10 AdminGuide Edit Node Groups</b>	<b>14</b>
10.1 Installer Node Group	14
10.1.1 Packaged Node Group	14
10.1.2 Imaged Node Group	14
10.1.3 Diskless Node Group	14
<b>11 AdminGuide File and Package Synchronization in the Cluster</b>	<b>15</b>
11.1 Overview	15
11.2 Package Synchronization	15
11.3 File Synchronization	15
11.4 Usage	15

# Table of Contents

<b>12 AdminGuide Kits</b>	<b>17</b>
12.1 What is a Kit?	17
12.2 What is a Component?	17
12.3 Types Of Kits	17
12.4 Working With Kits	17
12.4.1 Installing a UniCluster Kit	17
12.4.2 Installing a Non-UniCluster Kit	18
12.4.3 Installing Additional OS Kits	18
12.4.4 Removing Kits	20
<b>13 AdminGuide Modifying Node pxeboot Parameters</b>	<b>21</b>
13.1 Introduction	21
13.2 Temporarily Changing Configuration	21
13.3 Permanent Configuration Change	21
<b>14 AdminGuide Network Editor (netedit)</b>	<b>22</b>
14.1 Introduction	22
14.2 Restrictions	22
14.3 Usage and Examples	22
<b>15 AdminGuide Networking</b>	<b>23</b>
<b>16 AdminGuide Node Group Editor</b>	<b>24</b>
16.1 Overview	24
16.2 What it does	24
16.3 Usage	24
<b>17 AdminGuide Package, Imaged, and Diskless</b>	<b>25</b>
17.1 Overview	25
17.2 Diskless and Imaged Installation Method	25
17.2.1 Image Creation	25
17.2.2 Image Customization	25
17.2.3 Installation	25
17.2.4 Co-Existing With Other Operating Systems	25
17.2.5 Additional OS Support	26
17.3 Packaged Installation Method	26
17.3.1 Server Configuration	26
17.3.2 The Nodeinstaller Script	26
<b>18 AdminGuide Running Commands Across the Cluster (parallel shell)</b>	<b>27</b>
18.1 Introduction	27
18.2 Use and Examples	27
<b>19 AdminGuide Troubleshooting</b>	<b>28</b>
19.1 Troubleshooting	28
<b>20 AdminGuide Updating the Cluster</b>	<b>29</b>
20.1 Overview	29
20.2 Accessing Updates	29
20.2.1 CentOS	29
20.2.2 RedHat	29
20.2.3 Oracle Linux	29
20.3 Downloading Updates	29
20.4 Deploying Updates	29
<b>21 AdminGuide User Account Management</b>	<b>31</b>
21.1 Introduction	31
21.2 User Management	31
21.2.1 Adding a User	31
21.2.2 Removing Users	31
21.3 LDAP, NIS, and Winbind	32
21.3.1 Generic Standalone Strategy	32
21.3.2 Generic Integration Strategy	32
<b>22 AdminGuide Yum Repositories</b>	<b>33</b>
22.1 Repository Types	33
22.2 OS updates with repopatch	33

# Table of Contents

<b>23 AdminGuide nghosts.....</b>	<b>35</b>
23.1 The nghosts Utility.....	35
23.2 Usage.....	35
23.3 TUI Screens.....	35
23.4 Node Migration Workflow.....	35
23.5 Listing of Node Groups.....	35
<b>24 Administrator Guide.....</b>	<b>36</b>
24.1 Chapters.....	36
<b>25 Concepts and Terminology.....</b>	<b>37</b>
<b>26 Getting Started.....</b>	<b>39</b>
26.1 Installer Node Requirements.....	39
26.1.1 Recommendations.....	39
26.2 UniCluster Installation Instructions.....	39
<b>27 UniCluster Documentation.....</b>	<b>40</b>
27.1 Documents.....	40
<b>28 UniCluster Express 4.0.0.....</b>	<b>41</b>

# 1 Add Compute Nodes

## 1.1 Compute Node

### 1.1.1 Overview

UniCluster 4.0 does the majority of the work involved in bringing up new compute nodes. There are two main ways to install a new compute node. The first way is to have the hardware MAC address of the machines listed in a file and using the `-f` option of the "addhost" command. This will prepare the installer node to provision the machines with those MAC addresses when it sees them boot up. The second way is to run "addhost" with no options. This will bring up an interface that will sit and wait for new machines to be powered on and when they are, UniCluster will automatically find them, add their MAC address to its records and provision the machine according to the nodegroup you select.

### 1.1.2 Tools for Installing Compute Nodes

#### 1.1.2.1 addhost

The *addhost* tool is used to add nodes to, replace nodes in, remove nodes from, and update nodes in the cluster. The tool supports these four major tasks in both a non-interactive mode as well as node addition through an auto-discovery menu-driven interactive mode.

#### 1.1.2.2 ngedit

The *ngedit* command manipulates node groups. It allows the user to select the operating system, software components and all details of the node setup.

Run without any options, the *ngedit* command presents an interactive interface that allows for changing, copying and deleting nodegroups. The tool is intuitive to use and takes care of all of the operations necessary to facilitate that requested changes.

### 1.1.3 Installing Compute Nodes

For completeness, the steps for installing compute nodes are:

1. Run "addhost" command on installer node.
2. Power on compute nodes.

That's it, everything else is automatic.

## 1.2 Kits

The general workflow for installing a new kit located on the network is:

1. On the installer node, run the install for the new kit:  
`yum install <kit name>` (e.g. `unicluster-kit-xen`)  
This will install a kit installation script in `/opt/tortuga/sbin/install-kit-<kit name>`.
2. Execute the aforementioned installation install script:  
`/opt/tortuga/sbin/install-kit-<kit name>`

At this point you are ready to add the new kit to a node group. Use *ngedit* to verify that the new kit is available.

## 2 AdminGuide Adding Custom Kernels & initrds

### 2.1 Adding Custom Kernels & initrds

If a non-standard (not included on the OS distribution media) kernel or initrd is required for initial provisioning custom kernels and initrd's can be added to the system easily. To make a custom kernels or initrd available to compute nodes simply place the desired file in the "/tftpboot/tortuga" directory. Once the files are in "/tftpboot/tortuga" they will be visible in ngedit and can be associated with the appropriate nodegroups.

[ UniCluster Documentation / Administrator Guide ]

## 3 AdminGuide Adding Hosts to the Cluster

Now that you have downloaded and installed the UniCluster software onto your Installer Node (see Getting Started), and prepared your Node Groups (see AdminGuide Edit Node Groups), you are ready to add Compute Nodes into your cluster.

The simplest way to add nodes to a node group is to run the `addhost` command on the Installer Node, select the Node Group you want to add the new nodes to, then power on the new nodes. The `addhost` command watches for incoming DHCP requests and begins the process of provisioning and configuring the new nodes as they come on-line.

### 3.1 Add a Node

When run without any arguments, or run without enough arguments, `addhost` enters an interactive mode. Hence, it will query the user for the necessary information to add a new host to the cluster. The following information is gathered:

1. Node Group Selection : The Node Group to add detected hosts to
2. Network Interface Selection : Select the interface to listen for hosts on
3. Rack Number : If used by the node group, the rack the new hosts are on

At this point Auto-Discovery begins and detected hosts will be added to the cluster.

#### 3.1.1 Add an Unmanaged Node

An unmanaged node is not part of a node group. Rather, it is simply a machine that the cluster should know about.

```
addhost --static=<static hostname> --ip-address=<IP address>
```

#### 3.1.2 Update a Node

Update an existing host:

```
addhost -u <host name>
```

#### 3.1.3 Remove a Node

```
addhost -e <host name>
```

#### 3.1.4 List Nodes

```
nghosts --list
```

#### 3.1.5 Replace an Existing Host

The replace process requires the graphical new-host selection through auto-discovery.

```
addhost -p <host name>
```

#### 3.1.6 Pre-populate a Compute Node

```
echo 00-01-00-00-01-01 > <MAC address file>
```

```
echo 00-01-00-00-00-02 >> <MAC address file>
```

```
addhost --nodegroup=<node group name> --rack=<rack number> \  
--node-interface=<node interface> --file=<MAC address file>
```

#### 3.1.7 Addhost Plugins

Since the actual `addhost` script is pretty much an information gatherer and "plugin" executer most of the actual configuration operations performed as the result of running `addhost` are defined in an `addhost` plugin. These plugins are little configuration scripts designed to configure one component resulting from the addition or removal of a host from the cluster. The `addhost` plugins are stored at `/opt/tortuga/lib/plugins/addhost`.

#### Default Addhost Plugins

Plugin	Description
00-boothost.py	Update the PXE configuration for the specified hosts using boothost.
01-hosts.py	Run the genconfig hosts plugin.
02-dnsreverse.py	Configure the reverse DNS lookups for named.
03-dnszone.py	genconfig zone plugin.
04-dhcp.py	Run the genconfig dhcpd plugin.
05-hostspdsh.py	Run the genconfig hostspdsh plugin.
06-hostsequiv.py	Run the genconfig hostsequiv plugin.
07-ssh.py	Run the genconfig ssh plugin.
09-autofs.py	Run the S91TortugaAutomountCFM.rc.py plugin.

An addhost plugin is written by implementing the one or more of the methods in the *AddHostPlugin* class. These methods are:

- added
- updated
- replaced
- removed
- finished

When a node is added to the cluster, the "added" function is called. Similarly, when a node is removed from the cluster the "removed" function is called. The "finished" functions are called after the "add" methods have been called for all of the *addhost* plugins. This allows a plugin to have access to a fully updated configuration database. The plugins are always executed in the order as returned by the Python `sort()` method called against their file names.

### 3.1.7.1 Sample Plugin

```
import os
from tortuga.addhost import *

class AddHostPlugin(AddHostPluginBase):
    def updated(self):
        os.system("echo Updated config")

    def finished(self, nodelist, prePopulateMode):
        os.system("echo finished config")
```

[ UniCluster Documentation / Administrator Guide ]

# 4 AdminGuide Adding a Network Filesystem to the Cluster

## 4.1 Introduction

It may be desirable to add a dedicated unmanaged NAS device to a UniCluster 4.0 cluster. UniCluster 4.0 provides a simply way to add Linux mountable drives hosted on the NAS to nodes in the cluster.

