1 Cluster Overview and Terminology 1
  1.1 Documentation ............................................................... 1
  1.2 Architectural Overview .................................................... 1
  1.3 Compute Nodes .............................................................. 5

2 Release Notes 7
  2.1 Introduction ................................................................. 7
  2.2 Changelog ......................................................................... 7
  2.3 Known Issues And Workarounds .......................................... 14

3 Administrator’s Guide 16
  3.1 Introduction ...................................................................... 16
  3.2 Required Components ...................................................... 16
  3.3 First Installation of Scyld ClusterWare ................................. 17
    3.3.1 Download the ClusterWare install script and related files ........ 18
    3.3.2 Execute the ClusterWare install script .............................. 19
    3.3.3 Configure additional cluster administrators ..................... 22
  3.4 Subsequent Updates of Scyld ClusterWare ......................... 22
  3.5 Backing up and Restoring ClusterWare ............................... 23
  3.6 Post-Installation Configuration ......................................... 23
    3.6.1 Optionally enable Couchbase auto-failover ....................... 23
    3.6.2 Configure authentication .............................................. 23
    3.6.3 Optionally configure a job scheduler ............................... 24
    To Configure Slurm .............................................................. 25
    To Configure TORQUE ......................................................... 25
    3.6.4 Optionally configure firewall ......................................... 26
    3.6.5 Optionally install OpenMPI .......................................... 27
    3.6.6 Optionally allow IP Forwarding ..................................... 27
    3.6.7 Optionally install additional tools .................................. 28
  3.7 Managing Large Clusters .................................................... 29
    3.7.1 Improve scaling of node booting ................................... 29
  3.8 Boot Configurations and Images ......................................... 29
    3.8.1 The ClusterWare Database .......................................... 29
    3.8.2 Compute Nodes ........................................................... 29
    3.8.3 Boot Configurations .................................................... 30
    3.8.4 Creating PXEboot Images ............................................ 31
    3.8.5 Managing PXEboot Images ......................................... 32
    3.8.6 Capturing and Importing PXEboot Images ....................... 34
    3.8.7 Wrapper scripts .......................................................... 34
    Repos and Distros ............................................................... 35
    Using Archived Releases ..................................................... 35
4 Reference Guide

4.1 Introduction ................................................................. 70
4.2 Important Files on Head Nodes ........................................... 70
  4.2.1 The ~/scyldcw/ Folder .............................................. 70
    auth_tkt.cookie .......................................................... 70
    logs/ ........................................................................... 70
    workspace/ ................................................................. 70
    parse_failures/ ............................................................ 70
  4.2.2 The /opt/scyld/clusterware/ Folder ................................. 71
    /opt/scyld/clusterware/bin/ ............................................ 71
    /opt/scyld/clusterware/conf/ .......................................... 72
    /opt/scyld/env/, modules/, and src/ ................................. 72
    /opt/scyld/clusterware/parse_failures/ ......................... 72
    /opt/scyld/clusterware/storage/ .................................... 72
    /opt/scyld/clusterware/workspace/ ................................. 73
4.3 Compute Node Initialization Scripts .................................... 73
4.4 Database Objects Fields and Attributes ............................... 73
4.5 Reserved Attributes ....................................................... 74
  4.5.1 _boot_config ......................................................... 74
  4.5.2 _boot_rw_layer ....................................................... 74
  4.5.3 _boot_style ......................................................... 75
  4.5.4 _boot_tmpfs_size .................................................. 75
  4.5.5 _disk_cache ......................................................... 76
  4.5.6 _disk_root .......................................................... 76
  4.5.7 _ipxe_sanboot ....................................................... 76
  4.5.8 _hostname ........................................................... 77
  4.5.9 _hosts ................................................................. 77
  4.5.10 _ips ................................................................. 77
  4.5.11 _no_boot ........................................................... 77
  4.5.12 _preferred_head .................................................. 78
  4.5.13 _remote_user ...................................................... 78
  4.5.14 _status_secs ...................................................... 78
4.6 Introduction to Tools ..................................................... 78
  4.6.1 --all and --ids ................................................... 79
  4.6.2 --config .......................................................... 79
  4.6.3 --base-url and --user ........................................... 80
  4.6.4 --show-uids,--human,--json,--pretty/--no-pretty .......... 80
  4.6.5 --csv,--table,--fields ......................................... 80
4.7 Common Subcommand Actions ............................................ 80
  4.7.1 list (ls) ............................................................ 80
  4.7.2 create (mk) ........................................................ 80
  4.7.3 clone (cp) ........................................................ 81
  4.7.4 update (up) ........................................................ 81
  4.7.5 replace (re) ....................................................... 81
  4.7.6 delete (rm) ......................................................... 81
4.8 The --content argument .................................................. 81
4.9 Files in database objects ................................................. 82
4.10 Scyld ClusterWare Administrator Tools .................................. 83
  4.10.1 scyld-add-boot-config ......................................... 83
  4.10.2 scyld-adminctl ................................................... 84
CHAPTER ONE

