<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Documentation Overview</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Quickstart Guide</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Release Notes</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Supported Distributions and Features</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Cluster Overview and Terminology Guide</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5.1 Cluster Architecture Overview</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>5.2 Scyld ClusterWare Software Overview</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5.2.1 The ClusterWare Database</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5.2.2 Provisioning Compute Nodes</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>Installation &amp; Administrator Guide</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>6.1 Introduction</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>6.2 Required and Recommended Components</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>6.3 Initial Installation of Scyld ClusterWare</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>6.3.1 Download the ClusterWare install script and related files</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>6.3.2 Execute the ClusterWare install script</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>6.3.3 Configure additional cluster administrators</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>6.4 Common Additional Configuration and Software</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>6.4.1 Configure Hostname</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>6.4.2 Choosing An Alternate Database</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>6.4.2.1 Database Differences</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>6.4.2.2 Switching Between Databases</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>6.4.3 Couchbase auto-failover</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>6.4.4 Configure Authentication</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>6.4.5 Disable/Enable Chain Booting</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>6.4.6 Installing Optional ClusterWare Software</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>6.4.7 Job Schedulers</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>6.4.7.1 Slurm</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>6.4.7.2 PBS TORQUE</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>6.4.7.3 OpenPBS</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>6.4.8 Kubernetes</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>6.4.9 scyld-nss Name Service Switch (NSS) Tool</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>6.4.10 Firewall Configuration</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>6.4.11 Install OpenMPI, MPICH, and/or MVAPICH</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>6.4.12 Configure IP Forwarding</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td>6.4.13 Install Additional Tools</td>
<td>38</td>
</tr>
</tbody>
</table>
6.13.1.2 Join a ClusterWare head node using scyld-install
6.13.1.3 Join a non-ClusterWare server using curl
6.13.1.4 Join a ClusterWare head node using curl
6.13.1.5 After a Join
6.13.2 Removing a Joined Head Node
6.13.2.1 Configuring Support for Database Failover
6.13.2.2 Shared Storage and Peer Downloads
6.13.3 Booting With Multiple Head Nodes
6.13.4 Switching To Alternative ClusterWare Database
6.14 Managing Node Failures
6.14.1 Head Node Failure
6.14.2 Kickstart Failing
6.14.3 Head Node Filesystem Is 100% Full
6.14.3.1 Verify excessive storage is related to ClusterWare
6.14.3.2 Remove unnecessary objects from the ClusterWare database
6.14.3.3 Investigate InfluxDB retention of Telegraf data
6.14.3.4 Remove unnecessary PXEboot images, repos
6.14.3.5 Otherwise
6.14.4 Exceeding System Limit of Network Connections
6.14.5 etcd Database Exceeds Size Limit
6.14.6 Failing To Boot From Local Storage
6.14.7 IP Forwarding
6.14.8 Soft Power Control Failures
6.14.9 Head Nodes Disagree About Compute Node State
6.14.10 Finding Further Information
6.15 Managing Large Clusters
6.15.1 Improve scaling of node booting
6.16 Backup and Restore
6.16.1 Backup and Restore of ClusterWare
6.16.2 Backup and Restore of the Database
6.17 Updating Scyld ClusterWare
6.18 Troubleshooting ClusterWare
6.18.1 Failing PXE Network Boot
6.18.2 Kickstart Failing
6.18.3 Head Node Filesystem Is 100% Full
6.18.3.1 Verify excessive storage is related to ClusterWare
6.18.3.2 Remove unnecessary objects from the ClusterWare database
6.18.3.3 Investigate InfluxDB retention of Telegraf data
6.18.3.4 Remove unnecessary PXEboot images, repos
6.18.3.5 Otherwise
6.18.4 Exceeding System Limit of Network Connections
6.18.5 etcd Database Exceeds Size Limit
6.18.6 Failing To Boot From Local Storage
6.18.7 IP Forwarding
6.18.8 Soft Power Control Failures
6.18.9 Head Nodes Disagree About Compute Node State
6.18.10 Finding Further Information
6.18.11 Contacting Penguin Computing Support
6.19 IPMI
6.19.1 IPMITool
6.20 Services, Ports, Protocols
6.20.1 Apache
6.20.2 Couchbase
6.20.3 DHCP
6.20.4 etcd
6.20.5 iSCSI
6.20.6 OpenSSH
6.20.7 Telegraf / InfluxDB
6.20.8 TFTP

7 Reference Guide
7.1 Introduction
7.2 Important Files on Head Nodes
7.2.1 The ~/.scyldcw/ Folder
7.2.1.1 auth_tkt.cookie
7.2.1.2 logs/
7.2.1.3 workspace/
7.2.1.4 parse_failures/

7.2.1.2 parse_failures/
Scyld ClusterWare11 Documentation, Release 11.9.1

7.11.2 scyld-adminctl ................................................................. 119
7.11.3 scyld-attribctl ................................................................. 121
7.11.4 scyld-bootctl ................................................................. 122
7.11.5 scyld-cluster-conf .......................................................... 125
7.11.6 scyld-clusterctl ............................................................... 127
7.11.7 scyld-imgctl ................................................................. 131
7.11.8 scyld-install ................................................................. 132
7.11.9 scyld-kube ................................................................. 134
7.11.10 scyld-mkramfs ............................................................ 135
7.11.11 scyld-modimg .............................................................. 136
7.11.12 scyld-nodectl ............................................................... 139
7.11.13 scyld-nssctl ............................................................... 144
7.11.14 scyld-reports ............................................................. 145
7.11.15 scyld-sysinfo .............................................................. 146
7.11.16 scyld-tool-config ........................................................ 148

7.12 Scyld ClusterWare Maintenance Tools ........................................ 148
7.12.1 headctl ................................................................. 149
7.12.2 make-iso ................................................................. 150
7.12.3 managedb ............................................................... 151
7.12.4 take-snapshot ............................................................. 152

8 Appendices .............................................................................. 154
8.1 Appendix: Creating Local Repositories without Internet .................. 154
8.2 Appendix: Creating Arbitrary CentOS Images ................................. 155
8.3 Appendix: Creating Arbitrary RHEL Images .................................. 157
8.4 Appendix: Creating Ubuntu and Debian Images .............................. 159
8.5 Appendix: Converting CentOS 8 to Alternative Distro ..................... 160
8.6 Appendix: Using Ansible ...................................................... 160
8.7 Appendix: Using Kubernetes .................................................. 162
8.8 Appendix: Using Docker ....................................................... 167
8.9 Appendix: Using Singularity .................................................. 169
8.10 Appendix: Creating Diagnostic Test Images ................................. 170
8.11 Appendix: Booting From Local Storage Cache ............................... 171
8.12 Appendix: Validating ClusterWare ISOs .................................... 172
8.13 Appendix: Managing Zero-Touch Provisioning (ZTP) ..................... 173

9 Changelog .............................................................................. 175
9.1 Known Issues And Workarounds .............................................. 195

10 ClusterWare 6/7 vs. Current ClusterWare ...................................... 197

11 License Agreements .................................................................. 199
11.1 Scyld ClusterWare End-User License Agreement ......................... 199
11.2 Third-Party License Agreements ............................................. 202

12 Frequently Asked Questions (FAQ) ............................................ 210
12.1 Software Install/Update ....................................................... 210
12.2 Cluster Management ......................................................... 210
12.3 Manipulating Compute Node Images ....................................... 211
12.4 Issues with Interacting with Compute Nodes ............................... 211

13 Feedback .............................................................................. 212
CHAPTER ONE

DOCUMENTATION OVERVIEW

The Scyld ClusterWare documentation set:

- The Quickstart Guide contains a brief how-to for installing and configuring a basic functional cluster.
- The Release Notes is a summary of notable changes for the latest Scyld ClusterWare Release v11.9.1.
- The Supported Distributions and Features is a table summarizing the ClusterWare-supported distributions, security functionality, job schedulers, and OpenMPI-class middleware.
- The Cluster Overview and Terminology Guide is a general overview of a ClusterWare hardware and software architecture and terminology.
- The Installation & Administrator Guide describes in detail how to install, configure, use, maintain, and update the cluster.
- The Reference Guide describes in greater detail the commands to manage the cluster.
- The Appendices contains how-to examples of less common management topics.
- The Changelog contains the full ChangeLog of ClusterWare releases, plus a Known Issues And Workarounds list.
- ClusterWare 6/7 vs. Current ClusterWare is a brief guide for ClusterWare 6 and 7 administrators showing common ClusterWare 7 commands and current ClusterWare equivalents.
- The License Agreements contains specifics about software licenses for Scyld ClusterWare and ClusterWare-distributed 3rd-party packages.
- The Frequently Asked Questions (FAQ) contains quick cross-reference pointers into the documentation to answer some common questions.

The documentation is available in two formats: HTML and PDF. You can browse the latest documentation on the Penguin Computing Support Portal at https://www.penguincomputing.com/support/documentation, or find the documentation on any server on which the clusterware-docs RPM has been installed. The man pages are distributed in the clusterware-tools RPM.

Find the documentation PDF on a local server at /var/www/html/clusterware-docs.pdf or view the HTML at http://localhost/clusterware-docs. For a remote server cw11headnode, view the HTML at http://cw11headnode/clusterware-docs.
The following is a brief example of creating a minimal, yet functional, Scyld ClusterWare cluster. No attempt is made here to explain the full breadth and depth of ClusterWare, which is extensively discussed in the remainder of the documentation (see Documentation Overview). This Quickstart Guide assumes the reader is familiar with administering Red Hat RHEL or CentOS servers. For readers who are unfamiliar with clusters, see Cluster Architecture Overview.

Prerequisites for this minimal cluster:

- A minimum of two x86_64 servers, and preferably three: one becomes a ClusterWare head node and the remainder become ClusterWare compute nodes. This Quickstart example uses three servers.
- The head node can be a bare metal server, although there is greater flexibility if it is a virtual machine. It should have a minimum of 4GB of RAM and 16GB of storage, and it must have an Ethernet controller connected to a private cluster network to communicate with the compute node(s). These are minimal requirements. See Required and Recommended Components for recommendations for a production cluster.
- The head node must be running Red Hat RHEL or CentOS 7.6 (or newer) or an equivalent distribution (see Supported Distributions and Features). It must have access to a repo for that base distribution so that ClusterWare can yum install additional packages.
- If you do not already have a ClusterWare repo for ClusterWare packages, then the scyld-install installer prompts the user for an appropriate userid/password authentication and builds the ClusterWare repo in /etc/yum.repos.d/clusterware.repo. Typically access to the ClusterWare packages is through a second Ethernet controller connected to the "outside world", e.g., the Internet.
- The compute node(s) must have their BIOS configured to PXEboot by default, using either "Legacy" or "UEFI" mode. They should have a minimum of 4GB of RAM and one Ethernet controller that is also physically connected to the same private cluster network. See Required and Recommended Components for recommendations for a production cluster.

A ClusterWare cluster administrator needs root privileges. Common practice is to create non-root administrators and give them sudo capability. For example, create an administrator user admin1:

```
useradd admin1   # create the user
passwd admin1    # and give it a password

# Give "admin1" full root sudo privileges
echo "admin1 ALL=(root) NOPASSWD: ALL" >> /etc/sudoers

# Now execute as that user "admin1"
su - admin1
# And add SSH key pairs (defaulting to /home/admin1/.ssh/id_rsa)
ssh-keygen
```

Create a cluster configuration text file that names the interface to the private cluster network, the IP address on that network for the first compute node, and a list of MAC addresses to use as compute nodes. For example:
Now install the `clusterware-installer` package using the ClusterWare `/etc/yum.repos.d/clusterware.repo` file:

```
sudo yum install clusterware-installer
```

That package contains the `scyld-install` script that you will execute to install the software and create the `DefaultImage`, which consists of a basic compute node pxeboot image, and the `DefaultBoot` config file, which references that `DefaultImage` and contains various boot-time information such as a kernel commandline to pass to a booting node.

For a simple installation:

```
# Reminder: you should be executing as user "admin1"
scyld-install --config /tmp/cluster-conf
```

By default the `DefaultImage` contains a kernel and rootfs software from the same base distribution installed on the head node, although if the head node executes RHEL8, then no `DefaultImage` and `DefaultBoot` are created.

Alternatively, for more flexibility (especially with a RHEL8 head node), execute the installer with an additional option that identifies the base distribution to be used for the `DefaultImage`:

```
scyld-install --config /tmp/cluster-conf --os-iso <ISO-file>
```

where `<ISO-file>` is either a pathname to an ISO file or a URL of an ISO file of a specific base distribution release, e.g., `--os-iso rhel-8.5-x86_64-dvd.iso`. That ISO can match the head node's base distribution or can be any distribution supported by Penguin Computing (see Supported Distributions and Features).

Now you have a basic 2-node cluster that should PXEboot compute nodes. The installer has created a `DefaultImage` that contains basic compute node software and a `DefaultBoot` config file for booting that image, and has initialized every node to PXEboot using the `DefaultBoot`. Validate your current setup by rebooting both compute nodes, and check the status of the nodes as they boot and connect to the head node:

```
scyld-nodectl status --refresh
# Use ctrl-c to exit this display
```

which initially shows:

```
Node status          [ date & time ]
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
  n[0-1] new
```

for the nodes `n0` and `n1`, and automatically updates as each node's status changes from `booting` to `up`. The per-node transition from `new` to `booting` consumes a minute or more doing hardware initialization, PXEboot provisioning, and early software init. The transition from `booting` to `up` consumes another minute or more. If the nodes do not boot, then see Failing PXE Network Boot.

You can view information about the `up` nodes by executing:

```
scyld-nodectl ls -L
# which is shorthand for `scyld-nodectl list --long-long`
```

(continues on next page)
scyld-nodectl status -L

Now enhance the functionality of the compute node software by installing the Slurm job scheduler and an OpenMPI software stack into the image that PXEboots. Best practice is to retain the original DefaultImage and DefaultBoot as a pristine starting point for future additional software enhancements, so copy these Default objects and modify just the copies:

```
scyld-imgctl -i DefaultImage clone name=NewImage
scyld-bootctl -i DefaultBoot clone name=NewBoot
# The NewBoot clone is initially associated with the DefaultImage,
# so change that:
scyld-bootctl -i NewBoot update image=NewImage
# Instruct all compute nodes to use "NewBoot" (instead of "DefaultBoot"):
scyld-nodectl --all set _boot_config=NewBoot
```

Add the head node to /etc/hosts and then restart the clusterware-dnsmasq service. Suppose the head node's private cluster network IP address is "10.54.0.60":

```
echo "10.54.0.60 $(hostname)" | sudo tee -a /etc/hosts
sudo systemctl restart clusterware-dnsmasq
```

Now install and configure Slurm, which is one of the ClusterWare-supported job schedulers. The scyld-install installer has disabled the ClusterWare repo (for an explanation, see Installing Optional ClusterWare Software) so we must explicitly enable the repo:

```
sudo yum install slurm-scyld --enablerepo=scyld*
```

# Perform the Slurm initialization
slurm-scyld.setup init

# Add Slurm client software to the NewImage
slurm-scyld.setup update-image NewImage

# Reboot the nodes and view their status as they boot
scyld-nodectl --all reboot
scyld-nodectl status --refresh
# And ctrl-c when both rebooting nodes are again "up"

# Check the job scheduler status
slurm-scyld.setup status
# If the Slurm daemon and munge are not both executing, then:
slurm-scyld.setup cluster-restart
# And check status again
slurm-scyld.setup status
sinfo

Configure the cluster to support OpenMPI multi-threaded communication between compute nodes using the ssh transport mechanism, which requires user uid/gid and passphrase-less key-based access. For this Quickstart Guide we will continue to use admin1 as the user. Add admin1's authentication to the NewImage:

```
/opt/scyld/clusterware-tools/bin/sync-uids \
   -i NewImage --create-homes \
```

(continues on next page)
Install OpenMPI 4.0 into `NewImage` using `chroot`:

```
scyld-modimg -i NewImage --chroot --no-discard --overwrite --upload
  # Inside the chroot you are executing as user root
  yum install openmpi4.0

  # Set up access to Slurm and OpenMPI for "admin1"
  echo "module load slurm" >> /home/admin1/.bashrc
  echo "module load openmpi" >> /home/admin1/.bashrc

  # Build an example OpenMPI application
  cd /opt/scyld/openmpi/*/gnu/examples
  yum install make
  module load openmpi
  make hello_c
  # For simplicity, copy the executable to /home/admin1/hello_c
  cp hello_c /home/admin1/hello_c

  exit # from the chroot
```

Reboot the nodes with the updated `NewImage`:

```
scyld-nodectl --all reboot
  # Observe the node status changes
  scyld-nodectl status --refresh
  # And ctrl-c when both rebooting nodes are again "up"
```

From the head node, verify this by using Slurm to execute programs on the compute nodes:

```
module load slurm

  # Verify basic Slurm functionality by executing a simple command on each node
  srun -N 2 hostname

  # Use Slurm to execute one "Hello World" program on each of the two nodes
  srun -N 2 hello_c
```
Scyld ClusterWare Release v11.9.1 is the latest update to Scyld ClusterWare.

For the most up-to-date product documentation and other helpful information, visit the Penguin Computing Support Portal. In particular, the most recent ClusterWare documentation can be found on the Penguin Computing Support Portal at https://www.penguincomputing.com/support/documentation. The most recent version will accurately reflect the current state of the ClusterWare yum repository of RPMs that you are about to install.

Release Notes for Scyld ClusterWare:

- Assorted bug fixes on top of 11.9.0. See Changelog for details. Release notes from 11.9.0 continue below.
- As of RHEL 8.5 Red Hat has provided a mechanism to apply a Security Technical Implementation Guide (STIG) at the end of the install process. This ClusterWare release can be installed onto a head node with that STIG applied and includes an example kickstart file (basic-stig.ks) as part of the clusterware package. In order to kickstart an OS installation using the STIG please include the following lines in the kickstart file:

  ```
  %addon org_fedora_oscap
  content-type = scap-security-guide
  profile = xccdf_org.ssgproject.content_profile_stig
  %end
  ```

- Prior to this release ClusterWare could encrypt the image cache (specified by the _disk_cache node attribute) using a clear-text password provided as part of the attribute. Although this does provide encryption-at-rest, any administrator would be able see that password. Now setting a password of **TPM** will instruct scripts in the initramfs to generate a random password and store that password in the booting node’s Trusted Platform Module (TPM). This password is unique to the machine and can be bound to specific Platform Configuration Register (PCR) values and protected by the TPM owner password.

- This release introduces initial support for serving administrator-provided configuration scripts to Cumulus switches via ZTP. Future improvements will streamline a switch configuration backup and restore process.

- A new optional package, **scyld-nss**, provides a technology preview of a Name Service Switch (NSS) library that computes node IP addresses based on a configuration file mapping node names to IP ranges. Starting in ClusterWare 12 this package will be installed by default on head nodes and the scyld-install script will no longer attempt to manipulate the head node DNS configuration at install time.

See Changelog for a full history of ClusterWare releases, and Known Issues And Workarounds for a summary of notable known current issues.
SUPPORTED DISTRIBUTIONS AND FEATURES

Unless otherwise noted, ClusterWare is principally supported for the x86_64 architecture.

It has been additionally tested on the aarch64 architecture using Rocky 8.5. If you are interested in a cluster using that architecture or a mix of architectures, please contact Penguin Computing.

Entries marked as probable are RHEL clones and probably work, although they are not explicitly tested by Penguin Computing.

<table>
<thead>
<tr>
<th>Distro/CentOS</th>
<th>Version</th>
<th>Head Nodes</th>
<th>Compute Nodes</th>
<th>Node Image</th>
<th>Kickstart</th>
<th>Local Install</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL/CentOS</td>
<td>6</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>RHEL/CentOS</td>
<td>7.0 - 7.5</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RHEL/CentOS</td>
<td>7.6 - 7.9</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>RHEL</td>
<td>8.0 - 8.6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CentOS</td>
<td>8.0 - 8.6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Rocky</td>
<td>8.4 - 8.6</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>CentOS Stream</td>
<td>8.1</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Oracle</td>
<td>7.9</td>
<td>possible</td>
<td>possible</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Oracle</td>
<td>8.3 - 8.6</td>
<td>possible</td>
<td>possible</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>AlmaLinux</td>
<td>8.4 - 8.6</td>
<td>possible</td>
<td>possible</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>OpenSUSE</td>
<td>Leap 15.2 - 15.3</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Ubuntu</td>
<td>16 - 20 LTS</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Debian</td>
<td>stable, testing</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Footnotes
[1] CentOS 8 can be converted to an alternative distro.
See "Appendix: Converting CentOS 8 to Alternative Distro".
[2] CentOS Stream 8 was confirmed supported as of June 7, 2022.

Entries marked both indicate that Penguin Computing tests and supports both SELinux Targeted and MLS policies.
### PENGUIN-VERIFIED SECURITY

<table>
<thead>
<tr>
<th>OS Distro</th>
<th>Version(s)</th>
<th>FIPS Mode</th>
<th>SELinux</th>
<th>FIPS Mode</th>
<th>SELinux</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL/CentOS</td>
<td>7.0 - 7.5</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>RHEL/CentOS</td>
<td>7.6 - 7.9</td>
<td>yes</td>
<td>both</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>RHEL</td>
<td>8.0 - 8.6</td>
<td>yes(^3)</td>
<td>both</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>CentOS</td>
<td>8.0 - 8.6'</td>
<td>yes(^3)</td>
<td>both</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>Rocky</td>
<td>8.4 - 8.6</td>
<td>yes</td>
<td>both</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>CentOS Stream</td>
<td>8(^e)</td>
<td>probable</td>
<td>probable</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>Oracle</td>
<td>7.9</td>
<td>probable</td>
<td>probable</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>Oracle</td>
<td>8.3 - 8.6</td>
<td>probable</td>
<td>probable</td>
<td>yes</td>
<td>both</td>
</tr>
<tr>
<td>AlmaLinux</td>
<td>8.4 - 8.6</td>
<td>probable</td>
<td>probable</td>
<td>yes</td>
<td>both</td>
</tr>
</tbody>
</table>

**Footnotes**

[1] CentOS 8 can be converted to an alternative distro. See "Appendix: Converting CentOS 8 to Alternative Distro".

[2] CentOS Stream 8 was confirmed supported as of June 7, 2022.

[3] RHEL/CentOS 8 when using the Couchbase database can only install with FIPS disabled.

### CLUSTERWARE-DISTRIBUTED SCHEDULERS

<table>
<thead>
<tr>
<th>OS Distro</th>
<th>PBS TORQUE</th>
<th>Slurm</th>
<th>OpenPBS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL/CentOS 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHEL/CentOS 7</td>
<td>6</td>
<td>18-21</td>
<td></td>
</tr>
<tr>
<td>RHEL/CentOS 8</td>
<td></td>
<td>20-21</td>
<td>20</td>
</tr>
</tbody>
</table>

### CLUSTERWARE-DISTRIBUTED MIDDLEWARE

<table>
<thead>
<tr>
<th>OS Distro</th>
<th>OpenMPI</th>
<th>MPICH</th>
<th>MVAPICH</th>
</tr>
</thead>
<tbody>
<tr>
<td>RHEL/CentOS 6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RHEL/CentOS 7</td>
<td>1, 2, 3, 4</td>
<td>3.3</td>
<td>2.3</td>
</tr>
<tr>
<td>RHEL/CentOS 8</td>
<td>3, 4</td>
<td></td>
<td>2.3</td>
</tr>
</tbody>
</table>
5.1 Cluster Architecture Overview

A minimal ClusterWare cluster consists of a *head node* and one or more *compute nodes*, all interconnected via a *private cluster network*. User applications generally execute on the compute nodes, are often multithreaded across multiple compute nodes, and are usually coordinated by a job scheduler.
The head node is responsible for provisioning compute nodes, beginning with responding to a compute node's DHCP request for an IP address, and then (depending upon the compute node's BIOS settings) the compute node either boots from its local storage, or the compute node makes PXEboot requests for the kernel, initrd, and root filesystem images.

A ClusterWare head node usually also functions as a server for:

- The distributed Key/Value Database (implemented using etcd or Couchbase), that is accessed through the REST API or via command line or graphical tools and is the repository for information such as:
  - The MAC address to IP address and node number mappings.
  - The locations of the storage for the kernel, initrd, and root filesystem PXEboot images.
  - Compute node attributes, basic hardware and status, and configuration details.
• The storage for the images themselves.
• The compute node status information, which can be visualized by shell commands or by graphical tools. This is implemented by default by the TICK stack: Telegraf, InfluxDB, Chronograf, Kapacitor.
• Optional network storage, e.g., an NFS server.

A more complex cluster can be High Availability (HA), consisting of multiple head nodes where each head node has access to the distributed database and shared image storage. In an active-active(-active....) relationship, each head node can manage any compute node, providing it with boot files and forwarding its status information into the shared database. Since no particular head node is specifically necessary to manage an individual compute node, any head node can take over responsibility for the compute nodes that were previously communicating with a now-failed head node.

Note: Some network protocols, e.g. iSCSI, do not easily handle this sort of handoff, and any clusters using these protocols may experience additional difficulties on head node failure.

A complex cluster can also employ separate servers for the network storage, the compute node status information, and the boot images storage. For example:
An even more complex cluster may employ high-performance networking, such as Infiniband, Omni-Path, or even 40GB/sec or faster Ethernet, in addition to the typical 1Gb/sec or 10Gb/sec Ethernet that commonly interconnects nodes on the private cluster network. This faster (and more expensive) network fabric typically interconnects the compute nodes and commonly also shared cluster-wide storage.

The head node(s) commonly also have IPMI access to each compute node’s Base Management Controller (BMC), which provides for command-line or programmatic access to the compute nodes at a more basic hardware level, allowing for remote control of power, forcing a reboot, viewing hardware state, and more.

Also common is the use of Scyld Cloud Manager (SCM) to manage user access and accounting. User administrators (distinguished from cluster administrators, who have sudo-access to powerful ClusterWare tools) connect to the cluster portal to create virtual login node(s). Cluster users ssh to login nodes to build or install applications, download data...
into the cluster's data storage, submit jobs to a job scheduler for execution on compute nodes, and upload results back to the user's home system.

Some complex clusters connect head nodes to the public Internet via a gateway, e.g., to allow a cluster administrator to use `yum` to install or update software from Internet-accessible websites. Other complex clusters provide no head node access to the Internet and keep software hosted on a cluster-internal mirror server, where the local cluster administrator has precise control over updates.

For example:
5.2 Scyld ClusterWare Software Overview

A Scyld ClusterWare head node expects to execute in a Red Hat RHEL or CentOS 7.6 to 8.4, Oracle Linux 7.9 to 8.4, or Rocky 8.4 environment.


ClusterWare provides the tools (commonly named with prefix scyld-) and services (e.g., the key-value database) for a cluster administrator to install, administer, and monitor a Beowulf-style cluster. A cluster administrator commonly employs a shell on a head node to perform these functions. ClusterWare additionally distributes packages for an administrator to install an optional job manager (e.g., Slurm, OpenPBS, TORQUE), Kubernetes, and several varieties of OpenMPI-family software stacks for user applications. The Installation & Administrator Guide describes this with much greater detail.

5.2.1 The ClusterWare Database

The ClusterWare database is stored as JSON content within a replicated document store distributed among the ClusterWare head nodes. This structure protects against the failure of any single head node. Although the system originally used the community edition of Couchbase as the distributed database, the internal API is implemented using pluggable modules and now supports etcd as the default database and Couchbase as an alternative. The module API is intended to present a consistent experience regardless of the backend database, although some details, such as failure modes, will differ.

The server side (head node) responses to specific steps in the PXE boot process are controlled by the cluster configuration stored as JSON documents (aka objects) in the database. The following sections will follow the order of the boot steps described above to explore the definition and use of these database objects.

Internally, database objects are identified by unique identifiers (UIDs). These UIDs are also used to identify objects in ClusterWare command line and GUI tools, although as these strings tend to be cumbersome, an administrator should also assign a name and an optional description to each object. Even when objects are listed by name, the UID is available in the uid field returned by the object query tools.

Database objects generally consist of name-value pairs arranged in a JSON dictionary and referred to here as fields. These fields can be set via using the update argument of the appropriate scyld-* command line tools or by editing object details through the GUI. Field names are all lower case with underscores separating words. Not all fields on all objects will be editable, e.g. node names that are assigned based on the naming pool and node index.

Whenever a name-value pair is updated or added, a last_modified field in the mapping is also updated. These last_modified fields can be found scattered throughout the database objects.

5.2.2 Provisioning Compute Nodes

A principal responsibility of a head node is to provision compute nodes as they boot. A compute node's BIOS can be configured to boot from local storage, e.g., a harddrive, or to "PXEboot" by downloading the necessary images from a head node.

Each compute node is represented by a uniquely identified node object in the ClusterWare database. This object contains the basics of node configuration, including the node's index and the MAC address that is used to identify the node during the DHCP process. An administrator can also set an explicit IP address in the ip field. This IP address should be in the DHCP range configured during head node installation, although if none is specified, then a reasonable default will be selected based on the node index.

Each compute node is associated with a specific boot configuration, each stored in the ClusterWare database. A boot configuration ties together a kernel file, an initramfs file, and a cmdline, together with a reference to a root file system.
rootfs image. This rootfs is also known as a boot image, root image, or node image. A boot configuration also includes a configurable portion of the kernel command line that will be included in the iPXE boot script.

For a PXE boot, after the DHCP reply establishes the compute node’s IP address, the node requests a loader program, and the ClusterWare head node responds by default with the Open Source iPXE loader, and a configuration file that identifies the kernel and initramfs images to download, and a kernel command line to pass to the booting kernel.

This kernel executes and initializes itself, then launches the init user program (provided by dracut), which in turn executes various scripts to initialize networking and other hardware, and eventually executes a ClusterWare mount_rootfs script, which downloads the rootfs image and sets up the node’s root filesystem.

The mount_rootfs script may download and unpack a root filesystem image file, or alternatively may mount an iSCSI device or an image cached on a local harddrive, and then switch the node's root from the initramfs to this final root image. Other than when unpacking a root filesystem into RAM, images are shared and compute nodes are restricted to read-only access. In these cases compute nodes must use a writable overlay for modifiable portions of the file system. This is done toward the end of the mount_rootfs script via either the rwtab approach (for example, see https://www.redhat.com/archives/rhl-devel-list/2006-April/msg01045.html) or more commonly using an overlayfs (see https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt).
6.1 Introduction

The Cluster Overview and Terminology Guide describes the Scyld ClusterWare system architecture and design and basic terminology necessary to properly configure and administer a ClusterWare cluster.

This Scyld ClusterWare Installation & Administrator Guide is intended for use by Scyld ClusterWare administrators and advanced users. As is typical for any Linux-based system, the administrator must have root privileges (if only via sudo) to perform many of the administrative tasks described in this document.

This guide provides specific information about tools and methods for setting up and maintaining the cluster, the cluster boot process, ways to control cluster usage, methods for batching jobs and controlling the job queue, how load balancing is handled in the cluster, and optional tools that can be useful in administrating your cluster.

This guide is written with the assumption that the administrator has a background in a Unix or Linux operating environment; therefore, the document does not cover basic Linux system administration. If you do not have sufficient knowledge for using or administering a Linux system, we recommend that you first study other resources, either in print or online.

When appropriate, this document refers the reader to other parts of the Scyld documentation set for more detailed explanations for various topics, such as the Reference Guide, which provides greater details about commands, and various appendices.

6.2 Required and Recommended Components

Scyld ClusterWare head nodes are expected to use x86_64 processors running a Red Hat RHEL, CentOS, or similar distribution. See Supported Distributions and Features for specifics.

Important: ClusterWare head nodes currently require a Red Hat RHEL or CentOS 7.6 (or later) base distribution environment due to dependencies on newer libvirt and selinux packages. This requirement only applies to head nodes, not compute nodes.

Important: By design, ClusterWare compute nodes handle DHCP responses on the private cluster network (bootnet) by employing the base distribution's facilities, including NetworkManager. If your cluster installs a network file system or other software that disables this base distribution functionality, then dhclient or custom static IP addresses, and potentially additional workarounds, must be configured.
ClusterWare head nodes should ideally be "lightweight" for simplicity and contain only software that is needed for the local cluster configuration. Non-root users typically do not have direct access to head nodes and do not execute applications on head nodes.

Head node components for a production cluster:

- x86_64 processor(s) are required, with a minimum of four cores recommended.
- 8GB RAM (minimum) is recommended.
- 100GB storage (minimum) is recommended.

The largest storage consumption contains packed images, uploaded ISOs, et al. Its location is set in the file /opt/scyld/clusterware/conf/base.ini and defaults to /opt/scyld/clusterware/storage/.

The directory /opt/scyld/clusterware/git/cache/ consumes storage roughly the size of the git repos hosted by the system.

Other than the above storage/ and cache/, the directory /opt/scyld/ consumes roughly 300MB.

Each administrator's ~/.scyldcw/workspace/ directory contains unpacked images that have been downloaded by an administrator for modification or viewing.

- One Ethernet controller (required) that connects to the private cluster network which interconnects the head node(s) with all compute nodes.
- A second Ethernet controller (recommended) that connects a head node to the Internet.

Multiple Ethernet or other high-performance network controllers (e.g., Infiniband, Omni-Path) are common on the compute nodes, but do not need to be accessible by the head node(s).

We recommend employing virtual machines, hosted by "bare metal" hypervisors, for head nodes, login nodes, job scheduler servers, etc., for ease of management. Virtual machines are easy to resize and easy to migrate between hypervisors. See https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/virtualization_deployment_and_administration_guide/ for basic Red Hat documentation.

Note: A bare metal hypervisor host must contain the aggregated resources required by each hosted virtual server, and ideally the aggregated recommended resources, plus several additional CPUs/cores and RAM resources devoted to the hypervisor functionality itself.

Note: The nmcli connection add tool can be used to create network bridges and to slave physical interfaces to those newly created bridges. Once appropriate bridges exist, the virt-install command can attach the virtual interfaces to the bridges, so that the created virtual machines exist on the same networks as the physical interfaces on the hypervisor.

A High Availability ("HA") cluster requires a minimum of three "production" head nodes, each a virtual machine hosted on a different bare metal hypervisor. Even if an HA cluster is not required, we recommend a minimum of two head nodes - one functioning as the production head node, and the other as a development head node that can be used to test software updates and configuration changes prior to updating the production node to the validated final updates.

Compute nodes are generally bare metal servers for optimal performance. See Supported Distributions and Features for a list of supported distributions.

See Cluster Architecture Overview for more details.
6.3 Initial Installation of Scyld ClusterWare

The Scyld ClusterWare `scyld-install` script installs the necessary packages from the ClusterWare yum repositories, and installs dependency packages as needed from the base distribution (e.g., Red Hat RHEL or CentOS) yum repositories.

**Important:** Do not install ClusterWare as an upgrade to an existing ClusterWare 6 or 7 installation. Instead, install Scyld ClusterWare on a non-ClusterWare system that ideally is a virtual machine. (See Required and Recommended Components.)

**Important:** The head node(s) must use a Red Hat RHEL- or CentOS-equivalent base distribution release 7.6 or later environment, due to dependencies on newer `libvirt` and `selinux` packages.

**Note:** Clusters commonly employ multiple head nodes. The instructions in this section describe installing ClusterWare on the first head node. To later install ClusterWare on additional head nodes, see Managing Multiple Head Nodes.

`scyld-install` anticipates being potentially executed by a non-root user, so ensure that your userid can execute `sudo`. Additionally, if using `sudo` behind a proxy, then because `sudo` clears certain environment variables for security purposes, the cluster administrator should consider adding several lines to `/etc/sudoers`:

```plaintext
[Defaults]        env_keep += "HTTP_PROXY http_proxy"
[Defaults]        env_keep += "HTTPS_PROXY https_proxy"
[Defaults]        env_keep += "NO_PROXY no_proxy"
```

**Important:** Various commands that manipulate images execute as user root, thereby requiring that the commands internally use `sudo` and requiring that user root must have access to the administrator's workspace which contains the administrator's images. Typically the per-user workspace is `~/.scyldcw/workspace/`. If that directory is not accessible to the command executing as root, then another accessible directory can be employed, and the administrator can identify that alternative pathname by adding a `modimg.workspace` setting to `~/.scyldcw/settings.ini`.

**Important:** `scyld-install` uses the `yum` command to access Scyld ClusterWare and potentially various other repositories (e.g., Red Hat RHEL or CentOS) that by default normally reside on Internet websites. However, if the head node(s) do not have Internet access, then the required repositories must reside on local storage that is accessible by the head node(s). See Appendix: Creating Local Repositories without Internet.

6.3.1 Download the ClusterWare install script and related files

Most commonly, first download a ClusterWare yum repo configuration file that is already customized for your cluster, containing an appropriate `authentication token` granting access to the various ClusterWare yum repo directories:

- Click on the Assets tab, and then select a specific Asset Name.
- In the Asset Detail section, click on YUM Repo File, which downloads an asset-specific `clusterware.repo` file.
6.3.2 Execute the ClusterWare install script

If `/etc/yum.repos.d/clusterware.repo` exists, then `scyld-install`'s subsequent invocations of `yum` will employ that configuration file. If `/etc/yum.repos.d/clusterware.repo` does not exist, then `scyld-install` prompts the user for an appropriate authentication token and uses that to build a `/etc/yum.repos.d/clusterware.repo` that is customized to your cluster.

`sicyld-install` accepts an optional argument specifying a cluster configuration file that contains information necessary to set up the DHCP server. For example:

```
cat <<EOF > /tmp/cluster-conf
interface enp0s9 # names the private cluster interface
nodes 4 # max number of compute nodes
iprange 10.10.32.45 # starting IP address of node 0
node 08:00:27:f0:44:35 # node 0 MAC address
node 08:00:27:f0:44:45 # node 1 MAC address
node 08:00:27:f0:44:55 # node 2 MAC address
node 08:00:27:f0:44:65 # node 3 MAC address
EOF
```

where the syntax of this cluster configuration file is:

```
domain <DOMAIN_NAME>
    Optional. Defaults to "cluster.local".
interface <INTERFACE_NAME>
```

6.3. Initial Installation of Scyld ClusterWare
Optional. Specifies the name of head node’s interface to the private cluster network, although that can be determined from the specification of the <FIRST_IP> in the iprange line.

**nodes <MAX_COUNT>**

Optional. Specifies the max number of compute nodes, although that can be determined from the iprange if both the <FIRST_IP> and <LAST_IP> are present. The max will also adjust as-needed if and when additional nodes are defined. For example, see Node Creation with Known MAC address(es).

**iprange <FIRST_IP> [<LAST_IP>]**

Specifies the IP address of the first node (which defaults to n0) and optionally the IP address of the last node. The <LAST_IP> can be deduced from the <FIRST_IP> and the nodes <MAX_COUNT>. The <FIRST_IP> can include an optional netmask via a suffix of /<BIT_COUNT> (e.g., /24) or a mask (e.g., /255.255.255.0).

**<FIRST_INDEX> <FIRST_IP> [<LAST_IP>] [<via <FROM_IP>>] [gw <GATEWAY_IP>]**

This is a more elaborate specification of a range of IP addresses, and it is common when using DHCP relays or multiple subnets. <FIRST_INDEX> specifies that the first node in this range is node n<FIRST_INDEX> and is assigned IP address <FIRST_IP>; optionally specifies that the range of nodes make DHCP client requests that arrive on the interface that contains <FROM_IP>; optionally specifies that each DHCP’ing node be told to use <GATEWAY_IP> as their gateway, which otherwise defaults to the IP address (on the private cluster network) of the head node.

For example: 128 10.10.24.30/24 10.10.24.100 via 192.168.65.2 gw 10.10.24.254 defines a DHCP range of 71 addresses, the first starting with 10.10.24.30, and assigns the first node in the range as n128; watches for DHCP requests arriving on the interface containing 192.168.65.2; and tells these nodes to use 10.10.24.254 as the their gateway.

**node [<INDEX>] <MAC> [<MAC>]**

One compute node per line, and commonly consisting of multiple node lines, where each DHCP’ing node is recognized by its unique MAC address and is assigned an IP address using the configuration file specifications described above. Currently only the first <MAC> is used. An optional <INDEX> is the index number of the node that overrides the default of sequentially increasing node number indices and thereby creates a gap of unassigned indices. For example, a series of eight node lines without an <INDEX> that is followed by node 32 52:54:00:c4:f7:1e creates a gap of unassigned indices n8 to n31 and assigns this node as n32.

**Note:** ClusterWare yum repositories contain RPMs that duplicate various Red Hat EPEL RPMs, and these ClusterWare RPMs get installed or updated in preference to their EPEL equivalents, even if /etc/yum.repos.d/ contains an EPEL .conf file.

**Note:** ClusterWare employs userid/groupid 539 to simplify communication between the head node(s) and the backend shared storage where it stores node image files, kernels, and initramfs files. If the scyld-install script detects that this uid/gid is already in use by other software, then the script issues a warning and chooses an alternative new random uid/gid. The cluster administrator needs to set the appropriate permissions on that shared storage to allow all head nodes to read and write all files.

The ClusterWare database is stored as JSON content within a replicated document store distributed among the ClusterWare head nodes. This structure protects against the failure of any single head node. Although the system originally used the community edition of Couchbase as the distributed database, the internal API is implemented using pluggable modules and now supports etcd as the default distributed database and Couchbase as an alternative. The module API is intended to present a consistent experience regardless of the backend database but some details, such as failure modes, will differ.
During head node installation the cluster administrator can select the database using the DB_RPM environment variable. The current default is `clusterware-etcd`, although using a value of `clusterware-couchbase` will install the appropriate package and configuration for the Couchbase backend.

For example, using the `cluster-config` created above and installing the default etcd database:

```
scyld-install --config /tmp/cluster-conf
```

Or alternatively, choosing the alternative couchbase database:

```
DB_RPM=clusterware-couchbase scyld-install --config cluster.conf
```

The administrator can also switch between the available backend databases after the cluster is installed. See Choosing An Alternate Database for details.

By default scyld-install creates the DefaultImage that contains a kernel and rootfs software from the same base distribution installed on the head node, although if the head node executes RHEL8, then no DefaultImage and DefaultBoot are created.

Alternatively, for more flexibility (especially with a RHEL8 head node), execute the installer with an additional option that identifies the base distribution to be used for the DefaultImage:

```
scyld-install --config /tmp/cluster-conf --os-iso <ISO-file>
```

where `<ISO-file>` is either a pathname to an ISO file or a URL of an ISO file. That ISO can match the head node's distribution or can be any supported distribution.

`scyld-install` unpacks an embedded compressed payload and performs the following steps:

- Checks for a possible newer version of the `clusterware-installer` RPM. If one is found, then the script will update the local RPM installation and execute the newer scyld-install script with the same arguments. An optional argument `--skip-version-check` bypasses this check.
- An optional argument `--yum-repo /tmp/clusterware.repo` re-installs a yum repo file to `/etc/yum.repos.d/clusterware.repo`. This is unnecessary if `/etc/yum.repos.d/clusterware.repo` already exists and is adequate.
- Checks whether the `clusterware` RPM is installed.
- Confirms the system meets various minimum requirements.
- Installs the `clusterware` RPM and its supporting RPMs.
- Copies a customized Telegraf configuration file to `/etc/telegraf/telegraf.conf`
- Enables the tftpd service in `xinetd` for PXE booting.
- Randomizes assorted security-related values in `/opt/scyld/clusterware/conf/base.ini`
- Sets the current user account as a ClusterWare administrator in `/opt/scyld/clusterware/conf/base.ini`. If this is intended to be a production cluster, then the system administrator should create additional ClusterWare administrator accounts and clear this variable. For details on this and other security related settings, including adding ssh keys to compute nodes, please see the Installation & Administrator Guide section Securing the Cluster.
- Modifies `/etc/yum.repos.d/clusterware.repo` to change `enabled=1` to `enabled=0`. Subsequent executions of scyld-install to update ClusterWare will temporarily (and silently) re-enable the ClusterWare repo for the duration of that command. This is done to avoid inadvertent updates of ClusterWare packages if and when the clusterware administrator executes a more general `yum install` or `yum update` intending to add or update the base distribution packages.

Then scyld-install uses systemctl to enable and start firewalld, and opens ports for communication between head nodes as required by etcd (or Couchbase). See Services, Ports, Protocols for details.
Once the ports are open, scyld-install initializes the ClusterWare database and enables and starts the following services:

- **httpd**: The Apache HTTP daemon that runs the ClusterWare service and proxies Chronograf and Kapacitor.
- **xinetd**: Provides network access to tftp for PXE booting.
- **Telegraf**: Collects head node performance data and feeds into InfluxDB.
- **InfluxDB**: Stores node performance and status data for visualization in Chronograf.
- **Chronograf**: Displays the head node and compute node status data through a web interface.
- **Kapacitor**: The eventing software that works with Chronograf.

The script then:

- Opens ports in firewalld for public access to HTTP, HTTPS, TFTP, iSCSI, and incoming Telegraf UDP messages.
- Installs and configures the cluster administrator's clusterware-tools package (unless it was executed with the --no_tools option).
- Configures the cluster administrator's ~/.scyldcw/settings.ini to access the newly installed ClusterWare service using the scyld-tool-config tool.
- Creates an initial simple boot image DefaultImage, boot config DefaultBoot, and attributes DefaultAttribs using the scyld-add-boot-config tool.
- Loads the cluster configuration specified on the command line using the scyld-cluster-conf load command.
- Restarts the httpd service to apply the loaded cluster configuration.

**Important**: See the Node Images and Boot Configurations for details about how to modify existing boot images, create new boot images, and associate specific boot images and attributes with specific compute nodes. We strongly recommend not modifying or removing the initial DefaultImage, but rather cloning that basic image into a new image that gets modified further, or just creating new images from scratch.

**Important**: If you wish to ensure that the latest packages are installed in the image after the scyld-install, then execute `scyld-modimg -i DefaultImage --update --overwrite --upload`.

**Important**: See Common Additional Configuration and Software for additional optional cluster configuration procedures, e.g., installing and configuring a job scheduler, installing and configuring one of the MPI family software stacks.

**Important**: If this initial scyld-install does not complete successfully, or if you want to begin the installation anew, then when/if you re-run the script, you should cleanse the partial, potentially flawed installation by adding the --clear argument, e.g., `scyld-install --clear --config /tmp/cluster-conf`. If that still isn't sufficient, then `scyld-install --clear-all --config /tmp/cluster-conf` does a more complete clearing, then reinstalls all the ClusterWare packages.

Due to licensing restrictions, when running on a Red Hat RHEL system, the installer will still initially create a CentOS compute node image as the DefaultImage. If after this initial installation a cluster administrator wishes to instead
create compute node images based on RHEL, then use the `scyld-clusterctl repos` tool as described in Appendix: Creating Arbitrary RHEL Images, and create a new image (e.g., `DefaultRHELimage`) to use as a new default.

### 6.3.3 Configure additional cluster administrators

The ClusterWare administrator's command-line tools are found in the `clusterware-tools` package, which is installed by default on the head node by `scyld-install`. It can be additionally installed on any system that has HTTP (or HTTPS, see Securing the Cluster) access to a ClusterWare head node in the cluster.

To install these tools on a machine other than the head node, login to that other system, copy `/etc/yum.repos.d/clusterware.repo` from a head node to the same location on this system, then execute:

```
sudo yum install clusterware-tools
```

Once the tools are installed, each administrator must configure a connection to the ClusterWare service, which is controlled by variables in the user's `~/.scyldcw/settings.ini` file. The `scyld-tool-config` tool script is provided by the `clusterware-tools` package. The contents of the `settings.ini` file are discussed in the Reference Guide. Running that tool and answering the on-screen questions will generate a `settings.ini` file, although administrators of more advanced cluster configurations may need to manually add or edit additional variables.

Once the `settings.ini` is created, you can test your connection by running a simple node query:

```
scyld-nodectl ls
```

This query may complain at this time that no nodes exist or no nodes are selected, although such a complaint does verify that the requesting node can properly communicate with a head node database. However, if you see an error resembling the one below, check your `settings.ini` contents and your network configuration:

```
Failed to connect to the ClusterWare service. Please check that the service is running and your base_url is set correctly in /home/adminuser/.scyldcw/settings.ini or on the command line.
```

The connection URL and username can also be overridden for an individual program execution using the `--base-url` and `--user` options available for all `scyld-*` commands. The `settings.ini` file generated by `scyld-install` will contain a blank `client.authpass` variable. This is provided for convenience during installation, though for production clusters the system administrator will want to enforce authentication restrictions. See details in Securing the Cluster.

### 6.4 Common Additional Configuration and Software

Following a successful initial install or update of Scyld ClusterWare, or as local requirements of your cluster dictate, you may need to make one or more configuration changes.

#### 6.4.1 Configure Hostname

Verify that the head node hostname has been set as desired for permanent, unique identification across the network. In particular, ensure that the hostname is not `localhost` or `localhost.localdomain`. 

---

**6.4. Common Additional Configuration and Software**

29
6.4.2 Choosing An Alternate Database

Two backend databases are available: etcd and Couchbase. The current default is etcd. For a production cluster, the backend database should be chosen at the time of the first install, or simply allow the default database to prevail.

6.4.2.1 Database Differences

On head nodes with multiple IP addresses the current ClusterWare etcd implementation has no way to identify the correct network for communicating with other head nodes. By default the system will attempt to use the first non-local IP. Although this is adequate for single head clusters and simple multihread configurations, a cluster administrator setting up a multihead cluster should specify the correct IP. This is done by setting the `etcd.peer_url` option in the `/opt/scyld/clusterware/conf/base.ini` file. A correct peer URL on a head node with the IP address of 10.24.1.1 where the 10.24.1.0/24 network should be used for inter-head communications might look like:

```
etcd.peer_url = http://10.24.1.1:52380
```

If this value needs to be set or changed on an existing cluster, it should be updated on a single head node, then `managedb recover` run on that head node, and then other heads (re-)joined to the now correctly configured one. The `etcd.peer_url` setting should only be necessary on the first head as the proper network will be communicated to new heads during the join process.

Unlike Couchbase, the ClusterWare etcd implementation does not allow the second-to-last head node in a multihead cluster to leave or be ejected. Instead a cluster administrator can run the new `managedb recover` command on the remaining head node for the same effect. See Removing a Joined Head Node for details, and Managing Multiple Head Nodes for broader information about multiple headnode management.

**Important:** Prior to any manipulation of the distributed database, whether through `managedb recover`, joining head nodes to a cluster, removing head nodes from a cluster, or switching between database backends, the administrator is strongly encouraged to make a backup of the ClusterWare database using the `managedb` tool. See `managedb` in the Reference Guide.

The firewall requirements for etcd are much simpler, as only a single port needs to be opened between head nodes, whereas Couchbase requires six.

No etcd alternative to the Couchbase console exists, although the `etcdctl` command provides scriptable direct document querying and manipulation. ClusterWare provides a wrapped version of `etcdctl` located in the `/opt/scyld/clusterware-etcd/bin/` directory. The wrapper should be run as root and automatically applies the correct credentials and connects to the local etcd endpoint. Note that direct manipulation of database JSON documents should only be done when directed by Penguin support.

6.4.2.2 Switching Between Databases

The clusterware-provided `headctl` tool (found in `/opt/scyld/clusterware/bin/`) includes arguments to toggle between databases. We suggest taking a snapshot of the virtual machine before performing these operations, and preferably making a database backup with `managedb` as well.

For example, switching the database on a single head node cluster:

```
sudo /opt/scyld/clusterware/bin/headctl --use=<DATABASE>
```

where `<DATABASE>` is either `etcd` or `couchbase`, which performs a series of steps:

1. Install the `clusterware-<DATABASE>` package, if not already installed.
(2) Stop the clusterware service.

(3) Export the database to a temporary file.

(4) Toggle the base.ini plugins.database option.

(5) Purge any existing database.

(6) Load the exported database from the temporary file.

(7) Update the firewall for the appropriate <DATABASE>.

(8) Restart the clusterware service.

Once these steps complete, the head node will resume normal operations.

For a multihead cluster this should be performed on each head node in turn, first removing the head node from the joined cluster to make it independent, then using headctl as described above.

For example, to switch from couchbase to etcd:

```
sudo /opt/scyld/clusterware/bin/managedb leave
sudo /opt/scyld/clusterware/bin/headctl --use-etcd
```

Once all the now-independent head nodes are running the alternative database, you can pick one and join the others to it. You can confirm the head nodes are again working together by executing on each head node:

```
scyld-nodectl status
sudo /opt/scyld/clusterware/bin/managedb --heads
```

and verify that scyld-nodectl and managedb agree.

When you sure that everything is working as expected, then if switching from Couchbase to etcd, on each head node remove the clusterware-couchbase package and delete /opt/couchbase. If switching from etcd to Couchbase, then remove the clusterware-etcd package.