## 4.2 Configuration Steps

### 4.2.1 Add the NAS box to the Cluster

First the NAS box should be added to the cluster. The `addhost -s` (device hostname) and `-x` (device IP address) options will be used to add the node so that it will be placed in the hosts file and the named configuration.

```
[root@oracle-installer pxelinux.cfg]# addhost -x 172.17.0.10 -s NAS-Box-1
Reloading named: [ OK ]
Shutting down dhcpd: [ OK ]
Starting dhcpd: [ OK ]
Setting up automount maps for cfm: [ OK ]
Running plugin: /opt/tortuga/lib/plugins/cfmsync/getent-data.sh
Updating installer(s)
Done
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
```

### 4.2.2 Creating the `fstab.append` entry

UniCluster 4.0 supports the addition of managed mounts to nodes through the `fstab.append` file. This file exists for all nodegroups and is appended to the `/etc/fstab` file on each host. By default all non-primary installers will have a single entry for `/home`:

```
[root@oracle-installer pxelinux.cfg]# cat /etc/cfm/compute-oracle/etc/fstab.append
# Appended by CFM
# Entries below this come from the CFM's fstab.append
oracle-installer.cluster.univa.com:/home /home nfs defaults 0 0
```

Additional entries can be added...such as a NFS share for our NAS box...

```
[root@oracle-installer pxelinux.cfg]# cat /etc/cfm/compute-oracle/etc/fstab.append
# Appended by CFM
# Entries below this come from the CFM's fstab.append
oracle-installer.cluster.univa.com:/home /home nfs defaults 0 0
NAS-Box-1:/nfs-share /nfs-share nfs defaults 0 0
```

A new dummy placeholder file is needed so that the new mount directory gets created on each of the compute nodes and then sync the new `fstab.append`.

```
[root@oracle-installer ~]# mkdir -p /etc/cfm/compute-oracle/nfs-share
[root@oracle-installer ~]# touch /etc/cfm/compute-oracle/nfs-share/placeholder
[root@oracle-installer ~]# cfmsync -f
```

You can check the mount out on a compute node (if you have provisioned one) in a few seconds...

```
[root@compute-00-00 ~]# cat /etc/mtab
/dev/sda3 / ext3 rw 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
/dev/sda5 /var ext3 rw 0 0
/dev/sda6 /data ext3 rw 0 0
/dev/sda1 /boot ext3 rw 0 0
tmpfs /dev/shm tmpfs rw 0 0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
sunrpc /var/lib/nfs/rpc_pipefs rpc_pipefs rw 0 0
oracle-installer.cluster.univa.com:/home /home nfs rw,addr=172.19.0.1 0 0
NAS-Box-1:/nfs-share /nfs-share nfs rw,addr=172.17.0.10 0 0
```

And thats it...the `NAS-Box-1:/nfs-share` resource is now shared to all compute nodes inside the cluster.

# 5 AdminGuide Cluster monitoring

## 5.1 Introduction

UniCluster 4.0 provides 3 additional kits Cacti, Nagios, and Ganglia that provide monitoring functionality. Both Cacti and Ganglia target general node metric information that is useful for determining cluster utilization and performance while Nagios provides metrics that help diagnose cluster health.

### 5.1.1 Ganglia

To view the status of a cluster with Ganglia, the Ganglia kit must be installed and deployed within the cluster (see this article for details on deploying kits). Once it has been deployed, pointing a web browser at the node with the gmetad component installed on it (likely the installer node). For example, if the component is installed on "localhost", the URL for Ganglia will be <http://localhost/ganglia/>.

Also, if the gmetad component is installed on the Installer node there will be a link from the UniCluster homepage (<http://localhost>) on the Installer node directly to the Ganglia web interface.

### 5.1.2 Cacti

To view the status of a cluster with Cacti, the Cacti kit must be installed and deployed within the cluster (see this article for details on deploying kits). Once it has been deployed, pointing a web browser at the node with the cacti component installed on it (likely the installer node). For example, if the component is installed on "localhost", the URL for Cacti will be <http://localhost/cacti/>. The default username for cacti is "admin" and the default password is also "admin".

Also, if the cacti component is installed on the Installer node there will be a link from the UniCluster homepage (<http://localhost>) on the Installer node directly to the Cacti web interface.

### 5.1.3 Nagios

To view the status of a cluster with Nagios the Nagios kit must be installed and deployed within the cluster (see this article for details on deploying kits). Once it has been deployed, pointing a web browser at the node with the nagios installer component installed on it (likely the installer node). For example, if the component is installed on "localhost", the URL for Nagios will be <http://localhost/nagios/>. The default username for nagios is "admin" and the default password is also "admin". (Troubleshooting note: If the nagios kit is installed after all nodes have been added to the cluster running "addhost -u" is necessary in order to generate the nagios cluster configuration.)

Also, if the nagios component is installed on the Installer node there will be a link from the UniCluster homepage (<http://localhost>) on the Installer node directly to the Nagios web interface.

[ UniCluster Documentation / Administrator Guide ]

# 6 AdminGuide Configuring BMC on Nodes

## 6.1 Initial Configuration

In order for a node to talk to its BMC, a kernel driver must be loaded. The freeipmi tools support several types of interfaces to kernel drivers. However it was determined that the OPENIPMI freeipmi driver that utilizes OpenIPMI kernel modules was the most reliable and easiest to use. In order to get the OpenIPMI service installed on machines in a node group you must use ngedit to enable the component-ipmi-service that is contained within the base kit. Once this kit is installed a command such as:

```
bmc-tool -a 192.168.31.53 -s 255.255.255.0 -g 192.168.31.1
```

would set the BMC IP address, netmask, and gateway on the machine. If more information needs to be set (such as user accounts / passwords and Serial Over LAN (SOL) ) you can checkout the current BMC configuration, edit the configuration file and then check it back in. The checked out configuration file will be a bmc-config.conf file. To get the current configuration of your BMC you would run:

```
bmc-tool -c
```

or

```
bmc-tool -c > my-bmc-config.conf
```

to save it in a file. After you edit the file to your liking this bmc-config.conf file can be replicated to all of your nodes and then used as a template for configuring their BMC's. This way you could get all of your users added to all of your nodes while still being able to set your network information uniquely. The first example in this section would then become:

```
bmc-tool -a 192.168.31.53 -s 255.255.255.0 -g 192.168.31.1 -f my-bmc-config.conf
```

For more information on what you can set in bmc-conf please read the man page.

## 6.2 After LAN Access Is Enabled

Once LAN access is enabled, all bmc-tool functions can operate over the LAN. Simply passing in the target BMC IP address or hostname and an administrators username will allow you to configure the remote BMC. If you don't enter a password with the -p option it will be prompted. For example:

```
bmc-tool -n 192.168.31.53 -u administrator -c
```

would fetch the BMC configuraiton from the BMC at 192.168.31.53. And if you wanted to avoid the prompt:

```
bmc-tool -n 192.168.31.53 -u administrator -p <secretpw> -c
```

would fetch the configuration without the password prompt.

## 6.3 Post Configuration

Once LAN access has been configured the various freeipmi and OpenIPMI tools can be used to control and monitor the actual BMC's. For example, fetching the power with the freeipmi tools:

```
[root]# ipmipower -h 192.168.31.53 -u admin -p password --stat  
192.168.31.53: on
```

and with the OpenIPMI tools

```
[root]# ipmitool -H 192.168.31.53 -U admin -P password power status  
Chassis Power is on
```

[Back to Administrator Guide](#)

## 7 AdminGuide Configuring IPMI to Nodes

### 7.1 Overview

Once the BMC is configured on a compute node you can access IPMI via the freeipmi tools that are included in UniCluster 4.0. This is discussed in the BMC article.

[ UniCluster Documentation / Administrator Guide ]

# 8 AdminGuide Creating your own Kits

## Creating your own UniCluster Kits

### 8.1 Overview

Eventually you may want add new software to your cluster that isn't provided with UniCluster 4.0. If the new software is more complicated than simply installing and RPM on a whole bunch of nodes, creating a new kit may be the best way to cluster enable your software. The Admin Guide Kits Article provides an overview of kits and should probably be read as a base reference if you are unfamiliar with the kit concept.

### 8.2 An Example

Kits are stored on installation media as a set of files inside a single directory. The name of the directory typically mirrors the name of the Kit. For example, on the UniCluster 4.0.0 installation ISO there is a "base" directory that contains the "base" kit. Inside the directory there is a single RPM that begins with "kit-". This RPM holds the Kit meta-data as well as the plugin/configuration agent data. There are also RPM's that begin with "component-". These RPM's are the "Components" provided by this Kit. The other RPM's contain the new cluster software provided by the Kit.

The Kit metadata is stored within the **kitinfo** file which is provided by the kit-\*.rpm for the given kit. Here is the base kitinfo as an example:

```
kit = {'arch': 'noarch',
      'dependencies': [],
      'license': 'GPL',
      'name': 'base',
      'description': 'Base Kit',
      'pkgname': 'kit-base',
      'removable': False,
      'release': '0',
      'scripts': [],
      'version': '0.1'}
components = [{'arch': 'noarch',
              'comprelease': '0',
              'compversion': '0.1',
              'name': 'base-installer',
              'description': 'Component for Tortuga Installer Base',
              'ngtypes': ['installer'],
              'ostype': ,
              'osversion': ,
              'pkgname': 'component-base-installer'},
             {'arch': 'noarch',
              'comprelease': '0',
              'compversion': '0.1',
              'name': 'base-node',
              'description': 'Component for Tortuga Node Base',
              'ngtypes': ['installer', 'compute'],
              'ostype': ,
              'osversion': ,
              'pkgname': 'component-base-node'}]
```

This **kitinfo** file says that the "Base Kit" contains 2 components *base-installer - Component for Tortuga Installer Base* and *base-node - Component for Tortuga Node Base*. In the "base" directory you can see *component-base-installer-\*.noarch.rpm* and *component-base-node-\*.noarch.rpm* which are the RPM's that contain the components of defined in the **kitinfo** file listed above. If we dump the dependencies of *component-base-node-\*.noarch.rpm* we can see the actual RPM's that would get installed if this kit was installed. Here is an example of *rpm -qpR component-base-node-\*.noarch.rpm*:

```
[root@master base]# rpm -qpR component-base-node-*.noarch.rpm
tortuga-base-node >= 1.0
python
pdsh
pdsh-rcmd-exec
pdsh-rcmd-rsh
pdsh-rcmd-ssh
modules
tortuga-core
tortuga-path
tortuga-util
xinetd
rsh-server
rsh
/bin/sh
/bin/sh
/bin/sh
/bin/sh
rpmlib(PayloadFilesHavePrefix) <= 4.0-1
rpmlib(CompressedFileNames) <= 3.0.4-1
```

Notice how some of the RPM's are in the "base" directory, **tortuga-core**, while others are OS libraries, **xinetd**. The **tortuga-core** package and functionality is provided by the base Kit and component-base-node Component while xinetd is just a dependency.