CLUSTER OVERVIEW AND TERMINOLOGY

1.1 Documentation

The Scyld ClusterWare documentation set consists of:

- This *Cluster Overview and Terminology* containing a general overview of a ClusterWare cluster architecture and terminology.
- The *Release Notes* containing release-specific details.
- The *Administrator’s Guide* describing how to install, configure, use, maintain, and update the cluster.
- The *Reference Guide* describing in greater detail the commands to manage the cluster.

These product guides are 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`.

1.2 Architectural Overview

A minimal ClusterWare cluster consists of a *head node* (aka *master node*) and one or more *compute nodes* (aka *slave nodes*), all connected via a *private cluster network*. User applications generally execute on the compute nodes, and 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 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. NFS, 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 the compute nodes’ BIOS, providing for command-line or programmatic access to the compute nodes at a more basic hardware level, which allows 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 (who are not cluster administrators) connect to the cluster portal to create virtual login node(s), and users connect to login nodes to build or install applications, download data into the cluster’s data storage, submit jobs to a job scheduler,
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:

```
High-Availability HPC ClusterWare 11
```

```
1.3 Compute Nodes
```

A compute node’s BIOS can be configured to “pxeboot” or to boot from local storage, e.g., a harddrive.

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, but 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, along with a reference to a root file system image. This image 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 PXEboot, 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 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, an NFS root, or an image cached on a local harddrive, and then switch the node’s root from the initramfs to this final root image. Except 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 using an overlayfs (see https://www.kernel.org/doc/Documentation/filesystems/overlayfs.txt).
2.1 Introduction

Scyld ClusterWare Release v11.1.0 (released March 16, 2020) is the latest update to Scyld ClusterWare 11.

A Scyld ClusterWare v11.1.0 head node expects to execute in a Red Hat RHEL7 or CentOS7 base distribution environment.


For the most up-to-date product documentation and other helpful information, visit the Penguin Computing Support Portal.

Before continuing, make sure you are reading the most recent Scyld ClusterWare Release Notes, which 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 Scyld ClusterWare yum repository of RPMs that you are about to install.

The Administrator’s Guide describes how to perform an initial installation (see First Installation of Scyld ClusterWare) and how to update an existing installation (see Subsequent Updates of Scyld ClusterWare).

2.2 Changelog

- 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 -lIanplus 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 Appendix: Creating Nodes With Kickstart) and Debian preseed (see Appendix: 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-scylld 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-scylld 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-scylld 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-scyld, slurm-scyld, singularity-scyld, openmpi3.1, openmpi3.0, openmpi2.1, openmpi2.0, openmpi1.10, and openmpi1.8.

The torque-scyld and slurm-scyld packages are now split into three packages for each job scheduler. For example, torque-scyld (which requires torque-scyld-libs) installs on the server, and torque-scyld-node (which requires torque-scyld-libs) gets installed into a node image by the sched-helper script. (See Optionally configure a job scheduler.)

singularity-scyld 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 $PATH and $LD_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 Optionally configure a job scheduler.
  – 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*.
  – Support for booting “disked” compute nodes. See the *Administrator’s Guide* and *Reference Guide* for details.
  – 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-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.
  – Scyld ClusterWare 11 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.