This ability to toggle between databases is intended for administrators interested in experimenting with the alternate database or when troubleshooting problems. For production clusters the database should be selected at installation time and currently defaults to etcd.

### 6.4.3 Couchbase auto-failover

After the cluster administrator finishes configuring a cluster containing multiple head nodes and Couchbase, then you should enable auto-failover. See the Configuring Support for Database Failover for details.

### 6.4.4 Configure Authentication

ClusterWare administrator authentication is designed to easily integrate with already deployed authentication systems via PAM. By default cluster administrators are authenticated through the pam_authenticator tool that in turn uses the PAM configuration found in /etc/pam.d/cw_check_user. In this configuration, administrators can authenticate using their operating system password as long as they have been added to the ClusterWare system using the scyld-adminctl command. For example, to add username "admin1":

```
scyld-adminctl create name=admin1
```

If a ClusterWare administrator is running commands from a system account on the head node by the same name (i.e. ClusterWare administrator fred is also head node user fred), the system will confirm their identity via a Unix socket based protocol. Enabled by default, this mechanism allows the scyld tools to connect to a local socket to securely set
a dynamically generated one-time password that is then accepted during their next authentication attempt. This takes place transparently, allowing the administrator to run commands without providing their password. The client code also caches an authentication cookie in the user's `.scyldcw/auth_tkt.cookie` for subsequent authentication requests.

Managing cluster user accounts is generally outside the scope of ClusterWare and should be handled by configuring the compute node images appropriately for your environment. In large organizations this usually means connecting to Active Directory, LDAP, or any other mechanism supported by your chosen compute node operating system. In simpler environments where no external source of user identification is available or it is not accessible, ClusterWare provides a `sync-u ids` tool. This program can be found in the `/opt/scyld/clusterware-tools/bin` directory and can be used to push local user accounts and groups either to compute nodes or into a specified image. For example:

```
# push uids and their primary uid-specific groups:
sync-u ids --users admin1,tester --image SlurmImage

# push uid with an additional group:
sync-u ids --users admin1 --groups admins --image SlurmImage
```

The above pushes the users and groups into the compute node image for persistence across reboots. Then either reboot the node(s) to see these changes, or push the IDs into running nodes with:

```
sync-u ids --users admin1,tester --nodes n[1-10]
```

The tool generates a shell script that is then executed on the compute nodes or within the image chroot to replicate the user and group identifiers on the target system. This tool can also be used to push ssh keys into the `authorized_keys` files for a user onto booted compute nodes or into a specified image. Please see the tool's `--help` output for more details and additional functionality, such as removing users or groups, and controlling whether home directories are created for injected user accounts.

### 6.4.5 Disable/Enable Chain Booting

The default ClusterWare behavior is to perform *chain booting* for more efficient concurrency for servicing a flood of PXEbooting nodes that are requesting their large `rootfs` file. Without chain booting, the head node(s) serve the `rootfs` file for all PXEbooting nodes and thus become a likely bottleneck when hundreds of nodes are concurrently requesting their file. With chain booting, the head node(s) serve the `rootfs` files to the first compute node requesters, then those provisioned compute nodes offer to serve as a temporary `rootfs` file server for other requesters.

In the event that the cluster administrator wishes to disable chain booting, then the cluster administrator executing as user `root` should edit the file `/opt/scyld/clusterware/conf/base.ini` to add the line:

```
chaining.enable = False
```

To reenable chain booting, either change that `False` to `True`, or simply comment-out that `chaining.enable` line to revert back to the default enabled state.

### 6.4.6 Installing Optional ClusterWare Software

`scyld-install` installs and updates the basic ClusterWare software. Additional software packages are available in the ClusterWare repository.

`scyld-install` manipulates the `/etc/yum.repos.d/clusterware.repo` file to automatically enable the `scyld` repos when the tool executes and disable the repos when finished. This is done to avoid inadvertent updating of ClusterWare packages when executing a simple `yum update`.
Note: If the cluster administrator has created multiple /etc/yum.repos.d/*.repo files that specify repos containing ClusterWare RPMs, then this protection against inadvertent updating is performed only for /etc/yum.repos.d/clusterware.repo, not for those additional repo files.

Accordingly, the --enablerepo=scyld* argument is required when using yum for listing, installing, and updating these optional ClusterWare packages on a head node. For example, these optional installable software packages can be viewed using yum list --enablerepo=scyld* | grep scyld. After installation, any available updates can be viewed using yum check-update --enablerepo=scyld* | grep scyld.

Specific install and configuration instructions for various of these packages, e.g., job managers and OpenMPI middleware, are detailed in this chapter.

### 6.4.7 Job Schedulers

The default Scyld ClusterWare installation for RHEL/CentOS 7 includes support for the optional job scheduler packages Slurm and PBS TORQUE, and for RHEL/CentOS 8 includes support for the optional packages Slurm and OpenPBS. These optional packages can coexist on a scheduler server, which may or may not be a ClusterWare head node. However, if job schedulers are installed on the same server, then only one at a time should be enabled and executing on that given server.

All nodes in the job scheduler cluster must be able to resolve hostnames of all other nodes as well as the scheduler server hostname. ClusterWare provides a DNS server in the clusterware-dnsmasq package, as discussed in Node Name Resolution. This dnsmasq will resolve all compute node hostnames, and the job scheduler's hostname should be added to /etc/hosts on the head node(s) in order to be resolved by dnsmasq. Whenever /etc/hosts is edited, please restart the clusterware-dnsmasq service with:

```
sudo systemctl restart clusterware-dnsmasq
```

Installing and configuring a job scheduler requires making changes to the compute node software. When using image-based compute nodes, we suggest first cloning the DefaultImage or creating a new image, leaving untouched the DefaultImage as a basic known-functional pristine image.

For example, to set up nodes n0 through n3, you might first do:

```
scyld-imgctl -i DefaultImage clone name=jobschedImage
scyld-bootctl -i DefaultBoot clone name=jobschedBoot image=jobschedImage
scyld-nodectl -i n[0-3] set _boot_config=jobschedBoot
```

When these nodes reboot after all the setup steps are complete, they will use the jobschedBoot and jobschedImage.

See https://slurm.schedmd.com/rosetta.pdf for a discussion of the differences between PBS TORQUE and Slurm. See https://slurm.schedmd.com/faq.html#torque for useful information about how to transition from OpenPBS or PBS TORQUE to Slurm.

The following sections describe the installation and configuration of each job scheduler type.
6.4.7.1 Slurm

See Job Schedulers for general job scheduler information and configuration guidelines. See https://slurm.schedmd.com for Slurm documentation.

First install Slurm software on the job scheduler server:

```
sudo yum install slurm-scyld --enablerepo=scyld*
```

**Important:** For RHEL/CentOS 8, install Slurm with an additional argument:

```
sudo yum install slurm-scyld --enablerepo=scyld* --enablerepo=PowerTools
```

Now use a helper script slurm-scyld.setup to complete the initialization and setup the job scheduler and config file in the compute node image(s).

**Note:** The slurm-scyld.setup script performs the init, reconfigure, and update-nodes actions (described below) by default against all up nodes. Those actions optionally accept a node-specific argument using the syntax [--ids|-i <NODES>] or a group-specific argument using [--ids|-i %<GROUP>]. See Attribute Groups and Dynamic Groups for details.

```
slurm-scyld.setup init # default to all 'up' nodes
slurm-scyld.setup update-image slurmImage # for permanence in the image
```

Reboot the compute nodes to bring them into active management by Slurm. Check the Slurm status:

```
slurm-scyld.setup status

# If the Slurm daemon and munge are not both executing, then:
slurm-scyld.setup cluster-restart

# And check the status again
```

This cluster-restart is a manual one-time setup that doesn't affect the slurmImage. The update-image is necessary for persistence across compute node reboots.

Generate new slurm-specific config files with:

```
slurm-scyld.setup reconfigure # default to all 'up' nodes
```

Add nodes by executing:

```
slurm-scyld.setup update-nodes # default to all 'up' nodes
```

or add or remove nodes by directly editing the /etc/slurm/slurm.conf config file. Any such changes must be added to slurmImage by reexecuting:

```
slurm-scyld.setup update-image slurmImage
```

and then either reboot all the compute nodes with that updated image, or additional execute:

```
slurm-scyld.setup cluster-restart
```

to manually push the changes to the up nodes without requiring a reboot.
Inject users into the compute node image using the `sync-uids` script. The administrator can inject all users, or a selected list of users, or a single user. For example, inject the single user `janedoe`:

```
/opt/scyld/clusterware-tools/bin/sync-uids \
    -i slurmImage --create-homes \
    --users janedoe --sync-key janedoe=/home/janedoe/.ssh/id_rsa.pub
```

See `Configure Authentication` and `/opt/scyld/clusterware-tools/bin/sync-uids -h` for details.

To view the Slurm status on the server and compute nodes:

```
slurm-scyld.setup status
```

The Slurm service can also be started and stopped cluster-wide with:

```
slurm-scyld.setup cluster-stop
slurm-scyld.setup cluster-start
```

Slurm executable commands and libraries are installed in `/opt/scyld/slurm/`. Each Slurm user must set up the `PATH` and `LD_LIBRARY_PATH` environment variables to properly access the Slurm commands. This is done automatically for users who login when Slurm is running via the `/etc/profile.d/scyld.slurm.sh` script. Alternatively, each Slurm user can manually execute `module load slurm` or can add that command line to (for example) the user's `~/.bash_profile` or `~/.bashrc`.

### 6.4.7.2 PBS TORQUE

PBS TORQUE is only available for RHEL/CentOS 7 clusters. See `Job Schedulers` for general job scheduler information and configuration guidelines. See [https://www.adaptivecomputing.com/support/documentation-index/torque-resource-manager-documentation](https://www.adaptivecomputing.com/support/documentation-index/torque-resource-manager-documentation) for PBS TORQUE documentation.

First install PBS TORQUE software on the job scheduler server:

```
sudo yum install torque-scyld --enablerepo=scyld*
```

Now use a helper script `torque-scyld.setup` to complete the initialization and setup the job scheduler and config file in the compute node image(s).

**Note:** The `torque-scyld.setup` script performs the `init`, `reconfigure`, and `update-nodes` actions (described below) by default against all `up` nodes. Those actions optionally accept a node-specific argument using the syntax `[--ids|-i <NODES>]` or a group-specific argument using `[--ids|-i %<GROUP>]`. See [Attribute Groups and Dynamic Groups](#) for details.

```
torque-scyld.setup init # default to all 'up' nodes
torque-scyld.setup update-image torqueImage # for permanence in the image
```

Reboot the compute nodes to bring them into active management by TORQUE. Check the TORQUE status:

```
torque-scyld.setup status
```

# If the TORQUE daemon is not executing, then:
```
torque-scyld.setup cluster-restart
```

# And check the status again
This `cluster-restart` is a manual one-time setup that doesn't affect the `torqueImage`. The `update-image` is necessary for persistence across compute node reboots.

Generate new torque-specific config files with:

```
torque-scyld.setup reconfigure  # default to all 'up' nodes
```

Add nodes by executing:

```
torque-scyld.setup update-nodes  # default to all 'up' nodes
```

or add or remove nodes by directly editing the `/var/spool/torque/server_priv/nodes` config file. Any such changes must be added to `torqueImage` by reexecuting:

```
torque-scyld.setup update-image slurmImage
```

and then either reboot all the compute nodes with that updated image, or additional execute:

```
torque-scyld.setup cluster-restart
```

to manually push the changes to the `up` nodes without requiring a reboot.

Inject users into the compute node image using the `sync-uids` script. The administrator can inject all users, or a selected list of users, or a single user. For example, inject the single user `janedoe`:

```
/opt/scyld/clusterware-tools/bin/sync-uids \
  -i torqueImage --create-homes \ 
  --users janedoe --sync-key janedoe=/home/janedoe/.ssh/id_rsa.pub
```

See `Configure Authentication` and `/opt/scyld/clusterware-tools/bin/sync-uids -h` for details.

To view the TORQUE status on the server and compute nodes:

```
torque-scyld.setup status
```

The TORQUE service can also be started and stopped cluster-wide with:

```
torque-scyld.setup cluster-stop
```
```
torque-scyld.setup cluster-start
```

TORQUE executable commands are installed in `/usr/sbin/` and `/usr/bin/`, TORQUE libraries are installed in `/usr/lib64/`, and are therefore accessible by the default search rules.

### 6.4.7.3 OpenPBS

OpenPBS is only available for RHEL/CentOS 8 clusters.

See `Job Schedulers` for general job scheduler information and configuration guidelines. See [https://www.openpbs.org](https://www.openpbs.org) for OpenPBS documentation.

First install OpenPBS software on the job scheduler server:

```
sudo yum install openpbs-scyld --enablerepo=scyld*
```

Use a helper script to complete the initialization and setup the job scheduler and config file in the compute node image(s).
**Note:** The `openpbs-scyld.setup` script performs the `init`, `reconfigure`, and `update-nodes` actions (described below) by default against all *up* nodes. Those actions optionally accept a node-specific argument using the syntax `[--ids|-i <NODES>]` or a group-specific argument using `[--ids|-i %<GROUP>]`. See *Attribute Groups and Dynamic Groups* for details.

```
openpbs-scyld.setup init  # default to all 'up' nodes
openpbs-scyld.setup update-image openpbsImage # for permanence in the image
```

Reboot the compute nodes to bring them into active management by OpenPBS. Check the OpenPBS status:

```
openpbs-scyld.setup status

# If the OpenPBS daemon is not executing, then:
openpbs-scyld.setup cluster-restart

# And check the status again
```

This `cluster-restart` is a manual one-time setup that doesn’t affect the `openpbsImage`. The `update-image` is necessary for persistence across compute node reboots.

Generate new `openpbs`-specific config files with:

```
openpbs-scyld.setup reconfigure  # default to all 'up' nodes
```

Add nodes by executing:

```
openpbs-scyld.setup update-nodes  # default to all 'up' nodes
```

or add or remove nodes by executing `qmgr`.

Any such changes must be added to `openpbsImage` by reexecuting:

```
openpbs-scyld.setup update-image openpbsImage
```

and then either reboot all the compute nodes with that updated image, or additional execute:

```
openpbs-scyld.setup cluster-restart
```

to manually push the changes to the *up* nodes without requiring a reboot.

Inject users into the compute node image using the `sync-uids` script. The administrator can inject all users, or a selected list of users, or a single user. For example, inject the single user `janedoe`:

```
/opt/scyld/clusterware-tools/bin/sync-uids \
   -i openpbsImage --create-homes \ 
   --users janedoe --sync-key janedoe=/home/janedoe/.ssh/id_rsa.pub
```

See *Configure Authentication* and `/opt/scyld/clusterware-tools/bin/sync-uids -h` for details.

To view the OpenPBS status on the server and compute nodes:

```
openpbs-scyld.setup status
```

The OpenPBS service can also be started and stopped cluster-wide with:

```
```
openpbs-scyld.setup cluster-stop
openpbs-scyld.setup cluster-start

OpenPBS executable commands and libraries are installed in /opt/scyld/openpbs/. Each OpenPBS user must set up the PATH and LD_LIBRARY_PATH environment variables to properly access the OpenPBS commands. This is done automatically for users who login when OpenPBS is running via the /etc/profile.d/scyld.openpbs.sh script. Alternatively, each OpenPBS user can manually execute module load openpbs or can add that command line to (for example) the user's ~/.bash_profile or ~/.bashrc.

6.4.8 Kubernetes

ClusterWare administrators wanting to use Kubernetes as a container orchestration layer across their cluster can either choose to install Kubernetes manually following directions found online, or use scripts provided by the clusterware-kubeadm package. To use these scripts first install the clusterware-kubeadm package on a server that is a Scyld ClusterWare head node, a locally installed ClusterWare compute node, or a separate non-ClusterWare server. Installing the control plane on a RAM-booted or otherwise ephemeral compute node is discouraged.

The provided scripts are based on the kubeadm tool and inherit both the benefits and limitations of that tool. If you prefer to use a different tool to install Kubernetes please follow appropriate directions available online from your chosen Kubernetes provider. The clusterware-kubeadm package is mandatory, and the clusterware-tools package is recommended:

```
sudo yum --enablerepo=scyld* install clusterware-kubeadm clusterware-tools
```

**Important:** For a server to function as a Kubernetes control plane, SELinux must be disabled (verify with getenforce) and swap must be turned off (verify with swapon -s, disable with swapoff -a -v).

After installing the software, as a cluster administrator execute the scyld-kube tool to initialize the Kubernetes control plane. To initialize on a local server:

```
scyld-kube --init
```

Or to initialize on an existing booted ClusterWare compute node (e.g., node n0):

```
scyld-kube --init -i n0
```

Note that a ClusterWare cluster can have multiple control planes and can use multiple control planes in a Kubernetes High Availability (HA) configuration. See Appendix: Using Kubernetes for detailed examples.

You can validate this initialization by executing:

```
kubectl get nodes
```

which should show the newly initialized control plane server.

Next, join one or more booted ClusterWare nodes (e.g., nodes n[1-3]) as worker nodes of this Kubernetes cluster. The full command syntax accomplishes this by explicitly identifying the control plane node by its IP address:

```
scyld-kube -i n[1-3] --join --cluster <CONTROL_PLANE_IP_ADDR>
```

However, if the control plane node is a ClusterWare compute node, then the scyld-kube --init process defined Kube-specific attributes and a simpler syntax suffices:
The simpler join command can find the control plane node without needing to be told its IP address as long as there is only one compute node that functioning as a Kubernetes control plane.

Note that `scyld-kube --join` also accepts admin-defined group names, e.g., for a collection of nodes joined to the `kube_workers` group:

```
scyld-kube -i %kube_workers --join --cluster <CONTROL_PLANE_IP_ADDR>
```

See [Attribute Groups and Dynamic Groups](#) for details.

For persistence across compute node reboots, modify a node image (e.g., `kubeimg`), that is used by Kubernetes worker nodes so that these nodes auto-join when booted. If multiple control planes are present optionally specify the control plane by IP address:

```
scyld-kube --image kubeimg --join
or
scyld-kube --image kubeimg --join --cluster CONTROL_PLANE_IP_ADDR
```

After rebooting these worker nodes, you can check Kubernetes status again on the control plane node and should now see the joined worker nodes:

```
kubectl get nodes
```

You can test Kubernetes by executing a simple job that calculates pi:

```
kubectl apply -f https://kubernetes.io/examples/controllers/job.yaml
```

(ref: [https://kubernetes.io/docs/concepts/workloads/controllers/job/](https://kubernetes.io/docs/concepts/workloads/controllers/job/))

See [Appendix: Using Kubernetes](#) for detailed examples.

### 6.4.9 scyld-nss Name Service Switch (NSS) Tool

The optional package `scyld-nss` provides a Name Service Switch (NSS) tool that translates a hostname to its IP address or an IP address to its hostname(s), as specified in the `/etc/scyld-nss-cluster.conf` configuration file. These hostnames and their IP addresses (e.g., for compute nodes and switches) are those managed by the ClusterWare database, which automatically provides that configuration file at startup and thereafter if and when the cluster configuration changes.

**Note:** `scyld-nss` is currently only supported on head nodes.

Installing `scyld-nss` inserts the `scyld` function in the `/etc/nsswitch.conf hosts` line, and installs the symlink `/lib64/libnss_scylld.so.2` and library `/lib64/libnss_scylld-1.0.so` to functionally integrate with the other NSS tools.

Benefits include an expanded functionality of ClusterWare hostname resolution and increased performance of NSS queries for those hostnames. Install the `nscd` package for additional significant performance improvements on clusters with very high node counts.

The `scyld-nss` package includes a `scyld-nssctl` tool allowing a cluster administrator to manually stop or start the service by removing or inserting the `scyld` function in `/etc/nsswitch.conf`. Any user can employ `scyld-nssctl` to query the current status of the service. See `scyld-nssctl` for details.
6.4.10 Firewall Configuration

If you are not using the `torque-scyld` or `slurm-scyld` packages, either of which will transparently configure the firewall on the private cluster interface between the head node(s), job scheduler servers, and compute nodes, then you need to configure the firewall manually for both the head node(s) and all compute nodes.

6.4.11 Install OpenMPI, MPICH, and/or MVAPICH

Scyld ClusterWare distributes several versions of OpenMPI, MPICH, and MVAPICH, and other versions are available from 3rd-party providers. Different versions of the ClusterWare packages can coexist, and users can link applications to the desired libraries and execute the appropriate binary executables using `module load` commands. Typically one or more of these packages are installed in the compute node images for execution, as well as on any other server where OpenMPI (and similar) applications are built.

View the available ClusterWare versions using:

```
yum clean all  # just to ensure you'll see the latest versions
yum list --enablerepo=scyld* | egrep "openmpi|mpich|mvapich" | egrep scyld
```

The OpenMPI, MPICH, and MVAPICH packages are named by their major-minor version numbers, e.g., 3.0, 4.0, 4.1, and each has one or more available major-minor "point" releases, e.g., `openmpi4.1-4.1.1` and `openmpi4.1-4.1.4`.

A simple `yum install` will install the latest "point" release for the specified major-minor version, e.g.:

```
sudo yum install openmpi4.1 --enablerepo=scyld*
```

installs the default GNU libraries, binary executables, buildable source code for various example programs, and man pages for `openmpi4.1-4.1.4`. The `openmpi4.1-gnu` packages are equivalent to `openmpi4.1`.

Alternatively or additionally:

```
sudo yum install openmpi4.1-intel --enablerepo=scyld*
```

installs those same packages built using the Intel compiler suite. These compiler-specific packages can co-exist with the base GNU package. Similarly you can additionally install `openmpi4.1-pgi` for libraries and executables built using the PGI compiler suite.

The `openmpi*-psm2` packages are intended for use with QLogic Infiniband controllers.

**Important:** To install `openmpi*psm2` packages for RHEL/CentOS 8 and beyond, you must additionally enable the `PowerTools` repo, e.g., `sudo yum install openmpi4.1-psm2 --enablerepo=scyld* --enablerepo=PowerTools`

**Important:** The ClusterWare yum repo includes various versions of `openmpi` RPMs which were built with different sets of options by different compilers, each potentially having requirements for specific other 3rd-party packages. In general, avoid installing openmpi RPMs using a wildcard such as `openmpi4*scyld` and instead carefully install only specific RPMs from the ClusterWare yum repo together with their specific required 3rd-party packages.

Suppose `openmpi4.1-4.1.1` is installed and you see a newer "point" release `openmpi4.1-4.1.4` in the repo. If you do:

```
sudo yum update openmpi4.1 --enablerepo=scyld*
```
then 4.1.1 updates to 4.1.4 and removes 4.1.1. Suppose for some reason you want to retain 4.1.1, install the newer 4.1.4, and have both "point" releases coexist. For that you need to download the 4.1.4 RPMs and install (not update) them using rpm, e.g.,

```
sudo rpm -iv openmpi4.1-4.1.4*
```

You can add OpenMPI (et al) environment variables to a user's ~/.bash_profile or ~/.bashrc file, e.g., add module load openmpi/intel/4.1.4 to default a simple OpenMPI command to use a particular release and compiler suite. Commonly a cluster uses shared storage of some kind for /home directories, so changes made by the cluster administrator or by an individual user are transparently reflected across all nodes that access that same shared /home storage.

For OpenMPI, consistent user uid/gid and passphrase-less key-based access is required for a multi-threaded application to communicate between threads executing on different nodes using ssh as a transport mechanism.

For example, user root can set up access for each given username $user and target node $node:

```bash
# This script must execute as 'root'.
# Here we set up user "user1" on node n0:
user=user1
node=n0

user_uid=`id -u $user`
user_gid=`id -g $user`
user_gname=`id -gn $user`

ssh $node groupadd -g $user_gid $user_gname
ssh $node useradd -u $user_uid -g $user_gid $user

# copy ssh key .bashrc
ssh $node mkdir -p -m 700 /home/$user/.ssh >/dev/null
scp /home/$user/.ssh/id_rsa.pub $node:/home/$user/.ssh/authorized_keys >/dev/null
ssh $node chmod 600 /home/$user/.ssh/authorized_keys
scp /home/$user/.bashrc $node:/home/$user/ >/dev/null
ssh $node chown -R $user_gid:$user_uid /home/$user

ssh $node chown $user_gid:$user_uid /home/$user
```

To use OpenMPI (et al) without installing either torque-scyld or slurm-scyld, then you must configure the firewall that manages the private cluster network between the head node(s), server node(s), and compute nodes. See Firewall Configuration for details.

### 6.4.12 Configure IP Forwarding

By default, the head node does not allow IP forwarding from compute nodes on the private cluster network to external IP addresses on the public network. If IP forwarding is desired, then it must be enabled and allowed through each head node's firewall configuration.

On a head node, to forward internal compute node traffic through the `<PUBLIC_IF>` interface to the outside world, execute:

```
firewall-cmd --zone=public --remove-interface=<PUBLIC_IF>
firewall-cmd --zone=external --add-interface=<PUBLIC_IF>
# confirm it was working at this point then make it permanent
firewall-cmd --permanent --zone=public --remove-interface=<PUBLIC_IF>
firewall-cmd --permanent --zone=external --add-interface=<PUBLIC_IF>
```

Appropriate routing for compute nodes can be modified in the compute node image(s) (see scyld-modimg tool in

---

6.4. Common Additional Configuration and Software 41
6.4.13 Install Additional Tools

Cluster administrators may wish to install additional software tools to assist in managing the cluster.

**Name Service Cache Daemon (nscd)**

The Name Service Cache Daemon (nscd) provides a cache for most common name service requests. The performance impact for very large clusters is significant.

**/usr/bin/jq**

The jq tool can be downloaded from the Red Hat EPEL yum repository. It provides a command-line parser for JSON output.

For example, for the --long status of node n0:

```
[sysadmin@head1 /]$ scyld-nodectl -i n0 ls --long
Nodes
n0
  attributes
    _boot_config: DefaultBoot
    _no_boot: 0
    last_modified: 2019-06-05 23:44:48 UTC (8 days, 17:09:55 ago)
  groups: []
  hardware
    cpu_arch: x86_64
    cpu_count: 2
    cpu_model: Intel Core Processor (Broadwell)
    last_modified: 2019-06-06 17:15:59 UTC (7 days, 23:38:45 ago)
    mac: 52:54:00:a6:f3:3c
    ram_total: 8174152
  index: 0
  ip: 10.54.60.0
  last_modified: 2019-06-14 16:54:39 UTC (0:00:04 ago)
  mac: 52:54:00:a6:f3:3c
  name: n0
  power_uri: none
  type: compute
  uid: f7c2129860ec40c7a397d78bba51179a
```

You can use jq to parse the JSON output to extract specific fields:

```
[sysadmin@head1 /]$ scyld-nodectl --json -i n0 ls -l | jq '.n0.mac'
"52:54:00:a6:f3:3c"

[sysadmin@head1 /]$ scyld-nodectl --json -i n0 ls -l | jq '.n0.attributes'
{
  "_boot_config": "DefaultBoot",
  "_no_boot": "0",
  "last_modified": 1559778288.879129
}
```

(continues on next page)
6.5 Node Images and Boot Configurations

6.5.1 Compute Nodes with SSH banner

The cluster administrator may wish to add an SSH banner message to compute nodes. Follow the directions from your distro about how to accomplish this.

Because the `scyld-nodectl ssh`, `exec`, and soft power control commands trigger `ssh` executions from the ClusterWare head node to compute node(s), and because the remote banner text is written to stderr, this stderr text may be interpreted by `scyld-nodectl` as a fatal `ssh` error. To remedy, place an exact copy of the banner text into the head node's `/opt/scyld/clusterware/banner.txt` file to inform the head node that the specific banner response is not a true error. This remedy fixes certain `scyld-nodectl` ssh key and soft power control failures, and it also results in `scyld-nodectl` `exec` suppressing the display of the remote banner, although that suppression can be overridden with `scyld-nodectl -vv exec`.

On non-head node servers with ClusterWare tools installed, the banner text should be placed in `/opt/scyld/clusterware-tools/banner.txt` to filter the banner from `scyld-nodectl exec` commands.

**Important:** The compute node SSH banner text must be the same across all nodes.

6.5.2 Compute Node Fields

To view various compute node fields, e.g., for node n0:

```
# View the full list of fields, first using long-form arguments:
scyld-nodectl -i n0 list --long-long
# or the equivalent using shorthand arguments:
scyld-nodectl -i n0 ls -L

# View the abbreviated list of fields, instead using long-form args:
scyld-nodectl -i n0 list --long
# or the equivalent using shorthand args:
scyld-nodectl -i n0 ls -l
```

The `type` field is currently set to "compute", although future updates to Scyld ClusterWare may add additional values.

The `groups` and `attributes` fields are described in more detail in *Interacting with Compute Nodes* and in the *Reference Guide* commands `scyld-nodectl` and `scyld-attribctl`.

Prior to a node booting, the system will inform the DHCP server of MAC-to-IP address mappings for nodes known to the system. Changes to node indices, IP, or MAC addresses may affect these mappings and will cause updates to be sent to the DHCP server within a few seconds. When a node makes a DHCP request, the DHCP server maps that node's MAC address to the correct IP and provides additional options to the booting node, including where to find the correct boot files. These boot files are linked in *boot configurations* stored in the database.
6.5.3 Compute Nodes IPMI access

The node's `power_uri` field in the database is optional and informs the head node(s) how to control the power to a given node. A plugin interface allows for different forms of power control, currently supporting IPMI for bare metal nodes, and libvirt or VirtualBox (vbox) for different types of virtual nodes. For example, a `power_uri` for a VirtualBox virtual node might be:

```
vbox://192.168.56.1/CW_Compute0
```

Production system compute nodes are generally bare-metal nodes that can be controlled via the `ipmitool` command that communicates with the node's Baseboard Management Controller (BMC) interface. For such nodes the administrator should set a `power_uri` with the appropriate BMC IP address and username/password access credentials, e.g.,

```
ipmi:///admin:password@172.45.88.1
```

With such a `power_uri`, the head node communicates with that compute node's BMC located at 172.45.88.1 using the username "admin" and password "password" to perform a `scyld-nodectl power on`, `power off`, `power cycle`, `shutdown --hard`, or `reboot --hard`.

If for any reason only a specific remote machine can execute `ipmitool` to control a node, then add that server name, and an optional user name and password, to the `power_uri`, and the local head node will `ssh` to that remote server and execute the `ipmitool` command from there. For example, the `power_uri`:

```
ipmi:///remote_server/admin:password@172.45.88.1
```

sends the `ipmitool` command details to server "remote_server" for execution.

The `scyld-nodectl` so-called "soft" `shutdown --soft` and `reboot --soft` commands do not use the `power_uri`, but rather `ssh` to the compute node to execute the local `/usr/sbin/shutdown` or `/usr/sbin/reboot` command with appropriate arguments. A simple `scyld-nodectl -i <NODE> reboot` (or `shutdown`) first attempts a "soft" action if the node is "up" and the head node can communicate with the node. If the "soft" action is not possible or does not complete within a reasonable time, then the `scyld-nodectl` resorts to a "hard" action using the `power_uri` connection.

If compute node SSH banners have been configured, then the head node (or the non-head node server invoking `scyld-nodectl`) needs to ignore the banner message in response. See Compute Nodes with SSH banner.

6.5.4 Boot Configurations

The `scyld-install` script creates a basic boot configuration named `DefaultBoot` that references the initial `Default-Image` and is initially associated with all compute nodes. After installation, the cluster administrator can customize that configuration and/or create additional boot configurations and compute node images.

Administrators can modify configuration fields using the `scyld-bootctl` tool. For example, the administrator can change the name and description of the newly created boot configuration on a freshly installed system using the `update` argument:

```
scyld-bootctl -i DefaultBoot update name="NewName" description="New description"
```

The kernel and initramfs can also be set using the same command, although their paths must be prefixed with @ (which signifies that what follows is a local file path), e.g.:

```
scyld-bootctl -i DefaultBoot update kernel=@/boot/vmlinuz-3.10.0-862.el7.x86_64
```

Other database objects (Nodes, Images, etc.) are modified using similarly named tools, e.g. `scyld-nodectl` and `scyld-imgctl`. Each node associates with a specific boot configuration through its `_boot_config` attribute. Like other
attributes, this field may be inherited from an attribute group (including the global default attribute group) or set directly on the node. Details of manipulating node attributes are discussed in Interacting with Compute Nodes.

Boot configurations also contain two more fields, release and boot_style. The release field is not editable by the administrator and is populated by the system whenever the kernel file is uploaded, based on the Linux file command output. The boot_style dictates how the nodes will receive the root file system, although that can be overridden by the _boot_style attribute (see Reserved Attributes) set at the node level or in any attribute groups used by the node.

The possible values for boot_style are rwram, roram, iscsi, disked, live, next, and sanboot. The default rwram instructs the system to download the compressed image into compute node RAM where the mount_rootfs script unpacks it during the boot process. Alternatively, when the roram option is provided, the script downloads a squashfs image into compute node RAM, combines this with a writable tmpfs via overlayfs, and boots using that combined file system. The iscsi option instructs the node to mount a read-only image via iSCSI and similarly apply a writeable overlay.

The disked option allows a node with local storage to both employ a node-local persistent cache to retain downloaded images and unpack images onto a node-local partition. Using a cache avoids the need to download images at boot time, and booting from a local partition frees the RAM that would otherwise hold the compute node image. See Appendix: booting from local storage cache for details.

The live and next options are most useful when kickstarting locally installed nodes. The live option can be applied to a boot configuration that points to a repo based on an uploaded CentOS or RHEL ISO. Nodes booted live from such a configuration will use the kernel and initramfs from the ISO with an inst.repo kernel option to boot into the ISO’s Anaconda-based installer. Given access to the node console, a cluster administrator can manually install to the local disk, thereby generating a kickstart file that can be used to reinstall this or similar nodes at a later time. The BIOS of such kickstarted nodes should be configured to boot from the network and then from local disk. In this configuration the next boot style should cause the compute node(s) to initially attempt to PXE boot, but then fail and try to boot their local disk. Additional details of kickstarting locally installed nodes can be found in creating-nodes-with-kickstart.

Depending on BIOS details, some locally installed systems will not properly handle the next boot style and will halt instead of failing over to another boot device. In that case, the sanboot option can be used to trigger booting of the first partition of the first disk. The sanboot option behavior can be customized using the _ipxe_sanboot attribute described in Reserved Attributes.

The boot_style setting can be overridden for an individual or group of nodes by assigning a _boot_style attribute. Similarly, to avoid overlayfs and use the rwtab approach to providing write capabilities to read-only root file systems, an administrator can set a node’s (or attribute group’s) the _boot_rw_layer attribute to rwtab.

### 6.5.5 Creating PXEboot Images

**Important:** Various commands that manipulate images execute as user root, thereby requiring that the commands internally use sudo and requiring that user root must have access to the administrator's workspace which contains the administrator's images. Typically the per-user workspace is ~/.scyldcw/workspace/. If that directory is not accessible to the command executing as root, then another accessible directory can be employed, and the administrator can identify that alternative path by adding a modimg.workspace setting to ~/.scyldcw/settings.ini.

Administrators can use the scyld-modimg tool (and possibly together with the scyld-clusterctl tool) to create new and modify existing image files.

The scyld-install script creates an initial basic image with the default name DefaultImage based on the publicly available CentOS repositories. If these repositories are not accessible, the scyld-add-boot-config tool can be run later with locally accessible repositories as described in Appendix: Creating Local Repositories without Internet. Once the DefaultImage is created, the cluster administrator can use scyld-modimg to modify it directly, though a safer approach is to use scyld-imgctl to clone the DefaultImage to new name, and then use scyld-modimg to modify that cloned image, leaving the DefaultImage untouched. See Modifying PXEboot Images for details. The administrator can also re-create the DefaultImage. See Recreating the Default Image for details.
The administrator can also create a new image from an ISO or network accessible package repository. When doing that, consider the source of the components (aka packages) for that new image. A distro ties together a list of repos, i.e., package repositories, and an optional release. The package_manager is determined during image creation but can be overridden in the distro. The initial default distro is CentOS version 7 or 8 (matching the original head node’s version), uses package_manager yum, and downloads packages from a one item repos list containing “CentOS_base”:

```
[admin@virthead]$ scyld-clusterctl distros ls -l
Distros
 CentOS
   name: CentOS
   package_manager: yum
   release: 7
   repos
     CentOS_base

[admin@virthead]$ scyld-clusterctl repos ls -l
Repos
 CentOS_base
   keys: []
   name: CentOS_base
   urls
     http://mirror.centos.org/centos/$releasever/os/$basearch/
```

Create a new image named "NewImg" using the default distro:

```
scyld-modimg --create --set-name NewImg
```

which downloads packages from the latest CentOS 7 yum repo.

Create a CentOS 6 distro that downloads packages from the latest CentOS 6 yum repo:

```
scyld-clusterctl distros create name=CentOS6 release=6 repos=CentOS_base
```

Note that this new "CentOS6" distro for release 6 uses the same repos value of "CentOS_base" that is used for the "CentOS" distro for release 7. This works because a distro’s release value becomes the repo’s urls "$releasever" field, and the CentOS repositories for 6 and 7 have the same subdirectory hierarchies.

View both distros, and also see that the default remains the CentOS 7 distro:

```
[admin@virthead]$ scyld-clusterctl distros ls -l
Distros
 CentOS
   name: CentOS
   package_manager: yum
   release: 7
   repos
     CentOS_base

 CentOS6
   name: CentOS
   package_manager: yum
   release: 6
   repos
     CentOS_base
```

(continues on next page)
Create an image using this CentOS distro, overriding the default CentOS:

```
scyld-modimg --create CentOS6 --set-name CentOS6_image
```

or first switch the default distro to CentOS6 and do another simple create:

```
scyld-clusterctl --set-distro CentOS6
scyld-modimg --create --set-name CentOS6_image
```

Keep in mind that now every subsequent simple `--create` command will default to use the CentOS6 distro.

To create a CentOS image that contains something other than the latest CentOS 7 or 6 release, see Appendix: Creating Arbitrary CentOS Images. To create a RHEL image, see Appendix: Creating Arbitrary RHEL Images.

### 6.5.6 Recreating the Default Image

If you wish to recreate the `DefaultImage` that was built by the `scyld-install` tool, then you must first delete the components of the existing image and boot config:

```
scyld-attribctl -i DefaultAttribs rm
scyld-bootctl -i DefaultBoot rm
scyld-imgctl -i DefaultImage rm
```

Then create a new default. If there are no attribute groups defined on this cluster (see Node Attributes), then:

```
scyld-add-boot-config --make-defaults
```

Otherwise the administrator should first clear the attributes.

### 6.5.7 Modifying PXEboot Images

Once you have an existing image, you can install additional RPMs into that image. We suggest that Best Practices is to rarely and only very carefully modify `DefaultImage` and `DefaultBoot`, and to use them as starting points from which you clone new images and boot configurations.

The `scyld-modimg` tool supports a rich collection of options. See `scyld-modimg` for details.

For example:

```
scyld-imgctl -i DefaultImage clone name=mpiImage
scyld-add-boot-config --image mpiImage --boot-config mpiBoot
scyld-modimg -i mpiImage --install openmpi3.1
```

Suppose you want to create a new boot config `mpiAltBoot` that references the same `mpiImage` though is otherwise different than `mpiBoot`. For instance, suppose you want `mpiAltBoot` to have a different `cmdline`:

```
scyld-bootctl -i mpiBoot clone name=mpiAltBoot
# Note that an updated cmdline replaces the entire existing cmdline,
# so examine the current cmdline:
```

(continues on next page)
You can also manually customize an image, including installing or removing RPMs and modifying configuration files, by operating on the image inside a chroot:

```
scyld-modimg -i mpiImage --chroot
```

Or combine commands, ending inside a chroot:

```
scyld-modimg --create --set-name mpiImage --install openmpi3.1 --chroot
```

Inside the chroot you execute as user root and can manually add, update, or remove rpms with yum (or other appropriate package manager), modify configuration files, etc. When you exit the chroot, you are asked if you want to discard or keep the changes. If you keep the changes, then you are asked whether or not you want to replace the local image, to upload the image, and to replace the remote image.

**Note:** Keep in mind that several directories in the image do **not** get repacked and saved into the image file after an exit: `/tmp/`, `/var/tmp/`, `/var/cache/yum`.

If your intention is to answer yes to all the questions following your exit, then you can skip those questions by adding more arguments to the original command line:

```
scyld-modimg --create --set-name mpiImage --install openmpi3.1 --chroot --no-discard --overwrite --upload
```

You can examine the RPM contents of an image without going into a chroot by doing a simple query:

```
# Display the version of 'clusterware-node' in the image
scyld-modimg -i mpiImage --query clusterware-node

# Display the version of all RPMs in the image
scyld-modimg -i mpiImage --query
```

Finally, you must set the `_boot_config` attribute for specific nodes, or for all nodes, as desired to use this new boot config. For example, to have nodes n0-n15 use the `mpiBoot` boot config:

```
scyld-nodectl -i n[0-15] set _boot_config=mpiBoot
```

The `scyld-modimg` command prompts the user about whether to overwrite an existing image or create a new one, and whether to upload the resulting file to the head node, optionally overwriting the image stored on the ClusterWare head node. This tool operates on a local cache of the image and cannot be used to delete an image from the head nodes or to directly modify the name or description of an image on the head node. To modify these sorts of fields, use the `scyld-imgctl` tool.

Images are stored in the head node's `/opt/scyld/clusterware/storage/` in **cwsquash** format, which consists of a squashfs image offset inside a pseudo-disk image. This format is suitable for exporting via iSCSI.

Small homogeneous clusters may use a single node image across all compute nodes, although larger clusters that include compute nodes with differing hardware will require additional customization that may not be applicable to all nodes. Although cluster administrators may find that node attributes (discussed in more detail in **Interacting with Compute Nodes**) and customized boot-time scripting provide adequate image customization, it may be useful (or necessary) to create additional boot configurations and root file systems that meet specific hardware and/or software needs.
Customization can involve more than adding software drivers to support node-specific hardware and adding applications and their associated software stacks. It can also involve customizing configuration files in an image to deal with a non-standard networking environment. For example, if the compute node needs to use a networking route that is not the gateway defined in the head node's /opt/scyld/clusterware-isc-dhcp/dhcpd.conf.template, then the cluster administrator needs to edit that file to modify the default option routers <GATEWAY>; line, or edit the compute node image's appropriate /etc/sysconfig/network-scripts/ifcfg-* script to insert the desired GATEWAY IP address. For more details see https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Editing_Network_Configuration_Files.html or documentation for your base distribution.

### 6.5.7.1 Updating the kernel in an image

To update the kernel in an image, first install its RPM into the image. For example, using mpiImage and mpiBoot:

```bash
scyld-modimg -i mpiImage --install kernel-3.10.0-1160.24.1.el7.x86_64 \
   --no-discard --overwrite --upload
```

Then create a new initramfs file on the head node to match that kernel:

```bash
scyld-mkramfs --kver 3.10.0-1160.24.1.el7.x86_64 --update mpiBoot
```

**Note:** Note that this new initramfs file is not the same as a similarly named "initramfs" file in the head node /boot/ that is associated with a kernel in the head node's /boot/ directory. This ClusterWare initramfs file is associated with a specific image and boot config, and it contains custom scripts that execute at boot time.

### 6.5.8 Capturing and Importing PXEboot Images

Cluster administrators can also modify the files on a booted compute node and use the scyld-modimg --capture command to capture those changes into the image. You can capture the node into an existing image or into a new image. For example, to capture node n0:

```bash
scyld-modimg --capture n0 --set-name NewImage
```

This process will take several minutes. During that time the scyld-pack-node tool is executed on the compute node via the scyld-nodectl exec mechanism, and the result is streamed back to the scyld-modimg command that then uploads it to the head node, potentially replacing an existing NewImage contents. The scyld-pack-node tool captures all files on the node's / mount, but does not walk other mounted file systems to ensure that any shared storage is not accidentally captured.

You also need to create a boot config for this captured image. For example:

```bash
scyld-add-boot-config --image NewImg --boot-config NewBoot --batch
```

Please note that manual work will likely be required to generalize the captured image, as the process may capture details specific to the compute node. Due to this hazard, future ClusterWare releases may expand what files are excluded during image capture. Additionally, cluster administrators should confirm that the node being captured is idle to reduce the chance of capturing an image in some intermediate state.
6.5.9 Deleting unused images and boot configurations

Compute node images and boot configurations consume significant storage space. Remote images are replicated among cooperating head nodes and are the files downloaded by PXE booting compute nodes. A local image is a cached copy of a remote image that was downloaded when the cluster administrator viewed or modified the image. Deleting a local image does not affect its remote version and merely causes it to be re-downloaded from the head node if and when an administrator subsequently views or modifies it.

To view the list of local and remote images:

```bash
scyld-modimg ls
```

Delete a local image `xyzImage` with:

```bash
scyld-modimg -i xyzImage --delete
```

Permanently delete an unwanted remote image from the database with:

```bash
scyld-imgctl -i xyzImage delete
```

Boot configurations contain only a kernel and initramfs and consume only a few tens of megabytes. Permanently delete an unwanted boot configuration `xyzBoot` with:

```bash
scyld-bootctl -i xyzBoot delete
```

6.5.10 Copying boot configurations between head nodes

A multiple head node cluster contains cooperating head nodes that share a replicated database and transparent access to peer boot configurations, kernel images, and initramfs files. See Managing Multiple Head Nodes for details. There is no need to manually copy boot configs between these head nodes.

However, it may be useful to copy boot configurations from a head node that controls one cluster to another head node that controls a separate cluster, thereby allowing the same boot config to be employed by compute nodes in the target cluster. On the source head node the administrator “exports” a boot config to create a single all-inclusive self-contained file that can be copied to a target head node. On the target head node the administrator “imports” that file into the local cluster database, where it merges with the local head node’s existing configs, images, and files.

**Important:** Prior to exporting/importing a boot configuration, you should determine if the boot config and kernel image names on the source cluster already exist on the target cluster. For example, for a boot configuration named `xyzBoot`, execute `scyld-bootctl -i xyzBoot ls -l` on the source head node to view the boot config name `xyzBoot` and note its image name, e.g., `xyzImage`. Then on the target head node execute `scyld-bootctl ls -l | grep "xyzBoot |xyzImage"` to determine if duplicates exist.

If any name conflict exists, then either (1) on the source head node create or clone a new uniquely named boot config associated with a uniquely named image, then export that new boot config, or (2) on the target head node import the boot config using optional arguments, as needed, to assign unique name or names.

To export the boot configuration `xyzBoot`:

```bash
scyld-bootctl -i xyzBoot export
```

which creates the file `xyzBoot.export`. If there is no name conflict(s) with the target cluster, then on the target head node import with:
If there is a name conflict with the image name, then perform the import with the additional argument to rename the imported image:

```
scyld-bootctl import xyzImage.export --image uniqueImg
```

or import the boot config without importing its embedded image at all (and later associate a new image with this imported boot config):

```
scyld-bootctl import xyzImage.export --no-recurse
```

If there is a name conflict with the boot config name itself, then add:

```
scyld-bootctl import xyzImage.export --boot-config uniqueBoot
```

Associate a new image name to the imported boot config if desired, then associate the boot config with the desired compute node(s):

```
scyld-nodectl -i <NODES> set _boot_config=xyzBoot
```

### 6.5.11 Wrapper scripts

Since a cluster administrator creating a new boot image commonly wants to create a corresponding boot configuration and assign that configuration to a set of nodes, the `scyld-add-boot-config` tool wraps `scyld-modimg`, `scyld-mkramfs`, and the appropriate `scyld-*ctl` tools to perform the necessary steps. The tool will also optionally display the required steps so that administrators can learn about the usage of the underlying tools.

When executed with no arguments, the `scyld-add-boot-config` script asks a series of questions to define the various fields of the boot configuration, image, and attribute group that are being created. Default values are provided where possible.

**Important:** The default kernel command line sets SELinux on the compute nodes to permissive mode.

### 6.5.11.1 Repos and Distros

One of the steps in the `scyld-install` script is to run the `scyld-clusterctl` tool to define a *distro* prior to creating the first image. The `scyld-modimg` tool can only create images based on defined distros. A *distro* associates one or more repos together with their package manager and an optional release string. If no release string is provided, then any supplied URL should not include the string "$releasever", as that variable will not be defined during image creation.

On a CentOS or RHEL system the default repo and *distro* are created by:

```
scyld-clusterctl repos create name=CentOS_base \  
  urls=http://mirror.centos.org/centos/$releasever/os/$basearch/ 
scyld-clusterctl distros create name=CentOS repos=CentOS_base release=7
```

Together with the local `/etc/yum.repos.d/clusterware.repo` file, this information will be used at image creation time to generate a `/etc/yum.repos.d/clusterware-node.repo` file for the image containing sections referring to both the head node's ClusterWare repository and to the *distro's* repos.

A system administrator is welcome to create additional repos and distros to make node images based on different upstream sources. An administrator can provide multiple comma-separated URLs to the `scyld-clusterctl repos`
create command, or multiple repos to the scyld-clusterctl distros create command. Distros can also be imported from an existing yum repo files, e.g.:

```bash
scyld-clusterctl distros import --name CentOS7 /etc/yum.repos.d/CentOS-Base.repo
```

The import action will create repos based on the contents of the provided yum repo file(s) and then associate all of them with a newly created CentOS7 distro. Any string passed to --release will be saved into the distro release field and will be used by yum to replace any occurrences of "$releasever" in the repo file.

See the Reference Guide for additional details of the scyld-clusterctl repos and distros actions and of the scyld-modimg command that is used to actually create and modify images.

### 6.5.11.2 Using Archived Releases

Many distributions will archive individual releases after they have been superseded by a newer release, but for this discussion we will examine CentOS. The CentOS project provides packages and updates on their various mirror sites for the most recent release, i.e. 7.9.2009 as of this writing, but deprecates all previous point releases. This means that at the URL where a mirror would nominally keep the previous release, a readme file is provided explaining that the release has been deprecated and pointing users to the CentOS vault for packages. The packages located in the vault are unchanged from when they were "current". The CentOS project also deprecates the release that is two major releases back, meaning that as of the release of version 7, version 5 was deprecated. In this way there are always two currently supported versions of CentOS, the latest and the most recent of the previous major release, i.e. 7.9.2009 and 6.10 as of the time of this writing.

What this means for ClusterWare administrators is twofold. First, in order to create an image of an archived version of CentOS, an administrator must create the correct repo and distro objects in the ClusterWare database. Second, after creating an image from the vault, the administrator must manually modify the yum repo files present in the image. We will now explore these steps in more detail.

To create an image based on an archived version of CentOS, 7.3 in this example, the steps are:

```bash
scyld-clusterctl repos create name=CentOS-vault \
    urls=http://vault.centos.org/$releasever/os/$basearch/
scyld-clusterctl distros create name=CentOS_7.3 repos=CentOS-vault release=7.3.1611
scyld-modimg --create CentOS_7.3 --set-name CentOS_7.3_img
```

The first command creates a repo called CentOS-value pointing at the generic vault URL. The second command creates a distro that references the CentOS-vault repo and defining the release string. Once the distro exists, it can be referenced by name in the third command to actually create a new image.

Unfortunately, because the CentOS vault packages are identical to when they were the current release, the yum repo files located in the /etc/yum.repos.d/ directory will contain references to mirror.centos.org instead of vault.centos.org. The cluster administrator must manually modify these files after image creation and before running yum commands directly or through the scyld-modimg --install, --uninstall, --update, or --query. The above scyld-modimg --create command will also display an error referring back to this documentation:

```
[admin@virthead]$ scyld-modimg --create CentOS_7.3 --set-name CentOS_7.3_img
```

Executing step: Create
Preparing the chroot...
...done.
Initializing the chroot...
elapsed: 0:01:11.4
...initialized.
Installing core packages...
In order to manually modify the yum repo files, an administrator can use the `scyld-modimg --chroot` command on an already created image as follows:

```
[admin@virthead]$ scyld-modimg -i CentOS_7.3_img --chroot
Checksumming image 6a8947156e08402ba2ad6e23a7642f4f
  elapsed: 0:00:01.0
Unpacking image 6a8947156e08402ba2ad6e23a7642f4f
  100.0% complete, elapsed: 0:00:29.6 (62.2% compression)
Checksumming...
  elapsed: 0:00:01.0
Executing step: Chroot
Dropping into a /bin/bash shell. Exit when done.
[root@virthead /]# exit
exit
fixing SELinux file labels...
(K)eep changes or (d)iscard? [kd]
```

When you exit the shell, the tool will confirm that you want to keep the changes made and offer to upload the modified image to head node storage.