## 8.3 Building OS Kits

### 8.3.1 Overview

There are two ways to create an OS kit. The first is done automatically when Tortuga is installed on the installation node. During the install process, the OS distribution is required. This distribution is used to both install the operating system on the installation node and also to automatically create an OS kit of that operating system. The second method is more manual. Once the installer node is up and running, subsequent OS kits can be created in order to allow for multiple operating systems to be installed from the installer node.

### 8.3.2 OS Kits

In either the automatic or manual creation of an OS kit the ".discinfo" file on the installation media provides the meta-data about the OS. This is used to identify the distribution which is then used to determine the location of all the necessary files. The main component of creating the OS kit is copying the distribution RPMs to the installer repository. You will notice this phase as it takes several minutes to complete. The other important bit of work that is done is taking the initrd image and the kernel image from the distribution media and putting it into the TFTP directory. This is done so that when a new node is booting up it can boot from the appropriate images. The nodes DHCP PXE configuration will be setup to point to the correct kernel and initrd versions for their intended operating system.

An additional option available for installing OS kits on hardware which requires drivers that are not packaged with a distribution is the "driverpatch" utility.

### 8.3.3 Supporting New OSs

As stated earlier, OS kits are automatically created from the installation media. If you want to add support for additional OS's or different versions of OS's some internal libraries must be updated. The `distro.py` file contains a set of classes for each OS that Tortuga supports and can build an image out of. To add a new OS a new `DistroInstallSrcBase` based OS class must be created and added to this file. This new class must then be added to the `DistroFactory` so the new OS source media can be detected. Currently the sheer presence of the necessary source files for a given OS on the `distro` media is enough to match an OS class with the media. This should probably be changed to use the `getVersion` method as well to make sure the correct class is used.

## 8.4 Building Feature Kits

### 8.4.1 Overview

Feature Kits are built using the aptly named tool, `buildkit`. Typically the sources for a Kit exist in a single directory. This directory contains a "build.kit" file that describes the Components and packages within a Kit as well as the dependencies of the given Components. Here is an example of a **build.kit** file:

```
# build.kit template

# cacti rpm
cactiPkg = RPMPackage()
cactiPkg.name = 'cacti'
cactiPkg.version = '0.8.6j'
cactiPkg.release = '1'
cactiPkg.filename = 'cacti-0.8.6j-1.noarch.rpm'

# package cactid
cactidPkg = SRPMPackage()
cactidPkg.name = 'cacti-cactid'
cactidPkg.version = '0.8.6i'
cactidPkg.release = '1'
cactidPkg.filename = 'cacti-cactid-0.8.6i-1.src.rpm'

# cacti node rpm
cactiNodePkg = RPMPackage()
cactiNodePkg.name = 'cacti-node'
cactiNodePkg.version = '1'
cactiNodePkg.release = '1'
cactiNodePkg.filename = 'cacti-node-1-1.noarch.rpm'

# cacti component
cactiComp = DefaultComponent()
cactiComp.name = 'cacti'
cactiComp.ngtypes = ['installer']
cactiComp.description = 'component associated with cacti'
cactiComp.addDep(cactiPkg)
cactiComp.addDep(cactidPkg)
cactiComp.addScript('component-cacti.remove.sh', mode='postun')

# cacti node component
cactiNodeComp = DefaultComponent()
cactiNodeComp.name = 'cacti-monitored-node'
cactiNodeComp.ngtypes = ['installer', 'compute']
cactiNodeComp.description = 'component associated with cacti nodes'
cactiNodeComp.addDep(cactiNodePkg)
```

```

cactiNodeComp.addScript('component-cacti-monitored-node.remove.sh', mode='postun')

# define a default kit
k = DefaultKit()
k.name = 'cacti'
k.description = 'cacti kit.'
k.arch = getArch()

# Adding the component defined earlier
k.addComponent(cactiComp)
k.addComponent(cactiNodeComp)

```

Image:Kits-components.png

#### Kit Component Hierarchy

This **build.kit** file describes the "cacti" kit which contains 2 components, "cacti" and "cacti-monitored-node". Also listed are the "cacti" RPM, the "cacti-node" RPM, and the "cacti-cactid" source RPM. The "cacti" component contains the "cacti-node" RPM and the "cacti-cactid" source RPM while the "cacti-monitored-node" component contains the "cacti-node" RPM. The UniCluster 4.0.0 Kit and Component RPM's that get created from this build.kit file are:

- **kit-cacti-0.1-0.x86\_64.rpm**
- **component-cacti-0.1-0.noarch.rpm**
- **component-cacti-monitored-node-0.1-0.noarch.rpm**

And the actual third-party Cacti software is in:

- **cacti-0.8.6j-1.noarch.rpm**
- **cacti-cactid-0.8.6i-1.src.rpm**
- **cacti-node-1-1.noarch.rpm**

## 8.4.2 The Automated Way

You can build kits completely by hand, but **buildkit** provides the "new" method for creating a new skeleton kit directory. An example:

```

[root@localhost kits]# buildkit new kit=foobar
Creating foobar directory..
Creating kit for x86 architecture..
foobar directory created.

```

The **buildkit** tool just created a new foobar kit in the current directory. The contents of foobar/ are:

```

[root@localhost kits]# ls -lR foobar/
foobar/:
total 28
drwxr-xr-x 2 root root 4096 Apr 10 09:31 artifacts
-rw-r--r-- 1 root root 1106 Apr 10 09:31 build.kit
drwxr-xr-x 2 root root 4096 Apr 10 09:31 docs
drwxr-xr-x 2 root root 4096 Apr 10 09:31 packages
drwxr-xr-x 5 root root 4096 Apr 10 09:31 plugins
drwxr-xr-x 2 root root 4096 Apr 10 09:31 sources
drwxr-xr-x 2 root root 4096 Apr 10 09:31 tmp

foobar/artifacts:
total 0

foobar/docs:
total 0

foobar/packages:
total 0

foobar/plugins:
total 12
drwxr-xr-x 2 root root 4096 Apr 10 09:31 addhost
drwxr-xr-x 2 root root 4096 Apr 10 09:31 genconfig
drwxr-xr-x 2 root root 4096 Apr 10 09:31 ngedit

foobar/plugins/addhost:
total 0

foobar/plugins/genconfig:
total 0

foobar/plugins/ngedit:
total 0

foobar/sources:
total 8
-rw-r--r-- 1 root root 418 Apr 10 09:31 00-post-script.sh
-rw-r--r-- 1 root root 378 Apr 10 09:31 00-postun-script.sh

foobar/tmp:

```

total 0

### The default (and commented) build.kit looks like:

```
[root@localhost kits]# cat foobar/build.kit
# build.kit template

# Define your packages here by using a correct packageprofile class.
# Available types are SourcePackage(), RPMPackage(), SRPMPackage(),
# DistroPackage(), BinaryPackage()

#pkg1 = SourcePackage()
#pkg1.name = 'foo'
#pkg1.version = '1.0'
#pkg1.release = '0'
#pkg1.installroot = '/opt/foo'
#pkg1.filename = 'foo-1.0.tar.gz'

# Define a default component
comp = Fedora6Component()
comp.name = 'foobar'
comp.description = 'foobar component for Fedora Core 6.'

# Add any packages defined earlier by using the comp.addDep method
#comp.addDep(pkg1)

# Define a default kit
# Change arch if needed to fit your needs (x86, x86_64, noarch)
k = DefaultKit()
k.name = 'foobar'
k.description = 'foobar kit.'
k.arch = getArch()

# Adding the component defined earlier
k.addComponent(comp)

# Add post install script. You can modify this shell script or add new ones in the
# sources directory
k.addScript('00-post-script.sh',mode='post')

# Add post uninstall script. You can modify this shell script or add new ones in the
# sources directory
k.addScript('00-postun-script.sh',mode='postun')
```

The 00-post-script.sh and 00-postun-script.sh scripts are simple post install and post uninstall scripts that can be used to execute Kit specific actions.

# 9 AdminGuide DNS Configuration

## 9.1 DNS in UniCluster 4.0

Proper Domain Name System (DNS) configuration is vital for the operation of UniCluster 4.0. Not only does it provide a convenient way to access the many machines that may exist in the cluster, several software packages require specific DNS configurations to function properly.

## 9.2 General Configuration Philosophy

Many software packages require a DNS to IP relationship that is circular. When you look up a FQDN and get an IP address you should be able to reverse lookup that IP address and get the original FQDN. An example of this relationship is shown by using the `nslookup` and displayed below:

```
[user@host1 ~]$ nslookup host2.domain.com
Server:      10.0.0.10
Address:     10.0.0.10#53

Name:   host2.domain.com
Address: 10.0.0.50

[user@host1 ~]$ nslookup 10.0.0.50
Server:      10.0.0.10
Address:     10.0.0.10#53

50.0.0.10.in-addr.arpa      name = host2.domain.com.

[user@host1 ~]$
```

This approach works great if you only have one FQDN and one NIC per node. UniCluster handles DNS alias and multi-homed systems by making sure that the "hostname" of a given machine always maps to the same "provision" address and that "provision" address always maps back to the hostname.

## 9.3 Configuration Implementation

The implementation of UniCluster's DNS and name resolution functionality is done both with BIND and the `/etc/hosts` file. By default a BIND server is installed on the installer node and is configured as the master for the user provided provision domain. The BIND server forward lookup zone is configured with all of the nodes, including unmanaged nodes and aliases, that have been added to the cluster. The BIND server's reverse lookup zone contains an entry just for the hostname and the IP of the cluster nodes that have a provision network interface. Since BIND is scoped by its authoritative zone on forward lookups and its provision network address on reverse lookups there is a configuration hole for unmanaged nodes that have addresses outside of the provision network (ie corporate NAS) and the installer node if its hostname is not in the provision domain. The `/etc/hosts` file can handle both of configuration holes left by BIND.