# 2.2. Changelog
* 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.

### 2.3 Known Issues And Workarounds

The following are known issues of significance with the latest version of Scyld ClusterWare 11 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.

- If the head node root filesystem fills, subsequent `scyld-` commands will fail to execute and will terminate with:

```
OSError: [Errno 28] No space left on device.
```

A workaround that removes ephemeral files and likely allows subsequent `scyld-` commands to execute successfully is to stop clusterware, then `sudo rm -fr /opt/scyld/clusterware/workspace/*`, then restart clusterware. Then perhaps delete images to free more space. Execute `scyld-imgctl ls -l` to see the full information for all images, including their sizes, then use `scyld-imgctl -i <ImageName> rm` to delete selected unnecessary images.

After some root filesystem space is freed, if you continue to see `influxdb` messages in `/var/log/messages` complaining "no space left on device", then try `sudo systemctl restart influxdb`.

- When using a TORQUE or Slurm job scheduler (see Optionally configure a job scheduler), 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-scyld-node` or `NODE=n0 slurm-scyld-node`. Or to restart on all nodes: `torque-scyld.setup cluster-restart` or `slurm-scyld.setup cluster-restart`.

Ideally, compute node images are updated using `torque-scyld.setup update-image` or `slurm-scyld.setup update-image`, which installs the TORQUE/Slurm config file in the image and enables the appropriate service at node startup.

- Rarely, the `scyld-modimg` command can fail to upload a modified image, resulting in an error message similar to:

```
ERROR: Uploaded image size does not match provided size: 414948185 != 463875291
```

ERROR: Remote file may need to be re-uploaded.

Before reporting this error please retry just the upload: `scyld-modimg -i ImageName --upload`

If that retry completes successfully, then the previous error message can be safely ignored.

- The head node’s `GRUB_CMDLINE_LINUX` in `/etc/default/grub` must not contain `ipv6.disable=1`; otherwise, `memcached` 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 `NetworkManager-config-server` package includes a `NetworkManager.conf` config file with an enabled “no-auto-default” setting. That is incompatible with ClusterWare 11 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 `NetworkManager-config-server` package from compute node images.
- The `scyld-clusterctl repos create` command has a `urls=` argument that names 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*. 
3.1 Introduction

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

This Scyld ClusterWare Administrator’s 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 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.

3.2 Required Components

Scyld ClusterWare 11 head nodes are expected to use x86_64 processors running a Red Hat RHEL7 or CentOS7 base distribution.

**Important:** ClusterWare 11 head nodes currently require a 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 11 compute nodes handle DHCP responses on the private cluster network (boot-net) 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.

Required head node components:

- x86_64 processor(s), with a minimum of four cores recommended.
- 4GB RAM.
- One Ethernet controller connecting to the private cluster network that interconnects the head node(s) and all compute nodes, required.
- One Ethernet controller connecting to the Internet, recommended.
- 8GB storage specifically for ClusterWare software, its database, log files, and one compute node image set. Add 2GB for each additional image set. Add additional storage for the base distribution, job scheduler software and logging, user data, etc.

Significantly more CPU, RAM, and storage resources may be required for desired performance, and multiple Ethernet or other network controllers are common.

We recommend a minimum of two head nodes - one functioning as the production head node, and the other as a development head node used to test software updates and configuration changes prior to updating the production node to the validated final updates. A High Availability (“HA”) cluster should have a minimum of three “production” head nodes.

We recommend employing virtual servers, instead of “bare metal” servers, for ease of management of head nodes, login nodes, and other servers other than compute nodes. Compute nodes are generally bare metal servers for optimal performance. 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.