### 6.5.11.3 Using ISO Releases

Many distributions are distributed in ISO form. Use the `scyld-clusterctl` tool to create an image from an ISO. For example, for an ISO named `CentOS-7-x86_64-DVD-2009.iso`, first create a repo:

```
scyld-clusterctl repos create name=centos_7.9_iso
  iso=@/path/to/CentOS-7-x86_64-DVD-2009.iso
```

then create a distro that references the new repo:

```
scyld-clusterctl distros create name=centos_7.9_distro repos=centos_7.9_iso
```

then you can create an image using that repo and distro:

```
scyld-modimg --create centos_7.9_distro --set-name centos_7.9_image
```

When this image is booted, the ISO-based repo may not be accessible, and the `/etc/yum.repos.d/clusterware-node.repo` file will need to be modified to use a more permanent repo location.

**Note:** If the CentOS 7.9 ISO was downloaded from `https://www.centos.org/centos-linux/`, then that ISO contains CentOS 7.9 base distribution packages for the first release of 7.9, not packages for the latest CentOS 7.9.
6.5.11.4 Installing Software With Subscriptions

For distributions requiring subscriptions for access to updated packages, please note that subscription information in an image will be used by all nodes unless removed before upload:

```bash
hostname nodeTemplate
subscription-manager register --username=$RHUSER --password=$RHPASS
subscription-manager attach --pool=$POOL_ID
yum upgrade -y
yum install $REQUIRED_PACKAGE
subscription-manager remove --all
subscription-manager unregister
subscription-manager clean
```

6.5.12 Adding 3rd-party software

An existing compute node image may need to contain additional software (e.g., a driver and perhaps the driver's associated software) that has been downloaded from a 3rd-party vendor in the form of an RPM or a tarball.

Suppose a tarball named `driver-tarball.tgz` has been downloaded into the head node `/tmp/` directory, and you need to install its contents into an image. A cautious first step is to clone an existing image and add the new software to that clone, which leaves the existing image unmodified. For example, clone a new image:

```bash
scyld-imgctl -i DefaultImage clone name=UpdatedImage
```

Now enter the new `UpdatedImage` in a chroot environment:

```bash
scyld-modimg -i UpdatedImage --chroot
```

Suppose your clusterware administrator user name is `admin1`. Inside the chroot you are always user `root`. Copy the downloaded tarball from the head node into your chroot with a simple command from inside the chroot:

```bash
scp -r admin1@localhost:/tmp/driver-tarball.tgz /tmp
```

Unpack `/tmp/driver-tarball.tgz` and examine the contents, where you will likely find a script that manages the tarball-specific software installation.

**Important:** Carefully read the instructions provided by the 3rd-party software vendor before executing the script, and carefully read the output produced when executing the script.

There are several factors to keep in mind when executing the 3rd-party install script:

- A 3rd-party installation that involves a new kernel module requires linking that module to the kernel in the chroot. This requires the presence of the `kernel-devel` package that matches that kernel. If that RPM is not currently installed in the chroot, then inside the chroot manually `yum install` it, naming the specific kernel version, e.g.:

  ```bash
  yum install kernel-devel-3.10.0-957.27.2.el7.x86_64
  ```

  to match kernel-3.10.0-957.27.2.el7.x86_64.

- Some 3rd-party install scripts use the `uname` command to determine the kernel against which to link a new kernel module. However, when the `uname` command executes inside a chroot, it actually reports the kernel version of the host system that executes the `scyld-modimg --chroot` command, **not** the kernel that has been installed.
inside the chroot. This `uname` behavior only works properly for module linking purposes if the chroot contains only one kernel and if that kernel matches the kernel on the `scyld-modimg --chroot`-executing server. To specify an alternate kernel, either name that kernel as an optional argument of a `--chroot` argument, e.g.:

```
scyld-modimg -i NewImage --chroot 3.10.0-1160.45.1.el7.x86_64
```

or as a `KVER` variable value using the `--exec` argument, e.g., for a script inside the image that initializes a software driver module and links that module to a specific kernel:

```
scyld-modimg -i NewImage --execute 'KVER=3.10.0-1160.45.1.el7.x86_64 /path/to/script'
```

Otherwise, hopefully the 3rd-party install script supports an optional argument that specifies the intended kernel version, such as:

```
/path/to/install-script -k 3.10.0-1160.45.1.el7.x86_64
```

- If the 3rd-party install script encounters a missing dependency RPM, then the script reports the missing package name(s) and fails. You must manually `yum install` those missing RPM(s) within the chroot and reexecute the script.

- Some 3rd-party install scripts replace RPMs that were installed from the base distribution, e.g., Infiniband, OFED. If any currently installed ClusterWare packages declare these base distribution packages as dependencies, then the install script's attempt to replace those packages fails. You must then uninstall the specified ClusterWare package(s) (e.g., `openmpi3.1`, `openmpi3.1-intel`), then retry executing the install script. In some cases the 3rd-party tarball contains packages that replace the ClusterWare package(s). In other cases you can reinstall these ClusterWare package(s) after the 3rd-party install script successfully completes.

Finally, exit the chroot and specify to `Keep changes`, `Replace local image`, `Upload image`, and `Replace remote image`.

### 6.6 Using Kickstart

The `Node Images and Boot Configurations` section discusses the creation and modification of compute node images and how to add locally installed nodes to the system. To facilitate the installation of these locally installed compute nodes, ClusterWare provides mechanisms for ISO upload and node kickstart.

#### 6.6.1 Uploading ISO Images

ISO images of the installation DVDs for RHEL or CentOS systems can be downloaded from their respective websites and imported into ClusterWare as repos using the `scyld-clusterctl repos` command. For example:

```
scyld-clusterctl repos create name=CentOS iso=@CentOS-7-x86_64-DVD-1908.iso
```

Once the upload completes, the ISO will be automatically forwarded to all head nodes and will be locally mounted on each. Below are the repository details immediately after the upload completes:

```
[cwadmin@virthead]$ scyld-clusterctl repos -i CentOS ls -l
Repos
CentOS
  iso
    checksum: 6b71f450513c92e2ab6ea663a69989f8f3680f01
    mtime: 2020-03-05 17:19:41 UTC (0:01:39 ago)
    size: 4.3 GiB (4664064948 bytes)
```

(continues on next page)
ClusterWare will display URLs for the repositories identified on the ISO. For example, there is a repository in the root of the uploaded CentOS ISO. This RPM repository can be accessed at `<BASE_URL>/repo/CentOS/repo/`. The `<BASE_URL>` tag is used as a placeholder to signify that any head node can provide access. Replace the tag with the `base_url` for a head node before using this URL:

```
[cwadmin@virthead]$ curl http://localhost/api/v1/repo/CentOS/repo/EULA
CentOS Linux 7 EULA
CentOS Linux 7 comes with no guarantees or warranties of any sorts, either written or implied.
The Distribution is released as GPLv2. Individual packages in the distribution come with their own licenses.
```

The mounted contents of the ISO are accessible through each head node's webserver at `<BASE_URL>/repo/<REPO_NAME>/content/`, and the ISO itself is located at `<BASE_URL>/repo/<REPO_NAME>/iso`. The ISO can also be downloaded using the `scyld-clusterctl` command:

```
scyld-clusterctl repos -i CentOS download iso
```

Just like URL defined repos, repos created using ISOs can be referenced in distros. See `Creating PXEboot Images` in `Node Images and Boot Configurations` in this guide for details about using repos and distros to create compute node images.

### 6.6.2 Kickstarting Installations

In addition to providing content for distros, repos based on CentOS or RHEL ISO images can also be used to kickstart locally installed compute nodes. To prepare a kickstart configuration, create a boot configuration that references the repo directly:

```
scyld-bootctl create name=CentOS_iso repo=CentOS
```

The resulting boot configuration will automatically locate the kernel and initramfs on the ISO and default to using no image:

```
[cwadmin@virthead]$ scyld-bootctl -i CentOS_iso ls -l
CentOS_iso
    image: none
    initramfs: repo:images/pxeboot/initrd.img
    kernel: repo:images/pxeboot/vmlinuz
    last_modified: 2020-03-05 17:40:32 UTC (0:00:34 ago)
    name: CentOS_iso
    repo: CentOS
```

Initially this boot configuration can be used to boot a disked node with the `live` boot style assigned either by the boot configuration `boot_style` field or a `_boot_style` node attribute. When live booting a node, the cluster administrator will
need to access the node's console to proceed through the operating system installation steps. To use the serial-over-lan BMC feature, the administrator may need to provide an appropriate `console= cmd line:

```
scyld-bootctl -i CentOS_iso update cmd line=console/ttyS0,115200
```

The specific details of the console and other command line arguments depend on the target hardware and are beyond the scope of this document. Once the installation process is complete, the compute node should use a `next` boot style in order to skip the PXE boot process and instead boot from the next boot device. Cluster administrators are encouraged to configure the BIOS of locally installed compute nodes to attempt PXE boot first and then boot from the local disk so that the `next` boot style works as intended.

Once the freshly installed node has booted, there will be a `/root/anaconda-ks.cfg` file that can be used as a starting point for creating a more generalized kickstart file. If the cluster administrator would like to reinstall the node the exact same way, the simplest thing to do is copy that `anaconda-ks.cfg` file to the head node's kickstart directory and assign it to be used in the boot configuration:

```
cp anaconda-ks.cfg /opt/scyld/clusterware/kickstarts
scyld-bootctl -i CentOS_iso update kickstart=anaconda-ks.cfg
```

After that file is in place, any compute node booted from that boot configuration without the `next or live` boot style will boot using the kernel and initramfs from the ISO, and a URL to the kickstart file will be added to the kernel command line. Keep in mind that once a node starts the kickstart process, it is a good idea to change its boot style to `next` so that it does not reboot at the end of the install process and immediately reinstall. Configuring the kickstart process to end with a `shutdown` command (see your operating system documentation) is the current best practice.

If a cluster administrator wants to use a different kernel and/or initramfs for kickstarting instead of the ones found on the ISO, those can be replaced just like in any other boot configuration through the `update` action. Updating them with an empty string will reset them back to the detected paths:

```
scyld-bootctl -i CentOS update kernel= initramfs=
```

6.6.3 Kickstart Files

Kickstart configuration files are stored on the head node inside the `/opt/scyld/clusterware/kickstarts/` folder. In a multihead configuration this folder must currently be manually synced between head nodes. These kickstart files can also contain a limited set of variables that will be substituted at download time. For example, the ClusterWare package includes an example called `basic.ks` that starts with the following contents:

```
install
text
url --url <repo_url()>
lang <head[lang]>
keyboard <head[keymap]>
timezone <head[timezone]>

# Node fields, attributes, status, and hardware are also available in
# dictionaries referenced by name or initial.
#
# <node[ip]>
# <a[_boot_config]>
# <hardware[mac]>
```

Two different types of tags are supported, and both provide simple text substitution. The first, exemplified by the `<repo_url()>` tag, is a function call that accepts as an optional argument the name of a ClusterWare repo. That tag will
be replaced by an appropriate URL for the named repo. Similarly, certain values are represented as more of a dictionary-
like reference, e.g. `<head[timezone]>`. These tags provide a simple variable lookup for information such as the head
node timezone. The snippet above shows all the currently supported tags except for the `<include(partial.ks)>` tag. The include tag allows a cluster administrator to break the kickstart files into manageable hunks that can then be
included into a top-level kickstart file, much like a C or C++ `#include`.

The post-substitution contents of the kickstart file depend on the specific node and can be downloaded using the
`scyld-bootctl` tool with the node name appended:

```
scyld-bootctl -i CentOS_iso download kickstart.n1
```

Additional variables and functions are planned, and the development team is open to suggestions of substitutions that
would be useful to cluster administrators.

### 6.7 Using RHCOS

ClusterWare provides support for installing and using RHCOS.

First create a repo from a RHCOS ISO file. For example:

```
scyld-clusterctl repos create name=rhcos iso=@rhcos-4.10.3-x86_64-live.x86_64.iso
```

Once the repo is created, the ISO will be automatically forwarded to all head nodes and will be locally mounted on
each. Below are the repository details immediately after the upload completes:

```
[cwadmin@virthead]$ scyld-clusterctl repos -i rhcos ls -l
Repos
rhcos
iso
  chksum: ee4f06946822b55c81c8aa95e21df4f02b9699e8
  filename: 7e9666b05e914b85b59be21f23ce9136
  mtime: 2022-06-17 18:20:52 UTC (0:16:20 ago)
  size: 999.0 MiB (1047527424 bytes)
  isolabel: rhcos-410.84.202201251210-0
  keys: []
  name: rhcos
  urls: []
```

Now create a boot config that uses this repo:

```
scyld-bootctl create name=rhcosBoot repo=rhcos
```

Examine the details of the boot config:

```
[scyld-bootctl]$ scyld-bootctl -i rhcosBoot ls -l
Boot Configurations
rhcosBoot
  cmdline: coreos.live.rootfs_url=<BASE_URL>/repo/rhcos/content/images/pxeboot/rootfs.
  image: none
  initramfs: repo:images/pxeboot/initrd.img
  kernel: repo:images/pxeboot/vmlinux
```

(continues on next page)
Note the _coreos_install_dev and _coreos_ignition_url attributes in the cmdline. These attributes are set by the scyld-nodectl tool for the specific node(s) that use the rhcosBoot boot config.

For example:

```
scyld-nodectl -in0 set _boot_config=rhcosBoot \
    _coreos_ignition_url=http://10.20.30.40/path/path/ignition.ign \
    _coreos_install_dev=/dev/sda
```

For further information and examples about ignition files, see https://cloud.redhat.com/blog/provision-red-hat-coreos-rhcos-machines-with-custom-v3-ignition-files.

This variable replacement scheme is similar to the variable replacement in kickstart *.ks files.

### 6.8 Booting Diskful Compute Nodes

In addition to booting diskless clients, ClusterWare can also integrate with so-called "diskful" compute nodes that boot from full installations on local disk drives. (See creating-nodes-with-kickstart and creating-nodes-with-preseed for examples.)

The administrator can add a locally installed node to the cluster using the same mechanisms as done with a diskless node. For example, if the new node's network interface is physically connected to the private cluster network shared by existing head node(s) and compute nodes, and if that interface is configured to use a dynamic IP address that will be assigned by DHCP at boot time, then on the head node execute:

```
scyld-nodectl create mac=00:11:22:33:44:55
```

and ClusterWare will assign the next available IP address to that MAC address.

Or if the new node has a static IP address, first ensure that static address is in the defined ClusterWare DHCP range, then additionally specify that static address:

```
scyld-nodectl create mac=00:11:22:33:44:55 index=100 ip=10.10.42.100
```

For the new node to fully integrate as a ClusterWare compute node, the cluster administrator as user root should install on the new node the clusterware-node package and dependencies. Since the node will not be using the initrims provided by a ClusterWare boot configuration for diskless nodes, then after installing clusterware-node the administrator needs to update the configuration file /opt/scyld/clusterware-node/etc/node.sh on the node, which instructs the node how to communicate with a head node. This file must define the base_url of a head node and optionally the iface network interface name that assigned the MAC address used in the scyld-nodectl create that added the node to the database. For example, for interface eth0 that will connect to head node head-01, after editing:

```
[root@newnode]$ cat /opt/scyld/clusterware-node/etc/node.sh
# Specify a base_url for any head node
base_url=http://head-01/api/v1

# Specify the network interface used to reach the head node(s)
iface=eth0
```

(continues on next page)
On the node manually execute:

```
/usr/bin/update-node-status --hardware --upload
```

which sends the initial node hardware information to the head node, then reboot the node, which will now be fully integrated into the cluster.

The newly booted compute node can be controlled through the customary `scyld-nodectl reboot`, `shutdown`, and `exec` commands. To support `--hard` mode, the administrator must configure the node's `power_uri` field to provide appropriate `ipmitool` authentication and an IP address of the node's Baseboard Management Controller (BMC), e.g., `ipmi://admin:password@172.45.88.1`. See `Compute Nodes IPMI access` and `Database Objects Fields and Attributes` for details.

### 6.9 Interacting with Compute Nodes

The primary tool for interacting with nodes from the command line is `scyld-nodectl`. This tool is how an administrator would add a node, set or check configuration details of a node, see the basic node hardware, see basic status, cause a node to join or leave attribute groups, reboot or powerdown a node, or execute commands on the node.

In this section we will show a number of examples and discuss what information an administrator can both get and set through the `scyld-nodectl` tool, as well as reference other resources for further details.

Nodes are named by default in the form of `nX`, where `X` is a numeric zero-based index. More complicated clusters may benefit from more flexible naming schemes. See `Node Names and Pools` for details.

#### 6.9.1 Node Creation with Known MAC address(es)

When a new node's MAC address is known to the cluster administrator, the simplest method is add the node to the cluster is to use `scyld-nodectl create` action and supply that node's MAC address:

```
```

and the node is assigned the next available node index and associated IP address.

The administrator can also add the node at an index other than the next available index, e.g., to add a node n10:

```
```

Of course, if a node already exists for the specified MAC or index, then an error is returned and no node is created. Adding nodes one at a time would be tedious for a large cluster, so an administrator can also provide JSON formatted content to the create action. For example,

```
scyld-nodectl create --content @path/to/file.json
```

where that `file.json` contains an array of JSON objects, each object describing a single node, e.g., for two nodes:
The `content` argument can also directly accept JSON, or an INI formatted file, or a specially formatted text file. Details of how to use these alternative formats are available in the Reference Guide in Introduction to Tools.

### 6.9.2 Node Creation with Unknown MAC address(es)

A reset or powercycle of a node triggers a DHCP client request which embeds the node’s MAC address. A head node with an interface that is listening on that private cluster network and which recognizes that MAC address will respond with an IP address that is associated with that MAC, unless directed to ignore that node. A ClusterWare head node can be so directed to ignore the known-MAC node by using a `_no_boot` attribute (see `_no_boot`), and a ClusterWare 6 or 7 master node can employ a `/etc/beowulf/config` file `masterorder` configuration directive to consider this known-MAC node to be owned by another head/master node.

A ClusterWare DHCP server which does not recognize the incoming MAC will by default ignore the incoming DHCP client request. To override this default:

```bash
csyld-clusterctl --set-accept-nodes True
```

and then any head node that shares the same database will add that new MAC to the shared ClusterWare database, assign to it the next available node index and associated IP address, and proceed to attempt to boot the node.

If a ClusterWare 6 or 7 `beoserv` daemon is alive and listening on the same private cluster network, then that master node should have its `/etc/beowulf/config` specify `nodeassign locked`, which directs its `beoserv` to ignore unknown MAC addresses.

When all new nodes with previously unknown MAC addresses are thus merged into the ClusterWare cluster, then the cluster administrator should again reenable the default functionality with:

```bash
csyld-clusterctl --set-accept-nodes False
```

If multiple new nodes concurrently initiate their DHCP client requests, then the likely result is a jumbled assignment of indices and IP addresses. Cluster administrators often prefer nodes in a rack to have ordered indices and IP addresses. This ordered assignment can be accomplished by performing subsequent carefully crafted `scyld-nodectl update` actions, e.g.,

```bash
csyld-nodectl -i n10 update index=100
scyld-nodectl -i n11 update index=101
scyld-nodectl -i n12 update index=102
scyld-nodectl -i n10,n11,n12 reboot # at a minimum, reboot the updated nodes
```

**Note:** Desired ordering can more easily be accomplished by performing the initial node resets or powercycling for each individual node in sequence, one at a time, and allowing each node to boot and get added to the database before initiating the next node’s DHCP request.
6.9.3 Changing IP addresses

To change IP addresses on a cluster, generate a configuration file of the currently state of the nodes with their current IP addresses, edit the file to change one or more IP addresses as desired, re-load the file, and trigger the head node to recompute the new addresses and update the database. For example:

```
scyld-cluster-conf save new_cluster.conf
# manually edit new_cluster.conf to change IP addresses
scyld-cluster-conf load new_cluster.conf
scyld-nodectl -i <NODES_THAT_CHANGE> update ip=
```

The new addresses are not seen by compute nodes until they reboot or perform a dhcp renewal.

6.9.4 Replacing Failed Nodes

Since nodes are identified by their MAC addresses, replacing a node in the database is relatively simple. If the node (n23 in the following example) was repaired but the same network interface is still being used, then no changes are necessary; however, if it was the network card that failed and it was replaced, then the node’s MAC address can be updated with one command:

```
```

If the entire node was replaced, then instead of just updating the MAC address, the administrator would likely prefer to clear the node status and any history associated with that node. To do this, delete and recreate the failed node:

```
scyld-nodectl -i n23 delete
```

6.9.5 Node Name Resolution

The scyld-install script installs the *clusterware-dnsmasq* package which provides resolution services for head node names. Similar to the *clusterware-iscdhcp*, this package depends on a standard OS provided service, although runs a private instance of that service, configuring it through the templated configuration file `/opt/scyld/clusterware-dnsmasq/dnsmasq.conf.template`. Within that file, fields like `"<DOMAIN>"` are substituted with appropriate values from the cluster network configuration, and the resulting file is rewritten.

Specifically, the "domain" field (defaulting to `.cluster.local`) is appended to compute node names (n0, n1, etc.) to produce a fully-qualified domain name. That default value can be overridden in the cluster configuration provided at installation time or loaded via the `scyld-cluster-conf` command. Multiple domains can be defined in that configuration file and are applied to any subsequently defined network segments until a later line sets a new domain value. Note that when changing this value on an established cluster, the cluster administrator may want to only load the networking portion of the cluster configuration instead of recreating already configured compute nodes:

```
scyld-cluster-conf load --nets-only cluster.conf
sudo systemctl restart clusterware
```

By default, any hosts listed in the `/etc/hosts` file on the head node will also resolve on the compute nodes through dnsmasq. This setting and many others can be changed in the dnsmasq configuration template.

An administrator may modify the template file to completely remove the domain or to otherwise modify the dnsmasq configuration. Please see the dnsmasq project documentation for details of the options that service supports. Similarly, the dhcpd configuration template is located at `/opt/scyld/clusterware-iscdhcp/dhcpd.conf.template`, although as that service is much more integral to the proper operation of ClusterWare, changes should be kept to an
absolute minimum. Administrators of more complicated clusters may add additional "options" lines or similarly remove the "option domain-name" line depending on their specific network needs. Additional DNS servers can also be provided to compute nodes through the "option domain-name-servers" lines. As with dnsmasq, please see the ISC DHCP documentation for supported options.

During compute node boot, dracut configures the bootnet interface of the node with the DNS servers and other network settings. These settings may be changed by cluster administrators in startup scripts as long as the head node(s) remain accessible to the compute nodes and vice versa.

During initial installation, the scyld-install script attempts to add the local dnsmasq instance (listening on the standard DNS port 53) as the first DNS server for the head node. If this is unsuccessful, DNS resolution will still work on compute nodes, although the administrator may need to add local DNS resolution before ssh and similar tools can reach the compute nodes. Please consult your Linux distribution documentation for details. Note that DNS is not used for compute node name resolution within the REST API or by the ClusterWare administrative tools; rather, the database is referenced in order to map node ids to IP addresses.

6.9.6 Executing Commands

A cluster administrator can execute commands on one or more compute nodes using the scyld-nodectl tool. For example:

```
scyld-nodectl -i n0 exec ls -l /
```

passes the command, e.g. `ls -l /`, to the head node, together with a list of target compute nodes. The head node will then ssh to each compute node using the head node's SSH key, execute the command, and return the output to the calling tool that will display the results. Note that this relay through the REST API is done because the ClusterWare tools may be installed on a machine that is not a head node and is not able to directly access the compute nodes. An optional --direct option will bypass this step and ssh directly to the compute nodes to execute the command. For example:

```
scyld-nodectl -i n0 exec --direct ls -l /
```

This direct access requires that the IP addresses of the compute nodes be accessible from the machine where the tool is being executed, and that appropriate SSH keys are configured for password-less access and the machine does support the use of a local ssh-agent. Please refer to Securing the Cluster for key management details. Note that even if DNS resolution of compute node names is not possible on the local machine, scyld-nodectl exec will still work because it retrieves the node IP addresses from the ClusterWare database via the head node. Further, once an administrator has appropriate keys on the compute nodes and has DNS resolution of compute node names, they are encouraged to manage nodes either directly using the ssh or pdsh commands or at a higher level with a tool such as ansible.

Commands executed through scyld-nodectl exec are executed in parallel across the selected nodes. By default 64 nodes are accessed at a time, but this is adjustable by setting the ssh_runner.fanout to a larger or smaller number. This variable can be set in an administrator's `~/.scyldcw/settings.ini` or can be set in `/opt/scyld/clusterware/conf/base.ini` on a head node. All commands run with --direct will use the administrator's preference, while other commands that pass through the head node will use the value in base.ini. Setting the ssh_runner.fanout variable to a value less than or equal to 1 causes all commands to be executed serially across the nodes.

Some limited support is also provided for sending content to the stdin of the remote command. That content can be provided in a file via an option, e.g.:

```
scyld-nodectl -i n0 exec --stdin=@input.txt dd of=/root/output.txt
```

or the content can be provided directly:

```
scyld-nodectl -i n0 exec --stdin='Hello World' dd of=/root/output.txt
```
or the content can be piped to `scyld-nodectl`, and this time optionally using redirection on the compute node to write to the output file:

```bash
echo 'Hello world' | scyld-nodectl -i n0 exec cat > /root/output.txt
```

When a command is executed on a single node, the command's stdout and stderr streams will be sent unmodified to the matching file descriptor of the `scyld-nodectl` command. This allows an administrator to include remote commands in a pipe much like ssh. For example:

```bash
echo 'Hello world' | scyld-nodectl -i n0 exec tr 'a-z' 'A-Z' > output.txt
```

will result in a the local file `output.txt` containing the text "HELLO WORLD". The `scyld-nodectl` exec exit code will also be set to the exit code of the underlying command. When a command is executed on multiple nodes, the individual lines of the resulting output will be prefixed with the node names:

```bash
[admin@virthead]$ scyld-nodectl -in[0-1] exec ls -l
n0: total 4
  -rw-r--r-- 1 root root 13 Apr 5 20:39 output.txt
n1: total 0
```

When executing a command on multiple nodes, the exit code of the `scyld-nodectl` exec command will only be 0 if the command exits with a 0 on each node. Otherwise the tool return code will match the non-zero status of the underlying command from one of the failing instances.

The mechanism for passing stdin should not be used to transfer large amounts of data to the compute nodes, as the contents will be forwarded to the head node, briefly cached, and copied to all compute nodes. Further, if the data was passed as a stream either through piping to the `scyld-nodectl` command or passing the path to a large file via the `--stdin=@/path/to/file` mechanism, the nodes will be accessed serially, not in parallel, so that the stream can be rewound between executions. This is supported for convenience when passing small payloads, but is not efficient in large clusters. A more direct method such as `scp` or `pdcp` should be used when the content is more than a few megabytes in size. Also note that even when communicating with a single compute node, this is not truly interactive because all of stdin must be available and sent to the head node before the remote command is executed.

### 6.9.7 Node Attributes

The names and uses of the fields associated with each database object are fixed, although nodes may be augmented with attribute lists for more flexible management. These attribute lists are stored in the `attributes` field of a node and consist of names (ideally legal Javascript variable names) and textual values. Attribute names prefixed with an underscore such as `_boot_config` or `_boot_style` are reserved for use by ClusterWare. These attributes may be referenced or modified by administrator defined scripting, but changing their values will modify the behavior of ClusterWare.

Beyond their internal use, e.g. for controlling boot details, attributes are intended for use by cluster administrators to mark nodes for specific purposes, record important hardware and networking details, record physical rack locations, or whatever else the administrator may find useful. All attributes for a given node are available and periodically updated on the node in file `/opt/scyld/clusterware-node/etc/attributes.ini`. This directory `/opt/scyld/clusterware-node/etc/` is also symlinked to `/etc/clusterware`.

Attributes can also be collected together into `attribute groups` that are stored separately from the node database objects. Administrators can then assign nodes to these groups and thereby change the attributes for a selection of nodes all at once.

Each node has a list of groups to which it belongs, and the order of this list is important. Attribute groups appearing later in the list can override attributes provided by groups earlier in the list. For any given node there are two special groups: the global default group and the node-specific group. The global default group, which is defined during the installation process and initially named "DefaultAttribs", is always applied first, and the node-specific group contained
in the node database object is always applied last. Any attribute group can be assigned to be the default group through the `scyld-clusterctl` command, e.g.,

```
scyld-clusterctl --set-group GroupNameOrUID
```

An example should clarify how attributes are determined for a node. Immediately after installation the "DefaultAttribs" group contains a single value:

```
[example@head ~]$ scyld-attribctl ls -l
Attribute Groups
   DefaultAttribs
      attributes
        _boot_config: DefaultBoot
```

Note that fields extraneous to this example have been trimmed from this output, although some are discussed further in the *Reference Guide*. Looking at two nodes on this same cluster:

```
[example@head ~]$ scyld-nodectl ls -l
Nodes
   n0
      attributes:
        _boot_config: DefaultBoot
        groups: []
   n1
      attributes:
        _boot_config: DefaultBoot
        groups: []
```

By default no attributes are defined at the node level, although all nodes inherit the `_boot_config` value from the "DefaultAttribs" group. If an administrator creates a new boot configuration (possibly by using the `scyld-add-boot-config` script mentioned earlier) and calls it "AlternateBoot", then she could assign a single node to that configuration using the `scyld-nodectl` tool, e.g.,

```
scyld-nodectl -i n0 set _boot_config=AlternateBoot
```

Examining the same nodes after this change would show:

```
[example@head ~]$ scyld-nodectl ls -l
Nodes
   n0
      attributes:
        _boot_config: AlternateBoot
        groups: []
   n1
      attributes:
        _boot_config: DefaultBoot
        groups: []
```

Of course, managing nodes by changing their individual attributes on a per-node basis is cumbersome in larger clusters, so a savvy administrator can create a group and assign nodes to that group:

```
scyld-attribctl create name=AltAttribs
scyld-attribctl -i AltAttribs set _boot_config=ThirdBoot
```
Assigning additional nodes to that group is done by "joining" them to the attribute group:

```
scyld-nodectl -i n[11-20] join AltAttribs
```

After the above changes, node n0 is assigned to the "AlternateBoot" configuration, n11 through n20 would boot using the "ThirdBoot" configuration, and any other nodes in the system will continue to use "DefaultBoot". This approach allows administrators to efficiently aggregate a set of nodes in anticipation of an action against the entire set, for example when testing new images, or if some nodes need specific configuration differences due to hardware differences such as containing GPU hardware.

For a more technical discussion of setting and clearing attributes as well as nodes joining and leaving groups, please see the appropriate section of the Reference Guide.

### 6.9.8 Node Names and Pools

By default all compute nodes are named nX, where X is a numeric zero-based node index. This pattern can be changed using "nodename" lines found in a cluster configuration file. For example, a line "nodename compute{}" early in such a file will change the default node naming to computeX. This changes both the default node hostnames as well as the names recognized by the scyld-nodectl command.

For homogeneous clusters where all compute nodes are essentially the same, this is usually adequate, although in more complex environments there is utility in quickly identifying core compute node capabilities reflected by customized hostnames. For example, high memory nodes and general purpose GPU compute nodes could be named "hmX" and "gpgpuX". These names can be assigned via the _hostname attribute as described in Reserved Attributes, although the scyld-nodectl command will still refer to them as "nX".

To support multiple name groupings within the scyld-*ctl tools, the ClusterWare system includes the concept of a naming pool. These pools are defined and modified through the scyld-clusterctl pools command line interface. Once the appropriate pools are in place, then compute nodes can be added to those pools. Continuing the example described previously:

```
scyld-clusterctl pools create name=high_mem pattern=hm{} first_index=1
scyld-clusterctl pools create name=general_gpu pattern=gpgpu{} first_index=1
scyld-nodectl -i [37-40] update naming_pool=high_mem
scyld-nodectl -i [41,42] update naming_pool=general_gpu
```

After these changes the scyld-nodectl status and scyld-nodectl ls output will include the specified nodes as "hm[1-4]" and "gpgpu[1-2]". Any commands that previously used "nX" names will then accept "hmX" or "gpgpuX" names to refer to those renamed nodes. The first_index= field of the naming pool forces node numbering to begin with a specific value, defaulting to 0. Any nodes not explicitly attached to a naming pool will use the general cluster naming pattern controlled through the scyld-clusterctl --set-naming PATTERN command. This can be considered the default naming pool.

**Important:** Please note that when moving multiple compute nodes from one naming pool to another, that the node order may not be preserved. Instead, moving them individually, or specifying their MAC addresses in a cluster configuration file, may be more predictable.

When moving a node from one naming pool to another via the scyld-nodectl command, the node index will be reset to the next available index in the destination pool. Using an explicit index=X argument allows the cluster administrator to directly control the node renumbering. Note that nodes in different naming pools may have the same index, so in this configuration the index is no longer a unique identifier for individual nodes. Further, the --up, --down, --all node selectors are not restricted to a single naming pool and will affect nodes in all pools that match the selection constraint. Nodes in scyld-nodectl output will be ordered by index within their naming pool, although the order of the naming pools themselves is not guaranteed. For example:
Similarly, the nodes are grouped by naming pool in `scyld-cluster-conf` save output with "nodename" lines and explicit node indices inserted as needed:

```bash
[admin@head clusterware]$ scyld-cluster-conf save -
# Exported Scyld ClusterWare Configuration file
# This file contains the cluster configuration.
# Details of the syntax and semantics are covered in the
# Scyld ClusterWare Administrators Guide.
#
# nodename n{}

# 10.10.24.0/24 network
domain cluster.local
1 10.10.24.101/24 10.10.24.115
node 1 00:00:00:00:00:01 # n1
node 00:00:00:00:00:02 # n2
node 00:00:00:00:00:03 # n3
node 00:00:00:00:00:04 # n4
node 00:00:00:00:00:05 # n5
nodename login{}
node 6 00:00:00:00:00:06 # login6
node 00:00:00:00:00:07 # login7
node 00:00:00:00:00:08 # login8
node 00:00:00:00:00:09 # login9
```

The organization of node naming pools is intentionally independent of node networking considerations. The cluster administrator may choose to combine these concepts by creating separate naming pools for each network segment, although this is not necessary.

Secondary DNS names can also be defined using "nodename":

```
nodename <pattern> <ip> [pool_name]
```

A "nodename" line containing an IP address (or IP offset such as “0.0.1.0”) can define a name change at an offset within the IP space or define a secondary DNS name depending on whether the IP is within a defined network. For example:

```bash
iprange 10.10.124.100/24 10.10.124.250
node
node 08:00:27:F0:44:35 # n1 @ 10.10.124.101

nodename hello{}/5 10.10.124.105
```

(continues on next page)
Note that the "<pattern>/X" syntax defines the lowest node index allowed within the naming pool.

### 6.9.9 Attribute Groups and Dynamic Groups

The `scyld-install` script creates a default attribute group called `DefaultAttribs`. That group can be modified or replaced, although all nodes are always joined to the default group. The cluster administrator can create additional attribute groups, e.g.:

```bash
scyld-attribctl create name=dept_geophysics
scyld-attribctl create name=dept_atmospherics
scyld-attribctl create name=gpu
```

and then assign or remove one or more groups to specific nodes, e.g.:

```bash
scyld-nodectl -i n[0-7] join dept_geophysics
scyld-nodectl -i n[8-11] join dept_atmospherics
scyld-nodectl -i n[0-3,7-9] join gpu
scyld-nodectl -i n7 leave gpu
```

These group assignments can be viewed either by specific nodes:

```bash
scyld-nodectl -i n0 ls -l
scyld-nodectl -i n[4-7] ls -l
```

or as a table:

```bash
[admin]$ scyld-nodectl --fields groups --table ls -l
```

<table>
<thead>
<tr>
<th>Nodes</th>
<th>groups</th>
</tr>
</thead>
<tbody>
<tr>
<td>n0</td>
<td>['dept_geophysics', 'gpu']</td>
</tr>
<tr>
<td>n1</td>
<td>['dept_geophysics', 'gpu']</td>
</tr>
<tr>
<td>n2</td>
<td>['dept_geophysics', 'gpu']</td>
</tr>
<tr>
<td>n3</td>
<td>['dept_geophysics', 'gpu']</td>
</tr>
<tr>
<td>n4</td>
<td>['dept_geophysics']</td>
</tr>
<tr>
<td>n5</td>
<td>['dept_geophysics']</td>
</tr>
<tr>
<td>n6</td>
<td>['dept_geophysics']</td>
</tr>
<tr>
<td>n7</td>
<td>['dept_geophysics']</td>
</tr>
<tr>
<td>n8</td>
<td>['dept_atmospherics', 'gpu']</td>
</tr>
<tr>
<td>n9</td>
<td>['dept_atmospherics', 'gpu']</td>
</tr>
<tr>
<td>n10</td>
<td>['dept_atmospherics']</td>
</tr>
<tr>
<td>n11</td>
<td>['dept_atmospherics']</td>
</tr>
</tbody>
</table>
```

(continues on next page)
Scyld commands that accept group lists can reference nodes by their group name(s) (expressed with a % prefix) instead of their node names, e.g.:

\[
\begin{array}{c|c}
\text{n12} & [] \\
\text{n13} & [] \\
\text{n14} & [] \\
\text{n15} & [] \\
\end{array}
\]

Both the Kubernetes scyld-kube --init command (see *Kubernetes*) and the Job Scheduler $\{jobsched\}$-scyld. setup init, reconfigure, and update-nodes actions accept --ids %<GROUP> as well as --ids <NODES> (see *Job Schedulers*).

In addition to attribute groups, ClusterWare also supports admin-defined dynamic groups using a query language that allows for simple compound expressions. These expressions can reference individual attributes, group membership, hardware fields, or status fields. For example, suppose we define attribute groups "dc1" and "dc2":

\[
\begin{array}{c}
\text{scyld-attribctl create name=dc1 description='Data center located in rear of building 1'} \\
\text{scyld-attribctl create name=dc2 description='Data center in building 2'} \\
\end{array}
\]

and then add nodes to appropriate groups:

\[
\begin{array}{c}
\text{scyld-nodectl -i n[0-31] join dc1} \\
\text{scyld-nodectl -i n[32-63] join dc2} \\
\end{array}
\]

and for each node, identify its rack number in an attribute:

\[
\begin{array}{c}
\text{scyld-nodectl -i n[0-15] set rack=1} \\
\text{scyld-nodectl -i n[16-31] set rack=2} \\
\text{scyld-nodectl -i n[32-47] set rack=1} \\
\text{scyld-nodectl -i n[48-63] set rack=2} \\
\end{array}
\]

Note that all attribute values are saved as strings, not integers, so that subsequent selector expressions must enclose these values in double-quotes.

Now you can query a list of nodes in a particular rack of a particular building using a --selector (or -s) expression, and perform an action on the results of that selection:

\[
\begin{array}{c}
\text{scyld-nodectl -s 'in dc1 and attributes[rack] == "2"' status} \\
\text{# or use 'a' as the abbreviation of 'attributes'} \\
\text{scyld-nodectl -s 'in dc1 and a[rack] == "2"' set _boot_config=TestBoot} \\
\text{# Show the nodes that have 32 CPUs.} \\
\text{# These hardware _cpu_count values are integers, not strings, and are} \\
\text{# not enclosed in double-quotes.} \\
\text{scyld-nodectl -s 'hardware[cpu_count] == 32' ls} \\
\text{# or use 'h' as the abbreviation of 'hardware'} \\
\text{scyld-nodectl -s 'h[cpu_count] == 32' ls} \\
\text{# Show the nodes that do not have 32 CPUs} \\
\text{scyld-nodectl -s 'h[cpu_count] != 32' ls} \\
\end{array}
\]
You can also create a dynamic group of a specific selector for later use:

```
scyld-clusterctl dyngroups create name=b1_rack1 selector='in dc1 and a[rack] == "1"'
scyld-clusterctl dyngroups create name=b1_rack2 selector='in dc1 and a[rack] == "2"'
```

# Show the nodes in building 1, rack 2
scyld-nodectl -i %b1_rack2 ls

# Show only those %b1_rack2 nodes with 32 CPUs
scyld-nodectl -i %b1_rack2 -s 'h[cpu_count] == 32' ls

You can list the dynamic groups using scyld-clusterctl:

```
# Show the list of dynamic groups
[admin1@headnode1 ~]$ scyld-clusterctl dyngroups ls
Dynamic Groups
  b1_rack1
  b1_rack2
```

And show details of one or more dynamic group. For example:

```
# Show the selector associated with a specific dynamic group
[admin1@headnode1 ~]$ scyld-clusterctl dyngroups -i b1_rack1 ls -l
Dynamic Groups
  b1_rack1
    name: b1_rack1
    selector: in dc1 and a[rack] == "1"

# Or show the selector associated with a specific dynamic group in full detail
[admin1@headnode1 ~]$ scyld-clusterctl dyngroups -i b1_rack1 ls -L
Dynamic Groups
  b1_rack1
    name: b1_rack1
    parsed: ((in "dc1") and (attributes["rack"] == "1"))
    selector: in dc1 and a[rack] == "1"
```

The parsed line in the above output can be useful when debugging queries to confirm how Scyld parsed the provided query text.
6.10 Securing the Cluster

This *Installation & Administrator Guide* section discusses cluster security issues that are exclusive to Scyld ClusterWare. We assume that the cluster administrator is familiar with security issues that are not solely related to ClusterWare, such as securing the cluster from outside access, optionally enabling various Red Hat RHEL/CentOS functionalities for logging and auditing access to nodes and storage and for managing SELinux.

### 6.10.1 Authentication

The cluster administrator authentication method is controlled in the /opt/scyld/clusterware/conf/base.ini file by the `plugins.auth` variable and is initially set to "dummy". This plugin is the least secure and accepts any password for a known administrator, providing very little security. The initial list of known administrators is stored in the same file in the `auth.tmpadmins` variable. The `scyld-install` installation will (unless passed the `--no-tools` argument) add the current user to that comma separated list of user names.

Any administrator can add additional administrators through the `scyld-adminctl` command whose arguments match the other `scyld-*ctl` commands as described in the *Reference Guide*. We suggest that administrators add accounts for themselves through this tool, and thereafter clear the `auth.tmpadmins` variable. This variable is only intended to be used during early installation, for small experimental clusters, or when recovering from some sort of failure.

When deploying ClusterWare, the `plugins.auth` variable will be set to "appauth". This plugin executes the command defined in the `appauth.app_path` variable as user `root`. The default implementation of that command is provided by `/opt/scyld/clusterware/bin/pam Authenticator`. This implementation interfaces with the PAM authentication system using the `/etc/pam.d/cw_check_user` configuration file. The contents of this file initially use local system authentication, although this can be modified to authenticate against any mechanism available through the PAM system. Please see PAM documentation provided by your distro as well as the main PAM project. See the Red Hat [https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/pluggable_authentication_modules documentation](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/pluggable_authentication_modules documentation).

Administrators can provide authentication methods beyond PAM by implementing a script or application and providing it via the `appauth.app_path` variable. Appropriate applications should start with no arguments, read a username and password separated by a newline from `stdin`, and reply with either `yes` or `no` followed by a newline on `stdout`. For example, a test run of `pam Authenticator` looks like:

```
[example@head ~] sudo /opt/scyld/clusterware/bin/pam Authenticator

tester
not_the_password
no
```

### 6.10.2 Changing the Database Password

The `scyld-install` installation configures the ClusterWare database with a randomly generated password. This password is used when joining a new head node to the cluster and must be provided either through a command line or on request during the installation of the new head node. This password is stored in the `database.admin_pass` variable in the `/opt/scyld/clusterware/conf/base.ini` file. The details of changing this password depend on the specific database the cluster is using.

**Important:** Once this password is changed within the database, change the `database.admin_pass` variable in `base.ini` and restart the `clusterware` service on each head node.
6.10.2.1 Couchbase

Use the Couchbase console available on every head node on port 8091 to change the Couchbase password. Details of how to change this password can be found in the Couchbase documentation. See the "MANAGING CLUSTERS" section on the https://docs.couchbase.com/server/5.1/introduction/intro.html web page.

6.10.2.2 etcd

Use the etcdctl tool (provided in the clusterware-etcd package) to change the etcd password:

```
/opt/scyld/clusterware-etcd/bin/etcdctl user passwd root
```

6.10.3 Compute Node Remote Access

By default, remote access to compute nodes is provided through SSH using key-based authentication, although administrators may also enable password-based SSH in the compute node image by configuring a password for the root user. Every head node generates a public/private key pair and places these files in directory /opt/scyld/clusterware/.ssh/ using the names id_rsa.clusterware and id_rsa.clusterware.pub. These keys are used by the head nodes to execute commands on the compute nodes. All head node public keys are downloaded by compute nodes at boot time by the update_keys.sh script and appended to /root/.ssh/authorized_keys. This allows any head node to execute a command on any compute node. The /opt/scyld/clusterware/.ssh/id_rsa.clusterware key can be used by system administrators as an "automation" key for tasks like cron jobs. It is also useful in recovery situations where an administrator may need to use this private key to directly access compute nodes.

This same script that downloads the head node public keys will also download the public keys attached to every cluster administrator account. These accounts are created using the scyld-adminctl tool as follows:

```
scyld-adminctl create name=admin keys=@~/.ssh/id_rsa.pub
```

This would allow anyone with the corresponding id_rsa to SSH into the root account on any compute node booted after the key was added. The key can also be added as a string or updated for an existing administrator. For example,

```
scyld-adminctl -i admin update keys='ssh-rsa AAAAB3NzaC1yc2EAAA...'
```

Cluster administrators are also welcome to add SSH keys to compute node images in small private clusters, although adding administrator accounts with public keys simplifies management of larger clusters with multiple node images or cluster administrators. Note that administrator accounts stored in the database or listed in the base.ini both use the same authentication mechanisms described in the previous section.

**Important:** We urge cluster administrators to create their own administrator accounts and remove their usernames from the base.ini file after cluster installation.
6.10.4 Compute Node Host Keys

In most computer systems the SSH sshd daemon uses unique host keys to identify itself to clients, and host keys are not created during image creation. This means that each compute node will generate its own host keys during boot. Since the compute node changes are discarded on reboot, a new set of keys will be generated with each boot.

In an appropriately protected cluster, some administrators prefer for all compute nodes to share host keys. This can be achieved by storing host keys in the compute node image. For example, to generate host keys and repack the Default-Image, an administrator can run:

```
sclyld-modimg -i DefaultImage --exec sshd-keygen --overwrite --upload
```

All nodes that boot using this image after this change will use identical host keys, so ideally you should reboot the nodes with each node's updated image. To remove the host keys from an image, an administrator needs to delete the `/etc/ssh/ssh_host_*` files from the compute node image.

6.10.5 Encrypting Communications

By default the administrative tools communicate with the head node via HTTP, although they can also use HTTPS if appropriate certificates are configured on the head node's Apache web server. Please refer to documentation provided by your distro about how to properly enable HTTPS on the Apache server. Apache configuration files are located in `/opt/scyld/clusterware/conf/httpd/`. The Apache VirtualHost definition can be found in `vhost.conf`, and the proxy definition in that file will need to be included into the HTTPS VirtualHost.

Once HTTPS is enabled, the `~/.scyldcw/settings.ini` file of any existing ClusterWare tool installation should be updated. In that file the protocol of the `client.base_url` variable will need to be updated. It should be safe to leave HTTP enabled for localhost-only access, and in that case local tool installations can continue to use the original localhost-based URL.

6.10.6 Security-Enhanced Linux (SELinux)

Security-Enhanced Linux (SELinux) is a set of patches to the Linux kernel and various utilities that provide mandatory access control to major subsystems of a node. See https://en.wikipedia.org/wiki/Security-Enhanced_Linux for general discussion of SELinux.

ClusterWare supports SELinux on the head nodes and compute nodes.

6.10.6.1 SELinux On Compute Nodes

For Red Hat RHEL and CentOS compute nodes, the root file systems created by the `scyld-modimg` tool include SELinux support as part of the installation of the `@core` yum group. During the boot process the `mount_rootfs` script will, like the standard `dracut` based initramfs, load the SELinux policy before switching root. Note that the default cmdline in the boot configurations created through `scyld-add-boot-config` (including the DefaultBoot configuration) will contain `enforcing=0`, thereby placing all compute nodes in SELinux "permissive" mode. Only remove this option once you have completed testing to confirm that your applications will run as expected with SELinux in "enforcing" mode.

SELinux on compute nodes may be disabled in the standard ways through command line arguments or by changing the contents of the node's `/etc/selinux/config` configuration file. For details please refer to appropriate distro-provided documentation.

In addition to the default "targeted" SELinux policy provided by RHEL and CentOS, ClusterWare also supports the Multi-Level Security (MLS) policy for compute nodes. Enabling the MLS policy inside an image is done the same way as it would be done on a locally installed system. After entering the image chroot using `scyld-modimg`, first install...
the selinux-policy-mls package, and then modify the /etc/selinux/config file to reference the newly installed policy. Because the clusterware-node SELinux policy module is installed at image creation time, it may need to be re-installed after switching to the MLS policy:

```
semodule --install /opt/scyld/clusterware-node/clusterware-node.pp.bz2
```

The `semodule` command can also be used to check if the policy is loaded:

```
semodule --list | grep clusterware
```

When exiting the chroot, ClusterWare automatically relabels the file system based on the policy referenced in /etc/selinux/config.

**Important:** Fully configuring a cluster for MLS requires significant effort, including labeling objects on shared storage and defining additional policy around user workflows and tools. Please refer to your operating system documentation, as such details are beyond the scope of this document. Note that Scyld ClusterWare-provided schedulers, MPI implementations, and 3rd party applications may need additional custom permissions not covered here in order to configure a functional MLS cluster.

When creating boot configuration for an MLS enabled image, please be aware that the MLS policy, by default, does not allow the root user to log into the compute node via ssh. Because ssh is used by the ClusterWare soft power commands, please either enable the root login functionality or use the _remote_user node attribute to configure login as a user with sudo shutdown permission. The root login permission can be enabled through the `setsebool` command, and the boolean is named `ssh_sysadm_login`.

### 6.10.6.2 SELinux On Head Nodes

On head nodes, SELinux is detected to be in "enforcing" mode at both installation and service run time. To switch SELinux from "enforcing" to "permissive" mode, please see the documentation for your operating system. If this switch is made while the ClusterWare service is running, please restart that service:

```
sudo systemctl restart clusterware
```

### 6.10.6.3 MLS Policy On Head Nodes

For head nodes enforcing the MLS policy, the SELinux user `sysadm_u` should be used to install ClusterWare and run administrative tools.

To map a Linux user to the `sysadm_u` SELinux user, you can run:

```
sudo semanage login --add linux_user --seuser sysadm_u
```

By default, the `sysadm_u` user should run with the `sysadm_t` domain.
6.10.7 Security Technical Implementation Guides (STIG)


ClusterWare provides basic STIG support for kickstarted nodes by adding the following snippet to your kickstart *.ks file:

```bash
%addon org_fedora_oscap
  content-type = scap-security-guide
  profile = xccdf_org.ssgproject.content_profile_stig
%end
```

To configure a STIG head node, add the snippet to your kickstart config file and reboot the node using that *.ks file to enable STIG. Then install ClusterWare (Initial Installation of Scyld ClusterWare) on the STIG-enabled node in the usual way.

ClusterWare provides an example file /opt/scyld/clusterware/kickstarts/basic-stig.ks with that snippet appended for administrators who would like to kickstart infrastructure nodes or additional head nodes with that STIG applied at install time.

6.11 Monitoring the Status of the Cluster

ClusterWare provides two primary methods to monitor cluster performance and health: the command line scyld-nodectl status tool and the more extensive Chronograf GUI.

6.11.1 scyld-nodectl status

More basic node status can be obtained through the scyld-nodectl command. For example, a cluster administrator can view the status of all nodes in the cluster:

```bash
# Terse status:
[admin@virthead]$ scyld-nodectl status
n[0] up
n[1] down
n[2] new

# Verbose status:
[admin@virthead]$ scyld-nodectl status --long
Nodes
 n0
  ip: 10.10.24.100
  last_modified: 2019-04-16 05:02:26 UTC (0:00:02 ago)
  state: up
  uptime: 143729.68

n1
  down_reason: boot timeout
  ip: 10.10.42.102
  last_modified: 2019-04-15 09:03:20 UTC (19:59:08 ago)
  last_uptime: 59.61
```

(continues on next page)
From this sample output we can see that n0 is up and has recently (2 seconds earlier) sent status information back to the head node. This status information is sent by each compute node to its parent head node once every 10 seconds, although this period can be overridden with the _status_secs node attribute. The IP address shown here is the IP reported by the compute node and should match the IP provided in the node database object unless the database has been changed and the node has not yet been rebooted.

Compute node n1 is currently down because of a "boot timeout". This means that the node attempted to boot, and the node's initial "up" status message to the head node was not received. This could happen due to a boot failure such as a missing network driver, a networking failure preventing the node from communicating with the head node, or if the cw-status-updater service provided by the clusterware-node package is not running on the compute node. Other possible values for down_reason include "node stopped sending status" or "clean shutdown".

There is no status information about n2 because it was added to the system and has never been booted. Additional node status can be viewed with scyld-nodectl status --L (an abbreviation of --long-long) that includes the most recent full hostname, kernel command line, loaded modules, loadavg, free RAM, kernel release, and SELinux status. As with other scyld-*ctl commands, the output can also be provided as JSON to simplify parsing and scripting.

For large clusters the --long (or -l) display can be unwieldy, so the status command defaults to a summary. Each row of output corresponds to a different node status and lists the nodes in a format that can then be passed to the --ids argument of scyld-nodectl. Passing an additional --refresh argument will cause the tool to start an ncurses application that will display the summary in the terminal and periodically refresh the display:

```
scyld-nodectl status --refresh
```

This mode can be useful when adding new nodes to the system by booting them one at a time as described in Node Creation with Unknown MAC address(es).

### 6.11.2 Chronograf GUI

Access the Chronograf GUI through the ClusterWare GUI's "Monitoring" tab (see Graphical Interface). If the local server is a head node, you can use http://localhost/chronograf. To connect to another database-accessible node, e.g., named cwheadnode2, you can use http://cwheadnode2/chronograf.

**Note:** The URL http://cw11headnode/chronograf may differ if the cluster administrator has switched to HTTPS or otherwise modified the Apache configuration.

See https://docs.influxdata.com/chronograf/v1.7/ for extensive documentation about how to craft site-specific visualizations of cluster activity and status.

To facilitate monitoring of compute node GPU activity, first install into the GPU compute node image(s) the NVidia System Management Interface utility (nvidia-smi), which ships with NVidia GPU drivers. See https://developer.nvidia.com/nvidia-system-management-interface for details of that utility, and see https://www.cyberciti.biz/faq/how-to-install-nvidia-driver-on-centos-7-linux/ for a description of how to install NVidia drivers. Then in the compute node image(s) copy /etc/telegraf/telegraf.d/nvidia-smi.conf.example (distributed in the clusterware-node RPM) to /etc/telegraf/telegraf.d/nvidia-smi.conf.

In a browser click on the Log Viewer button on the left side of the window to see the head node and/or compute nodes' rsyslog.
To perform a simple setup, point your browser at http://localhost/chronograf, then click on the Configuration button at the lower left of the window (highlighted in blue as you pass your cursor over the vertical set of icons). On that Configuration screen, click on the green ClusterWare InfluxDB (Default) button:

The initial installation should have set up the connection to the database. Click on the blue Update Connection button to accept:
Click on the System dashboard to select it, then click on Create 1 Dashboard to create it:

Now confirm the Kapacitor connection to the database. Click on the blue Continue button:
And your setup is complete. Click on the green *Finish* button to confirm:

Back to the initial Chronograf window, click on the *Dashboards* button (highlighted in blue as you pass your cursor over the vertical set of icons), on the left side of the window:
And you see the System dashboard that was just set up, and its "Template Variables" is :host:

Click on the System dashboard to view it:
Note the :host: in the upper left of the window, which is a pulldown menu that shows the choices of which node to view. In this example there is a head node and two compute nodes.

Click on the Log Viewer button on the left side of the window to see the logs:
6.12 Graphical Interface

ClusterWare provides a graphical interface (GUI) for administrators to monitor and manage the cluster. Both the GUI and the command-line interfaces (detailed in Reference Guide) employ the same underlying interface to the ClusterWare database.

To use the GUI, point your browser at a head node (or at any node on which the clusterware-tools package has been installed), e.g., http://localhost if running on a head node, or http://cwheadnode2 to access another head node named cw-headnode2. You first see a login screen. If no cluster administrators have yet been declared (see the scyld-adminctl command), then login as root:

![Login Screen](image)

A successful login gets you to the home screen with its summary information. Note the top-level buttons across the top of the window, e.g., Monitoring, Nodes, Booting, etc., that presents additional information and features. You can always return to the home screen by clicking on the Penguin Computing Scyld ClusterWare logo in the top left of any GUI window.

The Monitoring button launches a Chronograf window. See Monitoring the Status of the Cluster for details.

![Cluster Status](image)

To add more administrators, click on Administrators and see the currently declared user names:

![Administrators](image)
After adding administrator *joe*:

Click on a specific node to see greater detail about that node:
The second top-level *Nodes* pulldown menu choice is *Status Grid*.

In that window you can choose *Filters* to display only *Up Nodes* or only *Down Nodes*, or for *Node Coloring* choose the *Code-Based Mappings*, or for *Odd-Node-Out* chose one of the *Variables* to highlight specific out-of-the-ordinary node states.

Click on the top-level *Images* to view the available *Node Images*: 
And click on a specific image to see more detail:

Click on the top-level Booting to view the available Boot Configurations:

And click on a specific Boot Configuration to see details:
Similarly, click on the top-level Attributes to view the available Attribute Groups:

And click on a specific Attribute Group to see details:
6.13 Managing Multiple Head Nodes

ClusterWare supports optional active-active(-active,...) configurations of multiple cooperating head nodes that share a single replicated database. Such multi-headnode configurations allow any head node to provide services for any compute node in the cluster. These services include cluster configuration using scyld-* tools, compute node booting and power control, and compute node status collection.

Currently the ClusterWare database requires a minimum of three cooperating head nodes to support full High Availability ("HA") in the event of head node failures. Couchbase HA can operate fully with two head nodes, albeit being unable to sustain full functionality with both head nodes failing concurrently. The etcd HA works in a limited manner with just two head nodes.

The command line tools provided by ClusterWare for head node management are intended to cover the majority of common cases, although head nodes using Couchbase can also be managed through the Couchbase console where more complicated recovery scenarios can be handled.

6.13.1 Adding A Head Node

After installing the first head node as described in Initial Installation of Scyld ClusterWare, additional head nodes can be installed and joined with the other cooperating head nodes using the same scyld-install tool or using curl.

On an existing head node view its database password:

```
sudo grep database.admin_pass /opt/scyld/clusterware/conf/base.ini
```

6.13.1.1 Join a non-ClusterWare server using scyld-install

For anon-ClusterWare server, you can use ClusterWare to join an existing head node that is identified by its IP address IP_HEAD:

```
sclyld-install --yum-repo clusterware.repo --database-passwd <DBPASS> --join <IP_HEAD>
```

where DBPASS is IP_HEAD's database password. If the --database-passwd is not provided as a scyld-install argument, then scyld-install queries the administrator interactively for that password.

The database type on all joined head nodes must be the same type. If needed, prefix scyld-install with (for example) DB_RPM=clusterware-couchbase to override the etcd database default if the existing head node(s) use Couchbase.

scyld-install needs to access a clusterware.repo file in order to install ClusterWare packages. This clusterware.repo file is created or accessed via one of the supported methods: a filename passed as a scyld-install --yum-repo argument that gets copied to /etc/yum.repos.d/clusterware.repo, an already existing /etc/yum.repos.d/clusterware.repo, or a file created by scyld-install using an authentication string that is passed as the scyld-install --token argument or when queried interactively by scyld-install.

scyld-install installs ClusterWare, replacing the local base.ini password with IP_HEAD's DBPASS password, then interacts with IP_HEAD to join with it and any other head nodes already joined.

A cluster configuration file is not required when joining a server to a head node because those settings are obtained from the existing head node's cluster database.
6.13.1.2 Join a ClusterWare head node using scyld-install

Alternatively, a "solo" ClusterWare head node can join an existing group of joined nodes.

**Important:** The join action discards the "solo" head node's current images and boot configs, then finishes leaving the "solo" head node with access to just the cooperating head nodes' images and boot configs. If you want to save any images or configs, then first use `scyld-bootctl export` (Copying boot configurations between head nodes) or `managedb save`.

For example, to join a ClusterWare head node:

```
scyld-install -u --database-passwd <DBPASS> --join <IP_HEAD>
```

Just as when joining a non-ClusterWare server, if no `--database-passwd` is provided as an argument, then `scyld-install` queries the administrator interactively for `IP_HEAD`'s database password.

6.13.1.3 Join a non-ClusterWare server using curl

The cluster administrator can use `curl` to download a modified version of the installer script with necessary parameters already embedded, and then pipe that script to `bash` to execute. This installs ClusterWare as a head node on the joining server and joins the cluster:

```
curl http://<IP_HEAD>/api/v1/install/head?passwd=<DBPASS> | bash
```

where `IP_HEAD` and `DBPASS` are described above.

6.13.1.4 Join a ClusterWare head node using curl

**Important:** The join action discards the "solo" head node's current images and boot configs, then finishes leaving the "solo" head node with access to just the cooperating head nodes' images and boot configs. If you want to save any images or configs, then first use `scyld-bootctl export` (Copying boot configurations between head nodes) or `managedb save`.

Alternatively, a "solo" ClusterWare head node can join an existing group of joined nodes using `curl` in the same manner a non-ClusterWare server joins a cluster, albeit with the additional `HEADNODE_JOIN=y` variable that passes to the downloaded installer script being executed by `bash`:

```
curl http://<IP_HEAD>/api/v1/install/head?passwd=<DBPASS> | HEADNODE_JOIN=y bash
```

where `IP_HEAD` and `DBPASS` are described above.
6.13.1.5 After a Join

**Important:** Every head node must know the hostname and IP address of every other head node, either by having those hostnames in each head node's `/etc/hosts` or by having their common DNS server know all the hostnames. Additionally, if using head nodes as default routes for the compute nodes, as described in Configure IP Forwarding, then ensure that all head nodes are configured to forward IP traffic preferably over the same routes.

**Important:** Every head node should use a common network time-sync protocol. The Red Hat RHEL default is `chronyd` (found in the `chrony` package), although `ntpd` (found in the `ntp` package) continues to be available.

After a Join, you should restart the `clusterware` service on all joined head nodes. Subsequent head node software updates are also accomplished by executing `scyld-install -u`. We recommend that all cooperating head nodes update to a common ClusterWare release. In rare circumstance a newer ClusterWare release on the head nodes also requires a compatible newer `clusterware-node` package in each compute node image. Such a rare coordinated update will be documented in the Release Notes and Changelog.

### 6.13.2 Removing a Joined Head Node

A list of connected head nodes can be seen with:

```
sudo /opt/scyld/clusterware/bin/managedb --heads
```

with more information visible doing:

```
scyld-clusterctl heads ls -l
```

For a cluster with two or more head nodes using a Couchbase database, or a cluster with three or more head nodes using an etcd database, you can remove one of the head nodes by doing:

```
sudo /opt/scyld/clusterware/bin/managedb leave
```

Or if that head node is shut down, then from another head node in the cluster doing:

```
sudo /opt/scyld/clusterware/bin/managedb eject <IP_HEAD_TO_REMOVE>
```

The now-detached head node will no longer have access to the shared database and will be unable to execute any `scyld-*` command, as those require a database. Either re-join the previous cluster:

```
sudo /opt/scyld/clusterware/bin/managedb join <IP_HEAD>
```

or join another cluster after updating the local `/opt/scyld/clusterware/conf/base.ini database.admin_pass` to the other cluster's database password:

```
sudo /opt/scyld/clusterware/bin/managedb join <IP_OTHER_HEADNODE>
```

or performing a fresh ClusterWare install by removing the current ClusterWare and continuing with a reinstall:

```
scyld-install --clear-all --config <CLUSTER_CONFIG>
```

However, for a cluster with only two head nodes using an etcd database, you cannot `managedb eject` or `managedb leave`, and instead must execute:
sudo /opt/scyld/clusterware/bin/managedb recover

on both head nodes. This severs each head node from their common coordinated access to the database.

**Important:** Keep in mind that following the `managedb recover`, both head nodes have autonomous and unsynchronized access to the now-severed database that manages the same set of compute nodes, which means that both will compete for "ownership" of the same booting compute nodes.

To avoid both head nodes competing for the same compute nodes, either execute `sudo systemctl stop clusterware` on one of the head nodes, or perform one of the steps described above to re-join this head node to the other head node that previously shared the same database, or join another head node, or perform a fresh ClusterWare install.

### 6.13.2.1 Configuring Support for Database Failover

When planning a multi-head cluster for true High Availability, a cluster administrator should allocate three or more head nodes. In this configuration, if one head node fails, then the database service on the remaining head nodes can be configured to automatically eject the failed node and recover with at most a short interruption in service. After one head node has failed, the cluster administrator must reset the auto-failover mechanism to avoid a single failure causing cascading ejections.

Complicated Couchbase recovery scenarios are managed by the cluster administrator interacting with the database console through a web browser: localhost:8091/ui.

The console username is `root` and the password can be found in the `database.admin_pass` variable in /opt/scyld/clusterware/conf/base.ini. Extensive documentation for this Couchbase console is available online on the Couchbase website: https://docs.couchbase.com/home/index.html. ClusterWare Couchbase is currently version 5.1.3.

To enable automatic failure of a head node in a multiple head node configuration, access the Couchbase console and click on **Settings** in the menu on the left side of the initial Dashboard window, then click on **Auto-Failure** in the horizontal list across the top of the Settings window. Then select **Enable auto-failure** and enter a preferred **Timeout** value, e.g., a default of 120 seconds. Finally, click the **Save** button.

In the discouraged dual-head configuration, a head node has no means to distinguish between a network bifurcation and the other node actually failing. To avoid a split-brain situation, the remaining head node must be explicitly told to take over for the failed node using the `managedb eject` command. Head node provided services will be interrupted until this ejection is triggered.

### 6.13.2.2 Shared Storage and Peer Downloads

Multi-head clusters can be configured to use shared storage among the head nodes, but by default each head will use its own local storage to keep a copy of each uploaded or requested file. The storage location is defined in /opt/scyld/clusterware/conf/base.ini by the `local_files.path` variable, and it defaults to /opt/scyld/clusterware/storage/.

Whenever a ClusterWare head node is asked for a file such as a kernel, the expected file size and checksum are retrieved from the database. If the file exists in local storage and has the correct size and checksum, then that local file will be provided. However, if the file is missing or incorrect, then the head node attempts to retrieve the correct file from a peer.

Note that local files whose checksums do not match will be renamed with a .old.NN extension, where NN starts at 00 and increases up to 99 with each successive bad file. This ensures that in the unlikely event that the checksum in the database is somehow corrupted, the original file can be manually restored.
Peer downloading consists of the requesting head node retrieving the list of all head nodes from the database and contacting each in turn in random order. The first peer that confirms that it has a file with the correct size provides that file to the requesting head node. The checksum is computed during the transfer, and the transferred file is discarded if that checksum is incorrect. Contacted peers will themselves not attempt to download the file from other peers in order to avoid having a completely missing file trigger a cascade.

After a successful peer download, the original requester receives the file contents after a delay due to the peer download process. If the file cannot be retrieved from any head node, then the original requester will receive a HTTP 404 error.

This peer download process can be bypassed by providing shared storage among head nodes. Such storage should either be mounted at the storage directory location prior to installation, or the /opt/scyld/clusterware/conf/base.ini should be updated with the non-default pathname immediately after installation of each head node. Remember to restart the clusterware service after modifying the base.ini file by executing `sudo systemctl restart clusterware`, and note that the systemd clusterware.service is currently an alias for the httpd.service.

When a boot configuration or image is deleted from the cluster, the deleting head node will remove the underlying file(s) from its local storage. That head node will also temporarily move the file's database entry into a deleted files list that other head nodes periodically check and delete matching files from their own local storage. If the clusterware service is not running on a head node when a file is marked as deleted, then that head node will not be able to delete the local copy. When the service is later restarted, it will see its local file is now no longer referenced by the database and will rename it with the .old.NN extension described earlier. This is done to inform the administrator that these files are not being used and can be removed, although cautious administrators may wish to keep these renamed files until they confirm all node images and boot configurations are working as expected.

### 6.13.3 Booting With Multiple Head Nodes

Since all head nodes are connected to the same private cluster network, any compute node's DHCP request will receive offers from all the head nodes. All offers will contain the same IP address by virtue of the fact that all head nodes share the same MAC-to-IP and node index information in the replicated database. The PXE client on the node accepts one of the DHCP offers, which is usually the first received, and proceeds to boot with the offering head node as its "parent head node". This parent head node provides the kernel and initramfs files during the PXE process, and provides the root file system for the booting node, all of which should also be replicated in /opt/scyld/clusterware/storage/ (or in the alternative non-default location specified in /opt/scyld/clusterware/conf/base.ini).

On a given head node you can determine the compute nodes for which it is the parent by examining the head node /var/log/clusterware/head_*. or /var/log/clusterware/api_error_log* files for lines containing "Boot- ing node". On a given compute node you can determine its parent by examining the node's /etc/hosts entry for parent-head-node.

Once a node boots, it asks its parent head node for a complete list of head nodes, and then thereafter the node sends periodic status information to its parent head node at the top of the list. If at any point that parent head node does not respond to the compute node's status update, then the compute node chooses a new parent by rotating its list of available head nodes by moving the unresponsive parent to the bottom of the list and moving the second node in the list up to the top of the list as the new parent.

The administrator can force compute nodes to re-download the head node list by executing `scyld-nodectl script fetch_hosts` and specifying one or more compute nodes. The administrator can also refresh the SSH keys on the compute node using `scyld-nodectl script update_keys`.
6.13.4 Switching To Alternative ClusterWare Database

To switch from couchbase to etcd or vice versa, see Switching Between Databases.

6.14 Managing Node Failures

In a large cluster the failure of individual compute nodes should be anticipated and planned for. Since many compute nodes are diskless, recovery should be relatively simple, consisting of rebooting the node once any hardware faults have been addressed. Disked nodes may require additional steps depending on the importance of the data on disk. Please refer to your operating system documentation for details.

A compute node failure can unexpectedly terminate a long running computation involving that node. We strongly encourage authors of such programs to use techniques such as application checkpointing to ensure that computations can be resumed with minimal loss.

6.14.1 Head Node Failure

To avoid issues like an Out-Of-Memory condition or similarly preventable failure, head nodes should generally not participate in the computations executing on the compute cluster. As a head node plays an important management role, its failure, although rare, has the potential to impact significantly more of the cluster than the failure of individual compute nodes. One common strategy for reducing the impact of a head node failure is to employ multiple head nodes in the cluster. See Managing Multiple Head Nodes for details.

6.15 Managing Large Clusters

Scyld ClusterWare head nodes generally scale well out-of-the-box, at least from the perspective of software, since the compute nodes’ demands on a head node are primarily during node boot, and thereafter nodes generate regular, modest Telegraf networking traffic to the InfluxDB server to report node status, and generate sporadic networking traffic to whatever cluster filesystem(s) are employed for shared storage.

Very large clusters may exhibit scaling limitations due to hardware constraints of CPU counts, RAM sizes, and networking response time and throughput. Those limitations are visible to cluster administrators using well known monitoring tools.

6.15.1 Improve scaling of node booting

The clusterware service is a multi-threaded Python application started by the Apache web server. By default, each head node will spawn up to 16 worker threads to handle incoming requests, but for larger clusters (hundreds of nodes per head node) this number can be adjusted as needed by changing the thread=16 value in /opt/scyld/clusterware/conf/httpd_wsgi.conf and restarting the clusterware service.
6.16 Backup and Restore

6.16.1 Backup and Restore of ClusterWare

The scyld-install script can also be used to back up and restore all cluster-specific data, including the cluster configuration, images, and node details. To back up the cluster:

```
scyld-install --save /path/to/backup.zip
```

By default the produced ZIP archive can be quite large, as it will contain all boot files and root file system images. If these files are archived by other means, e.g. as part of a backup solution for cluster-wide shared storage, then system administrators may want to include the `--without-files` option. The resulting ZIP file will contain only the Couchbase or etcd database. Please be aware that this option should only be used if those files are separately archived or when providing a copy of your Scyld ClusterWare database to Penguin Computing technical support.

A previously produced archive can also be loaded by the scyld-install script:

```
scyld-install --load /path/to/backup.zip
```

**Important:** Loading a ZIP backup will erase all data and all images and replace them with the corresponding contents from the archive.

During save and load, the scyld-install script is actually using the managedb tool that provides additional options and capabilities. For details please see managedb in the Reference Guide.

6.16.2 Backup and Restore of the Database

Cluster administrators may wish to capture the database state in the event that a database restore operation is desired in the future. This can be done by manually executing the `/opt/scyld/clusterware/bin/take-snapshot` tool, or more preferably by setting up a cronjob to periodically execute that tool.

For details please see `take-snapshot` in the Reference Guide.

6.17 Updating Scyld ClusterWare

**Important:** A simple `yum update` will not update Scyld ClusterWare packages, as the scyld-install tool has disabled `/etc/yum.repos.d/clusterware.repo` in order to prevent `yum update` from inadvertently updating Scyld ClusterWare. Instead, Penguin Computing strongly recommends using the scyld-install tool to perform updates of the basic Scyld ClusterWare packages that were originally installed by scyld-install, and then using `yum update --enablerepo=scyld* <PACKAGES>` to update (perhaps selectively) the optional Scyld ClusterWare packages that an administrator previously installed using `yum install --enablerepo=scyld* <PACKAGES>`, as described in Installing Optional ClusterWare Software.

From time to time, Scyld may release updates and add-ons to Scyld ClusterWare. Customers on active support plans for Scyld software products can access these updates on the Penguin Computing website. Visit [https://www.penguincomputing.com/support](https://www.penguincomputing.com/support) for details. This site offers answers to common technical questions and provides access to application notes, software updates, and product documentation.

The Release Notes contains brief notes about the latest release, and the Changelog provides a history of significant changes for each software release and a list of Known Issues And Workarounds.
The `scyld-install` tool is used to update Scyld ClusterWare software, just as it was used to perform the initial installation. This tool first determines if a newer `clusterware-installer` package is available, and if so will update `clusterware-installer` and then restart `scyld-install`.

**Important:** `scyld-install` uses the `yum` command to access Scyld ClusterWare and potentially various other repositories (e.g., Red Hat RHEL or CentOS) that by default normally reside on Internet websites. However, if the head node(s) do not have Internet access, then the required repositories must reside on local storage that is accessible by the head node(s). See Appendix: Creating Local Repositories without Internet.

**Note:** Executing `scyld-install` with no arguments presupposes that ClusterWare is not yet installed. If ClusterWare is currently installed, then the tool asks for positive confirmation that the user does intend to update existing software. You can avoid this interaction by instead executing `scyld-install` with `-u` or `--update`. That same degree of caution occurs if executing `scyld-install --update` on a server that does not have ClusterWare already installed: the tool asks for positive confirmation that the user does intend to install ClusterWare as a fresh install.

The `scyld-install` tool only updates basic ClusterWare software that was previously installed by the tool, plus any other dependency packages. After ClusterWare is updated, you can execute `yum check-update --enablerepo=scyld* | grep scyld` to view the optional ClusterWare packages that were previously installed using `yum install --enablerepo=scyld*`, and then use `sudo yum update --enablerepo=scyld* <PACKAGES>` to update (or not) as appropriate for your local head node.

You can also execute `yum check-update` to view the non-ClusterWare installed packages that have available updates, and then use `sudo yum update <PACKAGES>` to selectively update (or not) as appropriate for your local head node.

Alternatively, `scyld-install --clear` empties the database and clears the current installation. Just like during an initial installation, after a `--clear` the database should be primed with a cluster configuration. The cluster configuration can be loaded at the same time as the `--clear` using the `--config /path/to/cluster-conf` argument. This will use the `scyld-cluster-conf` tool to load the cluster configuration's initial declaration of private cluster interface, max number of nodes, starting IP address, and MAC address(es), as described in Execute the ClusterWare install script. For more details of the `scyld-cluster-conf` tool please refer to the Reference Guide.

Similar to using `scyld-install` to perform a fresh install on a non-Scyld head node, executing `scyld-install --clear --config /path/to/cluster-conf >` will invoke the `scyld-add-boot-config` script to create a new default boot image.

### 6.18 Troubleshooting ClusterWare

The `/var/log/clusterware/` folder contains several log files that may help diagnose problems. Additionally, the ClusterWare database service may have useful information in its logs.

For etcd, see `/var/log/clusterware/etcd.log`.

For Couchbase, see `/opt/couchbase/var/lib/couchbase/log/`. See https://docs.couchbase.com/home/index.html for details. The Couchbase console is available on port 8091 of any ClusterWare head node that employs Couchbase.

On a typical head node the `/var/log/clusterware/` folder contains `api_access_log` and `api_error_log` files. These are the Apache logs for the service providing the REST API. The log level available in this file is controlled by the Pyramid logging configuration in the `/opt/scyld/clusterware/conf/pyramid.ini` file. The Pyramid project documentation contains details of the pertinent variables https://docs.pylonsproject.org/projects/pyramid/en/latest/narr/logging.html
A selection of log statements from the api_error_log are also logged to the ClusterWare database and then copied to the logging folder on each head node. A separate log file is created for each head node and is named based on the head node UID, i.e. head_293aaf3f635448e9aaa76fc998ebc9c.log. This should allow a cluster administrator to diagnose many problems without needing to contact every head node individually. The log level for this file is controlled by the logging level variable in each head node's /opt/scyld/clusterware/conf/base.ini file. The default log level of WARNING should be useful but not overly verbose. The options from most terse to most verbose are AUDIT, ERROR, WARNING, INFO, DEBUG.

The various /var/log/clusterware/* logfiles are periodically rotated, as directed by the /etc/logrotate.d/clusterware, /etc/logrotate.d/clusterware-dnsmasq, and /etc/logrotate.d/clusterware-iscdhcp configuration files that distributed in the clusterware, clusterware-dnsmasq, and clusterware-iscdhcp RPMs, respectively.

**Note:** If the local cluster administrator modifies the /etc/logrotate.d/clusterware file, then a subsequent update of clusterware RPM will install a new version as /etc/logrotate.d/clusterware.rpmnew. The cluster administrator should merge this clusterware.rpmnew into the local customized /etc/logrotate.d/clusterware. Similar treatment of clusterware-dnsmasq and clusterware-iscdhcp is advised.

### 6.18.1 Failing PXE Network Boot

If a compute node fails to join the cluster when booted via PXE network boot, there are several places to look, as discussed below.

**Rule out physical problems.** Check for disconnected Ethernet cables, malfunctioning network equipment, etc.

**Confirm the node's MAC is in the database.** Search for the node by MAC address to confirm it is registered with the ClusterWare system:

```
scyld-nodectl -i 00:11:22:33:44:55 ls -l
```

**Check the system logs.** Specifically look for the node's MAC address in the api_error_log and head_*'.log files. These files will contain AUDIT statements whenever a compute node boots, e.g.,

Booting node (MAC=08:00:27:f0:44:35) as iscsi using boot config b7412619fe28424ebe1f7c5f3474009d.

Booting node (MAC=52:54:00:a6:f3:3c) as rwram using boot config f72edc4388964cd9919346dfe2b1cd2c.

If there are no "Booting node" log statements, then the failure is most likely happening at the DHCP stage, and the head nodes' isc-dhcpd.log log files may contain useful information.

As a last resort, check if the head node is seeing the compute node's DHCP requests, or whether another server is answering, using the Linux tcpdump utility. The following example shows a correct dialog between compute node 0 (10.10.100.100) and the head node.

```
[root@cluster ~]# tcpdump -i eth1 -c 10
Listening on eth1, link-type EN10MB (Ethernet),
capture size 96 bytes
18:22:07.901571 IP master.bootpc > 255.255.255.255.bootps:
  BOOTP/DHCP, Request from .0, length: 548
18:22:07.902579 IP .-1.bootps > 255.255.255.255.bootpc:
  BOOTP/DHCP, Reply, length: 430
18:22:09.974536 IP master.bootpc > 255.255.255.255.bootps:
  BOOTP/DHCP, Request from .0, length: 548
```

(continues on next page)
Verify that ClusterWare services are running. Check the status of ClusterWare services with the commands:

```
systemctl status clusterware
systemctl status clusterware-dhcpd
systemctl status clusterware-dnsmasq
```

Restart ClusterWare services from the command line using:

```
sudo systemctl restart clusterware
```

Check the switch configuration. If the compute nodes fail to boot immediately on power-up but successfully boot later, the problem may lie with the configuration of a managed switch.

Some Ethernet switches delay forwarding packets for approximately one minute after link is established, attempting to verify that no network loop has been created ("spanning tree"). This delay is longer than the PXE boot timeout on some servers.

Disable the spanning tree check on the switch. The parameter is typically named "fast link enable".

### 6.18.2 Kickstart Failing

If a node has been configured to kickstart using a boot configuration provided by a repo created from an ISO file but is failing, then check the console output for the node. If the node is entering the "Dracut Emergency Shell" from the dracut timeout scripts, then you will need to retry and see what messages were on screen prior to the "Warning: dracut-initqueue timeout" messages that flood the screen. One common error is "Warning: anaconda: failed to fetch stage2 from <URL>", where the URL points to a repo on the head node. If this message occurs, please check that you have uploaded the correct ISO into the system.

For CentOS and RHEL, the "boot" ISO files such as CentOS-8.1.1911-x86_64-boot.iso do contain the files necessary to initiate the kickstart process, but do not contain the full package repositories. The cluster administrator must provide appropriate URLs in the kickstart file, or must upload a more complete ISO such as CentOS-8.1.1911-x86_64-dvd1.iso using the scyld-clusterctl command. For example, to replace the ISO originally uploaded into a newly created centos8_repo repo:

```
scyld-clusterctl repos -i centos8_repo update iso=@CentOS-8.1.1911-x86_64-dvd1.iso
```
6.18.3 Head Node Filesystem Is 100% Full

If a head node filesystem(s) that contains ClusterWare data (typically the root filesystem) is 100% full, then the administrator cannot execute `scyld-*` commands and ClusterWare cluster operations will fail.

6.18.3.1 Verify excessive storage is related to ClusterWare

First determine whether or not the problem is due to ClusterWare-related files. Investigate with:

```bash
sudo du -sh /opt/* ; sudo du -sh /opt/scyld/*
sudo du -sh /var/lib/*
# For each cluster administrator with a home directory on local storage:
sudo du -sh /home/*/.scyldcw/workspace
```

and look for excessive storage consumption. If ClusterWare is not the problematic consumer, then broaden the search across the filesystem(s) for non-ClusterWare storage that can be reduced.

For ClusterWare storage, continue:

6.18.3.2 Remove unnecessary objects from the ClusterWare database

Remove any unnecessary objects in the database that may be lingering after an earlier aborted operation:

```bash
sudo systemctl stop clusterware
sudo rm /opt/scyld/clusterware/storage/*.old.00
sudo systemctl start clusterware
```

If that does not release enough space to allow the `scyld-*` commands to execute, then delete the entire local cache of database objects:

```bash
sudo systemctl stop clusterware
sudo rm -fr /opt/scyld/clusterware/workspace/*
sudo systemctl start clusterware
```

6.18.3.3 Investigate InfluxDB retention of Telegraf data

If you continue to see `influxdb` messages in `/var/log/messages` that complain "no space left on device", or if the size of the `/var/lib/influxdb/` directory is excessively large, then InfluxDB may be retaining too much Telegraf time series data, aka shards. Examine with:

```bash
sudo systemctl restart influxdb
# View the summation of all the Telegraf shards
sudo du -sh /var/lib/influxdb/data/telegraf/autogen/
# View the space consumed by each Telegraf shard
sudo du -sh /var/lib/influxdb/data/telegraf/autogen/*
```

If the `autogen` directory or any particular `autogen` subdirectory `shard` consumes a suspiciously large amount of storage, then examine the retention policy with the `influx` tool:
sudo influx

and now within the interactive tool you can execute influx commands:

```sh
> show retention policies on telegraf
```

The current ClusterWare defaults are a duration of 168h0m0s (save seven shards of Telegraf data) and a shardGroup-Duration of 24h0m0s (each spanning one 24-hour day). You can reduce the current retention policy, if that makes sense for your cluster, with simple command. For example, reduce the above 7-shard duration to five, thereby reducing the number of saved shards by two:

```sh
> alter retention policy "autogen" on "telegraf" duration 5d
```

You can also delete individual unneeded shards. View the shards and their timestamps:

```sh
> show shards
```

and selectively delete any unneeded shard using its id number, which is found in the show output's first column:

```sh
> drop shard <shard-id>
```

When finished, exit the influx tool with:

```sh
> exit
```

See https://docs.influxdata.com/influxdb/v1.8/ for more documentation.

6.18.3.4 Remove unnecessary PXEboot images, repos

If scyld-* commands can now execute, then view information for all images and repos, including their sizes:

```sh
cycld-imgctl ls -l
cycld-clusterctl repos ls -l
```

Consider selectively deleting unneeded images with:

```sh
cycld-imgctl -i <imageName> rm
```

and consider selectively deleting unneeded repos with:

```sh
cycld-clusterctl repos -i <repoName> rm
```

6.18.3.5 Otherwise

If scyld-* commands still cannot execute, and if your cluster really does need all its existing images, boot configs, telegraf history, and other non-ClusterWare filesystem data, then consider moving extraordinarily large directories (e.g., /opt/scyld/clusterware/workspace/, as specified in /opt/scyld/clusterware/conf/base.ini) to another filesystem or even to another server, and/or add storage space to the appropriate filesystem(s).
6.18.4 Exceeding System Limit of Network Connections

Clusters with a large number of nodes (e.g., many hundreds or more) may observe a problem when executing a workload that attempts to communicate concurrently with many or most of the nodes, such as scyld-nodectl --up exec or mpirun executing a multi-threaded, multi-node application. The problem exhibits itself with an error message that refers to being unable to allocate a TCP/IP socket or network connection, or arp_cache reporting a "neighbor table overflow!" error.

A possible solution is to increase the number of available "neighbor" entries. These are managed by a coordinated increase of \texttt{gc\_thresh1}, \texttt{gc\_thresh2}, and \texttt{gc\_thresh3} values. See \url{https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt} for the semantics of these variables. See the current values with:

\begin{verbatim}
sysctl net.ipv4.neigh.default.gc_thresh1
sysctl net.ipv4.neigh.default.gc_thresh2
sysctl net.ipv4.neigh.default.gc_thresh3
\end{verbatim}

Default CentOS/RHEL values are 128, 512, and 1024, respectively. Experiment with higher values until your workloads are all successful, e.g.:

\begin{verbatim}
sudo sysctl -w net.ipv4.neigh.default.gc_thresh1=2048
sudo sysctl -w net.ipv4.neigh.default.gc_thresh2=4096
sudo sysctl -w net.ipv4.neigh.default.gc_thresh3=8192
\end{verbatim}

See \texttt{man sysctl.conf} for how to make the successful values persistant across a reboot by putting them in a new /etc/sysctl.d/ file.

6.18.5 etcd Database Exceeds Size Limit

The etcd database has a hard limit of 2GB. If exceeded, then all scyld-* commands fail and /var/log/clusterware/api_error_log will commonly grow in size as each node's incoming status message cannot be serviced.

Normally a head node thread executes in the background that triggers the discarding of database history (called \texttt{compaction}) and triggers database defragmentation (called \texttt{defrag}) if that is deemed necessary. In the rare event that this thread stops executing, then the etcd database grows until its size limit is reached.

This problem can solved with manual intervention by an administrator. Determine if the etcd database really does exceed its limit. For example:

\begin{verbatim}
[admin@head1]$ sudo du -hs /opt/scyld/clusterware-etcd/
2.1G /opt/scyld/clusterware-etcd
\end{verbatim}

Subtract two or three thousand from the revision value 4752785 and compact to that new value:

\begin{verbatim}
[admin@head1]$ sudo /opt/scyld/clusterware-etcd/bin/etcdctl compaction 4750000
compacted revision 4750000
\end{verbatim}
and trigger a defragmentation to reclaim space:

```bash
[admin@head1]$ sudo /opt/scyld/clusterware-etcd/bin/etcdctl defrag
Finished defragmenting etcd member[http://localhost:52379]
```

Then clear the alarm and reload the `clusterware` service:

```bash
[admin@head1]$ sudo /opt/scyld/clusterware-etcd/bin/etcdctl alarm disarm
[admin@head1]$ sudo systemctl reload clusterware
```

This restarts the head node thread that executes in the background and checks the etcd database size. Everything should now function normally.

### 6.18.6 Failing To Boot From Local Storage

If a compute node is configured to boot from local storage, and yet after successfully booting it is actually instead using a RAM root filesystem, then the problem may be that the initramfs image does not contain a needed kernel module to mount the root filesystem on local storage. Examine `/opt/scyld/clusterware-node/atboot/cw-dracut.log` on the compute node to determine if the mount failed and why. If the problem is a missing kernel module, then add that to the initramfs. For example, add the `virtio_blk` module, and rebuild the boot config:

```bash
scyld-mkramfs --update DefaultBoot --kver 3.10.0-957.27.2.el7.x86_64 --drivers virtio_blk
```

### 6.18.7 IP Forwarding

If IP forwarding is desired and is still not working, then search for the line containing "net.ipv4.ip_forward":

```bash
grep net.ipv4.ip_forward /etc/sysctl.conf
grep net.ipv4.ip_forward /etc/sysctl.d/*
```

If that line exists and the assigned value is set to zero, then IP forwarding will be disabled.

### 6.18.8 Soft Power Control Failures

If the `scyld-nodectl reboot` or `shutdown` commands always fall back on hard power control, then this can point to one or two different problems. If compute nodes use an ssh banner, this can confuse the service that detects success and failure from the Linux `shutdown` command on the compute nodes. In that case, please see the discussion of banners at [Compute Nodes with SSH banner](#) and `power_uri` at [Compute Nodes IPMI access](#).

Alternatively, the shutdown process on the compute node may take too long. When this happens the `scyld-nodectl reboot` or `shutdown` commands will pause for several seconds waiting for the soft power change to take place before falling back to direct power control through the `power_uri`. A common cause for this is a network file system that is slow to unmount. The cluster administrator should address the problem delaying shutdown, but if it is unavoidable, then the `reboot` and `shutdown` commands accept options to adjust the timeout (`--timeout <seconds>`), or you can specify to use only the soft reboot (`--soft`) without falling back to direct power control.
6.18.9 Head Nodes Disagree About Compute Node State

If two linked head nodes disagree about the status of the compute nodes, this is usually due to clock skew between the head nodes. The appropriate fix is to ensure that all head nodes are using the same NTP / Chrony servers. The shared database includes the last time each compute node provided a status update. If that time is too far in the past, then a compute node is assumed to have stopped communicating and is marked as "down". This mark is not recorded in the database, but is instead applied as the data is returned to the calling process such as scyld-nodectl status.

6.18.10 Finding Further Information

If you encounter a problem installing your Scyld cluster and find that this Installation & Administrator Guide cannot help you, the following are sources for more information:

- The Changelog contains per-release specifics, and a Known Issues And Workarounds section.
- The Reference Guide contains a technical reference to Scyld ClusterWare commands.

6.18.11 Contacting Penguin Computing Support

If you choose to contact Penguin Computing Support, you may be asked to submit a system information snapshot. Execute scyld-sysinfo --no-tar to view this snapshot locally, otherwise execute scyld-sysinfo to produce the compressed tarball that can be emailed or otherwise communicated to Penguin Computing.

6.19 IPMI

Included in the RHEL/CentOS base distribution are tools that may be of interest to users, including the ipmitool command for monitoring and managing compute node hardware.

6.19.1 IPMITool

ipmitool is a hardware management utility that supports the Intelligent Platform Management Interface (IPMI) specification v1.5 and v2.0.

IPMI is an open standard that defines the structures and interfaces used for remote monitoring and management of a computer motherboard (baseboard). IPMI defines a micro-controller, called the "baseboard management controller" (BMC), which is accessed locally through the managed computer's bus or through an out-of-band network interface connection (NIC).

The root can use ipmitool for a variety of tasks, such as:

- Inventory a node's baseboards to determine what sensors are present
- Monitor sensors (fan status, temperature, power supply voltages, etc.)
- Read and display values from the Sensor Data Repository (SDR)
- Read and set the BMC's LAN configuration
- Remotely control chassis power
- Display the contents of the System Event Log (SEL), which records events detected by the BMC as well as events explicitly logged by the operating system
- Print Field Replaceable Unit (FRU) information, such as vendor ID, manufacturer, etc.
- Configure and emulate a serial port to the baseboard using the out-of-band network connection known as serial over LAN (SOL)

Several dozen companies support IPMI, including many leading manufacturers of computer hardware. You can learn more about OpenIPMI from the OpenIPMI project page at http://openipmi.sourceforge.net, which includes links to documentation and downloads.

### 6.20 Services, Ports, Protocols

#### 6.20.1 Apache

Apache serves the ClusterWare REST API via HTTP on port 80 using mod_wsgi through the httpd systemd service aliased as `clusterware`. HTTPS Encryption over port 443 can be enabled through standard Apache and operating system procedures. Apache is Open Source, and Penguin Computing contributes the REST API. The log files are `/var/log/clusterware/api_access_log` and `/var/log/clusterware/api_error_log`.

The ClusterWare GUI is also served through Apache from the `/var/www/clusterware/front/` directory.

#### 6.20.2 Couchbase

The replicated configuration key/value store Couchbase has the systemd service name `couchbase-server`. Log files are found in `/opt/couchbase/var/lib/couchbase/logs/` but may also be accessed through the Couchbase console on port 8091 of any head node.

The Couchbase ports:

```
4369/tcp  # erlang port mapper (for Couchbase join)
8091/tcp  # Couchbase GUI and command interface (for Couchbase join)
8092/tcp  # Couchbase JSON interface (for Couchbase queries)
21101/tcp # otp port (for Couchbase join)
11209/tcp # memcached (for Couchbase data synchronization)
11210/tcp # Also used for Couchbase data synchronization
```

#### 6.20.3 DHCP

DHCP provides dynamic host configuration, with a systemd service name `clusterware-dhcpd` and using port 68. The log file is `/var/log/clusterware/isc-dhcpd.log`. This service is configured, started, and stopped by the ClusterWare service based on the cluster configuration. The configuration file is generated from a template located at `/opt/scyld/clusterware-iscdhcp/dhcpd.conf.template`.

#### 6.20.4 etcd

The replicated configuration key/value store etcd has the systemd service name `clusterware-etcd`. Log files are found in `/var/log/clusterware/`. etcd uses port 52380 to communicate with other head nodes.
6.20.5 iSCSI

iSCSI optionally serves root filesystems to compute nodes and uses port 3260. Serving root file systems via iSCSI is configured by the ClusterWare service using the `targetcli` command line tool.

6.20.6 OpenSSH

OpenSSH provides services to remotely execute programs and to transfer files, with a systemd service name `sshd` and using port 22. Encryption is SSH. The log file is `/var/log/messages`.

6.20.7 Telegraf / InfluxDB

Telegraf and InfluxDB communicate and store compute node performance data, with a systemd service name `telegraf` and using port 8094. Encryption is HTTPS. The log files are found in `/var/log/telegraf/`.

6.20.8 TFTP

The TFTP Server provides downloads for early iPXE boot files, with a systemd service name `xinetd` and using port 69. This service can be replaced by appropriate network card firmware. The log file is `/var/log/messages`. 
7.1 Introduction

This document describes Scyld ClusterWare commands that are intended for the cluster administrator.

This Reference Guide is written with the assumption that the reader has a background in a Linux or Unix operating environment. Therefore, this document does not cover basic Linux system use, administration, or application development.

7.2 Important Files on Head Nodes

7.2.1 The ~/.scyldcw/ Folder

As described elsewhere in this document, ClusterWare administrator tools read some configuration details from the user's ~/.scyldcw/settings.ini file. This section describes the other common contents of the ~/.scyldcw/ folder. Although this is included in the Important Files on Head Nodes chapter, please note that this folder exists in the home directory of any user who executes the ClusterWare tools, and that these tools are intended to be installed not just on the head node, but also wherever an administrator finds convenient and has appropriate HTTP or HTTPS access to the head node.

7.2.1.1 auth_tkt.cookie

Whenever a user authenticates to the REST API running on a head node, an authentication cookie is generated and used for subsequent requests in the same session. Even though sessions typically end when the executed tool completes, the command line tools caches the authentication cookie in the ~/.scyldcw/auth_tkt.cookie file to allow for faster tool start times. A summary of the network requests are logged at the DEBUG level:

```
[sysadmin@virthead ~]$ scyld-nodectl -vv ls
DEBUG: GETing /node/{uid} through /mux
INFO: No value provided for global option 'client.auth_tkt'.
DEBUG: Starting new HTTP connection (1): localhost:80
DEBUG: http://localhost:80 "GET /api/v1/whoami HTTP/1.1" 200 74
DEBUG: http://localhost:80 "GET /api/v1/whoami HTTP/1.1" 200 109
INFO: Loaded authentication cookie from previous run.
DEBUG: http://localhost:80 "POST /api/v1/mux?log=GET-/node/UID HTTP/1.1" 200 7995
DEBUG: 0.0946: transaction prepared in 0.017, completed in 0.033
INFO: Expanded '*' into 2 nodes.
```

(continues on next page)
As can be seen in the above log, the authentication token from a previous run was loaded and used for the duration of command execution and then re-cached for later use.

### 7.2.1.2 logs/

The command line tools also log their arguments and some execution progress in the `~/.scyldcw/logs/` folder. By default each tool keeps logs of its previous five runs, though this number can be adjusted in the `settings.ini` file by resetting the `logging.max_user_logs` value. Set this value to zero to discard all logs, and set to a negative number to preserve logs indefinitely. Administrators may be asked to provide these logs (usually via the `scyld-sysinfo` tool) when requesting assistance from Penguin Computing technical support.

### 7.2.1.3 workspace/

The `~/.scyldcw/workspace/` folder is used by the `scyld-modimg` tool to store, unpack, and manipulate image contents. Root file system images are large, which means this local image cache can grow large. Administrators are encouraged to delete unneeded entries in the cache using the `scyld-modimg --delete` command, either with the `-i` (or `--image`) argument to name specific images, or with `--all` to delete all local images. This will not delete the remote copies of images stored on the head nodes, just delete the local cache. Within this folder, the `manifest.json` file contains JSON formatted information about the cached images, while the images themselves are stored as individual packed files with names based upon their UID. If the cached images are ever out of sync with the manifest, i.e. a file is missing or an extra file is present, then the `scyld-modimg` tool will print a warning:

```
WARNING: Local cache contains inconsistencies. Use --clean-local to delete temporary files, untracked files, and remove missing files from the local manifest.
```

This warning can be automatically cleared by running the tool with the `--clean-local` option. This is not done automatically in case some useful image or other data might be lost. Alternatively, if the `manifest.json` is somehow lost, a new file can be constructed for a collection of images using the `--register-all` option. See the command documentation for more details.

The location of the workspace folder can be controlled on the `scyld-modimg` command line or by the `modimg.workspace` variable in the `settings.ini` file.

### 7.2.1.4 parse_failures/

Several ClusterWare tools execute underlying Linux commands, such as `rpm` or `yum`, and parse their output to check for details of success or failure. During execution and parsing, the `stdout` and `stderr` of the Linux commands are cached in the `~/.scyldcw/parse_failures/` folder. If the parsing completes, regardless of the command success or failure, these files will be deleted, but when a tool crashes or parsing fails, these files will be left behind. Though not generally useful to an administrator during normal operation, these output files could be useful for debugging problems and may be requested by Penguin Computing technical support. Much like files in the `~/.scyldcw/logs/` folder, these parse failures can be periodically purged if no problems are encountered, though be aware that useful debugging information may be lost.
7.2.2 The /opt/scyld/clusterware/ Folder

The /opt/scyld/clusterware folder exists only on a head node and contains the core ClusterWare installation. Selected contents are described below.

7.2.2.1 /opt/scyld/clusterware/bin/

Tools located in the bin/ folder are intended to be run as root only on head nodes and are rarely executed directly. This is where the managedb tool is located, as well as the pam_authenticator application described in the Installation & Administrator Guide and the randomize_ini script executed during initial installation.

7.2.2.2 /opt/scyld/clusterware/conf/

The conf/ folder contains the principal configuration files for ClusterWare REST API. In that folder the httpd_*.conf files are used in the actual Apache configuration, while the INI files control the behavior of the Python Pyramid-based service. Modifications to any of these files requires the administrator to restart the clusterware service. Also note that modifications to these files only affect the one head node and may need to be replicated to other head nodes in multihread configurations. Because of this, future releases may move selected variables from the base.ini file into the ClusterWare database to provide a cluster-wide effect.

Many aspects of the REST service can be tweaked through changes to variables in the base.ini, and these are discussed throughout this documentation. To list all available variables please use the managedb tool:

```
sudo /opt/scyld/clusterware/bin/managedb --print-options
```

This command will list all options registered with the configuration system, and although many of these options are for internal use only, Penguin Computing technical support may suggest changes in individual cases. The specific variables available and their effects may change in future releases.

The variable names take a general form of SUBSYSTEM.VARIABLE or PLUGIN.VARIABLE. For example, the plugins subsystem is controlled through these variables, and a specific authentication plugin is selected by the plugins.auth variable. Further, what application the appauth plugin uses is controlled by the appauth.app_path variable. For a description of this specific plugin, please see Securing the Cluster. Other variables in the base.ini file follow similar patterns.

Variables in the production.ini file are used to control aspects of the Python Pyramid framework, specifically logging. Variables in this file are also for internal use and should not be modified except by the suggestion of Penguin Computing technical support.

7.2.2.3 /opt/scyld/env/, modules/, and src/

The env/, modules/, and src/ folders contain the Python virtual environment, including the libraries required by the scyld-* and other tools.
7.2.2.4 /opt/scyld/clusterware/parse_failures/

Similar to the individual administrator ~/scyldcw/parse_failures/, files in this folder will accumulate any parsing failures found while running underlying Linux commands and should generally be empty. If files are accumulating here, it is safe to delete them, but the ClusterWare developers should be informed and may request a sample of the files to diagnose the underlying failure.

7.2.2.5 /opt/scyld/clusterware/storage/

The storage/ folder is the default location used by the local_files plugin to store kernels, initramfs files, and packed root file systems. The actual location of this folder is controlled by the local_files.path variable in the base.ini configuration file.

This folder can grow relatively large depending on the size and quantity of root file systems in the cluster. Most organizations will want to include the storage folder in their backup planning along with the database contents obtained through scyld-install --save or the managedb save command. See Backup and Restore for additional discussion of backup up the database contents.

7.2.2.6 /opt/scyld/clusterware/workspace/

The REST service running on each head node requires a location to hold temporary files. This location is controlled by the head.workspace variable and defaults to /opt/scyld/clusterware/workspace/. Like the storage/ directory, workspace/ can grow to relatively large size, but unlike storage/ does not need to be backed up. Any files or directories found in this folder are temporary and should be deleted when the service is shut down or restarted. If files or folders accumulate in this folder, they are safe to remove, although this must be done carefully or when the REST service is stopped. If files do accumulate here, please notify Penguin Computing developers so that we may diagnose the underlying issue.

7.3 Compute Node Initialization Scripts

All compute node images should include the clusterware-node package. This package includes systemd services used for periodically reporting node status back to the head node as well as initialization scripts run as the node is booting.

At the end of the boot process described in Node Images and Boot Configurations, the mount_rootfs script hands control of the machine over to the standard operating system initialization scripts when it switches to the newly mounted root. Shortly after networking is established on the booting node, it contacts the parent head node, the compute node begins periodic pushes of status information to the parent, which stores that information in the ClusterWare database. The first data push includes detected hardware information, while subsequent data only contains the more ephemeral node status information. With each status update the node also retrieves its attribute list and stores this list as an INI file at /opt/scyld/clusterware-node/etc/attributes.ini. Code running on the compute node can use the contents of this file to customize the node configuration. A simple attributes.ini file:

```
[Node]
UID = c1bf15749d724105bce9e07a3d79cb69

[Attributes]
_boot_config = DefaultBoot
```

The [Node] section will include node-specific details, while the [Attributes] section contains the node attributes as determined from the node’s groups using the process described in Node Attributes. The clusterware-node package also contains a symlink at /etc/clusterware pointing to /opt/scyld/clusterware-node/etc/.
Shortly after the first status push, a series of shell scripts are executed on the node to perform ClusterWare-specific node initialization. These scripts are linked in `/opt/scyld/clusterware-node/scripts-enabled` and located in `/opt/scyld/clusterware-node/scripts-available`.

All such scripts should include `/opt/scyld/clusterware-node/functions.sh` for common variables and functions, and should use the `attributes.ini` described previously to determine what actions are necessary. Cluster administrators are invited to enable and disable these scripts in their root file system images as they see fit and to contribute improved or added scripts back to the ClusterWare developers for the continuing improvement of the product.

### 7.4 Database Objects Fields and Attributes

Various ClusterWare database objects (e.g., nodes, boot configurations, image configurations, administrators, attributes) each carry with them detailed descriptors called `fields`. Each field consists of a name-value pair and is relevant for its database object type. Fields are predefined by ClusterWare. The cluster administrator uses the `update` action to change a field value.

For instance, a compute node object for each node has fields `mac` with the node's MAC address, `name` with the node's alphanumeric name, and `power_uri` with a value denoting how to communicate via ipmi to that node. For example, the command `scyld-nodectl -i n0 ls -l` displays all the defined fields' name-value pairs for node n0.

Compute node and Attribute Groups object types have special fields called `attributes`, where an attribute is a collection of one or more attribute name-value pairs. Attribute names that begin with an underscore `"_"` are called `reserved attributes` or `system attributes`. The cluster administrator uses the `set` action to change an attribute value. See the following section `Reserved Attributes` for details.

Additional attributes can be added by a cluster administrator as desired, each with a custom name and value defined by the administrator. Any script on a compute node can access the local file `/etc/clusterware/attributes.ini` and find that node's attributes. On the node there are helper functions in `/opt/scyld/clusterware-node/functions.sh` for reading attributes, specifically the function `attribute_value`.

### 7.5 Reserved Attributes

Within the ClusterWare attribute system, administrators are encouraged to store whatever information they find useful for labeling and customizing nodes. For ease of use, attributes names should be valid Javascript variable names, i.e., meaning that they may begin with any uppercase or lowercase letter, followed by letters, digits, or underscores. Names that start with an underscore are used by ClusterWare and should be set by administrators to affect the behavior of the system. These will be referred to as `system attributes` throughout this discussion.

Attributes are stored internally as a Javascript dictionary mapping strings to strings, otherwise known as name-value pairs. Administrator-defined attribute values should be strings and relatively small in size. The ClusterWare backend database enforces some document size constraints, and collections of node attributes should be no more than tens to hundreds of kilobytes in size. Individual attributes can be any length as long as the overall attribute group or node object size does not exceed this limits. Generally, if a cluster configuration is approaching these sizes, a cluster administrator pursue moving data from the database into shared storage locations referenced by database entries.

Attributes can be applied directly to nodes, but may also be collected into groups, and then these groups applied to sets of nodes. Attributes passed to nodes through groups are treated no differently than those applied directly to a node. Attribute groups help cluster administrators create more scalable and manageable configurations. See `Node Attributes` for more details.

The remainder of this section is a list of system attributes describing their use and allowed values.
7.5.1 _ansible_pull

Default: none
Values: reference to an ansible git repo and a playbook in that repo
Depends: none
See Appendix: Using Ansible for details about format and usage.

7.5.2 _ansible_pull_now

Default: none
Values: reference to an ansible git repo and a playbook in that repo
Depends: none
the cluster administrator must systemctl enable cw-ansible-pull-now and systemctl start cw-ansible-pull-now. See Appendix: Using Ansible for details about format and usage.

7.5.3 _boot_config

Default: none
Values: boot configuration identifier
Depends: none
The _boot_config attribute defines what boot configuration a given node should use. For a detailed discussion of boot configurations and other database objects, please see Node Images and Boot Configurations.
A boot configuration identifier may be a, possibly truncated, UID or a boot configuration name.

7.5.4 _boot_rw_layer

Default: overlayfs
Values: overlayfs, rwtab
Depends: _boot_style == roram or iscsi
Use _boot_rw_layer to control the type of overlay used to provide read/write access to an otherwise read-only root file system image. The overlayfs provides a writable overlay across the entire file system, while the rwtab approach only allows write access to the locations defined in /etc/rwtab or /etc/rwtab.d in the node image.
Note that prior to kernel version 4.9, overlayfs does not support SELinux extended attributes and so cannot be used for compute nodes with SELinux in enforcing mode. The rwtab option does work with SELinux, but two additional changes need to be made when enabling rwtab. First, the cluster administrator must modify the /etc/sysconfig/readonly-root file in the node image to ensure READONLY is set to "yes":

    READONLY=yes

Second, the kernel cmdline in the appropriate boot configuration must include "ro":

    cmdline: enforcing=1 ro
7.5.5 _boot_style

Default: rwram
Values: rwram, roram, iscsi, disked, next, sanboot, live
Depends: none

Root file system images can be supplied to nodes through a variety of mechanisms, and this can be controlled on a per-node basis through the _boot_style attribute. In both the rwram and roram modes, the node will download the entire image into RAM and either unpack it into a tmpfs RAM file system (rwram) or apply a writable overlay (roram). These boot styles have the advantage of post-boot independence from the head node, meaning that the loss of a head node will not directly impact booted compute nodes.

The iscsi option uses less RAM as the boot image is not downloaded into node RAM, but depends on the head node even after the node is fully booted. Due to this dependence a head node crash may cause attached compute nodes to hang and lose work. This approach requires a writable overlay, as the images may be shared between multiple nodes.

With the disked option, the node boots with images read from local storage. See Appendix: Booting From Local Storage Cache for details.

Use the next option to exit the boot loader and allow the BIOS to try the next device in the BIOS boot order. Since this process depends on support in the BIOS, it may not work on every server model.

The sanboot option causes the booting node to boot using the iPXE sanboot command and defaults to booting the first hard disk. Please see the _ipxe_sanboot attribute for more details.

The live option only works for ISO-based configurations, e.g., those used for kickstart. For supported ISOs (e.g., RHEL-based) the node boots into the live installer, and the administrator needs to interact with it via the (likely graphical) system console.

7.5.6 _boot_tmpfs_size

Default: half of RAM
Values: 1g, 2g, etc.
Depends: _boot_style == rwram or _boot_rw_layer == overlayfs

During the node boot process, a tmpfs is used to provide a writable area for diskless compute nodes. For the rwram boot style this attribute controls the size of the root file system where the image is unpacked. When booting with overlayfs on a roram or iscsi style, this attribute controls the size of the writable overlay.

7.5.7 _coreos_ignition_url

Default: none
Values: The URL of a RHCOS *.ign ignition file.
Depends: none

Both _coreos_ignition_url and _coreos_install_dev are attributes that must be set to fill in variables in the associated boot config's cmdline. See Using RHCOS.
7.5.8 _coreos_install_dev

Default: none
Values: The device on the target node into which the image is installed.
Depends: none

Both _coreos_ignition_url and _coreos_install_dev are attributes that must be set to fill in variables in the associated boot config's cmdline. See Using RHCOS.

7.5.9 _disk_cache

Default: none
Values: local partition name + optional encryption
Depends: none

Specifies a persistent location where the node can store downloaded images. This location should be a local partition with sufficient size to hold a handful of compressed images.

If the specified location exists, then the node will retain there a copy of the downloaded image. During subsequent boots the node will first compare the checksum of a file previously saved with the expected checksum provided by the head node in order to avoid unnecessary downloads.

If the specified partition does not exist, then an error will be logged, although the node will download the image to RAM and still boot. If the partition exists but cannot be mounted, then it will be reformatted.

Optionally Linux Unified Key Setup (LUKS) encryption can also be specified for the partition. Append :encrypt to the partition name to encrypt with a random key, or append :encrypt=KEY to specify an encryption key.

If no key is specified, encryption is performed using standard LUKS tools with 128 bytes of data from /dev/urandom stored in a key file used as the passphrase. This key file is only briefly stored in RAM and deleted shortly before an Ext4 file system is created on the newly encrypted partition.

Alternatively, if the specified key is TPM then the random key will be stored in the booting system's Trusted Platform Module (TPM) and deleted out of RAM shortly before the file system is created. The key can also be bound to specific TPM Platform Configuration Register (PCR) values meaning that the TPM will not later reveal the key unless those PCRs hold the same values. Since these values include hashes of the BIOS code, configuration, kernel, and other boot-time binaries access to the encrypted partition can be restricted to specific boot-time configurations. If the TPM has an owner password set it must be provided in the _tpm_owner_pass attribute.

Note: The cryptsetup-luks package must be installed in the image being booted.

Specifying a KEY is essentially necessary for _disk_cache because without that after a subsequent reboot the partition contents will be lost as they were encrypted with an unknown random key.

For example:

```
scyld-nodectl -i n[0-63] set _disk_cache=/dev/nvme0n1p2:encrypt=Penguin
```

If _disk_cache is present but no _disk_root is provided, then if a roram-compatible image is downloaded, then the node will boot directly from the cached image with a writable overlay.

Important: Any data in the partition specified as a _disk_cache may be destroyed at boot time!
Similar to /etc/fstab, partitions can be identified by device path, UUID, PARTLABEL, or PARTUUID.

7.5.10 _disk_root

Default: none
Values: local partition name + optional encryption
Depends: ignored unless _boot_style == disked

Specifies a persistent location into which at boot time the node can unpack the root image. This will delete the contents of the partition before unpacking the root image. If the specified partition does not exist, then an error will be logged, although the node will still boot using the image unpacked into RAM.

Similar to _disk_cache, append :encrypt to the partition name to encrypt with a random key, or :encrypt=KEY to specify the encryption key. For _disk_root a random key is preferable, as the _disk_root contents are intended to be ephemeral across boots.

**Important:** All data in the partition specified as a _disk_root will be destroyed at boot time!

Similar to /etc/fstab, partitions can be identified by device path, UUID, PARTLABEL, or PARTUUID.

7.5.11 _disk_wipe

Default: none
Values: comma-separated list of local partition names + optional encryption
Depends: none

The listed partitions will be reformatted at every boot with an Ext4 file system. Similar to _disk_cache, append :encrypt to the partition name to enable "encryption at rest", or :encrypt=KEY to specify the encryption key. Like _disk_root the random key is preferable to ensure _disk_wipe partition contents are not retrievable from a physically removed storage device.

7.5.12 _gateways

Default: The default gateway for the node's interfaces
Values: <ifname>=IPaddress
Depends: None

Override the interface ifname's current gateway value with an alternative IP address. For example, _gateways=enpls0f0=10.20.30.40,enpls0f1=10.20.40.40.
### 7.5.13 _health

Default: none  
Values: node health status  
Depends: none

Cluster administrators commonly write node health checks and can use the _health attribute to relay the results back to the head nodes. Whenever the health check starts running on the node, it should execute `set-node-attribs _health=checking`. At completion of the health check the results should be stored in the _health attribute using the same mechanism. The special value `ok` will be interpreted as success, while any other value is recorded as failure. Cluster administrators can check the current status using `scyld-nodectl`:

```
scyld-nodectl status --health [--refresh]
```

Instead of reporting the node up / down / booting status, that command will report whether the health checks have reported ok / bad / checking. Adding `-l` will show the failure reason for 'bad' nodes.

### 7.5.14 _hostname

Default: none  
Values: Hostname or fully-qualified domain name  
Depends: none

Booting compute nodes will assign the value of _hostname as their hostname using the `hostnamectl` command. If the attribute value is a simple name (without periods), then the cluster domain will be appended to construct a FQDN. Changes to this variable take effect during the next reboot.

### 7.5.15 _hosts

Default: blank  
Values: download  
Depends: none

During the compute node boot process, a list of known hosts is downloaded from the head node and is appended to the compute node's `/etc/hosts`. By default this will only append a list of head nodes to ensure that each compute node can resolve all head nodes without DNS. If the _hosts attribute is set to 'download', then all compute node names and IP addresses will be appended to `/etc/hosts`.

### 7.5.16 _ips

Default: none  
Values: comma-separated IP assignments  
Depends: none

Compute nodes commonly define additional high-speed network interfaces other than the PXE boot network. These interfaces are commonly defined by `ifcfg-XXX` files located in `/etc/sysconfig/network-scripts` and differ between nodes only in the assigned IP address. Use the _ips attribute to specify what IP address should be assigned to an individual node on one or more interfaces. For example, a value of `_ips=en0=10.10.23.12,ib0=192.168.24.12` would cause the `prenet/write_ifcfg.sh` startup script to replace any `IPADDR=` line in `/etc/sysconfig/network-scripts/ifcfg-ib0` with
IPADDR=192.168.24.12 and would similarly modify the adjacent ifcfg-en0 file, replacing any IP assignment in that file with IPADDR=10.10.23.12.

7.5.17 _ipxe_sanboot

Default: none
Values: local disk or partition
Depends: _boot_style == sanboot

Use this attribute to cause a node to boot using the iPXE sanboot command. This is most commonly used to boot a locally installed disk, although administrators are cautioned to be extremely careful with stateful compute nodes as they will retain modifications from previous boots, leading to an unexpectedly heterogeneous cluster.

Nodes with this attribute set will not download an image from the head node and will instead boot based on the URL or other iPXE sanboot arguments provided. Please see the iPXE documentation for the details of what iPXE provides: http://ipxe.org/cmd/sanboot

In addition to the arguments and URLs supported by iPXE, ClusterWare also accepts a shorter URL for booting local disks of the form local://0xHH where 'HH' is a hexadecimal value specifying a local hard disk. The first disk is identified as 0x80, the second is 0x81, and so on. The provided hexadecimal value is then used in a sanboot --no-describe --drive 0xHH call.

7.5.18 _macs

Default: The default MAC address for each of the node's interfaces
Values: <ifname>=<MACaddress>
Depends: None

Override the interface ifname's current MAC address with an alternative value. For example, _macs=bond0=aa:bb:cc:dd:ee:ff. Generally only used for bonded interfaces. Ignored for the booting interface bootnet.

7.5.19 _no_boot

Default: false
Values: boolean equivalents (0 / 1, true / false, t / f, yes / no, y / n)
Depends: none

The _no_boot attribute controls whether information about a node is provided to the DHCP server. Any node with _no_boot set to true will not receive DHCP offers from any ClusterWare head node. This allows an administrator to temporarily remove a node from the cluster.
7.5.20 _preferred_head

Default: none
Values: head node UID
Depends: none

In a multihead configuration any head node can provide boot files to any compute node in the system. In most cases this is a desirable feature because the failure of any given head node will not cause any specific set of compute nodes to fail to boot. In some cases the cluster administrator may want to specify a preference of which head node should handle a given compute node. By setting a compute node’s _preferred_head attribute to a specific head node's UID, all head nodes will know to point that node toward the preferred head node. This is implemented during the boot process when the iPXE script is generated and passed to the compute node. This means that any head node can still supply DHCP, the iPXE binaries, and the iPXE boot script, but the subsequent kernel, initramfs, and root file system files will be provided by the preferred head node, and thereafter the node's boot status information will be sent to that _preferred_head.

7.5.21 _remote_pass

Default: none
Values: node account password for _remote_user attribute
Depends: none

Supports an alternative to the customary ClusterWare ssh-key functionality. It is useful to support scyld-nodectl exec to non-ClusterWare compute nodes which do not have clusterware-node installed, but which do accept user/password authentication.

To use, install the sshpass RPM on the head node. Set the _remote_pass attribute to the password of the _remote_user attribute user name (default root). Subsequent executions of scyld-nodectl exec to nodes that are set up with this attribute will employ this user/password pair to authenticate access to those target node(s).

**Note:** Use of sshpass is discouraged and is not a best practice. A clear text password is a significant security risk.

7.5.22 _remote_user

Default: root
Values: node account name
Depends: none

The _remote_user attribute controls what account is used on the compute node when executing the scyld-nodectl reboot/shutdown commands. Please ensure the specified account can execute sudo shutdown without a password or soft power control will not work. Similarly the scyld-nodectl exec and scyld-nodectl ssh commands will also use the specified remote user account and the boot-time script that downloads head node keys will store those keys in the _remote_user's authorized_keys file.
7.5.23 _status_hardware_secs

Default: 300

Values: seconds between checking for status hardware changes

Depends: none

A node sends its hardware state (viewed with scyld-nodectl list --long and list --long-long) as a component of its larger basic status information. See _status_secs above. This hardware component is typically only sent once at boot time. However, the node periodically reevaluates its hardware state every _status_hardware_secs seconds, and in the rare event that something has changed since it last communicated its hardware state to its parent head node, then the node includes the updated hardware information in its next periodic basic status message.

Changes to this value are communicated to an up node without needing to reboot the node.

7.5.24 _status_packages_secs

Default: 0

Values: seconds between checking for installed packages changes

Depends: none

The time interval in seconds between the relatively expensive search for what Scyld packages are installed. This value times 10 is the time interval between the even more expensive calculations of a sha256sum hash of the sorted list of names of all installed packages, distilled into a single hexadecimal value. These values are seen by executing scyld-nodectl -i<nodes> status -L on the head node.

A non-zero value should be longer than the _status_secs value, described below.

If the value is zero, then these packages searches and calculations are done just at node boot time, and additionally when (and if) the administrator executes /usr/bin/update-node-status --hardware on a compute node. Such run-time changes to a node's installed packages are relatively rare, so the default value is zero to minimize the performance impact of these operations.

Changes to this value are communicated to an up node without needing to reboot the node.

7.5.25 _status_secs

Default: 10

Values: seconds between status updates

Depends: none

Booted compute nodes periodically send basic status information to their parent head node. This value controls how often these messages are sent. Although the messages are relatively small, clusters with more compute nodes per head node will want to set this to a longer period to reduce load on the compute nodes.

Changes to this value are communicated to an up node without needing to reboot the node.
7.5.26 _tpm_owner_pass

Default: none
Values: Owner password for the compute node TPM
Depends: none

Certain TPM commands require authentication using the "owner" TPM password. This means that the clear-text password must be provided to systems using the TPM for disk encryption via this attribute.

7.6 Introduction to Tools

This section describes the commonly used arguments and subcommands used by the various Scyld ClusterWare tools. These tools can be used by the cluster administrator and are not intended for use by the ordinary user.

Certain arguments are shared among nearly all the scyld-*ctl tools, and instead of repeatedly describing these arguments, we will cover them here. Many of these arguments control the general operation of the tools, i.e. by printing help (--help or -h), selecting targets (--all or -a, --ids or -i), changing the verbosity or client configuration (--verbose or -v, --quiet or -q, --config or -c), allowing a user to override basic connection details (--base-url, --user or -u), or changing output formatting (--show-uids, --human, --json, --pretty or --no-pretty). Many of these arguments are self-explanatory, but others are described below:

7.6.1 --all and --ids

Tools that accept the --all (short name -a) and --ids (short name -i) arguments operate on corresponding database objects. For instance, scyld-nodectl is used for manipulating node objects in the database, and scyld-attribctl is used for manipulating attribute groups.

As one might expect, --all can be used to make an alteration to all of a given class of objects at once. For example, to remove a given attribute such as _boot_style from all attribute groups, e.g.:

`scyld-attribctl --all clear _boot_style`

Alternatively, an administrator can specify objects by name, or UID, or truncated UID (at least the first 5 characters of the UID are required to reduce the chance of accidental selection). Certain object types can also be selected based on some core fields, e.g. MAC, IP, or index for nodes. Further, nodes can be selected using the node query language, e.g.:

`scyld-nodectl --ids n[0-5] --ids 08:00:27:F0:44:35 ls`

For convenience, many tools can be executed without explicitly selecting any objects. Specifically, query tools such as list will default to --all if no selection arguments are used, and many other tools will operate on a single object if only one object of the expected type exists in the system.

7.6.2 --config

All client tools accept a --config argument which can be used to specify a client INI file. By default several locations are checked for configuration INI files with each able to override variables from the previous files. The client configuration search order is:

- /etc/scyldcw/settings.ini
- /etc/scyldcw/${TOOL}.ini
- ~/.scyldcw/settings.ini
- ~/.scyldcw/${TOOL}.ini
These configuration files should be INI formatted, and the [ClusterWare] section can contain the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>client.base_url</td>
<td><a href="http://localhost/api/v1">http://localhost/api/v1</a></td>
</tr>
<tr>
<td>client.sslverify</td>
<td>True</td>
</tr>
<tr>
<td>client.authuser</td>
<td>$USER</td>
</tr>
<tr>
<td>client.authpass</td>
<td>None</td>
</tr>
<tr>
<td>client.format</td>
<td>human</td>
</tr>
<tr>
<td>client.pretty</td>
<td>False</td>
</tr>
</tbody>
</table>

The base_url specifies the URL that the tools should use to connect to the head node's REST API and defaults to connecting to the standard location (http://localhost/api/v1) on the local machine. If the base_url specifies an HTTPS URL, then a client can disable SSL verification, but this is strongly discouraged as it bypasses the protections provided by HTTPS against impersonation and man-in-the-middle attacks. The authuser and authpass can be included to simplify authentication to the service, but be aware that specifying the authpass here may not be secure, depending on your environment.

The format argument affects the output format of data returned by the tools. The default value of "human" causes the tools to output an indented format with various computed values augmented with human-readable summaries. The alternative value of "json" will output the results as JSON formatted text, and the pretty argument can be used to turn on indentation for that JSON output.

### 7.6.3 --base-url and --user

Since an administrator may want to periodically connect to different head nodes or as a different user, command line arguments are provided to override those configuration settings. For example, the entire string passed to the --base-url argument is treated as a URL and is passed to the underlying Python requests library.

Any string passed to the --user argument will be split at its first colon, and the remainder of the string will be treated as the user's password. Providing a password this way is convenient, especially during testing, but is generally discouraged as the password could then be visible in /proc while the tool is running. If no password is provided either through command line or client configuration, then one will be requested when needed.

### 7.6.4 --show-uids, --human, --json, --pretty/--no-pretty

These arguments are used to change the tool output format, much like the corresponding client configuration variables described above. The --human and --json arguments override the client.format variable, and --pretty and --no-pretty can be used to override the client.pretty variable.

By default, tool output will show an object's name when referring to a named object, and the UID (or shortened UID) only if no name is defined. Using the --show-uids argument forces the display of full UIDs in place of more human-readable options. This is uglier, but occasionally useful to be absolutely certain about what object is being referenced.
7.6.5 --csv, --table, --fields

For ease of reading and automated parsing, the scyld tools can also produce output as CSV or in a table. Use the --fields argument to select fields to display and select from --csv or --table to print in your preferred format:

```
$ scyld-nodectl --fields "mac,Assigned IP=ip,BootConfig=attributes._boot_config" \  
  --table ls -l

Nodes |      mac | Assigned IP   | BootConfig
-----+----------+--------------+--------------
n0   | 08:00:27:f0:44:35 | 10.10.24.100 | DefaultBoot
n1   | 08:00:27:a2:3f:c9 | 10.10.24.101 | DefaultBoot
n2   | 08:00:27:e5:19:e5 | 10.10.24.102 | DefaultBoot
```

The above demonstrates how to both assign column names and select nested values such as individual attributes.

7.7 Common Subcommand Actions

In addition to the above arguments, some subcommand actions are common among the scyld-*ctl tools as well: list, create, clone, update, replace, delete. The precise details of what additional arguments these subcommand actions accept may differ between tools, but the generally supported arguments are discussed here.

7.7.1 list (ls)

List the requested object names, and optionally with --long or -l will display object details. The --raw option will display the actual JSON content as returned by the ClusterWare API call.

7.7.2 create (mk)

Create a new object using name-value pairs provided either on the command line or passed using the --content argument described below.

7.7.3 clone (cp)

Copy existing objects to new UIDs and names. Individual fields in the new objects can be overridden by name-value or a --content argument described below.

7.7.4 update (up)

Modify existing objects altering individual fields in name-value pairs or a --content argument described below.
7.7.5 replace (re)

Much like update, but completely replace the existing objects with new objects from fields defined in name-value pairs or a --content argument described below.

7.7.6 delete (rm)

Delete objects.

7.8 The then argument

Various tools (most commonly scyld-nodectl, although also scyld-adminctl, scyld-attribctl, scyld-bootctl, and scyld-imgctl) accept the then argument, which serves as a divider of a serial sequence of multiple subcommands for a single invocation of the scyld-* tool.

For example,

```
scyld-nodectl -i n0 reboot then waitfor up then exec uname -r
```

that initiates a reboot of node n0, waits for the node to return to an "up" state, and then executes `uname -r` on the node.

If any subcommand in the sequence fails, then the tool reports the error, skips any subsequent subcommands, and terminates.

7.9 The --content argument

The --content argument can be passed to several of the tools described earlier and is always paired with an argument to accept name-value pairs that can override content values. The --content argument can be followed by a JSON string or by a file containing JSON formatted data, INI formatted data, or a text file where each object is represented by rows of name-value pairs. If the argument to --content is a filename, it must be prefixed with an '@' symbol.

For example, an administrator could create a new boot configuration as follows:

```
scyld-bootctl create --content 
  "{"name": "TestBoot", "kernel": "/boot/vmlinuz-3.10.0-957.1.3.el7.x86_64"}"
```

Of course, a boot configuration also requires an initramfs:

```
cat > content.ini <<EOF
[BootConfig]
initramfs: @initramfs-3.10.0-957.1.3.el7.x86_64.img
EOF
scyld-bootctl -iTestBoot update --content @content.ini
```

Adding nodes to the database one at a time is tedious for large clusters, and the --content argument can streamline this process. Below are examples of three different files that could be passed via the --content argument to add nodes with explicit indices to the database:

JSON:
INI:

[Node0]
mac: 00:11:22:33:44:55
index: 1

[Node1]
mac: 00:11:22:33:44:66
index: 2

[Node2]
mac: 00:11:22:33:44:77
index: 3

Text:

mac=00:11:22:33:44:55 index=1
mac=00:11:22:33:44:66 index=2
mac=00:11:22:33:44:77 index=3

Although providing multiple objects at once makes sense for the create subcommand, the clone, update, and replace subcommands require a list of fields to alter and will collapse multiple objects into one set of variables. For example:

```
[
  { "name": "TestBoot" },
  { "kernel": "/boot/vmlinuz-3.10.0-957.1.3.el7.x86_64" },
  { "name": "AnotherBoot" }
]
```

when passed to scyld-bootctl would result in the selected boot configuration(s) being renamed to "AnotherBoot" and assigned the /boot/vmlinuz-3.10.0-957.1.3.el7.x86_64 kernel.

### 7.10 Files in database objects

In the ClusterWare database, boot configurations and images both contain references to files, either a kernel and an initramfs, or a root file system. The files themselves are not stored in the database but instead are referenced by the system on backend storage through plugins, such as the local_files plugin that works with locally mounted storage through the POSIX API.

When listing the details of a database object containing a file reference, the reference will be shown as a dictionary containing the file size, modification time, checksum, and an internal UID. To explore this we will start by listing a boot configuration created earlier in this document:

```
$ bin/scyld-bootctl ls -l
Boot Configurations
```

(continues on next page)
Files have been uploaded to both the initramfs and kernel fields. The `chksum` fields are the SHA-1 output and are used to detect data corruption, not as a security feature. The `mtime` is the UTC timestamp of the last time the underlying file was modified, and the `size` field is the size of the file in bytes. The `uid` field is how the object is referenced within the ClusterWare system and is the name passed to whatever plugin is interfacing with underlying storage. In the case of the `local_files` plugin, this is used as the name of the file on disk.

Because this output was generated for human readability, some fields (`last_modified`, `mtime`, `size`) have been augmented with human readable representations. Also, the `release` field was determined by examining the contents of the kernel file when it was uploaded.

### 7.11 Scyld ClusterWare Administrator Tools

This section of the Reference Guide describes the Scyld ClusterWare administrator tools. These tools are used by the cluster administrator and are not intended for use by the ordinary user.

#### 7.11.1 `scyld-add-boot-config`

**NAME**

`scyld-add-boot-config` -- Tool for creating ClusterWare boot configurations.

**USAGE**

```
```

**OPTIONAL ARGUMENTS**

- **-h, --help**
  
  Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

- **-v, --verbose**

  Increase verbosity.

- **-q, --quiet**

  Decrease verbosity.

- **-c, --config CONFIG**

  Specify a client configuration file `CONFIG`. 

<table>
<thead>
<tr>
<th>TestBoot</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>initramfs</strong></td>
</tr>
<tr>
<td><strong>chksum</strong>: aa1161a52b98287a3eac4677193c141a3648ebc</td>
</tr>
<tr>
<td><strong>mtime</strong>: 2019-02-09 21:13:08 UTC (0:00:52 ago)</td>
</tr>
<tr>
<td><strong>size</strong>: 20.0 MiB (20961579 bytes)</td>
</tr>
<tr>
<td><strong>uid</strong>: d247e4a1fde4ac3853e78c0f7683947</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>kernel</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>chksum</strong>: 5a464d2a82839d9c21c0fb7350d9cb0d5f8fed</td>
</tr>
<tr>
<td><strong>mtime</strong>: 2019-02-09 21:08:34 UTC (0:05:26 ago)</td>
</tr>
<tr>
<td><strong>size</strong>: 6.3 MiB (6639808 bytes)</td>
</tr>
<tr>
<td><strong>uid</strong>: fc96da551b94038b38d7ef662b34947</td>
</tr>
<tr>
<td><strong>last_modified</strong>: 2019-02-09 21:08:34 UTC (0:05:26 ago)</td>
</tr>
<tr>
<td><strong>name</strong>: TestBoot</td>
</tr>
<tr>
<td><strong>release</strong>: 3.10.0-957.1.3.el7.x86_64</td>
</tr>
<tr>
<td><strong>uid</strong>: 4978077e53b944b38c4cda007b9b97b7</td>
</tr>
</tbody>
</table>
If there are no attribute groups on this system, then automatically build an attribute group referencing a new boot configuration referencing a new image.

Select the pre-existing distro NAME to use when creating an image.

Create a repo and distro from the local or remote base distribution ISO, where PATH is a pathname or a URL.

Select the pre-existing image NAME this command should use.

Name the boot configuration as NAME.

Name the new attribute group as NAME.

Skip assigning the attribute group to nodes.

Run this command using the default options.

Specify the base URL of the ClusterWare REST API.

Masquerade as user USER with optional colon-separated password PASSWD.

This tool is used internally by the scyld-install tool to populate the initial (or cleared) database with the objects necessary to boot compute nodes. When run on a database with no attribute groups defined and passed the --auto-first argument this script will not ask the user any questions and will use default values. This should not be necessary for an administrator to run unless they have manually cleared the database using the managedb clear command:

```bash
scyld-add-boot-config --make-defaults
```

Rebuild the DefaultImage and DefaultBoot.

```bash
scyld-add-boot-config --iso CentOS-7-x86_64-DVD-1908.iso
```

Use the named ISO file to build a distro and repo named CentOS-7-x86_64-1908, and manually accept defaults that create a boot image and boot configuration, all named CentOS-7-x86_64-1908.

```bash
scyld-add-boot-config --iso CentOS-7-x86_64-DVD-1908.iso --image CentOS-7.7-Image --boot-config CentOS-7.7-boot --batch
```

Use the named ISO file in hands-off batch mode to build a repo and distro, both named CentOS-7-x86_64-1908, a boot image named CentOS-7.7-Image and a boot config named CentOS-7.7-boot.

Upon successful completion, **scyld-add-boot-config** returns 0. On failure, an error message is printed to stderr and **scyld-add-boot-config** returns 1.

### 7.11.2 scyld-adminctl

**NAME**

**scyld-adminctl** -- Query and modify administrators for the cluster.

**USAGE**

```
```
FIELDS] [--show-uids] [[-i | --ids] ADMINS | -a | --all] {list,ls, create,mk, clone, cp, update,up, replace,re, delete,rm}

DESCRIPTION
This tool does not control the details of authentication. For that, please consult Securing the Cluster in the Administrator's Guide.

OPTIONAL ARGUMENTS
-h, --help
Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

-v, --verbose
Increase verbosity.

-q, --quiet
Decrease verbosity.

-c, --config CONFIG
Specify a client configuration file CONFIG.

--show-uids
Do not try to make the output more human readable.

-a, --all
Interact with all administrators (default for list).

-i, --ids ADMINS
A comma-separated list of administrators to query or modify.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS
--base-url URL
Specify the base URL of the ClusterWare REST API.

-u, --user USER[:PASSWD]
Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS
--human
Format the output for readability (default).

--json
Format the output as JSON.

--csv
Format the output as CSV.

--table
Format the output as a table.

--pretty
Indent JSON or XML output, and substitute human readable output for other formats.

--no-pretty
Opposite of --pretty.

--fields FIELDS
Select individual fields in the result or error.

ACTIONS ON SPECIFIED ADMINISTRATOR(S)
list (ls)
List information about administrator(s).

create (mk) name=NAME
Add an administrator NAME.

clone (cp) name=NAME
Copy administrator to new identifier NAME.

update (up)
Modify administrator fields.

replace (re)
Replace all administrator fields. Deprecated in favor of "update".

delete (rm)
Delete administrators(s).

EXAMPLES
scyld-adminctl create name=hsolo
Add new administrator "hsolo".
scyld-adminctl -i hsolo clone name=cbaca
Copy the administrator properties for "hsolo" to a new administrator "cbaca".

RETURN VALUES
Upon successful completion, scyld-adminctl returns 0. On failure, an error message is printed to stderr and scyld-adminctl returns 1.

7.11.3 scyld-attribctl

NAME
scyld-attribctl -- Query and modify attribute groups for the cluster.

USAGE

OPTIONAL ARGUMENTS
  -h, --help     Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
  -v, --verbose  Increase verbosity.
  -q, --quiet    Decrease verbosity.
  -c, --config CONFIG Specify a client configuration file CONFIG.
  --show-uids    Do not try to make the output more human readable.
  -a, --all      Interact with all attribute groups (default for list).
  -i, --ids ATTRIBS A comma-separated list of attribute groups to query or modify.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS
  --base-url URL Specify the base URL of the ClusterWare REST API.
  -u, --user USER[:PASSWD] Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS
  --human Format the output for readability (default).
  --json Format the output as JSON.
  --csv Format the output as CSV.
  --table Format the output as a table.
  --pretty Indent JSON or XML output, and substitute human readable output for other formats.
  --no-pretty Opposite of --pretty.

ACTIONS ON SPECIFIED ATTRIBUTE GROUP(S)
list (ls) List information about attribute group(s).
create (mk) name=NAME Add an attribute group NAME.
clone (cp) name=NAME Copy attribute group to new identifier NAME.
update (up) Modify attribute group fields.
replace (re)  Replace all attribute group fields.
delete (rm)  Delete attribute groups.
get  Get attribute values.
set  Set attribute values.
clear  Clear attribute values.

EXAMPLES

scyld-attribctl create name=iScsi

Add a new attribute group.

scyld-attribctl -i iScsi set _boot_config=RebelBoot _boot_style=iscsi

Configure attributes to boot nodes using RebelBoot using iSCSI for root file system access.

RETURN VALUES

Upon successful completion, scyld-attribctl returns 0. On failure, an error message is printed to stderr and scyld-attribctl returns 1.

7.11.4 scyld-bootctl

NAME

scyld-bootctl -- Query and modify boot configurations for the cluster.

USAGE


OPTIONAL ARGUMENTS

-h, --help  Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
-v, --verbose  Increase verbosity.
-q, --quiet  Decrease verbosity.
-c, --config CONFIG  Specify a client configuration file CONFIG.
--show-uids  Do not try to make the output more human readable.
-a, --all  Interact with all boot configurations (default for list).
-i, --ids BOOTGROUPS  A comma-separated list of boot configurations to query or modify.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL  Specify the base URL of the ClusterWare REST API.
-u, --user USER[:PASSWD]  Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS

--human  Format the output for readability (default).
--json  Format the output as JSON.
--csv  Format the output as CSV.
```
--table     Format the output as a table.
--pretty    Indent JSON or XML output, and substitute human readable output for other formats.
--no.pretty Opposite of --pretty.

ACTIONS ON BOOT CONFIGURATION(S)

list (ls) [-l, --long | -L, --long-long | --raw]
List information about boot configurations.
   -l, --long        Show a subset of all optional information for each node.
   -L, --long-long  Show all optional information for each node.
   --raw            Display the raw JSON content from the database.

create (mk) [--content [JSON | INI_FILE]] [NAME=VALUE] ...
Add a boot configuration with optional JSON or INI_FILE content or with optional NAME / VALUE pairs.
   --content [JSON | INI_FILE]  Load this JSON or INI_FILE content into the database as a boot config.

clone (cp) [--content [JSON | INI_FILE]] [NAME=VALUE] ...
Copy boot configuration with optional JSON or INI_FILE content or with optional NAME / VALUE identifiers.
   --content [JSON | INI_FILE]  Overwrite fields in the cloned boot config.

update (up) [--content [JSON | INI_FILE]] [NAME=VALUE] ...
Modify boot configuration NAME field(s) with new value(s).
   --content [JSON | INI_FILE]  Overwrite this content into the database for a boot config.

replace (re) [--content [JSON | INI_FILE]] [NAME=VALUE] ...
Replace all boot configuration fields.
   --content [JSON | INI_FILE]  Overwrite this content into the database for a boot config.

delete (rm) [-r, --recurse]
Delete boot configuration(s).
   -r, --recurse    Optionally also delete the referenced image or iso-based repo.

download [--dest DIR] Filename ... Extract named file(s) (any of "initramfs", "kernel") from boot config and download to current working directory (or to directory DIR).

export [--no-recurse] [PATH]
Export the specified boot configuration NAME to the file NAME.export in the current working directory or in destination PATH.
   --no-recurse    Do not recurse through and include dependencies.

import [--no-recurse] [--boot-config NAME_BOOT] [--image NAME_IMG] NAME.export Import the NAME.export file into a local boot configuration (default embedded in NAME.export, or optionally renamed NAME_BOOT) and associated compute node image (or optionally renamed NAME_IMG).

EXAMPLES
```
Create a boot configuration with a premade kernel and initramfs.

```bash
scyld-bootctl create name=Fed29Boot \
    kernel=@/boot/vmlinuz-4.20.6-200.fc29.x86_64 \
    initramfs=@cw-ramfs-4.20.6-200.fc29.x86_64
```

Create a boot configuration with a premade kernel and initramfs.

```bash
scyld-bootctl -iFed29Boot download kernel
```

Download the kernel previously uploaded to the Fed29Boot configuration.

```bash
scyld-bootctl -iFed29Boot update \
    initramfs=@new-ramfs-4.20.6-200.fc29.x86_64 \
    description="Ramfs created Fed24"
```

Replace the initramfs with a new one.

```bash
scyld-bootctl -i DefaultBoot ls -l
```

Display details about the DefaultBoot configuration.

```bash
scyld-bootctl -i DefaultBoot update cmdline="enforcing=0 console=ttyS0,115200"
```

Update the `cmdline` that is passed to a booting kernel to a new value. Note that `update` changes the entire `cmdline`, so to append a new substring to an existing `cmdline`, first view the full boot config (as noted in the example above), then form a new `cmdline` string with existing pieces you wish to retain.

```bash
scyld-bootctl -i SlurmBoot export \
    mv SlurmBoot.export ExportSlurmBoot
```

Export the boot config SlurmBoot and associated image as file SlurmBoot.export, and rename that file to ExportSlurmBoot. Note that the boot config name is embedded in ExportSlurmBoot as “SlurmBoot”.

```bash
scyld-bootctl import ExportSlurmBoot
```

Import the ExportSlurmBoot contents to a different cluster as a new SlurmBoot boot config and associated compute node image.

```bash
scyld-bootctl import --boot-config Slurm19Boot ExportSlurmBoot
```

Import the ExportSlurmBoot contents to a different cluster as a new Slurm19Boot boot config and associated compute node image.

```bash
scyld-bootctl import --boot-config Slurm19Boot --image Slurm19Image ExportSlurmBoot
```

Import the ExportSlurmBoot contents to a different cluster as a new Slurm19Boot boot config and associate compute node image with new name Slurm19Image.

```bash
scyld-bootctl import --boot-config Slurm19Boot --no-recurse ExportSlurmBoot
```

Import the ExportSlurmBoot contents to a different cluster as a new Slurm19Boot boot config without including the embedded image.

**RETURN VALUES**

Upon successful completion, `scyld-bootctl` returns 0. On failure, an error message is printed to `stderr` and `scyld-bootctl` returns 1.
7.11.5 scyld-cluster-conf

NAME

scyld-cluster-conf -- load or save the cluster configuration file.

USAGE


OPTIONAL ARGUMENTS

- h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- v, --verbose Increase verbosity.
- q, --quiet Decrease verbosity.
- c, --config CONFIG Specify a client configuration file CONFIG.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL Specify the base URL of the ClusterWare REST API.

-u, --user USER[:PASSWD] Masquerade as user USER with optional colon-separated password PASSWD.

ACTIONS

load CLUSTER_CONFIG

Load CLUSTER_CONFIG as the new configuration file, optionally loading only nodes.

--dry-run Parse the file, but do not alter the database.

--nets-only Ignore other settings and only load networks.

--nodes-only Ignore other settings and only load nodes.

save CLUSTER_CONFIG Save the current configuration file to file CLUSTER_CONFIG.

CLUSTER CONFIGURATION FILES

The scyld-cluster-conf command is primarily used to load a cluster configuration into ClusterWare including the PXE boot network definition(s) and the node definitions. A minimal useful configuration file consists of at least an iprange and one or more nodes:

iprange 10.10.24.100
node 08:00:27:A2:3F:C9

The first IP address in the iprange will be used to identify a local interface on the head node in order to find networking details such as the network mask. The DHCP range will be assumed to cover from the first IP up to the network broadcast address, but a "last" address can also be provided to limit that range:

nodes 10
iprange 10.10.24.100/24 10.10.24.199
node 08:00:27:A2:3F:C9
node
node 08:00:27:A2:E4:A2

Note that the node count can be provided in the file and a warning will be printed if more than that many nodes are defined in the file. The netmask can also be supplied as shown in the iprange line. Nodes will be numbered in order...
starting with index 0 but a line with no MAC address will act as a placeholder meaning this file would define nodes n0 and n2.

**Important:** If multiple MAC addresses are included for a single node, only the first will be used.

Alternatively network definitions can specify where the node numbering actually starts:

```plaintext
1 10.10.24.100/24 10.10.24.199
node 08:00:27:A2:3F:C9
node 08:00:27:A2:E4:A2
```

This configuration file still defines a DHCP range of 100 IP addresses, now the nodes will be numbered starting with n1. In more complicated network configurations compute nodes may be split among multiple subnets:

```plaintext
1 10.10.24.100/24
node 08:00:27:A2:3F:C9
node 08:00:27:A2:E4:A2

21 10.10.25.100/24 10.10.25.199 via 10.10.24.4 gw 10.10.25.254
node 08:00:27:FE:A3:22
```

The first network definition will be limited to 20 IP addresses based on the first index of the second network definition. For networks that are not locally accessible to the head node(s), such as 10.10.25.0/24 in this case, the configuration file can also specify an optional route and compute node gateway. The route is specified through the `via` keyword and is only used to identify the appropriate interface for the DHCP server to listen to at run time. The gateway (`gw`) should be on the compute node network and will be provided to the booting nodes such that they can reach the head node cluster. A DHCP relay should be configured to forward DHCP traffic from the remote compute nodes to the head nodes and vice versa, and should populate the `giaddr` field of the DHCP request with an address on the compute node subnet. For directions on configuring DHCP relays, please see your switch or operating system documentation.

When defining multiple networks they must be defined in order of node indexing. Node indexes and IP addresses are assigned based on the most recently defined network so the above example defines 3 nodes, n1, n3, and n20. Additional nodes added dynamically will be assigned the lowest available index and the corresponding IP address.

**Important:** Note that loading a cluster configuration will completely overwrite any existing configuration, including deleting all previously defined nodes.

**Important:** We suggest restarting the clusterware service on all head nodes after loading a new cluster configuration.

**EXAMPLES**

```plaintext
scyld-cluster-conf save /root/cluster-conf-bak
    Save a copy of the current network configuration and node list.
scyld-cluster-conf load /root/cluster-conf-new
    Replace the existing node definitions with ones loaded from the /root/cluster-conf-new file.
```

**RETURN VALUES**
Upon successful completion, `scyld-cluster-conf` returns 0. On failure, an error message is printed to `stderr` and `scyld-cluster-conf` returns 1.

### 7.11.6 scyld-clusterctl

**NAME**

`scyld-clusterctl` -- Tool for manipulating global cluster settings.

**USAGE**

```
```

**DESCRIPTION**

Query and modify global cluster settings. This tool also includes commands for modifying the repositories and distributions used when making images, as well as commands to interact with cluster head nodes.

**OPTIONAL ARGUMENTS**

- `-h`, `--help`  
  Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- `-v`, `--verbose`  
  Increase verbosity.
- `-q`, `--quiet`  
  Decrease verbosity.
- `-c`, `--config` `CONFIG`  
  Specify a client configuration file `CONFIG`.

**ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS**

- `--base-url` `URL`  
  Specify the base URL of the ClusterWare REST API.
- `-u`, `--user` `USER[:PASSWD]`  
  Masquerade as user `USER` with optional colon-separated password `PASSWD`.

**FORMATTING ARGUMENTS**

- `--human`  
  Format the output for readability (default).
- `--json`  
  Format the output as JSON.
- `--csv`  
  Format the output as CSV.
- `--table`  
  Format the output as a table.
- `--pretty`  
  Indent JSON or XML output, and substitute human readable output for other formats.
- `--no-pretty`  
  Opposite of `--pretty`.

**CLUSTER-WIDE SETTINGS AND COMMANDS**

- `--get-group`  
  Print the default attribute group id.
- `--set-group` `ATTRIB_GROUP`  
  Set the default attribute group.
- `--get-naming`  
  Print the default node naming pattern.
- `--set-naming` `PATTERN`  
  Set the default node naming pattern.
- `--get-accept-nodes`  
  Display whether or not unknown nodes should be automatically added.
--set-accept-nodes \texttt{T|F} Set whether unknown nodes should be automatically added (T=true) or not (F=false).
\begin{itemize}
  \item [--distros] List distros supported by the head node(s).
  \item [--get-distro] Get the current default distro.
  \item [--set-distro \texttt{DISTRO}] Set the default distro to \texttt{DISTRO}.
  \item [--image-formats] List image formats supported by the head node(s).
  \item [--get-image-format] Get the current default image format.
  \item [--set-image-format \texttt{FORMAT}] Set the default image format to \texttt{FORMAT}.
\end{itemize}

DATABASE QUERYING AND MODIFICATION, SELECT A CLASS OF DATABASE OBJECT

\texttt{repos} \texttt{[list,ls, create,mk, clone,cp, update,up, replace,re, delete,rm, download]} Manipulate available repos using a subcommand:
\begin{itemize}
  \item \texttt{list (ls)}: List information about repo(s).
  \item \texttt{create (mk)}: Add a repo.
  \item \texttt{clone (cp)}: Copy repo to new identifier.
  \item \texttt{update (up)}: Modify repo fields.
  \item \texttt{replace (re)}: Replace all repo fields. Deprecated in favor of "update".
  \item \texttt{delete (rm)}: Delete repo(s).
  \item \texttt{download}: Download named files (any of 'iso').
\end{itemize}
\begin{itemize}
  \item \texttt{-i, --ids \texttt{REPOS}} A comma-separated list of repos to query or modify.
  \item \texttt{-a, --all} Interact with all repos. (Default for \texttt{list})
\end{itemize}

\texttt{distros} \texttt{[list,ls, create,mk, clone,cp, update,up, replace,re, delete,rm, import]} Manipulate available distros using a subcommand:
\begin{itemize}
  \item \texttt{list (ls)}: List information about distro(s).
  \item \texttt{create (mk)}: Add a distro.
  \item \texttt{clone (cp)}: Copy distro to new identifier.
  \item \texttt{update (up)}: Modify distro fields.
  \item \texttt{replace (re)}: Replace all distro fields. Deprecated in favor of "update".
  \item \texttt{delete (rm) [-r, --recurse]}: Delete distro(s).
\end{itemize}
\begin{itemize}
  \item \texttt{-r, --recurse} Optionally also delete any referenced repo.
  \item \texttt{import --name \texttt{NAME} [--release \texttt{REL}] \texttt{FILE} ...} Import one or more \texttt{FILE} repos into a distro \texttt{NAME}, and \texttt{REL} is an optional release string.
\end{itemize}
\begin{itemize}
  \item \texttt{-i, --ids \texttt{DISTROS}} A comma-separated list of distros to query or modify.
  \item \texttt{-a, --all} Interact with all distros. (Default for \texttt{list})
\end{itemize}

\texttt{heads} \texttt{[list,ls, clean, service, delete,rm]} Interact with cluster head nodes using a subcommand:
\begin{itemize}
  \item \texttt{list (ls)}: List information about services on the head node(s).
  \item \texttt{clean [ACTION]}: Clean unreferenced objects from head node database, where \texttt{ACTION} is:
delete (rm): Delete head nodes.

service [NAMES] [ACTION]: Interact with ClusterWare services (default: list), where ACTION is:

- --start: Start the service(s) NAMES.
- --stop: Stop the service(s) NAMES.
- --restart: Restart the service(s) NAMES.
- --enable: Enable the service(s) NAMES.
- --disable: Disable the service(s) NAMES.

-pools [list, ls, create, mk, clone, cp, update, up, replace, re, delete, rm] Manipulate compute node name pools using a subcommand:

- list (ls): List information about the name pools.
- create (mk): Add a name pool.
- clone (cp): Copy name pools to new identifiers.
- update (up): Modify name pool fields.
- replace (re): Replace all name pool fields. (Depreciated - use update.)
- delete (rm): Delete name pools.

-dyngroups [list, ls, create, mk, clone, cp, update, up, replace, re, delete, rm, nodes] Manipulate dynamic groups using a subcommand:

- list (ls): List information about the dynamic groups.
- create (mk): Add a dynamic group.
- clone (cp): Copy dynamic groups to new identifiers.
- update (up): Modify dynamic group fields.
- replace (re): Replace all dynamic group fields. (Depreciated - use update.)
- delete (rm): Delete dynamic groups.
- nodes: List nodes that currently meet the same selector.

-gitrepos [list, ls, create, mk, clone, cp, update, up, delete, rm] Manipulate git repos using a subcommand:

- list (ls): List information about the git repos.
- create (mk): Add a git repo.
clone (cp): Copy git repos to new identifiers.
update (up): Modify git repo fields.
delete (rm): Delete git repos.

-i, --ids GITREPOS A comma-separated list of git repos to query or modify.
-a, --all Interact with all git repos (default for list).

certs {list,ls, create, mk, clone, cp, update, up, delete, rm, assign} Manipulate certificate sources using a subcommand:
list (ls): List information about the certificate sources.
create (mk): Add a certificate source.
clone (cp): Copy certificate sources to new identifiers.
update (up): Modify certificate source fields.
delete (rm): Delete certificate sources.
assign: Assign the certificate sources to nodes, and create the certificates.

-i, --ids CERTS A comma-separated list of certificate sources to query or modify.
-a, --all Interact with all certificate sources (default for list).

EXAMPLES
scyld-clusterctl heads --help
Show the available subcommands: list (ls), clean, service, delete (rm).

scyld-clusterctl heads clean --help
Show the resources that can be cleaned: --all, -- files, --heads, --database, --dry-run.

scyld-clusterctl heads service
Display the names of all ClusterWare system services and their states.

scyld-clusterctl heads service --help
Show all the available actions on services: --start, -- stop, -- restart, -- enable, --disable.

scyld-clusterctl heads --all clean --all
Clean everything on all head nodes.

scyld-clusterctl pools --help
Show the available subcommands: list (ls), create (mk), clone (cp), update (up), delete (rm)

scyld-clusterctl pools create name=infiniband_nodes pattern=ib{} first_index=0
scyld-nodectl -i n[64-127] update naming_pool=infiniband_nodes
Create a node name group "infiniband_nodes" for nodes named "ibX", beginning with "ib0", and associate those names with nodes n64 to n127.

RETURN VALUES
Upon successful completion, scyld-clusterctl returns 0. On failure, an error message is printed to stderr and scyld-clusterctl returns 1.
7.11.7 scyld-imgctl

NAME

scyld-imgctl -- Query and modify images for compute nodes.

USAGE


OPTIONAL ARGUMENTS

-h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
-v, --verbose Increase verbosity.
-q, --quiet Decrease verbosity.
-c, --config CONFIG Specify a client configuration file CONFIG.
--show-uids Do not try to make the output more human readable.
-a, --all Interact with all node images (default for list).
-i, --ids IMAGES A comma separated list of node images to query or modify.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL Specify the base URL of the ClusterWare REST API.
-u, --user USER[:PASSWD] Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS

--human Format the output for readability (default).
--json Format the output as JSON.
--csv Format the output as CSV.
--table Format the output as a table.
--pretty Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty Opposite of --pretty.
--fields FIELDS Select individual fields in the result or error.

ACTIONS ON IMAGE(s)

list (ls) List information about node images.
create (mk) Add node image.
clon (cp) Copy node image to new identifiers.
update (up) Modify node image fields.
replace (re) Replace all node image fields.
delete (rm) Delete node image(s) from the remote cache.
download FILES
Download named files *FILES* (any of "content").

```
--dest DIR       Optional destination for the downloaded files. (Default is current working
directory.)
```

**stat**  Print the recorded file stats for an image.

```
capture [--save FILE] [-n, --node NODE] [--exclude PATHS] [--content JSON|INI_FILE] ...
```

Replace or create an image captured from a running system, adding optional name=value pairs.