The `/etc/hosts` file is generated on the installer node and distributed throughout the cluster using `File_Synchronization_-_CFM`. The `/etc/hosts` file will have an entry for each unmanaged node as well as an entry for the provision interface of the installer node that will map to the hostname of the installer node.

## 9.4 Integrating with an External DNS Server

Integration with an existing DNS server is quite simple as the BIND server on the installer node is already configured with forwarders pointing at the DNS servers that were available at installation time. For integration to work correctly the domain chosen for the cluster should be a subdomain of authoritative zone on external DNS server. All that needs to be done to perform the integration is simply adding a NS entry in the forward lookup zone of the external server. An example configuration snippet with the external domain being `company.com`, cluster domain being `cluster.company.com`, and the hostname of the installer being `installer` is provided below:

```
$ORIGIN company.com.
cluster      NS      installer
installer   A       10.0.0.75
```

UniCluster does not currently support zone transfers from its installer node BIND server to external DNS servers.

[Back to Administrator Guide](#)

## 10 AdminGuide Edit Node Groups

A Node Group is a non-overlapping, logical group of nodes that are provisioned with the identical set of software. Except for the underlying hardware, which may be different within the Node Group, all nodes within a Node Group have the same operating system, installed software, network configuration, disk partitioning, startup and shutdown procedures, etc.

The purpose of a Node Group is to define the software configuration once, and then quickly replicate, or deploy, that configuration across a large group of nodes.

You now need to create or edit a node group that describes the configuration of the Compute Nodes on which you wish to run a particular application.

During installation, UniCluster creates four default Node Groups:

### 10.1 Installer Node Group

All nodes must belong to a Node Group, and the Installer Node is no exception. The Installer Node is the only node in the Installer Node Group.

#### 10.1.1 Packaged Node Group

In a packaged installation each node is individually installed from RPM's using Anaconda/Kickstart to drive an unattended install. After installation each compute node will have its own persistent storage and can be updated independently from other compute nodes in the cluster or even the same node group. Of course all of the standard tools work with packaged based installations and new software can be deployed to already installed packaged based nodegroups using the ngedit tool.

#### 10.1.2 Imaged Node Group

In an imaged installation, a single image is created of an entire installation and that file is transferred to the compute node where it is written to the disk and then run. After the installation, this machine will look almost identical to a node installed with a package based install.

#### 10.1.3 Diskless Node Group

In a diskless installation, again, a single image is created and transferred to the compute node but that image is only written to RAM and this is where it is run from. After installation, this node will have the same software installed as a imaged or package based install but will have less available RAM for operation because RAM is being used to store what is usually stored on disk.

In the simplest case, you will add nodes into the Packaged Node Group. Use the Man:ngedit command to change the node group's operating system, network, disk partitioning, and applications.

[ UniCluster Documentation / Administrator Guide ]

# 11 AdminGuide File and Package Synchronization in the Cluster

## 11.1 Overview

UniCluster 4.0 provides simple distribution of UniCluster software components, bulk operating system updates or individual RPM packages to individual nodegroups.

UniCluster also provides infrastructure so that files that need to be distributed to each node on a nodegroup can be seamlessly deployed. It does so on a per nodegroup basis so different files can be deployed to different nodegroups or the same file names with different contents can be deployed correctly to different nodegroups.

## 11.2 Package Synchronization

Deploying packages to nodegroups is very simple in UniCluster. However, before deploying packages, you must select which packages you want on which nodegroups. This is done through ngedit. If you select changes to UniCluster software components or individual RPMs, "ngedit" will prompt you and ask if you would like to deploy the changes immediately. If you say yes, it will deploy the software automatically. If you choose no, or if you want to deploy bulk operating systems updates that have been downloaded via repopatch then you will need to use the cfmsync command (see below) with the "--package" option for UniCluster software components and individual RPMs select in "ngedit" or the "--repopupdate" option for operating system updates.

"cfmsync" can be run with the nodegroup name and only that nodegroup will be updated, or if it is run without a nodegroup name, it will update all nodegroups.

## 11.3 File Synchronization

Synchronizing packages is a mostly automatic process. The work of synchronizing files is also automatic but you need to place the files that you would like to synchronize in the correct place. In order to have a file synchronized to a nodegroup, the file needs to be placed under /etc/cfm/<nodegroup name>/<file system path on the target nodes>.

For example, if we have a file, "myFile.txt" that we would like to have on all of the nodes in nodegroup "compute-centos", in the location /opt/myApp/myFile.txt, then, on the installer node, simply put your "myFile.txt" file into the /etc/cfm/compute-centos/opt/myApp/ directory.

You may have symbolic links within the /etc/cfm/<nodegroup name>/ structure as well. For example, if we want to have the installer nodes /etc/hosts file propagated to each node in the compute-centos nodegroup, we can create a symbolic link like this:

```
ln -s /etc/hosts /etc/cfm/compute-centos/etc/hosts
```

Now anytime the /etc/hosts file is updated on the installer, the change will be ready to synchronize to the other nodes.

With this setup, distributing the files becomes very similar to distributing packages. You again use the cfmsync command, this time with the "--files" option to distribute the latest changes.

Changes made to files will be propagated the next time "cfmsync" is run. The command checks both the timestamp and a MD5 sum of all files that is synchronizes so it can tell if a file has changed and needs to be synchronized again.

## 11.4 Usage

The usage for "cfmsync" is:

```
usage: cfmsync [-h|-v| -f -p -n node_group]
```

The cfmsync tool will signal the nodes in the cluster to update files and or packages. When this tool is run the nodes in the cluster, or a specified node group, will contact the primary installer to determine which files and or packages need to be updated. The nodes will then download the newer files or packages and install them.

The arguments have the following meanings:

```
-h           - Provide help on this tool.
-v           - Print the tool version.
-f           - Perform a configuration file update.
-p           - Perform a package update.
-n node-group - The name of the node group to update. If not
              provided update all node groups.
```

options:

```
-h, --help          show this help message and exit
-n NODEGRP, --nodegroup=NODEGRP
-f, --files
-p, --packages
-v, --version
```



## 12 AdminGuide Kits

### 12.1 What is a Kit?

New cluster enabled software is integrated into UniCluster via Kits. Kits provide the top level software container in the UniCluster 4.0.0 software deployment model. Kits directly contain references to "Components" as well as plugins/configuration agents that integrate the given component into the cluster. An example of such a plugin would be a script that automatically updates /etc/hosts when a new node is added to the cluster (this plugin actually exists in the "base" kit of UniCluster 4.0.0).

### 12.2 What is a Component?

A Component, like a Kit, is type of software container. It sits between the Kits and the RPM's that actually contain the new software. Kits typically have several Components and each component may reference many RPM's (either core platform or Kit based) through its dependencies. A component is considered an Atomic bundling of software unlike a kit that may provide several independent "Components". Enabling a component implies installing all of the software defined by the given component.

### 12.3 Types Of Kits

There are only 2 major Kit types OS kits and feature kits. OS kits provide the complete set of RPM's for a given distribution and are used as a source for provisioning the given OS on a cluster node. These kits are created by using the original vendor supplied installation medium as the source. A feature kit, provides additional applications or libraries to the cluster. The one special feature kit, the "base" kit, is key to the functioning of the system it is installed as an irremovable kit.

To display the feature kits available from the UniCluster run the yum list command on the installer node.

```
[root@installer cfm]# yum list unicluster-kit-*
Loading "fastestmirror" plugin
Loading mirror speeds from cached hostfile
 * unicluster-repo: peaches.grid.org
 * base: styx.biochem.wfubmc.edu
 * updates: repo.bioinformatics.upenn.edu
 * addons: styx.biochem.wfubmc.edu
 * extras: mirror.raystedman.net
Installed Packages
unicluster-kit-SGE.noarch          1.0-0          installed
unicluster-kit-base.noarch        1.0-11         installed
Available Packages
unicluster-kit-cacti.noarch       0.8.6-9        unicluster-repo
unicluster-kit-ganglia.noarch     1.0-0          unicluster-repo
unicluster-kit-grid.noarch        1.0-0          unicluster-repo
unicluster-kit-hpc.noarch         1.0-7          unicluster-repo
unicluster-kit-intel-clck.noarch  1.0-0          unicluster-repo
unicluster-kit-intelruntime.noarch 1.0-0          unicluster-repo
unicluster-kit-nagios.noarch      2.10-7         unicluster-repo
unicluster-kit-ofed.noarch        1.0-7          unicluster-repo
unicluster-kit-xen.noarch         1.0-0          unicluster-repo
```

## 12.4 Working With Kits

### 12.4.1 Installing a UniCluster Kit

At a low level the kitops tool is used to install new kits however all of the default kits provided with UniCluster 4.0 are have wrapper scripts that facilitate the download and installation of the kit. An example is the ganglia kit which has the "unicluster-kit-ganglia" rpm that then provides the "install-kit-ganglia". To install this kit simply run "yum install unicluster-kit-ganglia" on the installer node.

```
[root@oracle-installer ~]# yum install unicluster-kit-ganglia
Loading "security" plugin
Setting up Install Process
Parsing package install arguments
Resolving Dependencies
--> Running transaction check
--> Package unicluster-kit-ganglia.noarch 0:1.0-0 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version      Repository      Size
=====
Installing:
 unicluster-kit-ganglia noarch    1.0-0        unicluster      2.7 k
=====
Transaction Summary
=====
```

```
Install      1 Package(s)
Update       0 Package(s)
Remove       0 Package(s)
```

```
Total download size: 2.7 k
Is this ok [y/N]: y
Downloading Packages:
(1/1): uniclust-er-kit-gan 100% |=====| 2.7 kB    00:00
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing: uniclust-er-kit-ganglia          ##### [1/1]

Installed: uniclust-er-kit-ganglia.noarch 0:1.0-0
Complete!
```