Although a single “bare metal” hypervisor can host multiple virtual servers, a High Availability cluster should have critical virtual servers each hosted by their own unique hypervisor to avoid having a failure of a single hypervisor cause a simultaneous failure of all its hosted virtual machines.

See Architectural Overview for details.

The `/etc/yum.repos.d/` configuration files should support `yum` access to RHEL7 or CentOS7 RPMs, since installing ClusterWare will likely need to install additional base distribution packages as dependencies. 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.

### 3.3 First Installation of Scyld ClusterWare

The Scyld ClusterWare 11 `scyld-install` script installs the necessary packages from the Scyld ClusterWare 11 yum repositories, and installs dependency packages as needed from the RHEL7 or CentOS7 yum repositories.

**Important:** Do not install Scyld ClusterWare 11 as an upgrade to an existing ClusterWare 6 or 7 installation. Instead, install 11 on a non-ClusterWare system, and ideally on a virtual server. (See Required Components.)

**Important:** 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-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:

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

**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.

### 3.3.1 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 11 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.
- Move that downloaded file to /tmp/.
- Fix the file's permissions and ownership for the yum command, and then move that file to /etc/yum.repos.d/. Once in place, install the clusterware-installer package, which contains the scyld-install script.

For example:

```
cd /tmp
# Expecting the desired clusterware.repo file to now reside in /tmp
sudo chmod 644 clusterware.repo
sudo chown root:root clusterware.repo
sudo cp -a clusterware.repo /etc/yum.repos.d/clusterware.repo
sudo yum install clusterware-installer
```

Alternatively, if Penguin Computing has transmitted (e.g., by email) a custom clusterware.repo file to you, then as described above, move that file to /etc/yum.repos.d/ and install the clusterware-installer RPM, then execute the scyld-install script contained in that RPM.

Less commonly, download the scyld-install script directly from the Penguin Computing yum repository. When executed, that script queries the user for the appropriate userid/password authentication that Penguin Computing has provided to you, and uses that to create an appropriate /etc/yum.repos.d/clusterware.repo.

For example:

```
cd /tmp
wget https://updates.penguincomputing.com/clusterware/11/installer/scyld-install
# or download with *curl* or equivalent
chmod +x scyld-install
```
3.3.2 Execute the ClusterWare install script

The `scyld-install` script requires write access to the current working directory, so consider executing the script from `/tmp` or `/var/tmp`.

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.

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

```sh
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>`
  - 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>`.

- `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.

- `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:

```sh
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.

Now execute scyld-install, for example using the cluster-conf created above:

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

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 tftp 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 Administrator's Guide section Securing the Cluster.

Then uses systemd to enable and start firewalld and opens ports for communication between head nodes as required by Couchbase. See Services, Ports, Protocols for details.

Once the ports are open, the script 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.
- nfs-ganesha: The user-space NFSv4 service for (optional) NFS booting of compute nodes.
- 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, NFSv4, 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 script.
- Creates an initial simple boot image DefaultImage using the scyld-add-boot-config command.
- 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 Boot Configurations and Images 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 the initial DefaultImage, but rather cloning that basic image into a new image that gets further modified, or creating new images from scratch.

**Important:** See Post-Installation Configuration for additional optional cluster configuration procedures, e.g., installing and configuring a job scheduler, installing and configuring one of the MPI family software stacks. We recommend that this additional software installation and configuration be done.

**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 RHEL7 system, the installer will still initially create a CentOS7 compute node image as the DefaultImage. If after installation a system administrator wishes instead to create a compute node image based on RHEL, they should use the scyld-clusterctl repos tool as described in Repos and Distros. The administrator will need to populate an additional field in the repos object, rhel_entitlement. This field should be set to a valid RHEL entitlement string, as used in the file names in the /etc/pki/entitlement/ directory. This string will be used when creating the .pem and -key.pem files in the image.

When a RHEL compute node boots, it looks for the file /etc/clusterware/rhel-vars.sh that must define the RHEL_USER and RHEL_PASS variables. The booting node uses these values to attempt to register (or re-register) using the subscription-manager tool. On a successful first-time registration, the node transmits the resulting consumerid to its parent head node, who in turn stores that value into the node’s _rhel_consumerid attribute in the ClusterWare database.