```
--save FILE     Save the image locally instead of uploading.
-n, --node NODE Select the node to capture.
--exclude PATHS Exclude additional paths during image capture, specifying either pathnames
or a file @FILE that contains a list of pathnames.
--content [ JSON|INI_FILE ] Overwrite fields in the specified image(s).
```

**EXAMPLES**

```
sycld-imgctl -i DefaultImage download content
```

Download the previously uploaded image named DefaultImage.

```
sycld-imgctl -i DefaultImage stat
```

Print the last modified time and size of the previously uploaded image.

```
sycld-imgctl -i DefaultImage clone name=NewImage
```

Clone the DefaultImage to a new NewImage.

**RETURN VALUES**

Upon successful completion, *sycld-imgctl* returns 0. On failure, an error message is printed to *stderr* and *sycld-imgctl* returns 1.

### 7.11.8 scylld-install

**NAME**

*sycld-install* -- Tool to install ClusterWare and perform initial basic configuration of a head node, and to update an existing head node installation.

**USAGE**

```
```

**OPTIONAL ARGUMENTS**

```
-h, --help          Print usage message and exit. Ignore trailing args, parse and ignore preceding
                    args.
--config CONF_FILE  Specify a cluster configuration file to load and to initialize the DHCP server for
                    private cluster network.
--token TOKEN       Specify a cluster serial number or other authentication to use in the yum repository
                    file.
--yum-repo REPO_FILE Provide a complete yum repository file for ClusterWare.
```
-u, --update

If ClusterWare is already installed, then by default `scyld-install` asks for a confirmation that the intention is to update software, not to perform a new install. This optional argument explicitly directs `scyld-install` to update ClusterWare.

DATABASE LOAD/SAVE OPTIONS

-l, --load DATABASE_FILE
Load the ClusterWare database with the specified DATABASE_FILE.

-s, --save DATABASE_FILE
Save the ClusterWare database to the specified DATABASE_FILE.

--without-files
Do NOT include the contents of images and boot files when loading or saving.

ADVANCED OPTIONS

--iso PATH
Install or update ClusterWare using the named PATH, which is either the path of a ClusterWare ISO file or the URL of a remote ClusterWare ISO file.

--os-iso PATH
Create the DefaultImage and DefaultBoot using the named PATH, which is either the path of a base distribution ISO file or the URL of a remote base distribution ISO file.

--clear
Clear the ClusterWare database. **DELETES ALL IMAGES AND BOOT CONFIGURATIONS!**

--clear-all
A last resort more severe reset than --clear that removes all ClusterWare (except for clusterware-installer) and libcouchbase RPMs, and deletes /opt/couchbase/, /opt/scyld/clusterware/*, and /var/log/clusterware/ directories, then optionally reinstalls ClusterWare.

--no-tools
Don't install the ClusterWare tools. The default is to install the tools.

--join EXISTING_HEAD_IP
Join this head node to the EXISTING_HEAD_IP IP address of an existing head node.

--skip-version-check
Use this installer and skip the online checking for a newer version.

--database-passwd PASSWD
Specify the database administrative password.

--reconfigure
During an update, most steps that alter the head node OS will be skipped by default, but this option overrides and updates the base distribution.

--non-interactive
Execute the installer non-interactively, choosing default answers to the interactive questions in a way that most users would do. This is appropriate for using the installer in a script.

EXAMPLES

```
scyld-install --clear
Clear the database, leaving it empty, and undo any existing ClusterWare installation.
```

```
scyld-install --clear --config cluster-conf
Clear the database, leaving it empty, and undo any existing ClusterWare installation, then reset the database to the specified cluster-conf parameters.
```

RETURN VALUES

Upon successful completion, `scyld-install` returns 0. On failure, an error message is printed to stderr and `scyld-install` returns 1.
7.11.9 scyld-kube

NAME

scyld-kube -- Tool for managing Kubernetes control plane and worker nodes.

USAGE

[--prep-lb OPTIONS] [--clear-lb] [--cluster [IP | NODE]] [--image IMAGE] [--token
TOKEN] [--cahash CAHASH] [--certificate-key KEY] [--all] [--up]

DESCRIPTION

To administer Kubernetes in a cluster, install the clusterware-kubeadm package on either a Scyld ClusterWare head
node, a full-install ClusterWare compute node, or a separate non-ClusterWare server. This package contains the scyld-
kube tool. See the Using Kubernetes appendix for detailed examples.

STANDARD OPTIONS

- -h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding
args.

--init Initialize the only control plane node.

--join Add worker node.

--init-ha Initialize the first control plane node for High Availability (HA).

--join-ha Add control plane node for High Availability (HA).

--version VERSION Optionally specify a major.minor.patch version for an --init, --init-ha,
--join, or --join-ha. Default is the latest version.

--prep-lb APISERVER_VIP[:APISERVER_PORT:ROUTER_ID:AUTH_PASS]

MASTER_ID:MASTER_IP,BACKUP1_ID:BACKUP1_IP,...

Re-generate High Availability (HA) load balancer config files from local template.

APISERVER_VIP The virtual IP address of the Application Programming interface (API) server within the
network subnet.

APISERVER_PORT An unused port (default 4200) for the API server.

ROUTER_ID: Default 51.

AUTH_PASS: Default 42

MASTER_ID:MASTER_IP,BACKUP1_ID:BACKUP1_IP,... A comma- and a colon-separated list of master
and backup hosts' unique ID and IP addresses.

--clear-lb Clear High Availability (HA) load balancer config files (under /opt/scyld/
clusterware-kubeadm).

--cluster [ IP | NODE ] Optionally specify the cluster to join.

--image IMAGE Optionally modify the specified image for persistence across compute node re-
boots, so that nodes with this IMAGE auto-join when booted.

--token TOKEN Optionally specify first control plane's IP or node name; use with --join or
--join-ha.

--cahash CAHASH Optionally specify first control plane's discover-token-ca-cert-hash; use with
--join or --join-ha.
--certificate-key KEY  Optionally specify first control plane's certificate-key; use with --join or --join-ha.

**NODE SELECTION OPTIONS**

- `-i, --ids NODES`  A comma-separated list of nodes or an admin-defined group of nodes to act upon.
- `--all`  Configure all nodes (rare).
- `--up`  Configure all up nodes (rare).

**ADVANCED OPTIONS**

- `--core-inst`  Only install core packages.

**EXAMPLES**

**RETURN VALUES**

Upon successful completion, `scyld-kube` returns 0. On failure, an error message is printed to `stderr` and `scyld-kube` returns 1.

### 7.11.10 scyld-mkramfs

**NAME**

`scyld-mkramfs` -- Tool to create an initial root file system image.

**USAGE**

`scyld-mkramfs [-h] [OPTION] ... -o, -output PATH`

**STANDARD OPTIONS**

- `-h, --help`  Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- `--update, -u BOOT`  Space-separated list of boot configurations to update.
- `--output, -o PATH`  Where to write the initramfs.
- `--image, -i IMGID`  Uses `scyld-modimg` to install the `clusterware-tools` package inside the image `IMGID`, and then executes `scyld-mkramfs` inside the image to create the root file system image.
- `--kver VERSION`  Specify the kernel version to use, overriding the default of the head node's current kernel (viewed with `uname -r`).

**ADVANCED OPTIONS**

- `--no-selinux`  Do not include SELinux support.
- `--stripped`  Exclude all network drivers not loaded on some node.
- `--drivers NAMES`  Space-separated list of additional kernel drivers to include.
- `--modules NAMES`  Space-separated list of additional dracut modules to include.
- `--ramfs-conf PATH`  Use a config file from a non-standard location `PATH`.

**EXAMPLES**

`scyld-mkramfs --update OpenMPI-Slurm-Boot`

Rebuild the initramfs used by the OpenMPI-Slurm-Boot boot configuration.

`scyld-mkramfs --update OpenMPI-Slurm-Boot --drivers mlx4_core`
Rebuild the initramfs used by the OpenMPI-Slurm-Boot boot configuration after adding the mlx4_core driver (and its dependencies).

```bash
scyld-mkramfs --update OpenMPI-Slurm-Boot --kver 3.10.0-1160.45.1.el7.x86_64
```

Rebuild the initramfs in the boot config after an administrator installs a new kernel into the image that the boot config is using. The `--kver` argument is needed if there are multiple kernels installed in the image.

**RETURN VALUES**

Upon successful completion, `scyld-mkramfs` returns 0. On failure, an error message is printed to `stderr` and `scyld-mkramfs` returns 1.

### 7.11.11 scyld-modimg

**NAME**

`scyld-modimg` -- Tool for manipulating image contents.

**USAGE**

```
```

**ACTIONS**

- `list (ls)` List information about node images.

**OPTIONAL ARGUMENTS**

- `-h, --help` Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- `-v, --verbose` Increase verbosity.
- `-q, --quiet` Decrease verbosity.
- `-c, --config CONFIG` Specify a client configuration file `CONFIG`.
- `-a, --all` Select all local images (default).
- `-i, --image IMAGE` Or select an image by its name `IMAGE`.
- `--download-only [PATH]` Download a new local copy and then exit. If `PATH` is provided, then it is overwritten; otherwise any cached changes are lost.
- `--freshen` Discard any cached changes.
- `--overwrite` Keep the same UID after modifications and overwrite any existing image on upload.
- `--no-overwrite` Opposite of `--overwrite`.
- `--upload` Upload the final version. NOTE: This must follow all image manipulations options.
- `--no-upload` Opposite of `--upload`. 

---

140 Chapter 7. Reference Guide
--discard                   Discard image changes.
--no-discard                Opposite of --discard.
--pkgmgr CONF              Specify a package config file (using the '@' prefix), or pass the config contents as a string CONF, to override the default config example seen in /opt/scyld/clusterware-tools/examples/pkgmgr.ini. A cluster administrator wishing to customize pkgmgr.ini should copy that example to another location, then add, delete, and/or modify that copy as desired.
--shell SHELL              Select the shell to use in the image for the mutation operations (default /bin/bash).

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS
--base-url URL            Specify the base URL of the ClusterWare REST API.
-u, --user USER[:PASSWD]  Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS
--human                   Format the output for readability (default).
--json                    Format the output as JSON.
--csv                     Format the output as CSV.
--table                   Format the output as a table.
--pretty                  Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty               Opposite of --pretty.
--show-uids               Do not try to make the output more human readable.

CACHE MANIPULATIONS
Make changes to the local image cache.
--clean-local             Delete local images not found in the manifest and any temporary files or directories.
--register-all            Record information about all locally stored images.

IMAGE MUTATIONS
The following steps are be performed on a selected image. Any failure terminates execution.

--chroot [KVER]           Chroot into the unpacked image to allow for manual modifications. Optionally specify KVER, which is the version of a kernel inside the image, which informs a `uname -r` inside the chroot to identify the specific kernel version if/when configuring software needing to link against that kernel. (Otherwise a `uname -r` inside a chroot names the kernel of the host system executing the `scyld-modimg`, not the kernel installed inside the chrooted image.) See --execute COMMAND and EXAMPLES.

--create [DISTRO]        Create a new image from scratch, optionally specifying a non-default distro name DISTRO.
--delete                 Delete the selected image(s) from the local cache.
--import FILE            Import an existing tar, squashfs, or singularity image.

--capture NODE [--set-name IMAGE] Capture image from a booted node. If the optional --set-name IMAGE is not supplied, then the tool prompts the user for an IMAGE name to create or overwrite.
--install PKGS           Install packages PKGS into the image.
--set-name NAME          Set the name of the image.
--set-description DESC  Set the description for the image.

--query [PKGS]  Query package versions from the image (default=ALL).

--update [PKGS]  Update specified packages in the image (default=ALL).

--uninstall PKGS  Uninstall packages from the image.

--unpack TARGZ  Unpack a tar.gz file into the image.

--execute COMMAND  Execute a command in the unpacked image. Note that this COMMAND can
include KVER=<kernelVersion>, thereby overriding the default behavior of a
uname -r executing inside the image. See --chroot KVER and EXAMPLES.

--copyin SRC DEST  Copy files or directories from SRC into the image as DEST.

--copyout SRC DEST  Copy files or directories SRC out of the image to destination DEST.

--mount PATH  Unpack the image into PATH and bind-mount various folders as if preparing for
--chroot. After the mount the image can be customized by other commands,
such as ansible, before being repacked.

--unmount PATH  Repack the image from a previously mounted PATH.

EXAMPLES

scyld-modimg -i NewImage --query kernel,clusterware-node

  Display the kernel and clusterware-node RPM versions installed in the image.

scyld-modimg -i NewImage --query

  Display all RPMs installed in the image.

scyld-modimg -i NewImage --chroot

  Examine and/or modify the contents of the image using chroot.

scyld-modimg -i NewImage --chroot 3.10.0-1160.45.1.el7.x86_64

  Explicitly override the uname -r output when executed inside the image.

scyld-modimg -i NewImage --execute 'KVER=3.10.0-1160.45.1.el7.x86_64
uname -r'

  Explicitly control the uname -r output inside the image when executing commands, e.g., in this case the
uname -r
command.

scyld-modimg --capture n8 --set-name CapturedN8image --upload

  Capture the image executing on node n8, give it the name "CapturedN8image", and upload it.

QUIRKS

Note that when exiting a --chroot, several directories do not get repacked and saved into the image, including /tmp/,
/var/tmp/, /var/cache/yum.

RETURN VALUES

Upon successful completion, scyld-modimg returns 0. On failure, any changes are discarded, an error message is
printed to stderr, and scyld-modimg returns 1.
7.11.12 scyld-nodectl

NAME

scyld-nodectl -- Query and modify nodes for the cluster.

USAGE


OPTIONAL ARGUMENTS

-h, --help
Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

-v, --verbose
Increase verbosity.

-q, --quiet
Decrease verbosity.

-c, --config CONFIG
Specify a client configuration file CONFIG.

--show-uids
Do not try to make the output more human readable.

-a, --all
Interact with all nodes (default for list).

-i, --ids NODES
A comma-separated list of nodes or an admin-defined group of nodes to act upon.

--up
Interact with all "up" nodes.

--down
Interact with all "down" nodes.

--booting
Interact with all "booting" nodes.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL
Specify the base URL of the ClusterWare REST API.

-u, --user USER[:PASSWD]
Masquerade as user USER with optional colon-separated password PASSWD.

FORMATTING ARGUMENTS

--human
Format the output for readability (default).

--json
Format the output as JSON.

--csv
Format the output as CSV.

--table
Format the output as a table.

--pretty
Indent JSON or XML output, and substitute human readable output for other formats.

--no-pretty
Opposite of --pretty.

ACTIONS

list (ls) [(-l | --long) | --long-long | --raw]
Show information about nodes.

-l, --long
Show a subset of all optional information for each node.

-L, --long-long
Show all optional information for each node.

--raw
Display the raw JSON content from the database.
**hardware**  Show the "hardware" information subset of scyld-nodectl ls -L.

**status**  [--long] [--long-long] [--health]

  Show node status.

    --health  Show status based on _health attribute.

    -l, --long  Show a subset of all optional information for each node.

    -L, --long-long  Show all optional information for each node.

    --raw  Display the raw JSON content from the database.

    --refresh  Show basic node states, refreshing for any state change.

**create**  (mk)  [--content JSON | INI_FILE] [NAME=VALUE ...]

  Add a node, commonly by specifying its MAC address (e.g., mac=MACaddr, that assigns the next available node number and associated IP address).

    --content JSON | INI_FILE  Load this content into the database as a node.

**clone**  (cp)  [--content JSON | INI_FILE] [NAME=VALUE ...]

  Copy node with new NAME/VALUE identifier pairs.

    --content JSON | INI_FILE  Overwrite fields in the cloned node.

**delete**  (rm)  Delete node(s).

**update**  (up)  [--content JSON | INI_FILE] [NAME=VALUE ...]

  Modify node NAME field(s) with new value(s).

    --content JSON | INI_FILE  Overwrite this content into the database for a node.

**replace**  (re)  [--content JSON | INI_FILE] [NAME=VALUE ...]

  Replace all node fields.

    --content JSON | INI_FILE  Replace all fields with the specified content.

**set**  [--content JSON | INI_FILE] [NAME=VALUE ...]

  Set attribute value(s).

    --content JSON | INI_FILE  Import the NAME/VALUE pairs from the file into the node attributes.

**clear**  [-a | --all] [NAME ...]

  Delete attribute name(s) and their value(s).

    -a, --all  Delete all attributes.

**join**  GROUP ...  Append GROUP(S) to the node group lists.

**leave**  [-a | --all] [GROUP ...]

  Remove GROUP(S) from the node group lists.

**reboot**  [--soft | --hard] [--kexec] [--force] [--timeout SECS]

  Reboot node(s) using either "soft" (using ssh) or "hard" (using ipmi) or kexec methods. If none is specified, then the default behavior is to initially attempt a "soft" reboot; and if after a short delay (default 5 seconds) the node does not appear to begin a reboot, then perform a "hard" power cycle. Ignore the reboot if the node's _no_boot is set to one, unless an overriding --force argument is supplied.
--soft  Reboot node(s) using ssh methods.
--hard  Reboot node(s) using ipmi methods.
--kexec  Boot directly into a new kernel without a full reboot which would include Power On Self Test (POST) and hardware initialization. See man kexec for details. This is implemented on a compute node using the ClusterWare reboot-kexec tool which installs from the clusterware-node package.
--force  Override the node's _no_reboot attribute value when set to 1.
--timeout SECS  Wait a non-default SECS seconds between "soft" and "hard" methods.

shutdown [--soft | --hard] [--timeout SECS]
Shutdown node(s) using either soft (using ssh) or hard (using ipmi) methods. If neither --soft nor --hard is specified, then the default behavior is to first attempt a soft shutdown; if after a short delay the node does not appear to begin a shutdown, then perform a hard power off.
--hard  Shutdown node(s) using ipmi methods.
--soft  Shutdown node(s) using ssh methods.
--timeout SECS  Wait SECS seconds between "soft" and "hard" methods.

power {on | off | cycle | status | setnext BOOTDEV}
Display or control the node power state through the plugin defined by the node's power_uri, usually ipmi. The options on, off, cycle, and status correspond to ipmitool actions.
The option setnext specifies the boot device or method to use for the next node boot. BOOTDEV choices are none, pxe, disk, and bios.

sol [--enable ID] [--steal]
Start a serial-over-lan connection using the local ipmitool.
--enable ID  If SOL payload is disabled, then attempt to enable for ID and retry.
--steal  If an SOL session is currently active for that node, then deactivate that session and retry.

scp See scyld-nodectl exec in EXAMPLES, below.

ssh [--pubkey FILE]
Create an SSH connection to the specified node as the user root. This is done using a local SSH key that is temporarily copied to the compute node through the head node and removed after the command completes. The user can provide their own public key, or one will be generated and stored in ~/.scyldcw/tempauth.key. See Administrator's Guide for necessary steps if compute nodes employ an SSH banner message.
--pubkey FILE  Specify a file containing a public key to use for this connection.

script [--direct] SCRIPT
Execute the specified ClusterWare SCRIPT (distributed in the clusterware-node package) on the specified compute node(s). The script name list (or ls) displays names of the available scripts, which generally execute automatically at boot time to facilitate various node initializations and have limited usefulness for later execution by a cluster administrator. However, the scripts fetch_hosts (re-download the list of head nodes) and update_keys (update SSH keys) may be useful in rare circumstances for a booted node.
--direct  Bypass the head node when executing the script.

Execute the `CMD` (double-quotes are optional) string on node(s). The `scyld-nodectl exec` command passes its current stdin, stdout, and stderr to the remote command, or uses the `--stdin`, `--stdout`, and/or `--stderr` arguments to override the default(s) with a file.

When run via an `ssh` command (e.g., `ssh cwhead scyld-nodectl --up exec uptime`), that stdin should be provided and closed with Ctrl-d, or `ssh` should be passed the `-t` argument to force tty allocation. Otherwise the command will detect stdin is a pipe and wait for end-of-file.

If the compute nodes are configured to show an SSH banner on login, then it will be printed on standard error, although that can be filtered out. Increasing the command verbosity with `--vv` will stop the banner filtering. See the Administrator's Guide for necessary steps if compute nodes employ an SSH banner message.

Commands executed on multiple nodes will execute in parallel. The degree of fan-out can be controlled through the `ssh_runner.fanout` configuration variable in `base.ini`. Because these commands execute in parallel, their output may be interleaved or not in node index order. Override this with `grouped` or `--in-order` arguments.

For `sshpass` functionality, see `_remote_pass` in the *Reserved Attributes* section of the Reference Guide.

---
---

**--direct** Bypass the head node when executing the command.

**--grouped** Results are locally buffered and printed grouped by node.

**--in-order** Output is printed in node index order, implies `--grouped`.

**--stdin IN** Optional @file or input string provided to `CMD` executing on the node(s) as stdin.

**--stdout OUT** Optionally write stdout to the file `OUT`. Any `{}` in the filename gets translated to the node name (see EXAMPLES).

**--binary** Treat `CMD` output as binary data.

**--stderr ERR** Optionally write stderr to the file `ERR`. Any `{}` in the filename gets translated to the node name (see EXAMPLES). An `ERR` value consisting of the string STDOUT will merge stderr into stdout.

**ping [COUNT]** ping the specified node(s) with `COUNT` packets (default 1).

**waitfor [Options] COND**

Complete when one or more of the specified nodes meet the condition `COND`, which is either an expression or a `@`-prefixed file name. If no nodes are specified, then defaults to `--all`.

**--failure COND** Also complete if the failure condition becomes true.

**--timeout SECS** Complete after `SECS` seconds if condition(s) never become true.

**--name NAME** Use the currently defined `COND` state known as `NAME`, or define a new `COND` and remember it as `NAME`.

**--load-only** Just save the state sets into the database.

**--delete NAME** Delete an existing state set `NAME`.

**--show [NAME]** Show a list of all state sets, or optionally just the details of one.

**--stream** Stream back ongoing results instead of returning the first result and exiting.

**--skip** Do not use or print the initial node states.

**--one-per** Stream node state changes with one node per line.

**--this-head** Only return state changes handled by the current head.
EXAMPLES

scyld-nodectl list
    List all node names.

scyld-nodectl status
    Shows the basic state of each node.

scyld-nodectl status n5
    Shows the basic state of node n5.

scyld-nodectl -i n5 ls -L
    Shows full information available for node n5.

scyld-nodectl -i %groupx ls -l
    Shows an expanded information available for each node joined to the admin-defined group groupx.

scyld-nodectl create mac=00:25:90:0C:D9:3C
    Add a new node to the end of the current list of nodes.

scyld-nodectl create mac=00:25:90:0C:D9:3C index=10
    Add a new node beyond the end of the current list of nodes as node n10.

scyld-nodectl -i n3 update mac=40:25:88:0C:B9:2C
    Replace the current MAC address for node n5 with a new MAC address.

scyld-nodectl -i n20 update power_uri=ipmi://admin:passwd@10.2.255.37
    Replace the current power_uri (defaults to "none") to an ipmitool authentication and BMC IP address.

scyld-nodectl -in2 ssh
    Use ssh to open a shell on node n2.

scyld-nodectl -i n2 exec ls /var/log
    Execute ls /var/log on node n2, directing stdout and stderr to scyld-nodectl’s stdout and stderr, respectively.

scyld-nodectl -i n2 exec --stdout /tmp/n2.var.log ls /var/log
    Execute ls /var/log on node n2, directing stdout to the head node file /tmp/n2.var.log.

scyld-nodectl -i n[2-4] exec --stderr STDOUT --stdout /tmp/{}.var.log ls /var/log
    Execute ls /var/log on nodes n2, n3, and n4, directing both stderr and stdout to the head node files /tmp/n2.var.log, /tmp/n3.var.log, and /tmp/n4.var.log, respectively.

scyld-nodectl --up exec --stderr STDOUT --stdout /tmp/{}.var.log ls /var/log
    Perform the same action as above, although this time for all the "up" nodes.

scyld-nodectl -in5 exec --stdout /tmp/n5-log.tar.gz tar -czf- /var/log
    Execute tar -czf- /var/log on node n5, directing the stdout of the packed result into the head node file /tmp/n5-log.tar.gz.

scyld-nodectl -in5 exec --stdin=@/tmp/n5-log.tar.gz tar -C /root -xzf-
    Send the local file /tmp/n5-log.tar.gz as the stdin to node n5 as it executes tar -C /root -xzf- to unpack the stdin contents at /root.
Reboot node n4, then wait until the node returns to the "up" state.

Reboot node n4, then wait until the node returns to the "up" state. Another supported shorthand is the conditional "down".

For node n0 establish a waitfor state condition described in that specified examples file, in which the state condition is named status. If no -i <NODE(s)> is specified, then defaults to --all.

For all nodes re-establish a waitfor state condition for the previously defined state named status. When the condition is true for any node, write the state to stdout and exit.

For all nodes re-establish a waitfor state condition for the previously defined state named status. When the condition is true for any node, write the state change to stdout and continue executing.

Initiate a reboot of node n0, wait for the node to return to an "up" state, and then execute `uname -r` on the node.

Upon successful completion, `scyld-nodectl` returns 0. On failure, an error message is printed to `stderr` and `scyld-nodectl` returns 1.

### 7.11.13 scyld-nssctl

**NAME**

`scyld-nssctl` -- Manage the `scyld-nss` service.

**USAGE**

`scyld-nssctl [-h] [-v] [start] [stop], [status]`

**DESCRIPTION**

A basic tool to start, stop, or show the status of the `scyld-nss` functionality without affecting the systemd status. The 'start' and 'stop' actions must be executed by user `root`.

**OPTIONAL ARGUMENTS:**

-h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

-v Increase verbosity.

**start** Insert `scyld` in the `/etc/nsswitch.conf hosts` line to enable (or reenable) `scyld-nss` functionality.

**stop** Disable `scyld-nss` functionality by removing `scyld` in the `/etc/nsswitch.conf hosts` line.

**status** Display the current status of `scyld-nss` functionality. (The default if no argument is supplied.)

**EXAMPLES**

`scyld-nssctl` Display the current state of `scyld-nss`.

`scyld-nssctl status` Display the current state of `scyld-nss`.
**scyld-nssctl stop** Disable *scyld-nss* functionality.

**scyld-nssctl start** Enable *scyld-nss* functionality.

**RETURN VALUES**

Upon successful completion, **scyld-nssctl** returns 0. On failure, an error message is printed to stderr and **scyld-nssctl** returns nonzero.

### 7.11.14 scyld-reports

**NAME**

*scyld-reports* -- Manage and generate cluster reports.

**USAGE**

```
```

**DESCRIPTION**

This tool manages and generates cluster reports.

**OPTIONAL ARGUMENTS**

- **-h, --help** Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- **-v, --verbose** Increase verbosity.
- **-q, --quiet** Decrease verbosity.
- **-c, --config CONFIG** Specify a client configuration file `CONFIG`.

**ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS**

- **--base-url URL** Specify the base URL of the ClusterWare REST API.
- **-u, --user USER[:PASSWD]** Masquerade as user `USER` with optional colon-separated password `PASSWD`.

**FORMATTING ARGUMENTS**

- **--human** Format the output for readability (default).
- **--json** Format the output as JSON.
- **--csv** Format the output as CSV.
- **--table** Format the output as a table.
- **--pretty** Indent JSON or XML output, and substitute human readable output for other formats.
- **--no-pretty** Opposite of --pretty.

**ACTIONS**

**setup** [--accountant [ACCOUNTANT]] Specify the accountant URL with credentials. Contact Penguin Computing Support or Professional Services for assistance.

**usage** [--start START] [--end END] [--users USERS] [--queues QUEUES]

Display cluster usage. This requires a prior *scyld-reports setup* of the accountant connection.

- **--start START** Specify a starting date. Defaults to first of current month.
Scyld ClusterWare11 Documentation, Release 11.9.1

unknown [--newer-than SECS] [--columns COLS] [--sort COLS] [--lookup]

Display MAC addresses of unknown nodes that have attempted to boot.

--newer-than SECS Only show MACs with contact within the last SECS seconds.

--columns COLS Display the columns in the desired comma-separated name order. Default is display all columns.

--sort COLS Sort entries based on the provided column names.

--lookup Attempt to identify MAC OUIs.

--as-creates [POOL] Print a specific scyld-nodectl create command for each unknown MAC, optionally associating a new node with a specific POOL.

--flush Flush the current unknown node list.

EXAMPLES

scyld-reports unknown

Display the full list of unknown nodes that have attempted to boot.

scyld-reports unknown --newer-than 1200 --columns mac,seen

Display only nodes which have made contact in the last 1200 seconds (20 minutes), showing each node's MAC address and last seen timestamp.

scyld-reports unknown --newer-than 600 --as-creates

Display only nodes which have made contact in the last 600 seconds (10 minutes), and for each show the scyld-nodectl create mac=<MACADDR> command that would add that node as a compute node.

RETURN VALUES

Upon successful completion, scyld-reports returns 0. On failure, an error message is printed to stderr and scyld-reports returns 1.

7.11.15 scyld-sysinfo

NAME

scyld-sysinfo -- Capture the system state information.

USAGE


DESCRIPTION

The tool works best when executed by a cluster administrator who is either user root or a user with sudo rights. The executing user must have write access to the current working directory.

The tool captures elements of the current system state into a subdirectory of the current working directory with the name sysinfo-$hostname-YY-MM-DD (using a 2-digit Year-Month-Day). This "capture" subdirectory is compressed by
default into a gzip'ed tarball; alternatively, the optional \texttt{--no-tar} argument skips that compression and allows the administrator to explore the "capture" subdirectory to view exactly what information the tool has captured.

The administrator can employ a \texttt{blacklist} file containing a list of files and directories to \texttt{not} capture, passing this \texttt{blacklist} path to the tool with the \texttt{no-save} argument. The administrator can also use \texttt{--no-tar} and manually delete captured files and subdirectories within \texttt{sysinfo-$(hostname)-YY-MM-DD}, then manually compress the final captured information for archival or for sending the file to others for examination.

The tool also optionally captures sysinfo state for compute nodes, for either all \texttt{up} nodes or a specific node or list of nodes.

If the optional \texttt{-d DIR_SUBSTR} string is specified, then the directory name contains that alphanumeric string, e.g., \texttt{sysinfo-DIR_SUBSTR-$(hostname)-YY-MM-DD.tar.gz}.

If \texttt{-m MESSAGE} is specified, then the \texttt{MESSAGE} string is retained as the contents of the file \texttt{DESCRIPTION} at the top of the output directory. If \texttt{-m MESSAGE} is not specified, then the script queries the user for optional multi-line input that is retained as file \texttt{DESCRIPTION} in the output directory.

In the rare event that the tool aborts while capturing data, note that a partial capture is still available as the subdirectory \texttt{sysinfo-$(hostname)-YY-MM-DD} in the current working directory.

\textbf{OPTIONAL ARGUMENTS:}

\begin{itemize}
\item \texttt{-h, --help} \texttt{Print usage message and exit. Ignore trailing args, parse and ignore preceding args.}
\item \texttt{-V} \texttt{Print the scyld-sysinfo version and Scyld package versions.}
\item \texttt{--no-save BLACKLIST} \texttt{Do not save files/directories listed in file \texttt{BLACKLIST}.}
\item \texttt{--no-tar} \texttt{Leave the output as a subdirectory, not as a gzip'ed tarball.}
\item \texttt{--up} \texttt{Optionally capture the state of all \texttt{up} compute nodes.}
\item \texttt{-i NODES} \texttt{Optionally capture the state of a specific node or nodes.}
\item \texttt{-d DIR_SUBSTR} \texttt{Insert the alphanumeric string \texttt{DIR_SUBSTR} into the output directory/tarball name.}
\item \texttt{-d MESSAGE} \texttt{If specified, then the \texttt{MESSAGE} string is retained as the contents of file \texttt{DESCRIPTION} at the top of the output directory.}
\end{itemize}

\textbf{EXAMPLES}

\begin{itemize}
\item \texttt{scyld-sysinfo} \texttt{Capture the state of the current node into a gzip'ed tarball, executed as user root.}
\item \texttt{scyld-sysinfo \texttt{--no-tar}} \texttt{Capture the state of the current node into a human-readable subdirectory of the current working directory.}
\item \texttt{scyld-sysinfo -I -d UMich} \texttt{The output directory name for the head node "headnode1" is "sysinfo-UMich-headnode1-YY-MM-DD".}
\item \texttt{scyld-sysinfo \texttt{-m "dhcpd fails with network error"}} \texttt{The output directory contains the file \texttt{DESCRIPTION} that contains the specified string.}
\item \texttt{scyld-sysinfo \texttt{--up}} \texttt{Capture the state of the current head node and all the \texttt{up} compute nodes.}
\item \texttt{scyld-sysinfo \texttt{-i n0-n10}} \texttt{Capture the state of the current head node and compute nodes n0 through n10.}
\end{itemize}
scyld-sysinfo -i n0,n2,n100

Capture the state of the current head node and compute nodes n0, n2, and n100.

RETURN VALUES

Upon successful completion, scyld-sysinfo returns 0. On failure, an error message is printed to stderr and scyld-sysinfo returns nonzero.

7.11.16 scyld-tool-config

NAME

scyld-tool-config -- "Command line tool" for ClusterWare

USAGE


DESCRIPTION

The generic command line tool for ClusterWare.

OPTIONAL ARGUMENTS

- -h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- -v, --verbose Increase verbosity.
- -q, --quiet Decrease verbosity.
- -c, --config CONFIG Specify a client configuration file CONFIG.
- --yes Answer yes or to defaults to all questions.
- --example Generate an example file (implies --yes).

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL Specify the base URL of the ClusterWare REST API.

- -u, --user USER[:PASSWD] Masquerade as user USER with optional colon-separated password PASSWD.

EXAMPLES

RETURN VALUES

Upon successful completion, scyld-tool-config returns 0. On failure, an error message is printed to stderr and scyld-tool-config returns 1.

7.12 Scyld ClusterWare Maintenance Tools

This section of the Reference Guide describes the Scyld ClusterWare low-level tools. These tools can be used by the cluster administrator, and are not intended for use by the ordinary user.
7.12.1 headctl

NAME
headctl -- Manage head node network communication settings.

USAGE
[--use-etcd]

DESCRIPTION
This is a low-level tool that directly manipulates configuration settings for head node network communication. When controlling HTTP/HTTPS settings, it modifies /opt/scyld/clusterware/conf/base.ini and two Apache configuration files in /etc/http/conf.d/: ssl.conf and clusterware.conf. When enabling XSendfile support, the tool may install necessary RPMs as well as update variables in the base.ini.

Since the earliest boot steps cannot use encrypted communications, DHCP and PXE booting are not affected by these settings. Communications starting with initramfs execution will use HTTP or HTTPS as instructed by this command.

The tool resides in /opt/scyld/clusterware/bin/headctl and must be executed by user root.

OPTIONAL ARGUMENTS
- -h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- --status Report the configuration of the ClusterWare service.
- --prefer-http Instruct compute nodes to use HTTP.
- --prefer-https Instruct compute nodes to use HTTPS where possible.
- --enable-https Proxy through /etc/http/conf.d/ssl.conf.
- --disable-https Do not proxy through /etc/http/conf.d/ssl.conf.
- --minimal-ipv6 Confirm the radvd is running on our interfaces by optionally installing radvd, adding blocks to the radvd.conf, and restarting the service if necessary.
- --enable-xsendfile Use the Apache XSendfile header when clients download files.
- --disable-xsendfile Do not use the Apache XSendfile header.
- --use-couchbase Switch to Couchbase as the backend database.
- --use-etcd Switch to etcd as the backend database.

EXAMPLES

RETURN VALUES
Upon successful completion, headctl returns 0. On failure, an error message is printed to stderr and headctl returns 1.
7.12.2 make-iso

NAME

make-iso -- Create an ISO file from a yum repo.

USAGE

make-iso [-h] <RPM-SOURCE> [ISOFILE]

DESCRIPTION

This is a low-level tool that creates an ISO file, optionally named ISOFILE, from a yum repo file or from collection of RPMs from RPM-SOURCE.

The tool resides in /opt/scyld/clusterware-installer/make-iso.

<RPM-SOURCE> OPTIONS

--from-yum Mirror RPMs from the baseurl(s) in /etc/yum.repos.d/clusterware.repo.
--rpm-dir DIR Copy the RPMs from the directory DIR.
--yum-repo REPOFILE Parse a specific repo file REPOFILE for RPM sources.

OPTIONAL ARGUMENTS

-h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

EXAMPLES

(Note that make-iso resides in /opt/scyld/clusterware/installer/)

make-iso --yum-repo /tmp/clusterware.repo

Use the RPMs identified by the yum repo file /tmp/clusterware.repo to create an ISO named clusterware.iso.

make-iso --from-yum

Equivalent to make-iso --yum-repo /etc/yum.repos.d/clusterware.repo.

make-iso --yum-repo /tmp/clusterware.repo cw11.1.iso

Use the RPMs identified by the yum repo file /tmp/clusterware.repo to create an ISO named cw11.1.iso.

make-iso --rpm-dir /mnt/clusterware/11.0/el7 cw11.0.iso

Use the RPMs found in /mnt/clusterware/11.0/el7/ to create an ISO named cw11.0.iso.

RETURN VALUES

Upon successful completion, make-iso returns 0. On failure, an error message is printed to stderr and make-iso returns 1.
7.12.3 managedb

NAME
managedb -- Directly manipulate the database.

USAGE

DESCRIPTION
This is a low-level tool that directly manipulates the database, generally only executed by other scyld-* tools.
The tool resides in /opt/scyld/clusterware/bin/managedb and must be executed by user root.

ACTIONS
join IP  Join this head node (referenced by IP address) to an existing cluster.
   --purge     Entirely delete the exiting database(s).
leave     Remove this head node from the cluster.
eject IP  Remove the specified head node IP address from the cluster.
clear     Reset the data back to a fresh, empty state.
   --reinit     Reinitialize the database server.
   --purge     Entirely delete the exiting database(s).
update     Update the internal database format.
recover     Attempt to recover the local database.
maintain    Perform a maintenance tasks.
   --compact     Force a compaction, regardless of current size.
   --defrag      Shrink the database after compacting.
save ARCHIVE  Save the database to an ARCHIVE file or directory.
   --without-boots     Exclude boot files.
   --without-images     Exclude root file system images.
   --without-isos     Exclude ISO images uploaded for kickstarting.
   --without-gits     Exclude git archives.
   --without-all     Exclude all files, i.e., only export the database.
   --format FMT     Select a file format: zip (default), dir, tar
load ARCHIVE  Load the database from an ARCHIVE file or directory.
   --without-boots     Exclude boot files.
   --without-images     Exclude root file system images.
   --without-isos     Exclude ISO images uploaded for kickstarting.
   --without-gits     Exclude git archives.
   --without-all     Exclude all files, i.e., only export the database.
merge ARCHIVE  Merge the contents of an ARCHIVE file or directory into the database.

- --without-boots  Exclude boot files.
- --without-images  Exclude root file system images.
- --without-isos  Exclude ISO images uploaded for kickstarting.
- --without-all  Exclude all files, i.e., only export the database.

OPTIONAL ARGUMENTS

- -h, --help  Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- -v, --verbose  Increase verbosity.
- -q, --quiet  Decrease verbosity.
- -c, --config CONFIG  Specify a client configuration file CONFIG.
- --print-options  Print all backend options, then exit.
- --as-ini  Use ini format when printing options.

EXAMPLES

(Reminder: managedb resides in /opt/scyld/clusterware/installer/managedb)

sudo managedb leave

  Detach the current head node from the cluster.

sudo managedb eject 10.54.0.2

  Eject head node at IP address 10.54.0.2 from the cluster.

RETURN VALUES

Upon successful completion, managedb returns 0. On failure, an error message is printed to stderr and managedb returns 1.

7.12.4 take-snapshot

NAME

take-snapshot  -- Perform a database backup

USAGE

take-snapshot

DESCRIPTION

This is a low-level tool that performs a database backup, typically executed periodically by cron. The tool uses optional "backups" configuration settings found in /opt/scyld/clusterware/conf/base.ini.

The tool resides in /opt/scyld/clusterware/bin/take-snapshot and must be executed by user root.

The base.ini optional settings and examples:

backups.user = CWADMIN  This setting optionally specifies the user name CWADMIN of a ClusterWare administrator. If unspecified, then the default is user root, although in that case root must be previously declared (e.g., via scyld-admintctl create name=root) as a ClusterWare administrator.
backups.path = ~CWADMIN/.scyldcw/database-backups  This setting optionally specifies the path to the directory into which the backups and associated files reside. If unspecified, then the default is ~CWADMIN/.scyldcw/database-backups for the CWADMIN user in effect, whether explicitly specified or whether using the default root. For example, if backups.user is unspecified, then CWADMIN defaults to root and the default backups.path defaults to -root/.scyldcw/database-backups. The take-snapshot tool creates the directory with owner CWADMIN.

Within the backups directory there is a subdirectory files that contains the various raw content, kernel, and initramfs files from the database, a database-backups.log logfile, and one or more snap-<timestamp> directories of database snapshots, each created by an execution of the take-snapshot tool. Within each of these snapshot directories is a managedb-generated zipfile of files other than the various raw image files in the files subdirectory, and symlinks with "pretty" names such as "DefaultBoot.kernel" and "DefaultImage.content" that point to specific raw files in the files subdirectory.

backups.retention = 1h/24h,1d/7d,1w/4w,4w/1040w  This setting optionally specifies the four retention tiers, which are comma-separated block and span time values separated by a '/'. A time value is a nonzero positive integer with a single letter suffix of h for hours, d for days, or w for weeks.

The above values are the default values for the tiers and specify:

<table>
<thead>
<tr>
<th>Tier</th>
<th>Description</th>
<th>Retention Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tier1</td>
<td>For the most recent 24 hours (&quot;24h&quot;), retain a max of one snapshot per hour (&quot;1h&quot;).</td>
<td>1h/24h</td>
</tr>
<tr>
<td>Tier2</td>
<td>Then for the previous 7 days (&quot;7d&quot;) prior to that Tier1 24 hour span, retain a max of one snapshot per day (&quot;1d&quot;).</td>
<td>1d/7d</td>
</tr>
<tr>
<td>Tier3</td>
<td>Then for the previous 4 weeks (&quot;4w&quot;) prior to that Tier2 7 day span, retain a max of one snapshot per week (&quot;1w&quot;).</td>
<td>1w/4w</td>
</tr>
<tr>
<td>Tier4</td>
<td>Then for the previous 1040 weeks (&quot;1040w&quot;, or about 20 years), prior to that Tier3 4 week span, retain a max of one snapshot per 4 weeks (&quot;4w&quot;).</td>
<td>4w/1040w</td>
</tr>
</tbody>
</table>

Any snapshots older than the Tier4 "span" are simply discarded.

backups.clean = 14d  This setting optionally specifies an interval between scans of the snap-<timestamp> directories to determine which of the raw files in the files subdirectory, if any, are no longer referenced by any snap-<timestamp>. If unspecified, then the default is once every 14 days.

EXEMPLARY
(Note that take-snapshot resides in /opt/scyld/clusterware/installer/)

```bash
sudo take-snapshot
```

Manually perform a single database backup.

```bash
sudo cat /var/spool/cron/root
```

A sample crontab to execute the tool once an hour at five minutes past the hour:

```bash
SHELL=/bin/bash
PATH=/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin
MAILTO=root@localhost
05 * * * * /opt/scyld/clusterware/bin/take-snapshot
```

RETURN VALUES

Upon successful completion, take-snapshot returns 0. On failure, an error message is printed to stderr and take-snapshot returns 1.
When scyld-install (and its underlying use of the yum command) do not have access to repositories that are accessible via the Internet, then repositories must be set up on local storage.

First ensure that the appropriate base distribution repositories (i.e., Red Hat RHEL or CentOS) are also accessible locally without requiring Internet access. An initial install of Scyld ClusterWare has dependencies on various base distribution packages, and a subsequent ClusterWare update may have dependencies on new or updated base distribution packages.

Next you need a ClusterWare ISO file that contains the desired software. Either contact Penguin Computing to obtain the ISO, or build the ISO on a local server that has access to the Internet. To build the ISO locally, you need a clusterware.repo file that contains a valid customer authentication token that allows access to Penguin Computing’s ClusterWare yum repo, then:

```
# Download the ClusterWare `make-iso` script:
curl -O https://updates.penguincomputing.com/clusterware/11/installer/make-iso

# Execute the `make-iso` script to create either an ISO named "clusterware.iso":
./make-iso --yum-repo ./clusterware.repo
# Or to create an arbitrarily named ISO:
sudo ./make-iso --yum-repo ./clusterware.repo clusterware-11.1.0.iso

# Note: `./make-iso --from-yum` is equivalent to
# `./make-iso --yum-repo /etc/yum.repos.d/clusterware.repo`
```

Suppose the ISO file clusterware-11.1.0.iso contains ClusterWare release 11.1.0:

```
# Mount the ClusterWare ISO, if not already mounted:
sudo mount -o loop clusterware-11.1.0.iso /mnt/cw11.1.0

For an initial install, use a cluster configuration file (e.g., named cluster-conf) that is described in Initial Installation of Scyld ClusterWare, and execute the scyld-install script that is embedded in the ISO to performs the basic first install of ClusterWare and create /etc/yum.repos.d/clusterware.repo, which points at the software in the ISO:

/mnt/cw11.1.0/scyld-install --config cluster-conf
```

Once the head node software has been installed, then subsequent ClusterWare commands need to find a base distribution defined repo and distro. See Appendix: Creating Arbitrary CentOS Images (or Appendix: Creating Arbitrary RHEL Images) for examples.
Suppose the base distribution ISO is accessible at http://<baseOSserver>/<baseOSiso>:

```shell
scyld-clusterctl repos create name=<baseOSrepo> iso=@/path/to/baseOSiso
scyld-clusterctl distros create name=<baseOSdistro> repos=<baseOSrepo>
```

Now finish the setup. The following expects to find a single `distro` and one or more `repo` repositories:

```shell
scyld-add-boot-config --make-defaults
```

**For a software update of an existing Scyld ClusterWare install**, rename the current `/etc/yum.repos.d/clusterware.repo`, then execute the script (which recreates `clusterware.repo` with the appropriate values):

```shell
(cd /etc/yum.repos.d; sudo mv -f clusterware.repo clusterware.repo.bak)
/mnt/cw11.1.0/scyld-install
```

**Important:** If the local repo has been created in a manner other than what is described above, then it is possible that `/etc/yum.repos.d/clusterware.repo` uses `baseurl` of the form `file:///` (e.g., `baseurl=file:///var/www/html/cw11.1.0`). This may cause future problems when attempting to create an image, so the administrator should edit this to a functionally equivalent form `http://` (e.g., `baseurl=http://localhost/cw11.1.0`).

## 8.2 Appendix: Creating Arbitrary CentOS Images

*Creating PXEboot Images* in the *Installation & Administrator Guide* describes how to create PXEboot images from the latest CentOS 7 and 6 repos. This section describes how to create images from CentOS 7 and 6 repos that are not the latest packages.

For example, using the CentOS-7-x86_64-DVD-1908.iso ISO file, you can use `scyld-clusterctl repos` and `scyld-clusterctl distros` to create a repo and distro for this CentOS version 7.7, then use `scyld-modimg` to create an image and a boot config.

Or more simply you can use `scyld-add-boot-config` to perform the same result in fewer steps. Execute the following and accept all the defaults:

```shell
scyld-add-boot-config --iso /mnt/isos/CentOS-7-x86_64-DVD-1908.iso
```

This creates a distro and repo both named `CentOS-7-x86_64-1908`, and an image and boot config both named `CentOS-7-x86_64-1908`.

Alternatively, you can avoid the manual acceptance of the defaults by specifying desired names and running the command in batch mode:

```shell
scyld-add-boot-config --iso /mnt/isos/CentOS-7-x86_64-DVD-1908.iso \
    --image CentOS-7.7-Image --boot-config CentOS-7.7-Boot --batch
```

View the result, which shows the default repo and the new repo, and the default boot config and the new boot config:

```
[admin@head]$ scyld-clusterctl distros ls -l
Distros
    CentOS
    name: CentOS
```

(continues on next page)
package_manager: yum
release: 7
repos
CentOS_base
CentOS-7-x86_64-1908
name: CentOS-7-x86_64-1908
package_manager: yum
release: none
repos
CentOS-7-x86_64-1908

[admin@head]$ scyld-bootctl ls -l
Boot Configurations
CentOS-7.7-Boot
cmdline: enforcing=0
image: CentOS-7.7-Image
initramfs
chksum: a85b01e91c26c52ebf549066c6c5fce544f3c75b
filename: 3684fbadf53f4c8bb8a3dea24ecf778d
mtime: 2022-03-18 16:49:06 UTC (1:14:23 ago)
size: 33.4 MiB (34978448 bytes)
kerneld
chksum: 73872862a49ee024bf44c4d796c96bed4d52ee43
filename: 1d6888add971485395d643df916d45c4
mtime: 2022-03-18 16:49:07 UTC (1:14:23 ago)
size: 6.4 MiB (6734016 bytes)
last_modified: 2022-03-18 16:49:07 UTC (1:14:23 ago)
name: CentOS-7.7-Boot
release: 3.10.0-1062.el7.x86_64

Note: Creating images from some older ISOs may produce an error message beginning with **ERROR: One or more repositories in the newly created image are invalid or unreachable.** The **scyld-modimg** tool will automatically retry the image creation, and if there is no subsequent error reported, then the administrator can assume that the resulting image is useable.

Important: If the CentOS image thus built is subsequently updated using **yum update**, then by default that updates packages to the latest minor release level, **not** to newer versions at the image's current minor release level. Also, **yum install** of additional packages may update dependency packages from their current minor release version level to the latest minor level. Such actions may result in a mixture of packages from different minor releases, which may have unintended consequences.
8.3 Appendix: Creating Arbitrary RHEL Images

The Appendix: Creating Arbitrary CentOS Images describes how to create and update PXEboot images using arbitrary CentOS repos. This appendix describes how to create arbitrary PXEboot RHEL images and register (or re-register) them to Red Hat. The repo is most commonly built from an ISO file that represents a specific RHEL major.minor version.

For this example we build a RHEL 7.8 image and boot config using the `rhel-computenode-7.8-x86_64-dvd.iso` ISO file.

You can use `scyld-clusterctl repos` and `scyld-clusterctl distros` to create a repo and distro for this RHEL version 7.8, then use `scyld-modimg` to create an image and a boot config.

Or more simply you can use `scyld-add-boot-config` to perform the same result in fewer steps. Execute the following and accept all the defaults:

```
scyld-add-boot-config --iso /mnt/isos/rhel-computenode-7.8-x86_64-dvd.iso
```

This creates a distro and repo both named `rhel-server-7.8-x86_64`, and an image and boot config both named `rhel-server-7.8-x86_64`.

Alternatively, you can avoid the manual acceptance of the defaults by specifying desired names and running the command in batch mode:

```
scyld-add-boot-config --iso=/mnt/isos/rhel-server-7.8-x86_64-dvd.iso \
   --image RHEL-7.8-Image --boot-config RHEL-7.8-Boot --batch
```

View the result, which shows the default repo and the new repo, and the default boot config and the new boot config:

```
[admin@head]$ scyld-clusterctl distros ls -l
Distros
   CentOS
      name: CentOS
      packaging: rpm
      release: 7
      repos
         CentOS_base
   rhel-server-7.8-x86_64
      name: rhel-server-7.8-x86_64
      packaging: rpm
      release: none
      repos
         rhel-server-7.8-x86_64

[admin@head]$ scyld-bootctl ls -l
Boot Configurations
   DefaultBoot
      cmdline: enforcing=0
      image: DefaultImage
      initramfs
         chksum: a623be752272166f47896d648689789359239ebf
         filename: b51e6d31a84a4f069c6a4a484b5b5264
         mtime: 2022-03-18 19:20:19 UTC (0:37:22 ago)
         size: 33.4 MiB (35046202 bytes)
      kernel
```

Important: If the cluster administrator wants to enable FIPS, then follow the directions provided by the base distribution provider. The Red Hat RHEL or CentOS repo must include @core, and any subsequently created compute node image must contain several additional packages, including dracut-fips. Verify the presence of @core by successfully executing `yum groupinfo core`.

To boot the new image, assign RHEL-7.8-Boot to node n0, and reboot n0:

```
[admin@head]$ scyld-nodectl -i n0 set _boot_config=RHEL-7.8-Boot
Results
 n0
  success: True