Now run the kit installation script.

```
[root@oracle-installer ~]# install-kit-ganglia
kit-ganglia-0.1-0.noarch. 100% |=====| 14 kB    00:00
component-ganglia-gmetad- 100% |=====| 2.3 kB    00:00
component-ganglia-gmond-0 100% |=====| 3.4 kB    00:00
ganglia-gmetad-3.0.7-1.x8 100% |=====| 84 kB    00:00
ganglia-gmetad-3.0.7-1.i3 100% |=====| 84 kB    00:00
ganglia-gmond-3.0.7-1.x86 100% |=====| 217 kB   00:00
ganglia-gmond-3.0.7-1.i38 100% |=====| 215 kB   00:00
ganglia-web-3.0.7-1.noarc 100% |=====| 118 kB   00:00
rrdtool-1.2.23-3.x86_64.r 100% |=====| 466 kB   00:00
rrdtool-1.2.23-3.i386.rpm 100% |=====| 464 kB   00:00
Adding kit using kitops...
10/10 - rrdtool-1.2.23-3.i386.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Added kit ganglia-0.1-noarch
```

At this point the kit has been installed and you can associate it with a nodegroup using `ngedit`.

```
[root@oracle-installer ~]# kitops -l
Kit:          oracle
Description:  OS kit for oracle 5.2 x86_64
Version:     5.2
Architecture: x86_64
OS Kit:      Yes
Removable:   Yes
Node Groups: installer, compute-oracle,
             compute-imaged, compute-diskless

Kit:          base
Description:  Base Kit
Version:     5.1
Architecture: noarch
OS Kit:      No
Removable:   No
Node Groups: installer, compute-oracle,
             compute-imaged, compute-diskless

Kit:          ganglia
Description:  Ganglia Monitoring Kit
Version:     0.1
Architecture: noarch
OS Kit:      No
Removable:   Yes
Node Groups:
```

## 12.4.2 Installing a Non-UniCluster Kit

Kits not hosted in the UniCluster 4.0 yum repository can be installed using `kitops` directly.

```
[root@oracle-installer tmp]# kitops -a -m intelruntime/
3/3 - component-intel-runtime-0.1-0.noarch.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Added kit intelruntime-1.1-x86_64
```

## 12.4.3 Installing Additional OS Kits

When the installer node is created, part of the installation process creates and installs the first OS kit. This kit will be for the operating system type of the installer node. In order to have compute nodes with a different operating system, additional OS kits must be installed.

To install an additional OS kit, put in the CD/DVD for the new operating system, or determine the path to the ISO file. Then run the appropriate "kitops" command.

If using a CD/DVD, simply run:

```
kitops -a
```

If using an ISO, run:

```
kitops -a -m /path/to/iso/file
```

The media will then be copied into a repository on the installer node:

```
[root@myinstaller ~]# kitops -a -m /storage/isos/rhel-5.2-server-x86_64-dvd.iso
/tmp/kitopsYPI_YE/.etc/terminfo/x
/tmp/kitopsYPI_YE/.etc/terminfo/v
/tmp/kitopsYPI_YE/.etc/terminfo/v/vt100-nav
/tmp/kitopsYPI_YE/.etc/terminfo/v/vt100
/tmp/kitopsYPI_YE/.etc/terminfo/v/vt102
/tmp/kitopsYPI_YE/.etc/terminfo/a
/tmp/kitopsYPI_YE/.etc/terminfo/a/ansi
/tmp/kitopsYPI_YE/.etc/terminfo/l
/tmp/kitopsYPI_YE/.etc/terminfo/l/linux
/tmp/kitopsYPI_YE/.etc/mtab
/tmp/kitopsYPI_YE/.sbin
/tmp/kitopsYPI_YE/.sbin/halt
/tmp/kitopsYPI_YE/.sbin/reboot
/tmp/kitopsYPI_YE/.sbin/insmod
/tmp/kitopsYPI_YE/.sbin/loader
/tmp/kitopsYPI_YE/.sbin/poweroff
/tmp/kitopsYPI_YE/.sbin/rmmod
/tmp/kitopsYPI_YE/.sbin/modprobe
/tmp/kitopsYPI_YE/.sbin/sh
/tmp/kitopsYPI_YE/.sbin/init
/tmp/kitopsYPI_YE/.bin
/tmp/kitopsYPI_YE/.tmp
/tmp/kitopsYPI_YE/.sys
/tmp/kitopsYPI_YE/.modules
/tmp/kitopsYPI_YE/.modules/modules.alias
/tmp/kitopsYPI_YE/.modules/module-info
/tmp/kitopsYPI_YE/.modules/pci.ids
/tmp/kitopsYPI_YE/.modules/modules.dep
/tmp/kitopsYPI_YE/.modules/modules.cgz
/tmp/kitopsYPI_YE/.profile
/tmp/kitopsYPI_YE/.init
/tmp/kitopsYPI_YE/.buildstamp
14979 blocks
/depot/kits/rhel/5/x86_64/.discinfo
/depot/kits/rhel/5/x86_64/.treeinfo
/depot/kits/rhel/5/x86_64/.Cluster
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-as-IN-5.1.0-7.noarch.rpm
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-bn-IN-5.1.0-7.noarch.rpm
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-de-DE-5.1.0-7.noarch.rpm
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-en-US-5.1.0-7.noarch.rpm
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-es-ES-5.1.0-7.noarch.rpm
/depot/kits/rhel/5/x86_64/.Cluster/Cluster_Administration-fr-FR-5.1.0-7.noarch.rpm
...
/depot/kits/rhel/5/x86_64/.isolinux/memtest
/depot/kits/rhel/5/x86_64/.isolinux/options.msg
/depot/kits/rhel/5/x86_64/.isolinux/param.msg
/depot/kits/rhel/5/x86_64/.isolinux/rescue.msg
/depot/kits/rhel/5/x86_64/.isolinux/splash.lss
/depot/kits/rhel/5/x86_64/.isolinux/vmlinuz
7043728 blocks
Any more disks for this OS kit? [y/n] n
1/1 - unicluster-release-1.0-0.noarch.rpm
Saving Primary metadata
Saving file lists metadata
Saving other metadata
Added kit rhel-5-x86_64
[root@myinstaller ~]#
```

At this point the OS kit has been added to the system. To use that OS kit, a nodegroup has to be created that uses that OS type. The simplest way to do this is to use "ngedit".

1. On the main screen choose a suitable nodegroup to copy (choose a packaged node group if you want the new one to be packaged, imaged nodegroup for imaged, etc) and press the "Copy" button.
2. The new nodegroup will now show up in the list of nodegroups but it will have the same OSTYPE as the original nodegroup. Select the new nodegroup and select "Edit".
3. Then select "OS Config" and press "Edit".
4. Now select the the OS type that you and the new nodegroup to be and press "Next".

5. Press "Next" on the subsequent screens until you arrive back at the "Node Group Category Selection" screen and select "Next" once more.
6. Press "Accept" and UniCluster will perform the necessary actions to transition the newly created nodegroup to your new OS type.

Nodes can now be added to the new nodegroup and they will be installed with the new OS kit.

## 12.4.4 Removing Kits

The kitops tool is used to remove kits from the system. Components within the kit must not be enabled on any nodegroups.

```
[root@oracle-installer tmp]# kitops -e -k intelruntime
+-----+-----+-----+
| Kit           | Version | Architecture |
+-----+-----+-----+
| intelruntime | 1.1     | x86_64        |
+-----+-----+-----+
The above kits will be removed.

Confirm [y/N]: y
```

[ UniCluster Documentation / Administrator Guide ]

## 13 AdminGuide Modifying Node pxeboot Parameters

### 13.1 Introduction

The PXE configuration used by compute nodes is generated by the boothost tool and uses information stored in the database to create the appropriate pxelinux.cfg file for a given node. The PXE configuration files for all of the compute nodes can be found in /tftpboot/tortuga/pxelinux.cfg/ on the installer node with each file being named based on the MAC address of a compute node.

### 13.2 Temporarily Changing Configuration

Simply updating a the appropriate pxelinux.cfg file for a given node will allow for a temporary update in the PXE boot parameters for that node. To edit the configuration for "compute-00-00" one could do the following:

```
[root@oracle-installer xinetd.d]# boothost -l compute-00-00
Node:    compute-00-00          Node Group: compute-oracle
State:   Installed             Boot:    Disk
Kernel:  kernel-oracle-5.2-x86_64
Initrd:  initrd-oracle-5.2-x86_64.img
Kernel Params: text noipv6 kssendmac selinux=0
MAC:     00:0c:29:c7:85:4b      IP:     172.19.0.2
-----
```

And then use the MAC address to get the pxelinux.cfg file:

```
[root@oracle-installer xinetd.d]# echo /tftpboot/tortuga/pxelinux.cfg/`echo "00:0c:29:c7:85:4b" | sed 's/^/01-/; s/\:\/-/g'`
/tftpboot/tortuga/pxelinux.cfg/01-00-0c-29-c7-85-4b
[root@oracle-installer xinetd.d]# cat /tftpboot/tortuga/pxelinux.cfg/01-00-0c-29-c7-85-4b
# PXE file for: compute-00-00
default Reinstall
prompt 0
label Reinstall
    kernel kernel-oracle-5.2-x86_64
    append initrd=initrd-oracle-5.2-x86_64.img syslog=172.19.0.1:514 niihost=172.19.0.1 ks=http://172.19.0.1/kickstart/kernel-oracle-5.2-
```

### 13.3 Permanent Configuration Change

To permanently change the configuration of a compute node you need to use ngedit to update the compute nodes nodegroup. In ngedit simply go to the "OS Config" tab for the desired nodegroup, select your nodegroup OS, and then update the "Kernel Params" option.

[ UniCluster Documentation / Administrator Guide ]

## 14 AdminGuide Network Editor (netedit)

### 14.1 Introduction

The Network Editor (netedit) allows for the manipulation of the UniCluster 4.0 managed network interfaces. Either new networks can be created or existing networks can be added. The networks that UniCluster manages need not be all physically attached to the cluster, only single network must be connected for provisioning.

### 14.2 Restrictions

While new networks can be added and then associated with node groups through ngedit after nodes have been provisioned existing nodes will not have configuration generated for the new networks. In order to have the interfaces created on the nodes inside the nodegroup you must first remove with addhost -e and then add them back in with the addhost tool. New nodes added to the nodegroup will automatically have their interface added and configured.

### 14.3 Usage and Examples

For usage instructions and examples please see netedit.