If a specific Pool ID is required, add the attribute _rhel_poolid.
3.3.3 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 the `scyld-install` script. 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` convenience script is provided by the `clusterware-tools` package and the contents of the `settings.ini` file are discussed in the Reference Guide. Running that script 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-*` programs. The `settings.ini` file generated by the `scyld-install` script will also 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.

### 3.4 Subsequent Updates of Scyld ClusterWare

**Important:** Even though a simple `yum update` will update Scyld ClusterWare packages, Penguin Computing strongly recommends using the `scyld-install` script to perform Scyld ClusterWare updates.

The `scyld-install` script first determines if a newer `clusterware-installer` package is available, and if so will update `clusterware-installer` and restart the `scyld-install` script.

**Note:** Executing `scyld-install` with no arguments presupposes that ClusterWare is not yet installed. If ClusterWare is currently installed, then the script 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 script asks for positive confirmation that the user does intend to **install** ClusterWare as a fresh install.

The `scyld-install` script just updates ClusterWare software and any other packages it depends upon. After ClusterWare is updated, you can execute `yum check-update` to view the non-Scyld ClusterWare packages that are available for updating, then use `yum update` to 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 /tmp/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 a new installation, running `scyld-install --clear` will also invoke the `scyld-add-boot-config` script to create the default boot image.

### 3.5 Backing up and Restoring 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 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 the Reference Guide.
```

### 3.6 Post-Installation Configuration

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

#### 3.6.1 Optionally enable Couchbase auto-failover

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

#### 3.6.2 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 through the `scyld-adminctl` command:
scyld-adminctl create name=username

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-uids 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-uids --users admin1,tester --image SlurmImage

# push uid with an additional group:
sync-uids --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-uids --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.

### 3.6.3 Optionally configure a job scheduler

The default Scyld ClusterWare installation includes support for two job scheduler packages: Slurm and TORQUE. Slurm and TORQUE are optional packages and both can coexist on a scheduler server, which may or may not be a ClusterWare head node. However, if Slurm and TORQUE are installed on the same server, then only one at a time should be enabled and executing on that given server.

For TORQUE documentation see [https://www.adaptivecomputing.com/support/documentation-index/torque-resource-manager-documentation](https://www.adaptivecomputing.com/support/documentation-index/torque-resource-manager-documentation). For Slurm documentation see [https://slurm.schedmd.com](https://slurm.schedmd.com). See [https://slurm.schedmd.com/rosetta.pdf](https://slurm.schedmd.com/rosetta.pdf) for a discussion of the differences between TORQUE and Slurm, and [https://slurm.schedmd.com/faq.html#torque](https://slurm.schedmd.com/faq.html#torque) for useful information about how to transition from PBS or TORQUE to Slurm.

All nodes in the Slurm or TORQUE 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 can itself 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
```

The slurm-scyld.setup and torque-scyld.setup scripts described below should always be executed on the scheduler server by a ClusterWare administrator who also has sudo privileges on that server.
To Configure Slurm

Install the Slurm software on the job scheduler server:

```
sudo yum install slurm-scyld
```

Installing and configuring a job scheduler requires making changes to the compute nodes. When using image-based compute nodes we suggest first cloning the DefaultImage or creating a new image, leaving the DefaultImage as a basic known-functional pristine image. For example, to set up nodes n0 through n3 as Slurm nodes:

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

When these nodes reboot, they will use the SlurmImage.

Then use a helper script to complete the initialization:

```
slurm-scyld.setup init
```

Restart the Slurm server software and node services, install the Slurm packages on compute nodes, push the server config to the compute nodes, sync the Slurm user uid/gid, and restart the compute node services with:

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

Note that these changes are not persistent across reboots until the Slurm configuration is added to the compute node image(s). For example, if the image for Slurm nodes is named SlurmImage:

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

Now you can reboot a Slurm node to bring it into active management by the Slurm job scheduler.