[admin@head]$ scyld-nodectl -i n0 reboot
Nodes
 n0: Soft reboot succeeded
```

A RHEL compute node can automatically register (or re-register) with Red Hat at boot time by adding the file `/etc/clusterware/rhel-vars.sh` to the image. That file must contain two lines that define values for the variables "RHEL_USER" and "RHEL_PASS". The booting RHEL node executes `/opt/scyld/clusterware-node/scripts-available/up/register_rhel.sh` (distributed in the `clusterware-node` package) which opens `/etc/clusterware/rhel-vars.sh` (if that exists) and parses the "RHEL_USER=" username and "RHEL_PASS=" password, then executes:

```
subscription-manager register --username <username> --password <password>
```

On a successful first-time registration, the node transmits the resulting `consumerid` to its parent head node, which in turn stores that value into the node's `_rhel_consumerid` attribute in the ClusterWare database.
If a specific Pool ID is required, then add the attribute `_rhel_poolid`.

**Important:** If the RHEL image thus built is subsequently updated using `yum update`, then by default that updates packages to the latest minor release level, **not** to newer versions at the image's current minor release level. Also, `yum install` of additional packages may update dependency packages from their current minor release version level to the latest minor level. Such actions may result in a mixture of packages from different minor releases, which may have unintended consequences.

### 8.4 Appendix: Creating Ubuntu and Debian Images

The following examples create Ubuntu and Debian images and associated boot configurations using the public Internet-accessible Ubuntu and Debian repos, both of which contain multiple releases.

**UBUNTU**

First an example of building an Ubuntu 20.04 LTS (Focal Fossa) image. Specify a local repo, arbitrarily naming it `ubuntu`, that serves as a shorthand reference to the public Ubuntu repo:

```
scyld-clusterctl repos create name=ubuntu urls=http://archive.ubuntu.com/ubuntu/
```

Next, specify a particular distribution within that Ubuntu repo. For this example we specify `focal`, which is the Focal Fossa 20.04 LTS release, and give this local distro the name `ubuntu_20.04`:

```
scyld-clusterctl distros create name=ubuntu_20.04 repos=ubuntu release=focal packaging=deb
```

Now create an image from the distro `ubuntu_20.04` and name it `UbuntuImg-20.04`:

```
scyld-modimg --create ubuntu_20.04 --set-name UbuntuImg-20.04 --upload
```

And create a boot configuration named `UbuntuBoot-20.04` that pxeboots that image:

```
scyld-add-boot-config --image UbuntuImg-20.04 --boot-config UbuntuBoot-20.04 --batch
```

The image thus created contains basic Ubuntu 20.04 LTS software. You can add software and modify configuration files in this image as needed, keeping in mind that Ubuntu's package manager expects software distributed as `*.deb` files, not `*.rpm` files.

For example, use `--chroot`:

```
scyld-modimg -i UbuntuImg-20.04 --chroot
```

and manipulate files and packages within the image.

**DEBIAN**

You can employ similar steps to create a Debian image, in this example creating an image from the `stable` distro:

```
scyld-clusterctl repos create name=debian urls=http://deb.debian.org/debian/
scyld-clusterctl distros create name=debian-stable repos=debian release=stable packaging=deb
scyld-modimg --create debian-stable --set-name debian-stable-img --upload
scyld-add-boot-config --image debian-stable-img --boot-config debian-stable-boot --batch
```

### 8.4. Appendix: Creating Ubuntu and Debian Images

163
8.5 Appendix: Converting CentOS 8 to Alternative Distro

CentOS 8 has reached its official End Of Life phase and now exists only in archived form at https://vault.centos.org/. To access software updates that track RHEL8 to one degree or another, you should convert to an alternative distribution.

Some alternatives choices are:

**Red Hat RHEL 8**

RHEL is the original source base of every CentOS release, so RHEL 8 is an obvious alternative to CentOS 8. Accessing the RHEL 8 repositories requires a paid subscription. Additionally, some RPMs found in the CentOS repository for a particular release are only found in the Red Hat EPEL repository. Contact Red Hat for details.

**CentOS Stream 8**

The CentOS Project recommends transitioning CentOS 8 to CentOS Stream 8. The CentOS Project defines that repository as containing RPMs that are in a development phase between a RHEL clone and the somewhat more experimental Fedora, i.e., as a form of “beta” release candidates for the next RHEL release.

The CentOS Project describes this transition at https://www.centos.org/news-and-events/convert-to-stream-8/ which today consists of two commands:

```
dnf --disablerepo '*' --enablerepo extras swap centos-linux-repos centos-stream-repos
dnf distro-sync
```

**Rocky 8**

Penguin Computing currently suggests considering Rocky 8 as a distribution similar to CentOS, i.e., tracking every new RHEL 8 release within days. See https://rockylinux.org/ for details.

See https://github.com/rocky-linux/rocky-tools/tree/main/migrate2rocky for details about a bash script that performs the conversion.

Note however that updating CentOS RPMs to Rocky RPMs will likely install updates of various configuration files, which will leave various *.rpmsave and *.rpmnew files that require the administrator to examine and potentially merge local changes that were made when running CentOS.

8.6 Appendix: Using Ansible

A compute node can be configured to execute an Ansible playbook at boot time or after the node is up. In the following example, the cluster administrator creates a git repository hosted by the ClusterWare head nodes, adds an extremely simple Ansible playbook to that git repository, and assigns a compute node to execute that playbook.

Install the `clusterware-ansible` package into the image (or images) that you want to support execution of an Ansible playbook:

```
scyld-modimg -i DefaultImage --install clusterware-ansible --upload --overwrite
```

The administrator should amend their PATH variable to include the git binaries that are provided as part of the `clusterware` package in `/opt/scyld/clusterware/git/`. This is not strictly necessary, but the git in that subdirectory is version 2.33.0 and is significantly more recent than the version normally provided by an el7 base distribution:

```
export PATH=/opt/scyld/clusterware/git/bin:${PATH}
```

The administrator should add their own personal public key to their ClusterWare admin account. This key will be populated into user root's (or _remote_user's) authorized_keys file for a newly booted compute node. See `Compute Node Remote Access` for details. In addition, this provides simple SSH access to the git repository:
Adding the localhost's host keys to a personal known_hosts file is not strictly necessary, but it will avoid an SSH warning that can interrupt scripting:

```
ssh-keyscan localhost >> ~/.ssh/known_hosts
```

Now create a ClusterWare git repository called "ansible". This repository will default to `public`, meaning it is accessible read-only via unauthenticated HTTP access to the head nodes and therefore should not include unprotected sensitive passwords or keys:

```
scyld-clusterctl gitrepos create name=ansible
```

Note that being unauthenticated means the HTTP access mechanism does not allow for `git push` or other write operations. Alternatively the repository can be marked `private` (`public=False`), although it then cannot be used for a client's `ansible-pull`.

Initially the repository will include a placeholder text file that can be deleted or discarded.

Now clone the git repo over an SSH connection to localhost:

```
git clone cwgit@localhost:ansible
```

The administrator could also create that clone on any machine that has the appropriate private key and can reach the SSH port of a head node.

Finally, create a simple Ansible playbook to demonstrate the functionality:

```
cat >ansible/HelloWorld.yaml <<EOF
---
- name: This is a hello-world example
  hosts: n*.cluster.local
  tasks:
    - name: Create a file called '/tmp/testfile.txt' with the content
      copy:
        content: hello world
        dest: /tmp/testfile.txt
EOF
```

and add that playbook to the "ansible" git repo:

```
bash -c "\n  cd ansible; \n  git add HelloWorld.yaml; \n  git -c user.name=Test -c user.email='<test@test.test>' commit --message 'Adding a test playbook' HelloWorld.yaml; \n  git push; \n"
```

Multiple playbooks can co-exist in the git repo.

In a multiple-head node cluster an updated git repository will be replicated to other head nodes in the cluster, so any client `ansible_pull` to any cluster head node will see the same playbook and the same commit history. This replication requires several seconds to complete.

With the playbook now available in the git repo, configure the compute node to execute `ansible-pull` to download it at boot time:

```
8.6. Appendix: Using Ansible
```
scyld-nodectl -i n1 set _ansible_pull=git:ansible/HelloWorld.yaml

Alternatively, to download the playbook from an external git repository on the server named gitserver:

scyld-nodectl -i n1 set _ansible_pull=http://gitserver//HelloWorld.yaml

Either format can optionally end with "@<gitrev>", where <gitrev> is a specific commit, tag, or branch in the target git repo.

You may now reboot the node and wait for it to boot to an up status after the playbook has executed:

scyld-nodectl -i n1 reboot
scyld-nodectl -i n1 waitfor up

You can verify that the HelloWorld.yaml playbook executed:

scyld-nodectl -i n1 exec cat /tmp/testfile.txt ; echo

Note that during playbook execution the node remains in the booting status, changing to an up status after the playbook completes, assuming the playbook is not fatal to the node. That status may timeout to down (with no ill effect) when executing a lengthy playbook before switching to up after playbook completion. Administrators are advised to log the ansible progress to a known location on the booting node, such as /var/log/ansible.log.

The clusterware-ansible package supports another attribute, _ansible_pull_now, which uses the same syntax as _ansible_pull. Prior to first use, this attribute requires the administrator to:

systemctl enable cw-ansible-pull-now
systemctl start cw-ansible-pull-now

When the attribute is present, the node will download and execute the playbook during the node's next status update event, which occur every 10 seconds by default. Once the node completes execution of the playbook, it directs the head node to prepend "done" to the _ansible_pull_now attribute to ensure the script does not run again.

8.7 Appendix: Using Kubernetes

See Kubernetes for a brief explanation of how to install and initialize a Kubernetes cluster. This appendix provides more detailed examples.

EXAMPLE 1: Create a minimum viable Kubernetes cluster with one control plane ClusterWare node and some worker nodes.

The control plane node in a production environment should be a full-install node to provide persistence across reboots.

Step 1: Initialize the control plane node on a ClusterWare node, specifically using Kubernetes version 1.22.0 instead of defaulting to the latest version.

For this example use n0 as the control plane node:

[admin@head]$ scyld-kube -i n0 --init --version 1.22.0

Upon successful initialization, you can check Kubernetes node and pod status on the control plane:

[admin@head]$ scyld-nodectl -i n0 exec kubectl get node
NAME     STATUS    ROLES                     AGE      VERSION
n0.cluster.local  Ready     control-plane, master 2m15s  v1.22.0

(continues on next page)
Step 2: Join worker node(s).

Join nodes n[3-4] to control plane node n0:

```
[admin@head]$ scyld-kube -i n[3-4] --join --version 1.22.0
```

If there are multiple Kubernetes clusters defined in the ClusterWare cluster, then you need to specify the control plane node:

```
[admin@head]$ scyld-kube -i n[3-4] --join --version 1.22.0 --cluster n0
```

Check Kubernetes node status to see nodes n[3-4] are joined as workers:

```
[admin@head]$ scyld-nodectl -i n0 exec kubectl get node
```

Suppose nodes n[3-4] are full-install nodes. They maintain their role as Kubernetes workers after a reboot. If the nodes are instead non-persistent, i.e., they PXEboot, then their Kubernetes worker role disappears after a reboot.

Suppose nodes n[5-7] PXEboot with an image `kubeimg`. Make the worker state persistent across reboots of every node that uses `kubeimg` by executing the `--join` with an additional argument `--image` specifying the image:

```
[admin@head]$ scyld-kube --image kubeimg --join --version 1.22.0
```

Now you can reboot nodes n[5-7] and observe nodes n[3-7] are all joined as Kubernetes workers:

```
[admin@head]$ scyld-nodectl -i n0 exec kubectl get node
```

As with the earlier `--join` example, if there are multiple Kubernetes clusters defined in the ClusterWare cluster, then execute the join with an additional argument identifying the specific control plane node:
EXAMPLE 2: Create a High-Availability (HA) Kubernetes cluster with 3 control planes and some worker nodes.

The control plane nodes in a production High-Availability environment should all be full-install nodes to provide persistence across reboots.

Step 1: Prepare the load balancer files.

For this example use the latest Kubernetes version (e.g., 1.22.4), use node n0 for the first control plane node and master load balancer, and use nodes n1 and n2 as other control planes and backup load balancers:

```
[admin@head]$ scyld-kube --prepare-lb 10.54.2.1[:4200:51:42] \
    n0:10.54.150.100,n1:10.54.150.101,n2:10.54.150.102
```

In the above configuration, 10.54.2.1 is an unused virtual IP address negotiated between n0, n1 and n2 within the network subnet. [:4200:51:42] are optional default values for [:API-\SERVER_PORT:ROUTER_ID:AUTH_PASS] and are needed only if you want to use non-default values.

Step 2: Initialize the Kubernetes cluster on the first control plane.

Initialize node n0 with an HA variation of the --init argument:

```
[admin@head]$ scyld-kube -i n0 --init-ha
```

Upon successful initialization the stdout should contain a proposed kubeadm join command, such as:

```
[admin@head]$ kubeadm join 10.54.2.1:6443 --token g9h7gm.qmg18h8evp7g0701 \ 
    --discovery-token-ca-cert-hash sha256:8a536515c02bb7f099f38be604c94a90b54d1ccce8422e8219c2680e379c9e14 \ 
    --control-plane --certificate-key b97c9c1ca0635ffef5a531b5ff41eaa55e0b379242ac85ef8028c0a184c190
```

and near the end of the stdout you will see:

```
A non-expiring Scyld ClusterWare token: sv1tb2.qyfuu8ehrbxk3tzu is generated.
```

The first "--token TOKEN" in the proposed kubeadm join output expires after 24 hours, so your upcoming scyld-kube --join-ha should instead use the non-expiring ClusterWare token.

The KEY expires in two hours. If needed, you can generate a new key:

```
[admin@head]$ scyld-nodectl -i n0 exec kubeadm init phase upload-certs \ 
    --upload-certs
```

Check the Kubernetes node and pod on n0:

```
[admin@head]$ scyld-nodectl -i n0 exec kubectl get node
NAME       STATUS     ROLES                  AGE       VERSION
n0.cluster.local Ready control-plane,master 2m10s   v1.22.4

[admin@head]$ scyld-nodectl -i n0 exec kubectl get pod -n kube-system
NAME                             READY STATUS   RESTARTS AGE
coredns-78fc69978-kkc8x          1/1    Running   0  2m19s
coredns-78fc69978-rlk4d          1/1    Running   0  2m19s
etcd-n0.cluster.local           1/1    Running   0  2m13s
haproxy-n0.cluster.local        1/1    Running   0  2m13s
```

(continues on next page)
keepalived-n0.cluster.local 1/1 Running 0 2m13s
kube-apiserver-n0.cluster.local 1/1 Running 0 2m13s
kube-controller-manager-n0.cluster.local 1/1 Running 0 2m13s
kube-flannel-ds-f97k5 1/1 Running 0 2m18s
kube-proxy-4mzrl 1/1 Running 0 2m18s
kube-scheduler-n0.cluster.local 1/1 Running 0 2m13s

Notice that the haproxy and keepalived pods are running. They are not running in Example 1.

Step 3: Join the other control plane nodes to the first control plane.

Join control plane nodes n[1-2] to the first control plane n0 using the "certificate-key" discussed above, e.g.:

```
[admin@head]$ scyld-kube -i n[1-2] --join-ha --certificate-key KEY
```

As before, if there are multiple Kubernetes clusters in ClusterWare cluster, then execute the --join-ha command with the additional --cluster n0 argument to identify the first control plane.

Now check the status:

```
[admin@head]$ scyld-nodectl -i n0 exec kubectl get node
NAME      STATUS    ROLES          AGE   VERSION
n0.cluster.local Ready control-plane,master 23m v1.22.4
n1.cluster.local Ready control-plane,master 14m v1.22.4
n2.cluster.local Ready control-plane,master 14m v1.22.4
```

```
[admin@head]$ scyld-nodectl -i n0 exec kubectl get pod -n kube-system
NAME                 READY STATUS    RESTARTS AGE
coredns-78fcd69978-kkc8x 1/1  Running 0 22m
coredns-78fcd69978-rlk4d 1/1  Running 0 22m
etcd-n0.cluster.local  1/1  Running 0 22m
etcd-n1.cluster.local  1/1  Running 0 22m
etcd-n2.cluster.local  1/1  Running 0 22m
haproxy-n0.cluster.local 1/1  Running 0 22m
haproxy-n1.cluster.local 1/1  Running 0 22m
haproxy-n2.cluster.local 1/1  Running 0 22m
keepalived-n0.cluster.local 1/1  Running 0 22m
keepalived-n1.cluster.local 1/1  Running 0 22m
keepalived-n2.cluster.local 1/1  Running 0 22m
kube-apiserver-n0.cluster.local 1/1  Running 0 22m
kube-apiserver-n1.cluster.local 1/1  Running 0 22m
kube-apiserver-n2.cluster.local 1/1  Running 0 22m
kube-controller-manager-n0.cluster.local 1/1  Running 1 (13m ago) 22m
kube-controller-manager-n1.cluster.local 1/1  Running 0 22m
kube-controller-manager-n2.cluster.local 1/1  Running 0 22m
kube-flannel-ds-262pd 1/1  Running 0 22m
kube-flannel-ds-b5scg 1/1  Running 0 22m
kube-flannel-ds-f97k5 1/1  Running 0 22m
kube-proxy-2swbv 1/1  Running 0 22m
kube-proxy-4mzrl 1/1  Running 0 22m
kube-proxy-ktlc9 1/1  Running 0 22m
kube-scheduler-n0.cluster.local 1/1  Running 1 (13m ago) 22m
kube-scheduler-n1.cluster.local 1/1  Running 0 22m
kube-scheduler-n2.cluster.local 1/1  Running 0 22m
```
Step 4: Join worker node(s)

Join the worker node(s) or modify the workers’ node image to the first control plane node using the same commands as in Step 2 in Example 1.

EXAMPLE 3: Create a High Availability Kubernetes cluster with 3 non-ClusterWare control plane servers and some ClusterWare worker nodes.

For this example use servers named `kube-1` as the first control plane node and master load balancer, and servers `kube-2` and `kube-3` as other control planes and backup load balancers. These servers must be running a full RHEL/CentOS distribution, must be connected to the ClusterWare private cluster network, and must know the names of the other servers and the prospective ClusterWare nodes that will be used as Kubernetes workers.

Step 1: Initialize the Kubernetes HA cluster on the first control plane.

Install the `clusterware-kubeadm` RPM on the first control plane, either using `yum install clusterware-kubeadm` from the ClusterWare repo, or downloading the RPM from the repo and installing it manually with `rpm -i`, then prepare the load balancer files:

```
[root@kube-1]$ scyld-kube --prepare-lb 10.54.2.1[:4200:51:42] \
kube-1:10.54.150.200,kube-2:10.54.150.201,kube-3:10.54.150.202
```

In the above configuration, 10.54.2.1 is an unused virtual IP address negotiated between `kube-1`, `kube-2`, and `kube-3` within the network subnet. `[:4200:51:42]` are optional default values for `[:API-SERVER_PORT:ROUTER_ID:AUTH_PASS]` and are needed only if you want to use non-default values.

Now initialize the Kubernetes cluster on the first control plane:

```
[root@kube-1]$ scyld-kube --init-ha
```

Upon successful initialization the stdout should contain a proposed `kubeadm join` command, such as:

```
kubeadm join 10.54.2.1:6443 --token g9h7gm.qmg18h8evp7g0701 \
    --discovery-token-ca-cert-hash \
    sha256:8a536515c02bb7fe999f3be604c94a90b54d1ccce8422e8219c2680e379c9e14 \
    --control-plane --certificate-key \
    b97c9c1ca0635ffef5a531b5fff41eaa55e0b379242ac85ef8028c0a184c190
```

and near the end of the stdout you will see:

```
A non-expiring Scyld ClusterWare token: sv1tb2.qyfuu8ehrbxk3tzu is generated.
```

The first "--token TOKEN" in the proposed `kubeadm join` output expires after 24 hours, so your upcoming `scyld-kube --join-ha` should instead use the non-expiring ClusterWare token.

The `KEY` expires in two hours. If needed, you can generate a new key with:

```
[root@kube-1]$ kubeadm init phase upload-certs --upload-certs
```

You will need these values on the other control planes `kube-2` and `kube-3` and the worker node(s) in order to perform their joins to `kube-1`.

Step 2: Initialize and join the other control plane nodes.
For each of the other control plane nodes, prepare the load balancer files in the same manner as was done with the first control plane. Install the `clusterware-kubeadm` package on the server, then prepare the load balancer files and join the server to the first. For example, for `kube-2`:

```
[root@kube-2] $ scyld-kube --prepare-lb 10.54.2.1[:4200:51:42] \ 
kube-1:10.54.150.200,kube-2:10.54.150.201,kube-3:10.54.150.202
[root@kube-2] $ scyld-kube --join-ha --cluster 10.54.150.200 --token TOKEN \ 
--cahash CAHASH --certificate-key KEY
```

Step 3: Join the worker nodes to the first control plane node.

On the ClusterWare head node, install the `clusterware-kubeadm` package and join the desired ClusterWare nodes as workers to the first control node. For example, for full-install nodes n[3-4]:

```
[admin@head]$ scyld-kube -i n[3-4] --join --cluster 10.54.150.200 \ 
--token TOKEN --cahash CAHASH
```

or for PXEbooting nodes n[5-7] that use image `kubeimg`:

```
[admin@head]$ scyld-kube --image kubeimg --join --cluster 10.54.150.200 \ 
--token TOKEN --cahash CAHASH
```

### 8.8 Appendix: Using Docker

Scyld ClusterWare also supports Docker, which is available from CentOS.

The following example shows Docker being used to execute the pre-built Docker "Hello World" image. First preferably create a new image:

```
# Clone a new image instead of modifying an existing image.
sclyld-imgctl -i DefaultImage clone name=DockerImage

# Install needed packages inside the new image.
# NOTE: This uses the default _boot_style=rwram and _boot_rw_layer=overlayfs
scyld-modimg -i DockerImage --freshen --overwrite --no-discard \ 
--install docker --exec "systemctl enable docker" --upload
```

Alternatively, the administrator may choose to use a _boot_style of roram or iscsi for nodes using this DockerImage. To accomplish this, more must be done to the `DockerImage` image and to all the nodes that use that image. For example:

```
# Additionally create file /etc/rwtab.d/docker in the image.
sclyld-modimg -i DockerImage --freshen --overwrite --no-discard \ 
--install docker --exec "systemctl enable docker" \ 
--exec "echo 'empty /var/lib/docker' >/etc/rwtab.d/docker" --upload

scyld-nodectl -i <NODES> set _boot_style=roram _boot_rw_layer=rwtab
```

You will also need to set up IP forwarding on the head node(s) for the node to access the external Internet, which may likely involve using `scyld-modimg` to add appropriate nameserver entries to the node's `/etc/resolv.conf`. See `Configure IP Forwarding` for details.

Now boot node n0 with the new DockerImage:

```
[admin@head]$ scyld-kube --image DockerImage --join --cluster 10.54.150.200 \ 
--token TOKEN --cahash CAHASH
```
scyld-bootctl -i DefaultBoot clone name=DockerBoot
scyld-bootctl -i DockerBoot update image=DockerImage
scyld-nodectl -i n0 set _boot_config=DockerBoot
  # Now reboot node n0
scyld-nodectl -i n0 reboot

When node n0 is up, you can initialize passphrase-less key-based access, as described in *Install OpenMPI, MPICH, and/or MVAPICH*, to allow your current administrator userid to ssh to the node, or you can simply login as root:

```
sudo ssh n0
```

# Now as user root on n0, and if n0 can access external Internet websites:

```
[root@n0 ]# docker run hello-world
Unable to find image 'hello-world:latest' locally
Trying to pull repository docker.io/library/hello-world ...
latest: Pulling from docker.io/library/hello-world
1b930d010525: Pull complete
Digest: sha256:41a65640635299bab090f783209c1e3a3f11934cf7756b09cb2f1e02147c6ed8
Status: Downloaded newer image for docker.io/hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://hub.docker.com/
```

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

Note the Hello from Docker! line in the above output.
8.9 Appendix: Using Singularity

Singularity is available in Scyld ClusterWare by installing the `singularity-scylld` RPM, which is built from source developed by Sylabs Inc., or by installing the `singularity` RPM found in the EPEL yum repository. See https://www.sylabs.io/docs for their extensive documentation.

The following example creates a Singularity container `openmpi.sif` containing `openmpi3.1`, and placing that container in a bootable image.

First create the `openmpi.def` Singularity definition file, then use that file to create the container:

```bash
# Use quoted "EOF" for bash to avoid % and $ expansions; just EOF for sh.
cat <<"EOF" >openmpi.def
Bootstrap: yum
OSVersion: 7
MirrorURL: http://mirror.centos.org/centos-%{OSVERSION}/%{OSVERSION}/os/$basearch/
Include: yum

%files
    /etc/yum.repos.d/clusterware.repo
    /etc/yum.repos.d/clusterware.repo

%environment
    PATH=/opt/scyld/openmpi/3.1.3/gnu/bin:$PATH
    LD_LIBRARY_PATH=/opt/scyld/openmpi/3.1.3/gnu/lib:/opt/scyld/slurm/lib64:$LD_LIBRARY_PATH
    MPI_HOME=/opt/scyld/openmpi/3.1.3/gnu
    MPI_LIB=/opt/scyld/openmpi/3.1.3/gnu/lib
    MPI_INCLUDE=/opt/scyld/openmpi/3.1.3/gnu/include
    MPI_SYSCONFIG=/opt/scyld/openmpi/3.1.3/gnu/etc

%post
    # IMPORTANT:
    # If instead using "OSVersion: 6" instead of "OSVersion 7" above,
    # then for any subsequent `rpm` or `yum`, add:
    # rpm --rebuilddb
    echo "Installing openmpi3.1-gnu rpm"
    yum -y install openmpi3.1-gnu
    exit 0

EOF

# Create the Singularity chroot "/tmp/openmpi" in which updates can be made.
sudo singularity build --sandbox /tmp/openmpi openmpi.def

# Make a sample update: build an openmpi test program inside the chroot.
sudo singularity exec -w /tmp/openmpi \
    mpicc -o /usr/bin/ring /opt/scyld/openmpi/3.1.3/gnu/examples/ring_c.c

# Finalize the sandbox chroot into the Singularity container "openmpi.sif".
sudo singularity build openmpi.sif /tmp/openmpi

Create a bootable image that hosts the Singularity container and can execute openmpi applications:

```
# Clone a new image instead of modifying an existing image.
sclyld-imgctl -i DefaultImage clone name=SingularityImage
```

(continues on next page)
# Install needed packages inside the new image.
scyld-modimg -i SingularityImage --freshen --overwrite --no-discard \
   --install singularity-scyld,openmpi3.1-gnu --upload

# Now get into the chroot of the Singularity image.
scyld-modimg -i SingularityImage --chroot --overwrite --upload --no-discard

# Inside the root, add your userid (e.g., "myuserid") if necessary, which
# creates a /home/myuserid/ directory, and import the Singularity container file.
useradd myuserid
scp myuserid@localhost:/home/myuserid/openmpi.sif /home/myuserid/
exit

Boot nodes n0 and n1 with SingularityImage:

scyld-bootctl -i DefaultBoot clone name=SingularityBoot
scyld-bootctl -i SingularityBoot update image=SingularityImage
scyld-nodectl -i n[0-1] set _boot_config=SingularityBoot
# Now reboot nodes n0 and n1
scyld-nodectl -i n[0-1] reboot

When the nodes are up, then initialize passphrase-less key-based access, as described in *Install OpenMPI, MPICH, and/or MVAPICH*.

Now you can run the *ring* program from n0 (or n1):

# logged into n0, or using a job scheduler
mpirun -np 2 --host n0,n1 singularity exec openmpi.sif /usr/bin/ring

Or from the head node:

# If not already installed
sudo yum install singularity-scyld openmpi3.1-gnu --enablerepo=scyld*
module load openmpi/gnu/3.1.3
mpirun -np 2 --host n0,n1 singularity exec openmpi.sif /usr/bin/ring

8.10 Appendix: Creating Diagnostic Test Images

In uncommon situations a cluster administrator may wish to execute a self-contained diagnostic program as a compute node image. "Self-contained" means the diagnostic program itself functions as a kernel and does not need an initrd.

An example is the *memtest86*+ memory diagnostic, which can be downloaded from www.memtest.org:

# Download the latest compressed binary:
wget https://www.memtest.org/download/5.01/memtest86+-5.01.bin.gz

# Uncompress the downloaded file to expose the binary:
gunzip memtest86+-5.01.bin.gz

# Clone the DefaultBoot into a new boot configuration that consists of just the binary:
scyld-bootctl -i DefaultBoot clone name=MemtestBoot kernel=@memtest86+-5.01.bin

# Configure the desired node (e.g., n123) to execute that new boot configuration:
sclyld-nodectl -i n123 set _boot_config=MemtestBoot

# And then reboot that node (with 'legacy' pxe booting!)
sclyld-nodectl -i n123 reboot

Note: The current memtest86+ only works for legacy PXE booting, not for uefi PXE booting, so the administrator needs to change the target node’s BIOS setting to legacy to boot memtest86+ -- and revert back to uefi if desired for normal ClusterWare compute node PXE booting.

8.11 Appendix: Booting From Local Storage Cache

Cluster designers sometimes include storage on compute nodes as scratch space or to fulfill the requirements of other cluster technologies such as caching in high speed storage systems. If a cluster administrator is able to partition off some of that space, ClusterWare can be configured to take advantage of this local storage. This can free up RAM that would otherwise be used to store the operating system and libraries, and in some circumstances of a very large node count may decrease boot-time network load for nodes which have local storage.

When a node boots using the disked boot_style, it checks two other attributes: _disk_cache and _disk_root. Each attribute should be set to a value that can be passed as a device to the mount command. This includes explicit partition paths such as /dev/sda2 or /dev/nvme0n1p4 as well as LABEL=X or UUID=Y aliases. Because UUIDs are randomly generated during partitioning or file system creation, they are less suitable for cluster use since every node would require a different value. Similarly, a heterogeneous cluster may have different physical disk configurations requiring a cluster administrator to specify different partition paths for different classes of nodes. For these reasons we encourage cluster administrators to label the target partitions using a tool appropriate to the file system, e.g. e2label. Because the _disk_cache and _disk_root attributes are ignored by other boot styles, setting nodes to the disked style can be used as a flag to enable and disable booting from local storage without otherwise altering the node’s boot configuration.

Early in the boot process a disked node will attempt to mount the partition specified by the _disk_cache attribute. If this attribute does not exist or if the partition specified cannot be mounted, an error will be logged and booting will continue without local caching. Shortly after the cache is mounted, the mount_rootfs script will attempt to mount the specified _disk_root partition. If this partition is not provided or cannot be mounted, an error is logged and booting continues in a rwram or roram style depending on the type of disk image downloaded. Log messages from this early boot process can be found in /var/log/messages on the node, and ClusterWare-specific early boot messages are also captured in the /opt/scyld/clusterware-node/atboot/cw-dracut.log file.

If the disk cache is successfully mounted, then prior to downloading any image the compute node will check if the image is already present in the cache. If the image is present, then the mount_rootfs script will compare the local file size and checksum to values provided by the head node. If both match, then the image download is skipped and the local copy will be used. Alternatively, if the image is not present in the cache or there is a size or checksum mismatch, then any local copy will be deleted and a fresh copy of the image will be downloaded into the cache partition.

During subsequent boots the booting node will confirm the cached image is valid and use the local copy whenever possible. Note that if the cache partition is large enough to hold several compressed images, then the local cache can provide a somewhat faster means to switch between images on consecutive boots. If the cache ever fills, thereby causing an image download to fail, then the cache will be cleared and the node will reboot to try again.

Important: Please note that a cache partition must be large enough to hold at least the compressed compute node
image plus a few megabytes, though ideally should be sized to hold a handful of compressed images.

If the disk root is successfully mounted, then when the image would usually be unpacked into RAM, the mount_rootfs script will instead delete the contents of the disk root and unpack the image into the now empty partition. Booting will then continue with that partition as the system root. Note that any changes made to the contents of this partition are intentionally discarded during the next disked boot. This is done to prevent cluster administrators from inadvertently creating a heterogeneous cluster with unexpected and unpredictable behavior.

**Important:** Root partitions must be large enough to hold the uncompressed image in addition to files that may be installed after boot. A rough minimum estimate is to provide 2.5 times the space required by the compressed image. We encourage administrators to err on the side of providing excess space, as storage is usually inexpensive.

In order to reduce the chances of automating destructive mistakes, ClusterWare does not provide tools to automatically partition compute node disks based on node attributes. Cluster administrators can manually partition disks in individual nodes for very small clusters and should research parallel management tools such as ansible when managing disk partitions on larger clusters: [https://docs.ansible.com/ansible/latest/modules/parted_module.html](https://docs.ansible.com/ansible/latest/modules/parted_module.html).

### 8.12 Appendix: Validating ClusterWare ISOs

To validate a downloaded Scyld ClusterWare ISO file, first import the gpg key that was used to sign the RPMs and ISOs:

```
```

Then download the CHECKSUM.asc file from the repo, e.g.,

```
wget https://<AUTHENTICATION_TOKEN>@updates.penguincomputing.com/clusterware/11/el7/iso/...
CHECKSUM.asc
```

and verify the CHECKSUM.asc file:

```
[admin@head]$ gpg --verify CHECKSUM.asc
gpg: Signature made Mon 07 Feb 2022 07:16:15 PM UTC using DSA key ID 0A1E1108
gpg: Good signature from "Penguin Computing <support@penguincomputing.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg: There is no indication that the signature belongs to the owner.
Primary key fingerprint: AEFA 2C55 EB4A 88EF BE71 022B 0722 4B0A 0A1E 1108
```

Confirm that the downloaded ISO is named in CHECKSUM.asc. For example, for `clusterware-11.6.0-g0000.el7.x86_64.iso`:

```
grep clusterware-11.6.0-g0000.el7.x86_64.iso CHECKSUM.asc
```

should find the ISO. Now compare the checksum of the ISO with the ISO named in CHECKSUM.asc:

```
diff <(sha256sum clusterware-11.6.0-g0000.el7.x86_64.iso) \
<(grep clusterware-11.6.0-g0000.el7.x86_64.iso CHECKSUM.asc)
```

and expect to see no differences.
8.13 Appendix: Managing Zero-Touch Provisioning (ZTP)

Important: Currently only supported for Cumulus switches.

Scyld ClusterWare supports ZTP (Zero-Touch Provisioning) of ONIE and related switches. Note that ZTP by itself does not provide a full, end-to-end control plane for cluster networking, but it is the first step in that direction, allowing for server-provided scripts to alter the configuration of connected switches.

Since the ZTP-capable switches are essentially Linux management systems attached to the switches, ClusterWare treats them as another node in the cluster. You can add them to the cluster using `scyld-nodectl create` and specifying the switch's MAC address. For example:

`scyld-nodectl create mac=aa:bb:cc:00:11:22`

which simplistically creates a new (switch) node in the default naming-pool and default group. This may not be the desirable approach, since it assigns a generic name like "n12" which is superficially indistinguishable from compute nodes "n0" through "n11". A better approach is to utilize the ClusterWare naming-pool and attribute-group functionality to assign a more self-identifying name and permit more efficient management of this and other ZTP-capable switches:

`scyld-clusterctl pools create name=ztpswitch pattern="switch{}"
scyld-nodectl create mac=aa:bb:cc:00:11:22 naming_pool=ztpswitch`

which creates a new naming pool "ztpswitch" and configures the new node inside that pool with the name "switch0". Subsequent ZTP-capable switches can use the same naming-pool, which names them "switch1", "switch2", etc.

The cluster administrator can then use:

`scyld-nodectl -i switch2 <action>
scyld-nodectl -i switch* <action2>`

to perform an action on a specific switch or a common action on all switches in that naming-pool.

Configure each ZTP node to boot using a ZTP boot script. A boot script may be written in Bash or Python. As with other scripts, the first line should be `#/path/to/interpreter`, e.g. `#!/bin/bash`. Some switches also allow Perl, Ruby, or a vendor-specific language. These scripts execute as user root on the switch and can execute commands supported by the switch, including triggering Puppet or Ansible runs, downloading files via `wget` or `curl` and manipulating or moving them on the switch, and more. After a successful execution, the script must return status 0.

ZTP boot scripts reside in `/opt/scyld/clusterware/kickstarts/`. Configure the boot script `ztp_config.sh` for the node `switch0` using the specific prefix "ztp":

`scyld-nodectl -i switch0 set _boot_config="ztp:ztp_config.sh"`

Since switch nodes are ClusterWare nodes, you can use attribute groups to configure this as well:

`scyld-attribctl create name=ZtpSwitches
scyld-attribctl -i ZtpSwitches set _boot_config="ztp:ztp_config.sh"
scyld-nodectl -i switch0 join ZtpSwitches`

which creates an attribute group "ZtpSwitches" and joins "switch0" into it. All members of that attribute group will boot the same `ztp_config.sh` script.

In a multi-headnode cluster, every head node should have the same ZTP boot script installed. Currently this must be done manually.
At boot time the ZTP-enabled node `switch0` executes a DHCP query. The server sees the query, identifies the node using the client's MAC address in the DHCP request, recognizes the client as a ZTP-enabled node and the node's `_boot_config`'s "ztp:", then builds a DHCP response that includes a URL of the form http://<SERVER_IPADDR>/boot/ztp_config.sh. The switch then uses standard web protocols to read the URL to download the script and execute it.

Per the Cumulus Linux guidelines, the script must include the phrase "CUMULUS-AUTOPROVISIONING", usually in a comment, in order to execute at ZTP boot. Other switch or NOS vendors may require similar keywords.

While the system may provide some limited logging that the ZTP script was run, it may make sense to log any/all command outputs to a known file for easier debugging and triage. A line such as `exec >> /var/log/autoprovision 2>&1` in a bash script writes output to that log file for subsequent commands in the script.

Once a ZTP-switch has been successfully configured and the script returns status 0, it will not execute the ZTP boot script again, not even at the next reboot of the switch node. To force the switch to re-execute the boot script on the next reboot, `ssh` to the switch and execute `sudo ztp --reset`. 
See Release Notes for summary information about the latest ClusterWare release. This section contains a more detailed ChangeLog history of all releases.

11.9.1-g0000 - November 4, 2022

- Assorted improvements to scyld-nss.
- Fix two GUI crashes when viewing image details.
- Remove dependency on libcggroup that caused image creation failures.
- Fix bug where a 11.9.0 head node could not join to a pre-11.9.0 head node.
- Add an SELinux module in the clusterware-ansible package.
- Fix scyld-modimg hang on system with SELinux disabled.
- Restrict ZTP-boot to Cumulus switches.
- Assorted other bug fixes.

11.9.0-g0000 - September 30, 2022

- Initial support for ZTP-boot for switches.
- Implement a couple fixes for the install-time el8 STIG.
- Include assorted tpm2_* tools into the initramfs for storing encryption keys in the compute node’s TPM.
- Include the nvme driver in the initramfs for disked booting on NVMe storage.
- Improved scyld-modimg SELinux labelling with a parallel setfiles.
- Fix a regression that broke the sshpass integration.
- Fix isc-dhcpd.log parsing in el8 and el9.
- Fix openSUSE image creation.
- Initial scyld-nss implementation for compute node name resolution on head nodes without DNS.
- Attempt to install nscd during image creation.
- Implement attribute substitution in power_uris.
- Initial support for a new _ansible_pull_now attribute.
- Enable repo_gpgcheck in our software repositories for el8 and later.
• Implement scyld-install --non-interactive for unattended installs.
• Disable and remove additional services during scyld-install --clear-all.
• Update to etcd 3.5.4
• Include a new GPU data collection script for TICK.
• Assorted other SELinux updates, bug fixes, and scaling improvements.

11.8.2-g0000 - August 19, 2022
• Fix a timeout during batch-create for nodes.
• Capture more log files in scyld-sysinfo.
• Greatly reduce calls to rpm in update-node-status.
• Ensure tftp starts after reboot on el8 head nodes.

11.8.1-g0000 - August 2, 2022
• Fix a regression that could cause a crash during image cloning.
• Fix bulk node creation when using a @contents.json source.
• Hide the BMC password in scyld-nodectl sol output.
• Fix handling of --binary in scyld-nodectl exec.
• Enable Git Smart HTTP for head-node hosted git repositories. Further Git improvements coming soon.
• Assorted other bug fixes.

11.8.0-g0000 - June 17, 2022
• Fix a scyld-nodectl ping crash where a down node is still ping-able.
• Default to not compressing data during image capture making the process significantly faster.
• Better checking that the ClusterWare installation source matches the system where the installation is running.
• Fix a bug that would render scyld-*ctl tools unusable when deleting an in-use naming pool.
• Fix long standing bug where a changing a node MAC would not be pushed to the DHCP server without another network change or a service restart.
• Use unsquashfs to unpack cwsquash file systems during rwram booting for a significant speedup.
• Implement variable replacement in boot configuration command lines.
• Add a node script to configure BMC settings based on a mixture of the power_uri field along with the _ips and _gateways attributes.
• Rewrite handling for adding unknown nodes to the cluster when they are seen making DHCP requests. Pre-loading MACs is still preferred.

• Fix scyld-clusterctl heads clean to clean unused files in more cases.

• Include more variables (groups, power_uri, etc.) in the [Node] section of the attributes.ini on compute nodes.

• Clean up basic.ks kickstart example file and separate ClusterWare related parts into an includable file.

• Fix down head handling on locally installed systems.

• Separate more IPv4 and IPv6 code and logging.

• Assorted changes to support Python 3.9 and later.

• Snap pip package versions forward.

• Add scyld-install --iso support to allow administrators to install from a downloadable ClusterWare ISO.

• Further refine support for installing RH COS from an uploaded ISO using their ignition system.

• Implement “then” support to allow multiple steps in a single command when using ClusterWare command line tools.

• Fix bugs with manually modifying an existing naming pool.

• Implement scyld-nodectl power setnext <bootdev> to set the next boot device for a node.

• Define parent-head-node inside scyld-modimg --chroot.

• Update /etc/hosts on compute nodes when they cycle from a down head to a working one.

• Implement scyld-modimg --write-repos to rewrite an image's clusterware-node.repo file based on the current head node configuration.

• Add a variety of new tests to better catch regressions.

• Assorted other bug fixes and scaling improvements.

11.7.2-g0000 - April 6, 2022

• Correct a pair of bash variable name collisions in update-node-status and the custom dracut module.

• Auto-reconnect for scyld-nodectl sol.

• Tweak the adjust-repos.sh script to work before find is installed.

• Catch a very unlikely case where nodes have the same system UUID.

• Assorted other bug fixes.

11.7.1-g0000 - March 18, 2022

• Small changes so _boot_rw_layer=rwtab works.

• Hide last_modified_on in the output from my tools.

• Fix script URL if the iPXE boots from the second network.
• Recognize uploaded RHEL CoreOS ISOS.
• Improve the export and iso download progress output.
• Administrators can use URLs as sources when uploading ISOS and other binaries.
• Implement scyld-install --os-iso and the corresponding scyld-add-boot-config --iso.
• Do not automatically use vault.centos.org as a DefaultImage installation source.
• Assorted other bug fixes.

11.7.0-g0000 - February 23, 2022
• Detect when a CentOS 8 image is being created and update the files in /etc/yum.repos.d to use the vault.centos.org server.
• Change the basic.ks example so the node sets its own _boot_style=next before rebooting instead of powering off.
• Catch a code path that could result in duplicate names for images.
• Support name= syntax to clear attributes and fields.
• Give administrators more control over image creation details via the --pkgmgr parameter.
• Implement a document cache to improve performance database performance.
• Support unconfiguring bootnet at the end of the initramfs.
• Significantly expand the scyld-nodectl waitfor functionality.
• Support per-node _gateways and _macs reserved attributes.
• Accept additional MAC address formats.
• Recognize nodes by DUID to support booting with IPv6 via dhcpv6.
• Split dhcpd.conf.template into multiple parts for easier management.
• Restore more specific success and error messages from node power control commands.
• Implement _disk_wipe and add encryption support to _disk_cache reserved attributes.
• Add a reboot-kexec command to the clusterware-node package to trigger kexec rebooting from inside the node.
• Include mdadm in the initramfs if it is installed in the image.
• Fix bug where scyld-modimg auto-deletes a cached item and then immediately tried to use it.
• Generalize IP code to properly handle IPv6 addresses.
• Properly handle interfaces with more than one address.
• Include the ipxe.iso for booting virtual machines using IPv6.
• Assorted Ubuntu compute node networking fixes.
• Remove unused URL parameters passed during the iPXE script download.
• Improve scyld-sysinfo data collection on Ubuntu compute nodes.
• Include Scyld TICK packages in scyld-install --clear-all / --update.
• Correctly terminate scripts with non-zero exit code on error.
• Remove the unused `ip=` kernel argument added during compute node boot.
• Better error handling during chain booting.
• Automatically update compute node hostnames when the node name or `_hostname` attribute changes.
• Allow for clearing selected database object fields by setting their value to an empty string, i.e. the node-level `cmdline` field.
• Log more information about each request in the `api_access_log`.
• Assorted other bug fixes and scaling improvements.

11.6.0-g0000 - October 31, 2021
• Fix a boot-time `systemd` service race condition by forcing `NetworkManager.service` to wait for `cw-boot-prenet.service` to complete.
• Require `scyld-adminctl` keys to be unique so they can be used to identify the user in `ssh` sessions during `git clone`.
• Update `headctl` to automatically open and close `firewalld` ports.
• Empty `~/.scyldcw/` during `scyld-install --clear-all` but leave the logs directory intact.
• Bump busybox version to 1.34.1 and remove unused build options.
• Update the `slurm-scylld` packages to version 21.08.3.
• Update the TICK packages: `telegraf` to version 1.20.2, `influxdb` to version 1.8.10, and `chronograf` to 1.9.1.
• Fix a boot chaining issue where peers waited on each other, and enable boot chaining by default with `chaining.enable` now defaulting to `True`.
• Make argument order optionally much more flexible.
• Better repository related error messages from `scyld-install`.
• Ensure node FQDNs are placed before the corresponding short names in our `dnsmasq` hosts file.
• Further improvements to `scyld-cluster-conf` to handle more config file formats.
• Rewrite naming pool pattern collision detection.
• FIPS fixes for Lark grammer parser.
• Add support for pushing IPs from secondary pools into node `_ips` attributes.
• Include multiple (default up to 3) head nodes for DNS through dhcp.
• Document head node support for AlmaLinux.
• Add a new optional `clusterware-ansible` package including boot-time `ansible-pull` support.
• Use symlinks when including `git` in ClusterWare packages.
• Implement `scyld-nodectl waitfor` using `--selector` logic.
• Expose and improve the `scyld-nodectl --selector` support.
• Client-side recursive delete for boot configuration and distros.
• Delete unmodified images from the scyld-modimg cache older than one hour.
• Add an optional uname wrapper inside scyld-modimg --chroot.
• Snap pip and npm package versions forward.
• SELinux improvements for MLS and RHEL registration.
• Replace systemctl restart with systemctl reload in the installer and headctl scripts.
• Set the new id_method status field based on how the head node identified the compute node during the status update.
• Add new scyld-clusterctl gitrepos and scyld-clusterware certs tools.
• Implement --version in scyld-kube so administrators can deploy a specific version.
• Assorted other improvements and bug fixes.

• 11.5.1-g0001 - October 5, 2021
  • SELinux changes for manipulating MLS images.
  • Correct scyld-modimg discard behavior.
  • Correct scyld-nodectl ping and related failures.
  • Assorted other bug fixes.

• 11.5.1-g0000 - October 4, 2021
  – Significant changes to scyld-cluster-conf to improve handling of less common configurations and make future maintenance simpler.
  – Check for rpmsave / rpmnew files post upgrade and notify the cluster administrator that they should be addressed.
  – Fix a typo in the influxdb rsyslog configuration.
  – Add parent-head-node to /etc/hosts inside chroots to more closely match deployed image behavior.
  – Significant rearrangement of scyld-modimg to make future maintenance simpler.
  – Fix uid/gid handling in scyld-modimg --import. Tool now expects any tar being imported to contain correct numeric uid / gid instead of matching names against the host system.
  – Remove deprecated replace code in favor of update calls.
  – Use inst.ks= instead of deprecated ks=. Note that this does break kickstarting for el6.
  – Fix a bug in scyld-nodectl exec for nodes that changed IP addresses after a ClusterWare head node upgrade.
  – Remove most references to the cwtar format but ensure scyld-modimg can still import and repack the format.
  – Test Rocky Linux 8.4 head nodes.
  – Update busybox used in the custom initramfs.
– Remove remaining traces of NFS Ganesha integration.
– Small improvements to image capture functionality.
– Customize the iPXE user-class (now CWiPXE) to more tightly control the iPXE boot process.
– Force iPXE to use low numbered ports when fetching boot files.
– Version bumps for many third-party packages.
– Include a custom-built version of git for future features.
– Remove long deprecated code.
– Assorted other improvements and bug fixes.

• 11.5.0-g0001 - September 2, 2021
  – Fix a regression that broke kickstart and live booting.

• 11.5.0-g0000 - August 26, 2021
  – Default to etcd for new installs.
  – Add HA kubernetes support to the scyld-kube utility.
  – Implement automatic etcd compact and defrag on heads nodes as well as auto-eject and auto-rejoin for the head node cluster.
  – Both general and etcd specific performance improvements.
  – Ensure _no_boot stops soft and hard power control commands unless --force is specified.
  – Improve scyld-mkramfs --update to accept a --kver <VERSION> option when updating a boot configuration after a kernel upgrade.
  – Implement scyld-nodectl reboot --kexec in preparation for image previewing.
  – Reduce resource usage in scyld-nodectl status --refresh.
  – Ongoing effort to encrypt more compute node to head node and compute node to compute node communications.
  – Reduced polling by implementing waiting on database content changes.
  – Much cleaner logging during service shutdown.
  – Code and endpoint removal and cleanups.
  – Assorted other improvements and bug fixes.

• 11.4.3-g0000 - July 8, 2021
  – Confirm image creation from an ISO is limited to packages on the ISO.
– Improved FIPS support for earlier RHEL and CentOS releases.
– Improved proxy handling during scyld-install --update.
– Rewrite scyld-nodectl status --refresh to handle more corner cases and terminal resizing.
– Include scyld package versions in scyld-nodectl ls -L output.
– Add support for redirecting stdout and stderr into per-node local files when using scyld-nodectl exec.
– Fix bugs around manually setting network database parameters.
– Simplify URLs used during the early boot process.
– Change kubeadm runtime to use containerd instead of docker.
– Change dnsmasq SRV request handling to immediately return an invalid response to requests from Couchbase.
– Update the slurm-scyld packages to version 20.11.8.
– Update the TICK packages: telegraf is version 1.18.3, influxdb is version 1.8.6, chronograf is 1.8.10, and kapacitor is version 1.5.9.
– Assorted other improvements and bug fixes.

11.4.2-g0000 - June 4, 2021
– Upgrade OpenMPI removing the Slurm library version dependency.
– Initial support for MLS on RHEL 7 and CentOS 7 head nodes.
– Support scyld-nodectl ping to ping nodes on demand.
– Support URL encoding of the password section of the power_uri to handle additional characters. Any power_uri currently containing % may need to be updated.
– Use excludes.txt to exclude specific directories in the squashfs packer.
– Exclude paths from the setfiles call when exiting the chroot.
– Add additional security related HTTP headers.
– Small updates to the ReactJS GUI including fixing checkbox behavior.
– Fix a managedb save regression that defaulted to saving into a directory instead of a file.
– Assorted other improvements and bug fixes.

11.4.1-g0000 - April 30, 2021
– Significant performance improvements from reducing database contention in high node count clusters.
– Technology preview: Adding etcd as an alternative database backend. etcd should allow for further scale improvements in later releases.
– Announcing support for Oracle Linux 7 and 8. These operating systems are now supported for both head nodes and compute nodes in all the configurations supported for RHEL and CentOS.
• Expand the information collected about hardware and firmware versions at boot time. This information expands the possibilities for cluster administrators to detect and track cluster changes.

• Improved FIPS and MLS support. ClusterWare now supports compute nodes CentOS and RHEL 8 images in MLS enforcing mode, and FIPS 140-2 is fully supported on head nodes and compute nodes across the cluster.

• Removing NFS compute node root file system integration. In modern, scalable clusters the benefits of separating compute nodes from head nodes (e.g. simplicity, performance, security, and independence from head nodes post boot) significantly outweigh the costs of running the core operating system from RAM.

• Support switching database backends on existing clusters using headctl.

• Streamline the peer download process used to pass files between head nodes.

• Improve initramfs LUKS support when booting ephemeral compute nodes with _disk_root and boot style disked.

• Fix multiple problems with CentOS / RHEL 8 kickstart support. The example basic.ks kickstart file now works for versions 7 and 8.

• Print any unexpected errors from the setfiles call when exiting the scyld-modimg chroot.

• Correct IP calculations in more complicated cluster configurations.

• Improve scyld-nodectl exec across large node counts.