[ UniCluster Documentation / Administrator Guide ]

## 15 AdminGuide Networking

A typical cluster has two networks:

### Compute Network

To manage the cluster, deploy software, etc., the Installer Node must be able to reach each Compute Node. The Compute Nodes may also need to communicate with each other. In the simplest configuration, all nodes in a cluster (the installer node and the compute nodes) connect to a single network – the Compute Network. More complicated configurations might have a separate Compute Network for each Node Group. We strongly suggest that the Compute Network be a dedicated network segment, separate from any user or desktop traffic.

### Public Network

System administrators and users access the cluster through the gateway provided by the Installer Node. Typically, the Installer Node has two network interfaces: 1) connects to the Compute Network, and 2) connects to the Public Network for admin/user/desktop access.

[ UniCluster Documentation / Administrator Guide ]

# 16 AdminGuide Node Group Editor

## 16.1 Overview

The Node Group Editor "ngedit" tool is used to administer node groups. The tool can modify, copy, list, create, or delete node groups.

## 16.2 What it does

"ngedit" manipulates node groups. What this really means is that the tool does a lot of interfacing to the database that holds all of the information that makes up the node group. In fact, the majority of the work this tool does in pulling data from and updating the database. Copying one node group to another is the most complicated operation. It first pulls down all of the configuration data from the database and then creates a new database record with the information that it just gathered. The tool also handles creating the initrd and kernel images for the new node group and places them in the appropriate directory with the correct names.

As far as updating node groups is concerned, the functionality is covered by the copying functionality. To do an update does the same database activity but without creating additional record.

Deleting is also not complex, it simply removes the entries from the database and the initrd and kernel images from the TFTP directory.

## 16.3 Usage

usage: Node Group Editor is an administrative tool for nodegroup related operations, including their copying/deletion/modification. It allows ultimate control over the specific behavior/configuration of node groups. ngedit is used as follows:  
ngedit [ -l | { -p | -s | -d } <NGName> | -c <NGName> -n <NewNGName> ]

options:

```
-h, --help          show this help message and exit
-d NGDEL, --delete=NGDEL
                    Delete specified nodegroup
-c NGSRC, --copy=NGSRC
                    Copy specified nodegroup
-n NGNEW, --new=NGNEW
                    NG name to use for a new nodegroup
-s NGSTALE, --stale=NGSTALE
                    Print all nodes with expired state for a given NG
-l, --list          List all nodegroups
-p NGPRINT, --print=NGPRINT
                    Provide detailed printout of a specified NG
```

[ UniCluster Documentation / Administrator Guide ]

# 17 AdminGuide Package, Imaged, and Diskless

## 17.1 Overview

UniCluster 4.0 supports three methods for installing compute nodes, packaged, diskless, and imaged. The imaged and diskless methods are quite similar to each other but are fairly different from the manner in which a packaged installation takes place.

In an imaged installation, a single image is created of an entire installation and that file is transferred to the compute node where it is written to the disk and then run. After the installation, this machine will look almost identical to a node installed with a package based install.

In a diskless installation, again, a single image is created and transferred to the compute node but that image is only written to RAM and this is where it is run from. After installation, this node will have the same software installed as a imaged or package based install but will have less available RAM for operation because RAM is being used to store what is usually stored on disk.

In a packaged installation each node is individually installed from RPM's using Anaconda/Kickstart to drive an unattended install. After installation each compute node will have its own persistent storage and can be updated independently from other compute nodes in the cluster or even the same node group. Of course all of the standard tools work with packaged based installations and new software can be deployed to already installed packaged based nodegroups using the ngedit tool.

## 17.2 Diskless and Imaged Installation Method

### 17.2.1 Image Creation

Until a imaged or diskless image is actually run on a compute node, the process of using a imaged or diskless image are quite similar. There are two images that are build, an initrd image that is used to provide an initial environment to boot from, and an image of the complete operating system and applications. The initrd image for both methods are the same in content but are compressed differently. The main images are the same for imaged and diskless nodes.

The initrd image is created by extracting the modules from the kernel RPM into a new directory. Then the tortuga libraries are copied into the directory structure and the imageinit.sh file is copied into the root of the directory structure. After that the directory structure is compressed into an image file. For a imaged node the image is created using a mkfs call which is then gzipped. If a diskless initrd is being created it is done with cpio and then compressed. After packaging, the compressed files are put into /tftpboot/tortuga/ where the compute nodes will look for them when they PXE boot.

The main image is created the same for imaged or diskless machines. This image is created by setting up a new directory and then taking all of the RPMs that are needed for the machine and installing them into this new directory. This creates an installation tree that will become the tree on the compute node. After that, some system files such as /etc/passwd and /etc/shadow are updated within the directory structure and then the tortuga files are copied into the directory structure. The directory structure is then compressed and put into the /depot/images directory on the installer node and this is where the compute nodes will look for it.

### 17.2.2 Image Customization

Both Imaged and Diskless root file system images can be customized using the imageedit tool. This tool allows for the creation of an "overlay" that is applied to the "base" image during installation. This overlay can add delete or change the permissions, ownership, and timestamps of files inside the image.

### 17.2.3 Installation

When the above images are created they are put into their own node groups. This is done so that a node is simply assigned to a node group and based on the node group when it boots up in that node group it boots in the method defined by the node group. Once the images are in place, as nodes are placed in a node group their PXE boot configuration is set according to the node group. Then when a node is powering on it will retrieve the initrd and kernel for its node group. The kernel is the same but the initrd is what defines how to boot and where, if necessary, to get the main image from.

From there the imageinit.sh file gets run, which calls the imageinit.py Python file which does the work needed to bring up the new compute nodes operating system and in the case of the imaged install, copy it to disk.

### 17.2.4 Co-Existing With Other Operating Systems

Diskless images due to their nature can be used on a machine that has an OS installed on a fixed disk without interfering with that OS. To switch between a UniCluster diskless os and the physical OS simply remove the node from UniCluster management or add it back in. Both operations can be done with the addhost tool. Another strategy would be simply switching the machines to the "unmanaged" node group to get the nodes to boot off of their physical disks.

Imaged nodes can also co-exist with existing operating systems with a bit of extra configuration. Imaged nodes offer the ability to "preserve" partitions. A preserved partition will not be touched by the imaged node installation process including partition flags. Imaged nodes also allow for the configuration of where the GRUB bootloader is installed. By default it will be installed in the MBR of the first fixed disk but if you want to preserve the existing MBR bootloader (ie the Microsoft Windows bootloader) you would want to install GRUB on the "boot" partition of your imaged OS. Both the preserve and

bootloader location options are configured in the partition editor of the ngedit tool. It is important to set the size of the preserver partition to a size at least equal to the size of the partition you wish to preserve as this number is used to pick the starting point of the next partition.

### **17.2.5 Additional OS Support**

Imaged and diskless nodes support an additional compute operating system, RHEL 4. When using RHEL 4 as an OS the component-base-node-legacy must be used in place of the standard component-base-node. The compute nodes running RHEL4 does not support updates while running thus a rebuild of the image is necessary if new software is to be included. Furthermore, many of the add on kits will not work with RHEL 4; the SGE kit is known to work.

## **17.3 Packaged Installation Method**

### **17.3.1 Server Configuration**

Like the imaged and diskless methods of installation, the packaged method of installation uses PXE to deliver the base kernel and initrd to the compute node being installed. The initrd used for packaged installations is the standard distribution initrd. The major customization of the standard Anaconda/Kickstart method is the use of an updates.img on the server that allows additional software to be deployed to the compute node during installation initialization.

### **17.3.2 The Nodeinstaller Script**

The updates.img file that gets deployed contains a base tortuga environment, nodeinstaller script and an simple anaconda wrapper The anaconda wrapper drives the installation by calling the nodeinstaller script. The nodeinstaller script talks to the installer node to retrieve its node specific configuration and then generates an appropriate Anaconda Kickstart file. This kickstart file is then passed to the stock anaconda installer to drive the installation.

[ UniCluster Documentation / Administrator Guide ]

## 18 AdminGuide Running Commands Across the Cluster (parallel shell)

### 18.1 Introduction

UniCluster 4.0 contains the Parallel Distributed Shell (pdsh) for running parrallel jobs across the cluster. This tool ties into the UniCluster 4.0 configured ssh and authentication environment to provide an easy mechanism for running tasks on several cluster nodes.

### 18.2 Use and Examples

For use instructions and examples of pdsh please see the MAN page at pdsh.

[ UniCluster Documentation / Administrator Guide ]

## 19 AdminGuide Troubleshooting

### 19.1 Troubleshooting

UniCluster 4.0.0 is an open-source software product backed by a community of developers and users. If you encounter issues with the software please visit the UniCluster Group hosted on Grid.org look for some answers. If you can't find what you are looking for please post your questions to the community.

[ UniCluster Documentation / Administrator Guide ]

## 20 AdminGuide Updating the Cluster

### 20.1 Overview

All of the vendors of the supported operating systems provide updates to their releases in the form of either a Yum repository hosted on the Internet, or via a web service that is accessible with a paid subscription. UniCluster 4.0 provides a means to interface to these services, download the updates and deploy the updates throughout the cluster.

### 20.2 Accessing Updates

#### 20.2.1 CentOS

CentOS provides public access to their Yum updates repository. UniCluster can mirror these updates locally by using rsync to directly pull down the updates. Edit the "centos" entry in the /opt/tortuga/etc/updates.conf to point to a geographically nearby mirror.

#### 20.2.2 RedHat

To use the RedHat update service you will need a RedHat subscription. This can be obtained from <http://rhn.redhat.com/> Once you create and setup your account you will need to put your username and password into the "rhel" section of the /opt/tortuga/etc/updates.conf file. The additional systemidfile4 and systemidfile5 attributes are provided so you can download updates for RHEL 4 or RHEL 5 on the same machine. In order to use the multi-version downloading feature you must have a system id file from a machine running the desired RHEL version.

#### 20.2.3 Oracle Linux

To access the Oracle Linux update service you will need to subscribe to Oracles ULN. Instructions for registering and setting up your machine to access Oracles updates can be found at:

[https://linux.oracle.com/uln\\_faq.html](https://linux.oracle.com/uln_faq.html)

If you wish to have RHEL and Oracle Linux updates on the same machine you must configure the systemid attributes of the rhel section of /opt/tortuga/etc/updates.conf to point to something other than /etc/sysconfig/rhn/systemid as the Oracle update agent must use /etc/sysconfig/rhn/systemid to talk to ULN.