The cluster administrator can add or remove nodes by manually editing /etc/slurm/slurm.conf, or can add nodes to that file by executing:

```
slurm-scyld.setup update-nodes
```

Any such changes to slurm.conf must be pushed to the up nodes and added to SlurmImage by reexecuting:

```
slurm-scyld.setup cluster-restart
slurm-scyld.setup update-image SlurmImage
```

Inject user and group IDs into compute nodes using the sync-uids script. See Configure authentication for details.

To view slurm status on the server and compute nodes:

```
# view Slurm status
slurm-scyld.setup status
```

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

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

Each Slurm user must setup the PATH and LD_LIBRARY_PATH environment variables to properly access the Slurm commands. This is done automatically for users who login when the slurm service is running and the pbs_server is not 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.

3.6. Post-Installation Configuration
To Configure TORQUE

Alternatively, to use TORQUE as the job scheduler, install the TORQUE software on the job scheduler server:

```
sudo yum install torque-scylld
```

Installing and configuring a job scheduler requires making changes to the compute nodes. If using image-based compute nodes, then we suggest first cloning the `DefaultImage` or creating a new image, which leaves the `DefaultImage` as a basic known-functional pristine image. For example, to set up nodes n0 through n3 as TORQUE nodes:

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

When these nodes reboot, they will use the `TorqueImage`.

Then use a helper script to complete the initialization:

```
torque-scylld.setup init
```

Restart the TORQUE server software and node services, install the TORQUE packages on compute nodes, push the server config to the compute nodes, sync the TORQUE user uid/gid, and restart the compute node services with:

```
torque-scylld.setup cluster-restart
```

Note that these changes are not persistent across reboots until the TORQUE configuration is added to the compute node image(s). For example, if the image for TORQUE nodes is named `TorqueImage`:

```
torque-scylld.setup update-image TorqueImage
```

Now you can reboot a TORQUE node to bring it into active management by the TORQUE job scheduler.

The cluster administrator can add or remove nodes by manually editing `/etc/torque/torque.conf`, or can add nodes to that file by executing:

```
torque-scylld.setup update-nodes
```

Any such changes to `torque.conf` must be pushed to the `up` nodes and added to `TorqueImage` by reexecuting:

```
torque-scylld.setup cluster-restart
torque-scylld.setup update-image TorqueImage
```

Inject user and group IDs into compute nodes using the `sync-uids` script. See *Configure authentication* for details.

To view status:

```
# view TORQUE status
torque-scylld.setup status
```

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

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

### 3.6.4 Optionally configure firewall

If you are not using the `torque-scylld` or `slurm-scylld` 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.
3.6.5 Optionally install OpenMPI

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 and on any other server where OpenMPI applications are built.

View the available ClusterWare versions using `yum list | egrep "openmpi|mpich|mvapich" | egrep scyld`. For example, `openmpi3.1` is built from the latest available Open Source version, e.g., 3.1.4. The basic `install openmpi3.1` installs the default GNU libraries, binary executables, buildable source code for various example programs, and man pages. The `openmpi3.1-gnu` package is equivalent to `openmpi3.1`. Alternatively, `yum install openmpi3.1-intel` installs those same things, except it includes libraries and executables that were built using the Intel compiler suite. These files can co-exist with the base package built by the GNU compiler suite. Similarly, you can alternatively (or additionally) install `openmpi3.1-pgi` with libraries and executables built using the PGI compiler suite. The `openmpi3.1-psm2` packages are intended for use with QLogic Infiniband controllers.

You can add OpenMPI environment variables to a user’s `.bash_profile` or `.bashrc` file, e.g., add `module load openmpi/intel/3.1.4` 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 share the `/home` storage.

For OpenMPI, consistent user uid/gid and passphrase-less key-based access is also 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`:

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

If using OpenMPI 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 Optionally configure firewall for details.