• Better banner filtering from scyld-nodectl exec and soft power control via scyld-nodectl <reboot|shutdown>.

• Attempt to install rdma-core and fipscheck packages in newly created compute node images.

• Improve backend caching and hinting for IP/MAC/name to UID translations.

• Improved FIPS support on compute nodes. This change requires upgrading clusterware-node inside images and rebuilding the initramfs files.

• Deprecate NFS Ganesha integration and obsolete the clusterware-ganesha package.

• Correct CentOS 6 images to point at the CentOS vault during creation.

• Improve scyld-install to stop earlier on error.

• Include mlx5_core by default in initramfs files.

• Implement stored selectors and dynamic groups.

• Names of dynamic groups cannot collide with attribute group names.

• Dyngroups can reference other dyngroups but can be slow to evaluate.

• Properly report group join / leave failures.

• Update the parser used on node specifications.

• Upgrading and refreeze all pip packages to latest versions.

• Remove unnecessary pip packages from the virtual environment.

• Assorted other improvements and bug fixes.

• 11.4.0-g0000 - January 22, 2021
– Initial kubeadm support. This adds a new clusterware-kubeadm package providing a scyld-kube command. See Kubernetes for details.

– Support passing \%group to scyld-nodectl in place of a node specification to affect all nodes in the named group.

– When installing from a ClusterWare ISO, upload that ISO into the ClusterWare system as a repo and update any file:// URLs in clusterware.repo accordingly.

– Properly copy gpgcheck values from /etc/yum.repos.d/clusterware.repo on the head node to the clusterware-node.repo during image creation.

– Do not ask for a ClusterWare password when stdout is not a terminal.

– Support CentOS Stream for head nodes and compute node images.

– The Job Scheduler -scyld.setup scripts now support optionally naming specific nodes (vs. presuming all up nodes) for the actions init, reconfigure, and update-nodes. See Job Schedulers.

– Remove some unused Couchbase-related Requires and BuildRequires from the spec file.

– Default to using the current $USER in scyld-* commands when no client.authuser is defined in settings.ini.

– Provide ISO contents over HTTP via a /repo/<name>/content/ URL.

– Do not record virt-what output on non-virtual nodes.

– Update libvirt-python to 6.10.0. Expect many other Python and NPM packages to be updated in following releases.

– Enable HTTPS communication during head node installation.

– Fix set-node-attrs command line parsing, and treat arguments without ‘=’ as a request to delete the named attribute.

– Do not install clusterware-ganesha during head node installation.

– Fix kernel version detection when multiple /lib/modules/<kernel>/ directories exist.

– Switch backend subexec from multiprocessing to threading, thereby making some deadlocks much less likely.

– Assorted other improvements and bug fixes.

• 11.3.0-g0001 - December 2, 2020

– Simplify initramfs rwram booting with SELinux by fully preserving rather than restoring SELinux contexts from the image.

– Compute IPs at node creation time instead of waiting for the leases daemon to compute the same. Clearing the ip field via scyld-nodectl up ip= will trigger immediate recomputation.

– Confirm incoming _boot_config and _boot_style strings are usable before accepting them.

– Adapt initramfs scripts to boot Ubuntu and Debian images.

– Improved support for customizing initramfs files through scyld-mkramfs.

– Add scyld-mkramfs --update <bootconfig> to simplify the common case where a cluster administrator wants to update the initramfs in an existing boot configuration.
– Initial implementation of `scyld-chroot` inside `scyld-modimg` chroots including `copyin`, `copyout`, and `info`.

– Fully disable backend image repacking since we now only use a single image format.

– Capture more information about compute node storage and Infiniband hardware.

– Expand the `yum` and `dnf` handler to also support `zypper` systems, i.e. openSUSE.

– Try to install `less`, `iperf3`, and `cryptsetup` when creating images.

– Initial implementation of `scyld-bootctl import` to match the existing export command.

– Assorted other improvements and bug fixes.

• 11.2.2-g0000 - October 30, 2020

  – Add `/opt/scyld/clusterware/bin/headctl` script to enable / disable Apache features on the head node. Can enable / disable HTTPS and set compute nodes to prefer HTTPS communication. Will default to preferring HTTPS in future release.

  – Compute nodes verify server identity provided by HTTPS when possible, but default to accepting unverified head nodes.

  – Further address a low probability file corruption bug when `scyld-modimg` unpacks images.

  – Fix IP collision bug introduced in 11.2.1 so that `X.X.X.1` is not detected as matching `X.X.X.1[0-9]+`.

  – The `scyld-tool-config` tool will generate a HTTPS `base_url` field when connecting to any server other than localhost.

  – Assorted SELinux updates for basic MLS policy.

  – Increase default password lengths as they are rarely manually entered.

  – Rearrange Apache configuration files to simplify changes in `/etc/httpd` and add a `CW-Proxy-Secret` header to confirm when backend system can trust other forward-related headers.

  – Double Python thread count to 32.

  – Initial LUKS in the initramfs providing encryption-at-rest for ephemeral compute node boot style `disked` with `_disk_root`.

  – Initial implementation of compute node peer downloads for boot chaining. Controlled by `chaining.enable` `base.ini` variable that defaults to `False`.

  – Add `arping` to `busybox` for early `dhcp` client scripts.

  – Remove deprecated arguments and content from `dhcpd.conf.template`, `scyld-clusterctl`, `mount_rootfs`, etc.

  – Add `scyld-nodectl` `sol` `[--enable|--steal]` options.

  – Include node hostnames in `dhcp` offers and more aliases in `dns`.

  – Expanded support for `_ips` to create `ifcfg-IFACE` files.

  – Include public ssh host keys in compute node status.

  – Pass the head node's gateway to compute nodes on the same network.

  – Capture more hardware (IB, NVMe) details during node boot.
- Assorted other improvements and bug fixes.

• 11.2.1-g0000 - September 24, 2020
  - Add mount/umount back into sudoers.d for Ganesha exports.
  - Fix Ganesha export permissions.
  - Disable backend repacking.
  - Disable zypper detection that triggered in odd circumstances.
  - Fix parsing of distribution major number.
  - Exclude tests folders from clusterware-tools.
  - Fix percent sign use in _boot_tmpfs_size.

• 11.2.0-g0000 - September 4, 2020
  - Support for CentOS / RHEL 8 head nodes.
  - Remove cwtar as a backend image format, leaving only cwsquash.
  - Fix scyld-modimg crash on bad --query.
  - Fix scyld-nodectl ls -l (and ls -L) ram_total and scyld-nodectl status -L ram_free output.
  - Fix permissions when creating files in sync-uids.
  - Fix scyld-modimg --create for CentOS 8.0 / 8.1.
  - Wait for rebalance to complete when joining head nodes.
  - Allow for zero-padding of node names.
  - Add more scyld-nodectl ls -l and ls -L output fields.
  - Rework scyld-add-boot-config to be more flexible.
  - Include example node.sh for locally installed compute nodes.
  - Only use local authentication when connecting to local server.
  - Improve locally installed compute node hostname handling.
  - Combine and improve calls to file to identify objects.
  - Remove remaining bits of bpstat and other legacy tools.
  - Install an example settings.ini during scyld-install.
  - Shorten paths in some output to make output more readable.
  - Trick mksquashfs into providing more detailed progress.
  - Clean up and standardize database failure cases, and resume daemons when database recovers.
  - Implement database purge and improve scyld-install --clear.
  - Improve package removal during scyld-install --clear-all.
– Change the cwsquash format to use a GPT partition table.
– Move ganesha SELinux rules into the clusterware-ganesha package.
– Improve the take-snapshot tool, which performs database backups and manages retention of those backups, typically executing as a cronjob. See take-snapshot in the Reference Guide.
– Improvements to scyld-sysinfo, including no longer requiring setup of user root authentication to capture state of compute nodes.
– Assorted other improvements and bug fixes.

• 11.1.2-g0001 - July 8, 2020
  – Patch pyramid in the virtual environment to allow a non-security use of md5.

• 11.1.2-g0000 - July 1, 2020
  – Initial implementation of node naming pools.
  – scyld-install update now calls managedb update.
  – Head and compute node status includes their "now" timestamp.
  – Initial implementation of head nodes as a chrony pool defaults to disabled.
  – Squashfs tools now use 50% of the available processors, although this is configurable.
  – Boot time set_hostname.sh script now uses hostname instead of hostnamectl on CentOS 6.
  – Fix an authentication race that triggered password prompts.
  – Initial support for CentOS 8.2 compute node images.
  – Add a short (0.03s) cache in the database layer.
  – Improved kickstart menu generation.
  – Use enabled=0/1 in /etc/yum.repos.d/clusterware.repo to avoid inadvertent yum updates.
  – Changes to scyld-install in preparation for CentOS 8.
  – Expanded variable substitution in kickstart files.
  – Improved SELinux permissions on enforcing compute nodes.
  – Fix file descriptor leak causing "too many open files" error.
  – Support X-Sendfile when downloading images and boot files.
  – scyld-modimg --query lists all installed packages.
  – Fixes to scyld-modimg discard and upload logic.
  – Assorted other improvements and bug fixes.
• 11.1.1-g0002 - May 27, 2020
  – Only updating clusterware-tools and these release notes.
  – Remove a log statement that caused a crash in scyld-nodectl exec when providing stdin.
  – Conditionally reinstate some initramfs code that is required to successfully boot a cwsquash image with style rwram.

• 11.1.1-g0001 - May 21, 2020
  – Use cgroups to identify and terminate child processes from a chroot.
  – Ignore /tmp and /var/tmp when correcting SELinux contexts in a chroot.
  – Use the head IP instead of the gateway IP in iscsi boot style.
  – Database cleaning code is now aware of uploaded ISO files.
  – Cleaning code will not attempt to connect to a down head node.

• 11.1.1-g0000 - May 19, 2020
  – Clearer errors from the client tools when the head node is unresponsive.
  – Handle when a large upload times-out, fixing the "size does not match" error.
  – Add mechanism for starting a long running task and checking for results in separate calls with a custom HTTP header.
  – Rewrite remotely deleted files detection to reduce the chances of leaving .old.00 files.
  – More daemons now clean up their leftovers in the workspace/ directory.
  – Add storage cleaning support via the scyld-clusterctl heads clean command. See scyld-clusterctl for details.
  – The status of ClusterWare services on a head node or nodes can now be checked and changed via the scyld-clusterctl heads service command. See scyld-clusterctl for details.
  – Fix a case that failed to find the disk during iscsi booting.
  – Improvements to libvirt power control for VM compute nodes.
  – Improved logging in SSH and Couchbase failure cases.
  – Nodes can be reordered using scyld-cluster-conf load without losing configuration.
  – Fix a cloning failure that left file copies in /opt/scyld/clusterware/storage/.
  – Display "[deleted]" when a database link is broken in scyld-bootctl or scyld-attribctl.
  – More consistent error and success messages from power on/off/status.
  – Reduce database calls in common code paths.
– When exiting `scyld-modimg`, move the stdout of "fixing SELinux file labels" to after choosing to keep an image, not prior to that choice.
– Document booting memtest86+ on compute nodes.
– Better error handling in clusterware-node scripts and head initialization.
– Assorted other improvements, code clean ups, and bug fixes.

• 11.1.0-g0001 - March 16, 2020
  – Default to `rwram` booting even when using the `cwsquash` format.
  – Improvements to the code that pulls images, ISOs, and boot files between heads.
  – More useful error messages from `scyld-modimg` package commands.
  – Better iSCSI device detection at boot time.
  – Default the authentication cookie lifetime to 20 minutes.
  – Initial support for capturing images from running nodes.
  – Support for the SELinux MLS policy on compute nodes.
  – Support `tar` input and output for `managedb`.
  – Expanded ISO upload and kickstart support.
  – Add `_boot_style` `live` and `next` for booting CentOS / RHEL ISOs.
  – Improved support for re-assigning compute node indices.
  – Compute nodes will re-fetch keys and head nodes if their head node is replaced with a new installation.
  – Simplify steps to switch head node SELinux status.
  – Include more tools in the initramfs busybox build.
  – `scyld-install` is more forgiving when creating the first user.
  – Adding `--grouped` and `--in-order` support to `scyld-nodectl exec`.
  – Officially support `scyld-modimg` `--mount` / `--unmount`.
  – Capture any modified installed file in `scyld-sysinfo`.
  – Include rsyslog and network information from telegraf.
  – Include progress meters on all `scyld-*ctl` uploads or downloads.
  – Support uploading larger files such as full DVD ISO files.
  – Add initial support for creating ClusterWare installation ISO images.
  – Assorted other improvements, code clean ups, and bug fixes.

• 11.0.8-g0000 - November 8, 2019
  – `dhcpd.conf.template` improvements to simplify bootstrapping systems.
– Initial implementation of `take-snapshot` for backing up the database and images.
– Pass more power command errors up to the user.
– Fix SELinux permissions for chronograf proxying.
– Move port numbers into named services for firewalld.
– FIPS fixes for ISC dhcpd to allow and default OMAPI to hmac-sha1.
– Default to using `-Ilanplus` for ipmitool calls.
– Support for filtering banners out of `scyld-nodectl exec`.
– Add a `_remote_user` attribute so we no longer require root ssh to control compute nodes.
– Improvements to the Slurm and TORQUE helper scripts.
– Add the `sync-uids` script to inject user accounts.
– Generate longer passwords for Couchbase.
– Replace most periodic sudo calls with long-lived scripts to reduce logging to `/etc/log/secure`.
– Default authentication to `pam_authenticator + maplocal`.
– Assorted other improvements and bug fixes.

• 11.0.7-g0001 - October 2, 2019
  – Add SELinux rule for ClusterWare service to query service status.
  – Fix a small bug where `scyld-sysinfo` was not capturing modified ClusterWare files (`rpms_clusterware_verify`).
  – Add a missing line to the clusterware-installer REVISIONS file.

• 11.0.7-g0000 - October 1, 2019
  – `scyld-sysinfo` now optionally captures compute node state.
  – Add 20-second keep-alive when wrapping ssh commands.
  – `scyld-nodectl ssh` command is an alias for `scyld-nodectl exec` if a command is passed.
  – Expand the head node information stored in the database.
  – Various `scyld-*ctl` commands support field selection with new `--field` arguments.
  – Various `scyld-*ctl` commands support two new output formats: `--csv` and `--table`.
  – Include `sanboot` as a `_boot_style` to boot local disks or URLs that iPXE sanboot supports.
  – `scyld-install` doing an upgrade will not run steps that were performed when doing the initial ClusterWare install and which may have been subsequently altered by the local administrator.
  – `scyld-install` prints version information for each installed or upgraded packages.
  – `scyld-install` passes `http_proxy/https_proxy` to underlying calls.
- Assorted other improvements and bug fixes.

• 11.0.6-g0000 - September 6, 2019
  - Include version number in REVISIONS files.
  - Fix a `scyld-modimg` problem that rejected any attempt to create a new image with a name that was a subset of an existing image name.
  - Add `scyld-clusterctl heads` that treats head nodes as database objects that can be viewed or deleted. More features to come.
  - Support socket-based admin authentication for local user accounts.
  - Fix `scyld-cluster-conf save`.
  - Eliminate an innocuous "Failure" message "No power URI provided for node" seen when doing `scyld-nodectl power cycle` or `power off`.
  - Add `nfs-utils` to the base image.
  - Pass more `ipmitool` error messages back to the caller.
  - Catch some exceptions that would unnecessarily stop daemons, and instead handle more gracefully.
  - `initramfs dhclient` should not survive the `switch_root`.
  - Add `_hostname` as a reserved attribute to override specific compute node hostnames. See `Reserved Attributes`.
  - Allow administrators to set a boot configuration image to "None" for new kickstart/preseed support, and add new appendices in the ClusterWare documentation that provides examples of how to use Red Hat kickstart for Ubuntu and CentOS (see `creating-nodes-with-kickstart`) and Debian preseed (see `creating-nodes-with-preseed`).
  - Assorted other fixes and improvements.

• 11.0.5-g0001 - August 6, 2019
  - Temporarily disable automatic renaming of unreferenced files.

• 11.0.5-g0000 - August 1, 2019
  - Fix the `--soft` then `--hard` behavior when rebooting or shutting down nodes.
  - Simplify and improve human readable tool output unless `--no-pretty` is passed.
  - Add a new `ssh` action to `scyld-nodectl`; details in documentation.
  - Include `/etc/systemd/system/couchbase-server.service.d/override.conf` to allow Couchbase to use MD5 even when FIPS mode is enabled.
  - Suppress FIPS mode messages from `scyld-nodectl exec`. 
- Support for locally installed compute nodes; details in documentation.
- Fixes when passing binary data to stdin of scyld-nodectl exec.
- Move the dhcpd.leases file from the default location to /opt/scyld/clusterware-iscdhcp/conf/dhcpd.leases.
- Give other head nodes a better chance to delete local copies of deleted content.
- Detect and rename files in storage that are not referenced in the database.
- Update resolv.conf if the only nameserver was a head node that goes down.
- Assorted other fixes and improvements.
- The slurm-scyld packages are updated to version 19.05.1, and openmpi2.0, openmpi1.10, and openmpi1.8 packages are rebuilt as version g0004 for compatibility with the newer slurm-scyld library. The openmpi3.1 packages are updated to version 3.1.4; openmpi3.0 updated to version 3.0.4; openmpi2.1 updated to version 2.1.6; and openmpi4.0 version 4.0.1 has been added to the distribution, all also compatible with the new slurm-scyld library and rebuilt as version g0004.
- 11.0.4-g0001 - July 3, 2019
  - Support CentOS 6 images for compute nodes.
  - Fix problem of root authorized keys being overwritten on compute node at boot time.
  - Require node status updates to arrive on privileged ports.
  - Improved api_error_log capture of IP addresses.
  - Make --summary the default scyld-nodectl status output.
  - Various scyld-sysinfo improvements, including requesting a comment from the user that gets added to the output.
  - Pass remote IPs through ProxyPass to get them to the logs.
  - Link dracut statically to simplify supporting different compute node OSes.
  - Enable automatic --soft then --hard behavior for scyld-nodectl reboot and shutdown, and document the difference.
  - Convert more exceptions to errors due to bad command line arguments.
  - Wrap ipmitool sol activate in a new scyld-nodectl option.
  - Add an empty /etc/fstab during image creation.
  - Modify the prompt when inside a chroot.
  - Fix a scyld-bootctl clone bug: copy the release field.
  - Better error messages when a Couchbase member is unreachable.
  - Log the head's hostname when starting the service.
  - Add a syncer daemon that fetches remote files in the background.
  - Add managedb update to fix Couchbase after out-of-diskspace conditions.
  - Add scyld-nodectl power on/off/cycle/status and scyld-nodectl sol.
  - If a small file is passed as stdin to scyld-nodectl, then exec the contents instead of streaming it.
– Cleanups to scyld-modimg around setting name, distro, and description.
– Rename scyld-modimg --export to --copyout, and implement a new inverse action --copyin.
– Assorted other fixes and improvements.
– Various other packages have been released in coordination with Scyld ClusterWare 11.0.4-g0001 and should be updated, if installed: torque-sclyld, slurm-sclyld, singularity-sclyld, openmpi3.1, openmpi3.0, openmpi2.1, openmpi2.0, openmpi1.10, and openmpi1.8.

The torque-sclyld and slurm-sclyld packages are now split into three packages for each job scheduler. For example, torque-sclyld (which requires torque-sclyld-libs) installs on the server, and torque-sclyld-node (which requires torque-sclyld-libs) gets installed into a node image by the sched-helper script. (See Job Schedulers.)

singularity-sclyld updates to version 3.2.1, and it no longer install files into /opt/scyld/, thus no longer requiring the user to module load singularity. The installed files are now accessible via the standard SPATH and SLD_LIBRARY_PATH.

• 11.0.3-g0020 - June 6, 2019
  – Fixes to peer download so that only one thread will download at a time.

• 11.0.3-g0014 - May 24, 2019
  – Stopping the clusterware service now also stops the clusterware-dhcpd and clusterware-dnsmasq services.
  – Include the pciutils package and an empty /etc/sysconfig/network file when creating the base image.
  – Fix various scyld-install --clear-all problems of overly aggressive deletions.
  – Add write_ifcfg.sh to the prenet startup on compute nodes.
  – Move the location of the scyld-helper script and add functionality to improve the configuration of Slurm or TORQUE. See Job Schedulers.
  – Minor fixes to managedb leave and eject.
  – Improve scyld-sysinfo error handling.
  – Expanded documentation around failover.
  – The sched-helper script can now push changes into compute node images.
  – Switch default gateway for compute nodes during head node failover.
  – Implement peer downloads for head node's missing files.
  – scyld-cluster-conf save now handles nodes on multiple networks.

• 11.0.3-g0000 - May 8, 2019
  – First General Availability release.
– Mark dnsmasq.conf.template and dhcpd.conf.template as configuration files.
– Support dhcp relays.
– Reduce log messages in api_error_log.
– Fix an early boot issue that was causing yum to fail on nodes booted using roram style.
– Fix the squashfs packer to work on images up to 100GB.
– Default to 16 threads in the Apache wsgi configuration.
– Add --clear-all argument to the installer.
– Python daemons will now attempt to automatically restart with an exponential backoff.
– Implement the _preferred_head attribute.
– Fix a bug where results were listed per node instead of collapsed.
– Other assorted documentation and tool fixes.
– Fixes for SELinux on head nodes:
  * dnsmasq properly starts and serves compute node addresses.
  * The repacker daemon disables itself due to required permissions.
– scyld-cluster-conf load improvements:
  * Multiple PXE boot networks can be loaded from a single configuration file.
  * Nodes will be assigned to the most recently defined network during parsing.
  * Support 'gw', 'via', and 'as' when parsing remote network definitions.
– scyld-nodectl improvements:
  * Parallelize power control commands.
  * Improved output streaming and parallelization.
  * Improved handling of stdin and --stdin.
  * Default the ssh_runner fanout value to 16 nodes at a time.
  * More documentation and examples.

• 11.0.1-b0209 - April 19, 2019
  – Third restricted release.
  – Includes the new clusterware-dnsmasq package, which supports resolving host names from /etc/hosts on the head node. See Node Name Resolution.
  – Support for establishing remote access between the head node(s) and compute nodes, or between compute nodes, by distributing SSH keys. See Compute Node Remote Access.
  – Excludes /boot/initramfs-* files, and does not exclude /etc/ssh/ssh_host_ * files, when packing images.
  – The Penguin serial number now appears in node hardware info, if it exists.
Scyld ClusterWare 11 Documentation, Release 11.9.1

-- scyld-nodectl exec improvements:
  * Command now exits with the subcommand's exit code.
  * Command can now operate through the head node (default) or --direct.
  * Hide some ssh warning messages.

• 11.0.1-b0197 - April 5, 2019
  - Second restricted release.
  - Numerous bug fixes and enhancements.

• 11.0.1-b0183 - March 22, 2019
  - First restricted release.
  - ClusterWare TORQUE reverts some changes that were made to the original Adaptive Computing distribution for Legacy ClusterWare 6 and 7:
    * Includes the built-in pbs_sched job scheduler, and does not include the maui scheduler.
    * Includes "LimitCORE=infinity" that 6 and 7 has removed.
    * Reverts the name pbs_trqauthd back to the original trqauthd, and pbs_mom and trqauthd are now systemd daemons.

9.1 Known Issues And Workarounds

The following are known issues of significance with the latest version of ClusterWare and suggested workarounds.

• The head node(s) must use a RHEL7 or CentOS7 base distribution release 7.6 or later environment, due to dependencies on newer libvirt and selinux packages.

• Scyld OpenMPI versions 4.0 and 4.1 for RHEL/CentOS 8 require ucx version 1.9 or greater, which is available from CentOS 8 Stream and RHEL 8.4.

• When using a TORQUE or Slurm job scheduler (see Job Schedulers), if a node reboots whose image was not created using /opt/scyld/clusterware-tools/bin/sched-helper, then the cluster administrator must manually restart the job scheduler. For example, if needed for a single node n0: NODE=n0 torque-sclyld-node or NODE=n0 slurm-sclyld-node. Or to restart on all nodes: torque-sclyld.setup cluster-restart or slurm-sclyld.setup cluster-restart.

   Ideally, compute node images are updated using torque-sclyld.setup update-image or slurm-sclyld.setup update-image, which installs the TORQUE/Slurm config file in the image and enables the appropriate service at node startup.

• If administrators are using scyld-modimg to concurrently modify two different images, then one administrator will see a message of the form:

9.1. Known Issues And Workarounds
WARNING: Local cache contains inconsistencies.
Use --clean-local to delete temporary files, untracked files,
and remove missing files from the local manifest.

then use scyld-modimg --clean-local.
However, only execute --clean-local after all scyld-modimg image manipulations have completed.

- The head node’s GRUB_CMDLINE_LINUX in /etc/default/grub must not contain ipv6.disable=1; otherwise, the memcached daemon cannot start (seen in /opt/couchbase/var/lib/couchbase/memcached. log.* logs), which means that Couchbase cannot start, despite the fact that Couchbase does not actually use IPv6.

- Ensure that /etc/sudoers does not contain the line Defaults requiretty; otherwise, DHCP misbehaves.

- The NetworkManger-config-server package includes a NetworkManager.conf config file with an enabled "no-auto-default" setting. That is incompatible with ClusterWare compute node images and will cause nodes to lose network connectivity after their boot-time DHCP lease expires. Either disable that setting or remove the NetworkManger-config-server package from compute node images.

- The scyld-clusterctl repos create command has a urls= argument that specifies where the new repo's contents can be found. The most common use is urls=http://<URL>. The alternative urls=file://<pathname> does not currently work. Instead, you must first manually create an http-accessible repo from that pathname. See Appendix: Creating Local Repositories without Internet.

- When moving a head node from one etcd-based cluster to another using the managedb join command, please reboot the joining head once the join is complete.

- If a new head node is failing to join an existing etcd-based cluster check /var/log/clusterware/etcd.log and look for repeated lines of the form:

  <DATE> <SERVER> etcd: added member <HEX> [<URL>:52380] to cluster <HEX>

If the log file contains multiple of these line per join attempt please try running managedb recover on an existing head node and joining all head nodes back into the cluster one-at-a-time. Re-joining heads that were previously in the cluster may require a --purge argument, i.e. managedb join --purge <IP>

- scyld-install performs its early check to determine if a newer clusterware-installer RPM is available by parsing the appropriate clusterware repo file (typically /etc/yum.repos.d/clusterware.repo) to find the first base_url= line. If there are multiple such lines, i.e., specifying multiple ClusterWare repos, then the cluster administrator should order the repos so that the repo containing the newest RPMs is the first repo in the file.

- Any compute node booting from a head node upgraded to 11.7.0 but using a version of clusterware-node older than 11.2.2 may not successfully send status. Please upgrade the clusterware-node package inside the image to resolve this problem.
CLUSTERWARE 6/7 VS. CURRENT CLUSTERWARE

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpsh 0-127 date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>pdsh -w n[0-127] date</td>
</tr>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl -i n[0-127] exec date</td>
</tr>
</tbody>
</table>

Attempt to remotely execute the `date` command on all nodes in the range of n0 to n127.

bpsh executes on the master node or compute nodes. Authentication is set up out-of-the-box for user root, though it must be set up by the local cluster administrator for non-root users.

pdsh executes on the head node or compute nodes and requires proper user authentication between nodes.

scyld-nodectl executes on any node that has the `clusterware-tools` package installed and is only available to cluster administrators.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpsh -a date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl --up exec date</td>
</tr>
</tbody>
</table>

Perform the same remote execution of the `date` command, although this time just for nodes in the "up" state.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpcp</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>rsync, scp, pdcp</td>
</tr>
</tbody>
</table>

bpcp executes on the master node or compute nodes. Authentication is set up out-of-the-box for user root, though it must be set up by the local cluster administrator for non-root users.

rsync, scp, pdcp execute on the head node or compute nodes, and each require proper user authentication between nodes.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpstat</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpstat -U</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl status --refresh</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpstat --long</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl status --long</td>
</tr>
</tbody>
</table>

bpstat only executes on the master node.
scyld-nodectl executes on any node that has the *clusterware-tools* package installed and is only available to cluster administrators.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>beostat, beostatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>Chronograf GUI</td>
</tr>
</tbody>
</table>

See *Chronograf GUI* for a brief discussion of Chronograf graphical monitoring and a further reference to full Chronograf documentation, and *scyld-nodectl status* for examples of the commandline interface for accessing partial per-node information.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>beomap</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>unavailable</td>
</tr>
</tbody>
</table>

Legacy ClusterWare anticipated users executing multi-threaded jobs by using beomap and beostat to determine the availability and current load-levels of the compute nodes, and then executing MPI-type commands (e.g., mpirun) from the head node to execute a job on selected nodes. Modern ClusterWare users instead rely upon a job scheduler (e.g., Slurm or PBS TORQUE) to optimally manage executing such multi-threaded jobs across multiple nodes.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpctl -S 0-127 -R</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl -i n[0-127] reboot</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>bpctl -S all -R</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl -a reboot</td>
</tr>
</tbody>
</table>

bpctl only executes on the master node.

scyld-nodectl executes on any node that has the *clusterware-tools* package installed and is only available to cluster administrators.

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>beoconfig node</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-nodectl ls -l</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClusterWare 6/7</th>
<th>beoconfig kernelimage</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare</td>
<td>scyld-bootctl -a ls -l</td>
</tr>
</tbody>
</table>

beoconfig only executes on the master node.

scyld-nodectl, scyld-bootctl, scyld-imgctl execute on any node that has the *clusterware-tools* package installed and is only available to cluster administrators.
**11.1 Scyld ClusterWare End-User License Agreement**

Penguin Computing Software End User License Agreement  
Last revised: 1/4/2012

LEGAL NOTICE - READ CAREFULLY BEFORE INSTALLING OR OTHERWISE USING THIS SOFTWARE.

This License Agreement (the "Agreement") is a legal agreement between you, a single legal entity ("End User"), and Penguin Computing ("Penguin"). This Agreement governs your use of the Scyld software defined below (the "Software") and any accompanying written materials (the "Documentation"). You must accept the terms of this Agreement before installing, downloading, accessing or otherwise using such Software and documentation.

By "ACCEPTING" at the end of this Agreement, you are indicating that you have read and understood, and assent to be bound by, the terms of this Agreement. If you are an individual working for a company, then you represent and warrant you have all necessary authority to bind your company to the terms and conditions of this Agreement.

If you do not agree to the terms of the Agreement, you are not granted any rights whatsoever in the Software or Documentation. If you are not willing to be bound by these terms and conditions, do not "ACCEPT" the EULA and remove the software from the system immediately.

END USER LICENSE AGREEMENT FOR SCYLD SOFTWARE

1. Definitions

   • "Clustered System" means a collection of computer systems managed by the Software and for which the total number of computers in the system is specified in the End User purchase order.

   • "Master Node" means the computer or computers designated as the Master Node(s) in the applicable End User purchase order, where the Software is initially installed and from which the total number of computers comprising the Clustered System are managed.

   • "Software" means the software provided under this Agreement by Penguin or its authorized distributor or reseller and for which the applicable End User purchase order specifies: (i) the Software to be licensed by End User; (ii) the Master Node(s); (iii) the license fees; and (iv) the total number of computers in the Clustered System for which End User has paid applicable license fees and the term of the Software usage.

   The Software is comprised of a collection of software components that fall into three (3) categories: (a) "Scyld Unpublished Software" which is owned by Penguin and/or its licensors and licensed under the terms of this Agreement; (b) "Scyld Published Software" which is owned by Penguin and licensed under the GPL version 2 open source license or such other open source license as Penguin may elect in its sole discretion; and (c) "Open Source Software" which is owned by various entities other than Penguin and is subject to the "open source" or "free software" licenses, including but not limited to General Public Licenses (GPL), Lesser General Public License (LPGL), Apache, Artistic, BSD, IBM Public, Mozilla, Omron, Open Group Public License, and Python licenses.
• "Client Connections" means the simultaneous connections between any software client and Software, where a connection creates a persistent and unique Software session per software client.

2. License

• License Grant. Subject to the terms and conditions of this Agreement, Penguin grants to End User a non-exclusive, non-transferable, non-sub licensable right and license to (a) reproduce (solely to download and install), perform, and execute the Scyld Unpublished Software on the specified Master Node(s), solely for End User's internal purposes, and (i) solely for use on the number of computers in the Clustered System and (ii) not to exceed the maximum number of Client Connections for which End User has paid the required license fees for the authorized term; and (b) make one (1) copy of the Scyld Unpublished Software and Documentation for backup and/or archival purposes only.

• Restrictions. The End User shall not, and shall not permit any third party to: (a) sell, lease, license, rent, loan, or otherwise transfer the Scyld Unpublished Software or Documentation, with or without consideration; (b) permit any third party to access or use the Scyld Unpublished Software or Documentation; (c) permit any third party to benefit from the use or functionality of the Scyld Unpublished Software via a timesharing, service bureau, or other arrangement; (d) transfer any of the rights granted to End User under this Agreement; (e) reverse engineer, decompile, or disassemble the Scyld Unpublished Software; (f) modify or create derivative works based upon the Scyld Unpublished Software or Documentation, in whole or in part; (g) reproduce the Scyld Unpublished Software or Documentation, except as expressly permitted in Section 2.1 above; (h) remove, alter, or obscure any proprietary notices or labels on the Scyld Unpublished Software or Documentation; (i) use the Scyld Unpublished Software for any purpose other than expressly permitted in Section 2.1 above; or (j) use the Scyld Unpublished Software for more than the total number of computers, or longer than the authorized term the End User is licensed for pursuant to Section 2.1 above.

• Open Source Software. The Open Source Software and Scyld Published Software are not subject to the terms and conditions of Sections 2.1, 2.2, or 6. Instead, each item of Open Source Software and Scyld Published Software is licensed under the terms of the end-user license that accompanies such Open Source Software and Scyld Published Software, as may be located in the product packaging or available on-line. End User agrees to abide by the applicable license terms for any such Open Source Software and Scyld Published Software. Nothing in this Agreement limits End User's rights under, or grants End User rights that supersede, the terms and conditions of any applicable end user license for the Open Source Software or Scyld Published Software. In particular, nothing in this Agreement restricts End User's right to copy, modify, and distribute any of the Open Source Software and Scyld Published Software that is subject to the terms of the GPL and LGPL. For the Open Source Software and Scyld Published Software subject to the GPL and LGPL, for a period of three (3) years following End User's receipt of the Software, End User may contact Penguin at the address below in writing and request a copy of the source code for such Open Source Software or Scyld Published Software at Penguin's then-current fees.

3. Ownership. The Software is licensed, not sold. Penguin and its licensors retain exclusive ownership of all applicable worldwide copyrights, trade secrets, patents, and all other intellectual property rights throughout the world, and all applications and registrations relating thereto, in and to the Scyld Unpublished Software, Scyld Published Software, and Documentation, and any full or partial copies thereof, including any additions or modifications to the Scyld Unpublished Software and Documentation. End User acknowledges that, except for the limited license rights expressly provided in this Agreement or the Open Source Licenses, as applicable, no right, title, or interest to the intellectual property in the Software or Documentation is provided to End User, and that End User does not obtain any rights, express or implied, in the Software or Documentation. All rights in and to the Software not expressly granted to End User in this Agreement or the Open Source Licenses, as applicable, are expressly reserved to Penguin and its licensors. The “Scyld”, “Scyld Beowulf”, “Scyld ClusterWare” and “Scyld Computing” trademarks and associated logos are the trademarks of Penguin and its affiliates. This Agreement does not permit End User to use the Penguin trademarks.

4. Limited Warranty. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, THE SOFTWARE IS PROVIDED AND LICENSED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESSED OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, TITLE OR FITNESS FOR A PARTICULAR PURPOSE. PENGUIN DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET THE END USER'S REQUIREMENTS
Scyld ClusterWare11 Documentation, Release 11.9.1

5. Limitation of Liability. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, PENGUIN NOR ANY OF ITS AUTHORIZED DISTRIBUTORS, RESELLERS AND LICENSORS WILL BE LIABLE TO END USER FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, LOST PROFITS, LOST OPPORTUNITIES, LOST SAVINGS, OR LOST DATA OR COST OF COVER ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE OR DOCUMENTATION OR ANY SERVICES HEREUNDER, HOWEVER CAUSED ON ANY THEORY OF LIABILITY (INCLUDING CONTRACT, STRICT LIABILITY, OR NEGLIGENCE), EVEN IF PENGUIN, ITS AUTHORIZED DISTRIBUTORS, RESELLERS OR LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT SHALL PENGUIN'S AGGREGATE LIABILITY UNDER THIS AGREEMENT EXCEED THE AMOUNT ACTUALLY PAID BY END USER TO PENGUIN FOR THE SOFTWARE GIVING RISE TO THE CLAIM.

END USER ACKNOWLEDGES THAT THE AGREEMENT REFLECTS AN ADEQUATE AND ACCEPTABLE ALLOCATION OF RISK.

6. Confidential Information. Scyld Unpublished Software and the structure, organization, and code of the Scyld Unpublished Software, including but not limited to the shell scripts of the Scyld Unpublished Software, are confidential and proprietary information ("Confidential Information") of Penguin and/or its licensors. End User agrees to safeguard such Confidential Information with a degree of care commensurate with reasonable standards of industrial security for the protection of trade secrets and proprietary information such that no unauthorized use is made of such information and no disclosure of any part of its contents is made to anyone other than End User's employees whose duties reasonably require such disclosure in order to effectuate the purposes of this Agreement.

7. Term and Termination. This Agreement will remain in effect until terminated or for the authorized term of license usage. End User may terminate this Agreement by removing the Scyld Unpublished Software from End User's computers, ceasing all use thereof, and destroying all copies of the Scyld Unpublished Software and Documentation and certifying to Penguin that it has done so. Any breach of this Agreement by End User will result in the immediate and automatic termination of this Agreement and licenses granted by Penguin herein, and End User shall cease all use of and destroy all copies of the Scyld Unpublished Software and Documentation and certify to Penguin that it has done so. In addition to termination, Penguin will have the right to pursue any other remedies available to it under law or in equity.

8. Export Controls. End User acknowledges and agrees that the Software and Documentation which is the subject of this Agreement may be controlled for export purposes. End User agrees to comply with all United States export laws and regulations including, but not limited to, the United States Export Administration Regulations, International Traffic in Arms Regulations, directives and regulations of the Office of Foreign Asset Control, treaties, Executive Orders, laws, statutes, amendments, and supplement thereto. End User assumes sole responsibility for any required export approval and/or licenses and all related costs and for the violation of any United States export law or regulation.


10. Miscellaneous. This Agreement is the final, complete and exclusive agreement between the parties relating to the Software and Documentation, and supersedes all prior or contemporaneous proposals, representations, understandings, or agreements relating thereto, whether oral or written. Software shall be deemed irrevocably accepted by End User upon installation. No waiver or modification of the Agreement will be valid unless signed by each party. The waiver of a breach of any term hereof will in no way be construed as a waiver of any other term or breach hereof. The headings in this Agreement do not affect its interpretation. End User may not assign or transfer any of its rights or obligations under this Agreement to a third party without the prior written consent of Penguin. Any attempted assignment or transfer in violation of the foregoing will be null and void. If any provision of this Agreement is held by a court of competent jurisdiction to be unenforceable, the remaining
provisions of this Agreement will remain in full force and effect. This Agreement is governed by the laws of the State of California without reference to conflict of laws principles that would require the application of the laws of any other state. The United Nations Convention on Contracts for the International Sale of Goods shall not apply to this Agreement. All disputes arising out of this Agreement will be subject to the exclusive jurisdiction of the state and federal courts located in San Francisco County, California, and the parties agree and submit to the personal and exclusive jurisdiction and venue of these courts. Should you have any questions about this Agreement, or if you desire to contact Penguin, please contact us by mail at Penguin Computing, Inc., 45800 Northport Loop West, Fremont, CA 94538.

11.2 Third-Party License Agreements

GNU General Public License (GPL)

GNU GPL 1: http://www.gnu.org/copyleft/gpl.html
GNU GPL 2: https://opensource.org/licenses/gpl-2.0.php
GNU GPL 3: https://opensource.org/licenses/gpl-3.0.html

Red Hat RHEL License

Visit https://www.redhat.com/en/about/agreements to view the license for each specific location.

Couchbase

Apache 2.0 license.

etcd

Apache 2.0 license.

TICK: Telegraf, InfluxDB, Chronograf, Kapacitor

https://github.com/alerta/tick-stack-config/blob/master/LICENSE

MIT License:

Copyright (c) 2-15-2018 InfluxData, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

jemalloc
Unless otherwise specified, files in the jemalloc source distribution are subject to the following license:

Copyright (C) 2002-2018 Jason Evans <jasone@canonware.com>. All rights reserved.

Copyright (C) 2007-2012 Mozilla Foundation. All rights reserved.

Copyright (C) 2009-2018 Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice(s), this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice(s), this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDER(S) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER(S) BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NFS-Ganesha

The nfs-ganesha*scyld* packages are deprecated and will not be distributed in Scyld ClusterWare 11 after the end of 2021.

Copyright (c) 2005, 2006 Matthew D. Fuller <fullermd@over-yonder.net>

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR(S) "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
This package also contains in tools/lorder a script borrowed from the FreeBSD <http://www.freebsd.org/> operating system which is necessary to build (but not to run) libcidr. See that file for its license terms.

**MPICH (formerly MPICH2)**

https://github.com/pmodels/mpich/blob/master/COPYRIGHT

The following is a notice of limited availability of the code, and disclaimer which must be included in the prologue of the code and in all source listings of the code.

Copyright Notice

- 2002 University of Chicago

Permission is hereby granted to use, reproduce, prepare derivative works, and to redistribute to others. This software was authored by:

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL 60439

(and)

Department of Computer Science, University of Illinois at Urbana-Champaign

**GOVERNMENT LICENSE**

Portions of this material resulted from work developed under a U.S. Government Contract and are subject to the following license: the Government is granted for itself and others acting on its behalf a paid-up, nonexclusive, irrevocable worldwide license in this computer software to reproduce, prepare derivative works, and perform publicly and display publicly.

**DISCLAIMER**

This computer code material was prepared, in part, as an account of work sponsored by an agency of the United States Government. Neither the United States, nor the University of Chicago, nor any of their employees, makes any warranty express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights.

**MUNGE**

GNU GPL 3, from https://slurm.schedmd.com/disclaimer.html

Copyright (C) 2007-2018 Lawrence Livermore National Security, LLC.

Copyright (C) 2002-2007 The Regents of the University of California. UCRL-CODE-155910.

MUNGE is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Additionally for the MUNGE library (libmunge), you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

MUNGE is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License and GNU Lesser General Public License for more details.

**MVAPICH**

http://mvapich.cse.ohio-state.edu/static/media/mvapich/LICENSE-MV2.TXT

Copyright 2003-2018 The Ohio State University.
Portions Copyright 1999-2002 The Regents of the University of California, through Lawrence Berkeley National Laboratory (subject to receipt of any required approvals from U.S. Dept. of Energy). Portions copyright 1993 University of Chicago.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

(1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

(2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

(3) Neither the name of The Ohio State University, the University of California, Lawrence Berkeley National Laboratory, The University of Chicago, Argonne National Laboratory, U.S. Dept. of Energy nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

OpenMPI

https://www.open-mpi.org/community/license.php

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below are in no particular order and generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

Copyright (c) 2004-2010 The Trustees of Indiana University and Indiana University Research and Technology Corporation. All rights reserved.

Copyright (c) 2004-2017 The University of Tennessee and The University of Tennessee Research Foundation. All rights reserved.

Copyright (c) 2004-2010 High Performance Computing Center Stuttgart, University of Stuttgart. All rights reserved.

Copyright (c) 2004-2008 The Regents of the University of California. All rights reserved.

Copyright (c) 2006-2017 Los Alamos National Security, LLC. All rights reserved.

Copyright (c) 2006-2017 Cisco Systems, Inc. All rights reserved.

Copyright (c) 2006-2010 Voltaire, Inc. All rights reserved.

Copyright (c) 2006-2017 Sandia National Laboratories. All rights reserved.

Copyright (c) 2006-2010 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms.

Copyright (c) 2006-2017 The University of Houston. All rights reserved.

Copyright (c) 2006-2009 Myricom, Inc. All rights reserved.
Copyright (c) 2007-2017 UT-Battelle, LLC. All rights reserved.
Copyright (c) 2007-2017 IBM Corporation. All rights reserved.
Copyright (c) 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing Centre, Federal Republic of Germany
Copyright (c) 2005-2008 ZIH, TU Dresden, Federal Republic of Germany
Copyright (c) 2007 Evergrid, Inc. All rights reserved.
Copyright (c) 2008 Chelsio, Inc. All rights reserved.
Copyright (c) 2008-2009 Institut National de Recherche en Informatique. All rights reserved.
Copyright (c) 2007 Lawrence Livermore National Security, LLC. All rights reserved.
Copyright (c) 2007-2017 Mellanox Technologies. All rights reserved.
Copyright (c) 2006-2010 QLogic Corporation. All rights reserved.
Copyright (c) 2008-2017 Oak Ridge National Labs. All rights reserved.
Copyright (c) 2006-2012 Oracle and/or its affiliates. All rights reserved.
Copyright (c) 2009-2015 Bull SAS. All rights reserved.
Copyright (c) 2010 ARM ltd. All rights reserved.
Copyright (c) 2016 ARM, Inc. All rights reserved.
Copyright (c) 2010-2011 Alex Brick. All rights reserved.
Copyright (c) 2012 The University of Wisconsin-La Crosse. All rights reserved.
Copyright (c) 2013-2016 Intel, Inc. All rights reserved.
Copyright (c) 2011-2017 NVIDIA Corporation. All rights reserved.
Copyright (c) 2016 Broadcom Limited. All rights reserved.
Copyright (c) 2011-2017 Fujitsu Limited. All rights reserved.
Copyright (c) 2014-2015 Hewlett-Packard Development Company, LP. All rights reserved.
Copyright (c) 2013-2017 Research Organization for Information Science (RIST). All rights reserved.
Copyright (c) 2017-2018 Amazon.com, Inc. or its affiliates. All Rights reserved.
Copyright (c) 2018 DataDirect Networks. All rights reserved.
Additional copyrights may follow

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

pdsh

"GPL", from https://github.com/chaos/pdsh/blob/master/pdsh.spec

Python 2.2 and beyond

https://docs.python.org/3/license.html

Python versions 2.2 and beyond are distributed under the Python Software Foundation (PSF) license, which PSF deems "GPL-compatible". This license is not the GNU GPL license because PSF allows for distributing a modified version of Python without making those changes open source. The license makes it possible to combine Python with other software that is released under the GPL; the others don't allow that.

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") in source or binary form and its associated documentation.


3. In the event Licensee prepares a derivative work that is based on or incorporates Python or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee agrees to be bound by the terms and conditions of this License Agreement.

**Slurm Workload Manager - SchedMD**

https://slurm.schedmd.com/disclaimer.html

Slurm is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

Slurm is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

**TORQUE Resource Manager**

OpenPBS (Portable Batch System) v2.3 Software License

Copyright (c) 1999-2000 Veridian Information Solutions, Inc. All rights reserved.

For a license to use or redistribute the OpenPBS software under conditions other than those described below, or to purchase support for this software, please contact Veridian Systems, PBS Products Department ("Licensor") at:

<table>
<thead>
<tr>
<th><a href="http://www.OpenPBS.org">www.OpenPBS.org</a></th>
<th>+1 650 967-4675</th>
<th><a href="mailto:sales@OpenPBS.org">sales@OpenPBS.org</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>877 902-4PBS (US toll-free)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This license covers use of the OpenPBS v2.3 software (the "Software") at your site or location, and, for certain users, redistribution of the Software to other sites and locations. Use and redistribution of OpenPBS v2.3 in source and binary forms, with or without modification, are permitted provided that all of the following conditions are met. After December 31, 2001, only conditions 3-6 must be met:

1. Commercial and/or non-commercial use of the Software is permitted provided a current software registration is on file at www.OpenPBS.org. If use of this software contributes to a publication, product, or service, proper attribution must be given; see www.OpenPBS.org/credit.html

2. Redistribution in any form is only permitted for non-commercial, non-profit purposes. There can be no charge for the Software or any software incorporating the Software. Further, there can be no expectation of revenue generated as a consequence of redistributing the Software.

3. Any Redistribution of source code must retain the above copyright notice and the acknowledgment contained in paragraph 6, this list of conditions and the disclaimer contained in paragraph 7.

4. Any Redistribution in binary form must reproduce the above copyright notice and the acknowledgment contained in paragraph 6, this list of conditions and the disclaimer contained in paragraph 7 in the documentation and/or other materials provided with the distribution.

5. Redistributions in any form must be accompanied by information on how to obtain complete source code for the OpenPBS software and any modifications and/or additions to the OpenPBS software. The source code must either be included in the distribution or be available for no more than the cost of distribution plus a nominal fee, and all modifications and additions to the Software must be freely redistributable by any party (including Licensor) without restriction.
6. All advertising materials mentioning features or use of the Software must display the following acknowledgment:

"This product includes software developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and Veridian Information Solutions, Inc. Visit www.OpenPBS.org for OpenPBS software support, products, and information."

7. DISCLAIMER OF WARRANTY

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT ARE EXPRESSLY DISCLAIMED.

IN NO EVENT SHALL VERIDIAN CORPORATION, ITS AFFILIATED COMPANIES, OR THE U.S. GOVERNMENT OR ANY OF ITS AGENCIES BE LIABLE FOR ANY DIRECT OR INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This license will be governed by the laws of the Commonwealth of Virginia, without reference to its choice of law rules.

Addendum

To obtain complete source code for OpenPBS and modifications/additions provided in torque visit www.openpbs.org and/or www.supercluster.org/downloads.
The following is a set of common questions and cross-reference pointers to the answers in the Scyld ClusterWare documentation.

### 12.1 Software Install/Update

**How do I install or update ClusterWare RPMs?**

Always use `scyld-install` to install or update the basic ClusterWare packages. See *Initial Installation of Scyld ClusterWare* and *Updating Scyld ClusterWare*.

For optional ClusterWare packages that are not managed by `scyld-install`, see *Installing Optional ClusterWare Software*.

Use a simple `yum install` or `yum update` to install or update non-ClusterWare base distribution packages.

**How do I install or update software without head node Internet access?**

See *Appendix: Creating Local Repositories without Internet*.

### 12.2 Cluster Management

**What if all ``scyld-*`` commands fail?**

One reason may be the root filesystem is full. See *Head Node Filesystem Is 100% Full*.

Another reason may be the etcd database exceeds its size limit. See *etcd Database Exceeds Size Limit*.

**What are hardware requirements for Scyld ClusterWare?**

See *Required and Recommended Components*.

**How do I add a compute node?**

See *Node Creation with Known MAC address(es)* or *Node Creation with Unknown MAC address(es)*.

**How do I replace a compute node?**

See *Replacing Failed Nodes*.

**How do I configure multiple head nodes?**

See *Managing Multiple Head Nodes*.

**How do I configure a job scheduler, like Slurm, TORQUE, or OpenPBS?**
How do I install and configure OpenMPI?
See *Install OpenMPI, MPICH, and/or MVAPICH*.

How do I keep the host keys consistent across all compute nodes?
See *Compute Node Host Keys*.

How do I change a node name?
See *Node Names and Pools*.

How do I change IP addresses?
See *Changing IP addresses*.

### 12.3 Manipulating Compute Node Images

How do I create an image containing a non-default kernel?
See *Modifying PXEboot Images*.

How do I recreate the default image, boot config, and attributes?
See *Recreating the Default Image*.

How do I create an image containing a non-default base distribution?
See *Appendix: Creating Arbitrary CentOS Images* or *Appendix: Creating Arbitrary RHEL Images*.

How do I delete unused images or boot configurations to free storage space?
See *Deleting unused images and boot configurations*.

### 12.4 Issues with Interacting with Compute Nodes

What if all ``scyld-*`` commands fail?
One reason may be the etcd database exceeding its size limit. See *etcd Database Exceeds Size Limit*.

Why does ``scyld-nodectl -i <NODE_NAME> ssh`` fail?

Why does ``scyld-nodectl -i <NODE_NAME> shutdown`` or ``reboot`` fail?

Why does ``scyld-nodectl -i <NODE_NAME> exec <COMMAND>`` show an SSH Banner?

Does <NODE_NAME> have an SSH Banner? See *Compute Nodes with SSH banner*.
We welcome any reports on errors or difficulties that you may find. We also would like your suggestions on improving this document. Please direct all comments and problems to support@penguincomputing.com.

When writing your email, please be as specific as possible, especially with errors in the text. Please include the chapter and section information. Also, please mention in which version of the manual you found the error. This version is Scyld ClusterWare Release v11.9.1.