### 20.3 Downloading Updates

Once you've configured or subscribed to updates, UniCluster will do the work of downloading the latest packages and make them available to the entire cluster. UniCluster can have multiple OS kits and can create compute node groups of any of the operating systems for which it has an OS kit. Each OS kit is updated individually with the repopatch command.

To see the list of available OS kits to be updated, run:

```
[root@oracle Server]# repopatch -l
```

```
The following OS kits are installed:
```

```
centos-5.1-x86_64
oracle-5.2-x86_64
```

```
The following snapshots exist:
```

Snapshot ID	Description
4	8/8/08 CentOS 64 bit Update

Here you can see that the OS kit "centos-5.1-x86\_64" is installed. To have UniCluster download all of the latest updates for this OS kit, run (NOTE: this will look different for different operating systems):

```
[root@oracle ~]# repopatch -u -o centos-5.1-x86_64
Getting updates for centos-5.1-x86_64. This may take a while...
cmd=/usr/bin/rsync -art rsync://mirror.anl.gov/centos/5/updates/x86_64 --exclude=debug/ /depot/kits/centos_updates/5.1
```

When the "repopatch -u" command finishes, all of the latest updates have been downloaded to the installer node and are ready to be deployed.

### 20.4 Deploying Updates

Updates are deployed on a per nodegroup basis and as such are configured via the ngedit command. To configure updates, run "ngedit", select the nodegroup that you wish to configure and press "Edit", then select "Updates" and press "Edit" again.

Initially there will be two entries to choose from, one for the base OS with no updates, and one for the latest updates that have been pulled down to the installer node via repopatch. You also have the option to create snapshots of the current set of updates. This is useful in the case where you would like to keep a nodegroup on the existing set of software update packages but would like to run "repopatch" again to get the very latest updates from the OS vendor without having these latest updates immediately deployed. To create a new snapshot, simply press the "Snapshot" button\*. You will be prompted to give the snapshot a description and then the new entry will appear in the list. Simply highlight the entry that you want associated with the nodegroup and press "Next".

Once you've completed your changes within "ngedit", run:

```
cfmsync -u <nodegroup>
```

to have each of the nodes in that nodegroup pull the current updates from the installer node.

\*It is also possible to use the "repopatch" command to create snapshots from the command line.

[ UniCluster Documentation / Administrator Guide ]

# 21 AdminGuide User Account Management

## 21.1 Introduction

UniCluster 4.0 is a cluster management and provisioning systems and has built in support for handling user account synchronization. By default the system is configured using MD5 password encryption and shadow passwords. The cfmsync tool is used to sync the master copy of /etc/passwd, /etc/group, and /etc/shadow stored on the primary installer to all of the compute nodes.

## 21.2 User Management

### 21.2.1 Adding a User

Adding a user to a UniCluster cluster is quite simple. Simply running useradd and passwd at the command line on the primary installer will create a new user with a home directory in /home/<username> and entries in the master password files.

```
[root@oracle-installer ~]# useradd testuser
[root@oracle-installer ~]# passwd testuser
Changing password for user testuser.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
[root@oracle-installer ~]# ls /home -l
total 4
drwx----- 4 testuser testuser 4096 Aug 20 09:45 testuser
```

Since the /home directory is NFS shared to all nodes in the cluster the users home directory is immediately available on all cluster nodes. The final step in adding a new user is running cfmsync -f which will replicate the updated /etc/passwd, /etc/group, and /etc/shadow files to the nodes in the cluster.

```
[root@oracle-installer ~]# cfmsync -f
Running plugin: /opt/tortuga/lib/plugins/cfmsync/getent-data.sh
Distributing 10 KBytes to all nodes.
Updating installer(s)
Done
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
```

New users will automatically have ssh keys created for them the first time the login due to the default /etc/skel/.bashrc file.

```
[root@oracle-installer ~]# ssh testuser@compute-00-00
testuser@compute-00-00's password:
No public/private RSA keypair found.
Generating public/private rsa key pair.
Created directory '/home/testuser/.ssh'.
Your identification has been saved in /home/testuser/.ssh/id_rsa.
Your public key has been saved in /home/testuser/.ssh/id_rsa.pub.
The key fingerprint is:
9c:67:88:cc:f8:71:4f:d7:6e:95:74:74:f2:24:4d:c5 testuser@compute-00-00.cluster.univa.com
```

Since the new users home directory is shared across all nodes the ssh key is now available and passwordless subsequent logins are enabled.

```
[testuser@compute-00-00 ~]$ ssh oracle-installer
Warning: Permanently added 'oracle-installer,172.19.0.1' (RSA) to the list of known hosts.
UniCluster 4.0 Installer Node
[testuser@oracle-installer ~]$
```

### 21.2.2 Removing Users

Removing users is done in a very similar way as adding users; the standard userdel and passwd tools are used followed by a cfmsync -f.

```
[root@oracle-installer ~]# userdel -r testuser
[root@oracle-installer ~]# cfmsync -f
Running plugin: /opt/tortuga/lib/plugins/cfmsync/getent-data.sh
Distributing 9 KBytes to all nodes.
Updating installer(s)
Done
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
Sending to 172.19.0.255
```

Now the user has been removed from the cluster.

```
[root@oracle-installer ~]# ls /home -l
total 0
[root@oracle-installer ~]# ssh compute-00-00
Last login: Wed Aug 20 09:35:37 2008 from oracle-installer.univa.com
[root@compute-00-00 ~]# su - testuser
su: user testuser does not exist
```

## 21.3 LDAP, NIS, and Winbind

UniCluster 4.0 does not configure either LDAP, Winbind, or NIS out of the box, but there should be no issues enabling either of these network based authentication and authorization services with UniCluster 4.0.

### 21.3.1 Generic Standalone Strategy

A basic standalone deployment strategy would be to install the server component on the installer node and then configure the installer node to use the server. The various client authentication configuration files that were generated on the installer node then be replicated using cfmsync to the other nodes in the cluster. For information on how to add new files to be replicated by cfmsync please see this article.

### 21.3.2 Generic Integration Strategy

A basic integration strategy would be very similar to the stand alone strategy. First configure the installer node to use the upstream network authentication service and then replicate the appropriate client authentication configuration files to other nodes in the cluster using cfmsync. For information on how to add new files to be replicated by cfmsync please see this article. Since your compute nodes may not have direct network access to the upstream authentication server it may be necessary to install and configure the proxy or slave component of the appropriate server on the installer node to be used as a gateway to the upstream server.

[ UniCluster Documentation / Administrator Guide ]

## 22 AdminGuide Yum Repositories

Yum, which means *Yellow dog Updater, Modified*, is a program used by a number of Linux distributions (including RedHat) to manage the downloading, installation and updating of RPM-based software packages. UniCluster makes extensive use of Yum to manage the software installed in your cluster.

### 22.1 Repository Types

There are four different types of Yum repositories used by UniCluster:

#### Public UniCluster Repository

UnivaUD, Inc. hosts a repository on the public internet for downloading the UniCluster software onto your Installer Node. Once you have set up your Installer Node (see Getting Started) UniCluster creates the following Yum repositories on the Installer Node and makes them available to all nodes in the cluster:

#### Operating System Base Repository

An Operating System Base Repository contains the base distribution of an operating system. The `unicluster-setup` command copies the files from the operating system distribution media onto the Installer Node, creates a Yum repository out of it, and makes the repository available to all nodes in the cluster.

#### Operating System Updates Repository

An Operating System Updates Repository contains all updates released by the operating system's vendor since the base distribution. The `repatch` command downloads these updates from the vendor, stores them in the updates repository, and makes the repository available to all nodes in the cluster. The `cfmsync -u` command instructs all nodes in a given nodegroup to update their packages from this repository.

#### Feature Repository

A Feature Repository contains a UniCluster Kit. Each UniCluster Kit is stored in its own Feature Repository. If there are multiple versions or multiple architecture of the same kit, there will be a separate repository for each.

### 22.2 OS updates with repopatch

`repatch` is a tool used to update the OS packages within an OS update repository. `repatch` will pull down via `rsync` and locally mirror the operating system vendors update repository if the vendor makes it available (such as with Fedora or Cent OS), or it can connect to Redhat's RHN to retrieve updates via Redhat's subscription service.

To configure `repatch` to update a Redhat repository from the RHN, simply modify the file `?/opt/tortuga/etc/updates.conf?`, with the username and password for the RHN subscription. See the sample `updates.conf` file below. The config file also allows the selecting of which mirror to use in the case of mirroring with `rsync`.

Sample `updates.conf` file:

```
[fedora]
url=http://download.fedora.redhat.com/pub/fedora/linux/

[centos]
url=rsync://mirror.anl.gov/centos/5.1

[rhel]
username=
password=
url=https://xmlrpc.rhn.redhat.com/XMLRPC
yumrhn=https://rhn.redhat.com/rpc/api
```

Usage:

```
options:
-h, --help           show this help message and exit
-o OSKIT, --oskit=OSKIT
                    <osname>-<osvers>-<osarch>
-v, --version       Display version of tool
-l, --list          Display list of installed OS kits and snapshots
-u, --update        Update the selected OS kit
-s DESCRIPTION, --snapshot=DESCRIPTION
                    Snapshot the selected OS kit and give it a description
-e DELETE, --delete=DELETE
                    Delete/erase a snapshot. Takes a snapshot ID
-y, --yes           Assume "yes"
--dbdriver=DBDRIVER Database driver (sqlite, mysql)
--dbdatabase=DBDATABASE
                    Database
--dbuser=DBUSER     Database username
--dbpassword=DBPASSWORD
                    Database password
```

The db options are only used, when connecting to an alternate database. They are not needed for normal operations on the active database.

[ UniCluster Documentation / Administrator Guide ]

## 23 AdminGuide nghosts

### 23.1 The nghosts Utility

The nghosts utility provides a single step process for moving a node from one group to another as well as a method for listing the node group membership for nodes in the cluster. It can then leverage the pdsh tool to send the "reboot" command to all migrated nodes.

The nghosts utility can be run interactively with a TUI or, provided all of the necessary input is provided, non-interactively from the command line.