3.6.6 Optionally allow 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, IPv4 Packet Forwarding must be enabled, and forwarding must be allowed by the head node’s firewall.
To enable IPv4 Packet Forwarding:

```
firewall-cmd --zone=external --add-masquerade
```

When IP forwarding is confirmed to work, make the current rule permanent so that the rules will survive a reboot:

```
firewall-cmd --runtime-to-permanent
```

Appropriate routing for compute nodes can be modified in the compute node image(s) (see `scyld-modimg` tool in the Reference Guide). Limited changes may also require modifying the DHCP configuration template `/opt/scyld/clusterware-iscdhcp/dhcpd.conf.template`.

### 3.6.7 Optionally install additional tools

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

**/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:

```
[nosadmin@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:

```
[nosadmin@head1 /]$ scyld-nodectl --json -i n0 ls -l | jq ".0.mac"
"52:54:00:a6:f3:3c"
```

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

Scyld ClusterWare 11 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.

3.7.1 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.

3.8 Boot Configurations and Images

3.8.1 The ClusterWare Database

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 Couchbase 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, but 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 are assigned based on the 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.

3.8.2 Compute Nodes

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, and currently supports 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`
Given a proper `power_uri`, the system can turn node power on and off when prompted by administrator commands. Without this setting the system can still reboot or shutdown nodes through `reboot` and `shutdown` commands executed on them. In a production system computes nodes are generally bare-metal nodes and can be controlled via `ipmitool`. For such nodes please provide a `power_uri` such as:

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

Any head node that is asked to perform a `power on`, `power off`, `power cycle`, `hard shutdown`, or `hard reboot` on a node with this `power_uri` will communicate with the compute node’s BMC located at 172.45.88.1 using the admin/password credentials.

**Important:** If for any reason only a certain machine can execute `ipmitool` to control a node, the cluster administrator can add a server to the node’s `power_uri` and allow ssh access from the head nodes to that server. The `ipmitool` will be executed on the remote server via ssh: `ipmi://remote_server/admin:password@172.45.88.1`.

The soft `shutdown` and `reboot` commands do not use the `power_uri` but instead ssh to the compute node and execute the `/usr/sbin/shutdown` command with appropriate arguments. Because `ssh` banners are sent through standard errors, these commands can cause the system to think that the remote command failed and will cause the system to attempt hard shutdowns or resets. To fix this, please place a copy of the banner into `/opt/scyld/clusterware/banner.txt` and restart the ClusterWare service. Similarly, you can put the same content into `/opt/scyld/clusterware-tools/banner.txt` to filter the banner content from `scyld-nodectl` exec output.

Other node fields include `type` (currently set to “compute”, but updates to Scyld ClusterWare will add additional options), `groups`, and `attributes`. These 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.

### 3.8.3 Boot Configurations

The `scyld-install` script creates a basic `boot configuration` that is associated with all compute nodes. After installation, the cluster administrator can customize that configuration and create additional boot configurations and associated compute node images.

Administrators can modify configuration fields using the `scyld-bootctl` tool. For example, one can change the name and description of the newly create 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 `@`, 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`. Nodes are assigned to boot configurations based on their `_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, but 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`, `nfs`, `disked`, `live`, and `next`. The default `rwram` instructs the system to download the compressed image into compute node RAM where the `mount_rootfs` script unpacks it into `/sysroot` during the boot process. Alternatively, when the `roram` option is provided, the script downloads a squashfs image from the head node into node RAM, combines this with a writable tmpfs via overlayfs, and boots the combined image. The `iscsi` and `nfs` options cause the node to mount a read-only image via iSCSI and NFS respectively. Similar to the downloaded squashfs image, a writeable tmpfs is applied with overlayfs prior to booting the node.

The `disked` option instructs a node to employ a node-local persistent cache to retain downloaded images and thereby avoid the need to download images at boot time. This option is likely only useful for clusters with very large node counts and where these nodes have local storage. See Appendix: Booting From Local Storage Cache for details.

The `live` option 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 Anaconda-based install. Given access to the node console, a cluster administrator can manually install to the local disk thereby generating a kickstart file. This kickstart file can then 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 kickstart locally installed nodes can be found in the======@jhawkes need a link to somewhere=======

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`.

3.8.4 Creating PXEboot Images

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, this tool can be run later with locally accessible repositories as described in Appendix: Creating Local Repositories. Once the DefaultImage is created, the cluster administrator can use `scyld-modimg` to modify it directly, though a safer approach is to use `scyld-modimg` to clone the DefaultImage to new name, and then use `scyld-modimg` to modify that cloned image, leaving the DefaultImage untouched. See Managing PXEboot Images for details.

The administrator can also create a new image from nothing. When doing that, consider the source of the components (aka packages) for that new image. A distro ties together a `package_manager`, a `release`, and a list of `repos`, and each repo is a repository that contains various packages. The initial default distro is CentOS 7: CentOS version 7, using `package_manager yum`, and downloading packages from the single `repos “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
```
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