### 23.2 Usage

```
usage: nghosts [-hvl]
        [-g {node group}]
        [-m {hosts} -t {node group} [-f {node group}] [-a rack#] [-r]]
        [-f {node group} -t {node group} [-a rack#] [-r]]

options:
-h, --help            show this help message and exit
-v, --version         Display version of tool
-l, --list-all-nodegroups
                    List all the node groups
-g LISTNODEGROUP, --list-nodegroup=LISTNODEGROUP
                    List a specific node group
-f, --from-group      Specify node group to move from
-t TOGROUP, --to-group=TOGROUP
                    Move the nodes to the specified node group. Must be
                    used with option -m or -f
-m, --move-hosts     Specify a list of nodes to move
-r, --reinstall       Specify to mark the nodes as expired for
                    reinstallation
-a RACKNUMBER, --rack=RACKNUMBER
                    The rack number to use (if applicable)
```

### 23.3 TUI Screens

There are only two screens in the TUI. The first screen allows you to select whether you wish to move all nodes from one node group to another or a set of nodes from one node group to another. The second screen allows the user to select the source nodes (either by node or group based on the first screen) and the new node group. There is also a checkbox on the second screen that controls the "--reinstall" option.

### 23.4 Node Migration Workflow

Really all that is done by nghosts is:

1. Acquire input from either the interactive TUI, the command line, or both
2. Remove the selected nodes from their current node group
3. Add the selected nodes to their new node group
4. If specified input, reboot the selected nodes via pdsh for reinstall
5. Exit

### 23.5 Listing of Node Groups

When listing node groups, only node groups that actually have member nodes are listed. The primary installer node is also excluded as a qualifying node and is never listed. Since the primary installer is typically the only node in the default installer node group the default installer node group is typically not displayed either.

[ UniCluster Documentation / Administrator Guide ]

## 24 Administrator Guide

Now that you have downloaded and installed the UniCluster software onto your Installer Node (see Getting Started), you are ready to begin provisioning and configuring your cluster for use.

### 24.1 Chapters

1. Edit Node Groups
2. Adding Hosts to the Cluster
3. Add Compute Nodes
4. Networking
5. Troubleshooting
6. Updating the Cluster
7. YUM Repositories
8. Kits and Components
9. Node Group Editor
10. nghosts
11. Network Editor (netedit)
12. Package, Imaged, and Diskless
13. Cluster monitoring
14. Creating your own Kits
15. Adding Custom Kernels & initrds
16. Modifying Node pxeboot Parameters
17. Configuring BMC on Nodes
18. Configuring IPMI to Nodes
19. File and Package Synchronization in the Cluster
20. User Account Management
21. Adding a Network Filesystem to the Cluster
22. Running Commands Across the Cluster (parallel shell)
23. DNS Configuration

[ UniCluster Documentation ]

# 25 Concepts and Terminology

This section describes concepts and terms that appear throughout the UniCluster documentation. We suggest you have a clear understanding of these items before continuing.

- Cluster

A Cluster is a collection of Nodes, or computers.

- Node

A node is a single computer within a cluster. There are two different types of nodes:

- ◊ Installer Node

There is one Installer Node in a cluster. It holds all provisioning and configuration data for all nodes in the cluster. Its purpose is to provision and deploy software to the other nodes (the Compute Nodes). You perform all provisioning and configuration on the Installer Node, which then "pushes" these changes out to the Compute Nodes. System administrators and users typically have direct network access to the Installer Node, which acts as a gateway between the users and the compute nodes. The Installer Node may also be referred to as the "Master" or "Main" node.

- ◊ Compute Nodes

Compute Nodes are the workers in the cluster; they run the deployed applications. They are typically connected to a single, common network. System administrators and users typically do NOT have direct access to this network. User access to a Compute Node is usually done through the gateway provided by the Installer Node.

- Node Group

A Node Group is a non-overlapping, logical group of nodes that are provisioned with the identical set of software. Except for the underlying hardware, which may be different within the Node Group, all nodes within a Node Group have the same operating system, installed software, network configuration, disk partitioning, startup and shutdown procedures, etc. The purpose of a Node Group is to define the software configuration once, and then quickly replicate, or deploy, that configuration across a large group of nodes.

See the "ngedit" command.

- Compute Network

To manage the cluster, deploy software, etc., the Installer Node must be able to reach each Compute Node. The Compute Nodes may also need to communicate with each other. In the simplest configuration, all nodes in a cluster (the installer node and the compute nodes) connect to a single network – the Compute Network. More complicated configurations might have a separate Compute Network for each Node Group. We strongly suggest that the Compute Network be a dedicated network segment, separate from any user or desktop traffic.

- Public Network

System administrators and users access the cluster through the gateway provided by the Installer Node. Typically, the Installer Node has two network interfaces: 1) connects to the Compute Network, and 2) connects to the Public Network for admin/user/desktop access.

- Repository

See "Yum Repository".

- Yum Repository

Yum stands for Yellow dog Updater, Modified and is the way a number of Linux distributions (including RedHat) allow users to install or update software. Univa UD hosts yum repositories so that UniCluster can connect to them and download new or updated software. Once software has been downloaded to the installer node, UniCluster 4.0 hosts yum repositories internally on the installer node for the compute nodes to update against. See [\[this\]](#) link for more information on how yum repositories work.

- Kit

A "kit" in UniCluster terms is a complete software package. If there are more than one chunks of software that make up a particular software package (such as SGE which has a master piece and a compute piece), then they are all included with the kit. Software is brought down to the installer node in the form of a kit. A kit contains "components" (see Components) and these components are what get deployed to nodes.

- Component

A "component" is a chunk of software that runs on one machine. If a software package has a master and a compute piece then there will be a master component and a compute component. Components must be selected individually for installation on specific node groups and this is done through ngedit.

- PXE Boot

PXE is a standard of booting up a computer where the computer will use a form of DHCP to request an initial kernel to boot from. This is what is used to provision nodes in UniCluster. It is sometimes necessary to change the boot order in the BIOS so that "Network Boot" or "PXE Boot" is the first item.

- Network Boot

See PXE Boot.

- CFM

CFM stands for Configuration File Management and is used to distribute configuration changes to the various node groups. This is generally not something that users or administrators need to interact with directly but there are instances where some interaction is necessary.

[ UniCluster Documentation ]

## 26 Getting Started

This document provides the instructions for downloading and installing the UniCluster software into your cluster. You will perform this work on a single node in the cluster, called the `Installer Node`. There is one `Installer Node` in a cluster. It holds all provisioning and configuration data for all nodes in the cluster. Its purpose is to provision and deploy software to the other nodes (the `Compute Nodes`). You perform all provisioning and configuration on the `Installer Node`, which then "pushes" these changes out to the `Compute Nodes`. System administrators and users typically have direct network access to the `Installer Node`, which acts as a gateway between the users and the compute nodes. The `Installer Node` may also be referred to as the "Master" or "Main" node.

### 26.1 Installer Node Requirements

- Installed Linux Operating System - choose one of these supported operating systems:

- ◊ RedHat Enterprise Linux 5, 32 or 64 bit.
- ◊ Centos 5, 32 or 64 bit.
- ◊ Oracle Enterprise Linux, 32 or 64 bit.

- 2 gigabytes RAM
- 10 gigabytes free disk space
- 1 network interface (2 recommended)
- The `yum-utils` package
- The `Development Tools` package group

#### 26.1.1 Recommendations

We strongly recommend configuring the `Installation Node` with two network interfaces:

1. an interface for connecting the installer node to the public network for admin/user/desktop access, and
2. an interface to reach the compute nodes on a dedicated network. Since DHCP will be running on this interface, it must have a static IP address.

### 26.2 UniCluster Installation Instructions

1. Create the provision network for your cluster. You will need to know the gateway IP address, the broadcast, and the netmask for this network.
2. Set up the network interface on the installation node from which compute nodes will boot. Use a static IP address; this will become the address from which the compute nodes will boot. The easiest way to modify the network is using the `setup` program.
3. Download <http://peaches.grid.org/unicluster/unicluster-repo-0.1-0.noarch.rpm>.
4. Install the RPM:

```
rpm --install unicluster-repo-0.1-0.noarch.rpm
```

5. Begin installing UniCluster:

```
yum install unicluster
```

6. Set shell environment:

```
source /etc/profile.d/unicluster.sh
```

7. Complete installing UniCluster:

```
/opt/tortuga/sbin/unicluster-setup
```

The `unicluster-setup` command begins by checking the network interfaces on the `Installer Node` to identify the one(s) to use for provisioning the `Compute Nodes`. Part of the provisioning process involves running a DHCP server on the `Installer Node` to respond to requests from the `Compute Nodes`. `Unicluster-setup` needs to know which interface(s) to run the server on. `Unicluster-setup` warns you if it appears that you risk running DHCP on a public network, and gives you a chance to stop; you probably do not want the `Installer Node` to run a DHCP server on any network(s) other than those dedicated to the `Compute Nodes`.

If `unicluster-setup` finds more than one network interface configured with static IP addresses, it will ask you which one(s) you want to use for provisioning.

Once the `unicluster-setup` command completes, you are ready to begin adding and provisioning `Compute Nodes`.

[ UniCluster Documentation ]

## 27 UniCluster Documentation

UniCluster 4.0, developed by Univa UD, integrates best of breed open source solutions to provide a complete cluster software stack. UniCluster simplifies the deployment, configuration and management of clusters, from installation and configuration, to job scheduling, monitoring and remote access.

- For information on available UniCluster add-on kits, professional support or UniCluster updates please visit <http://www.univaud.com/hpc/products/unicluster.php>
- For community support visit the UniCluster group on Grid.org at <http://groups.grid.org/content/unicluster>.

### 27.1 Documents

- **Concepts and Terminology**

This describes concepts and terms that appear throughout the documentation. We suggest you begin here, and have a clear understanding of these items before continuing.

- **Getting Started**

This describes how to download and install UniCluster software onto your cluster.

- **Administrator Guide**

This describes additional features and functions to configure and operate your cluster once you have downloaded and installed the UniCluster software.

- **User Guide**

This guide is for users of the cluster. It describes how to log into the cluster and run jobs.

- **Reference Guide**

The reference guide is useful for both administrators and users. It gives the details of all of the available commands.

Note: To view the currently installed software and kits you can return to the front page by entering the IP address or hostname of the main installer into your web browser.

# 28 UniCluster Express 4.0.0

1. REDIRECT Main Page