[admin@virthead]$ scyld-clusterctl --get-distro
Default distro: CentOS
```

Create an image using this CentOS6 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 commands 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.

### 3.8.5 Managing PXEboot Images

Once you create a new image, you can install additional RPMs:

```
scyld-modimg -i NewImg --install openmpi3.1
```

Or manually customize this image, including installing or removing RPMs, by operating on the image inside a chroot:
scyld-modimg -i NewImg --chroot

Or combine commands, ending inside a chroot:
scyld-modimg --create --set-name NewImg --install openmpi3.1 --chroot

After manipulations inside a chroot, you exit the chroot, specifying to keep the changes, to replace the local image, to upload the image, and to replace the remote image.

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 NewImg --install openmpi3.1 --chroot \  --no-discard --overwrite --upload

To update the kernel in the image, first install it into the image:
scyld-modimg -i NewImg --install kernel-3.10.0-957.27.2.el7.x86_64 \  --no-discard --overwrite --upload

Next, create a new initramfs file on the head node to match that kernel:
scyld-mkramfs -i NewImg --kver 3.10.0-957.27.2.el7.x86_64 \  --output initramfs-3.10.0-957.27.2.el7.x86_64.img

Now extract the kernel file from NewImg, and create a new boot image that contains that kernel and that initramfs:
# Extract into the current working directory.
scyld-modimg -i NewImg --copyout /boot/vmlinuz-3.10.0-957.27.2.el7.x86_64

# Identify the kernel boot file and initramfs file in the current directory.
scyld-bootctl create name=NewBoot kernel=@vmlinuz-3.10.0-957.27.2.el7.x86_64 \  initramfs=@initramfs-3.10.0-957.27.2.el7.x86_64.img \  image=NewImg

Finally, set the _boot_config attribute for specific nodes, or for all nodes, to use this new boot image:
scyld-nodectl -i n0 set _boot_config=NewBoot

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 two formats, cwtar and cwsquash. The first format is the default on systems where the head nodes are not running with SELinux in enforcing mode and primarily consists of a gzipped tar archive with appropriate extended attributes preserved. This cwtar format is directly downloaded by nodes booting with the rwram boot style and is faster to create and modify than the alternative cwsquash format.

The cwsquash format consists of a squashfs image offset inside a pseudo-disk image. This format is suitable for exporting via iSCSI or mounting and exporting via NFS. Head nodes will dynamically repackage images uploaded as cwtar as cwsquash, but due to security restrictions, this repackaging is not done when SELinux is in enforcing mode on the head node. Note that this repackaging may take several minutes during which time nodes will only be able to boot using the rwram style. The status of this repackaging process can be seen in the scyld-imgctl ls -l output.

The default format is determined at head node installation time based on the SELinux status, though may be changed through the scyld-clusterctl command. See the Reference Guide for command details. Note that if the SELinux status of a head node is changed, this default format will need to be manually updated.
Small homogeneous clusters may use a single node image, but larger clusters that include compute nodes with differing hardware will require additional customization. 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, but can also create additional boot configurations and root file systems.

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-iscdhcp/dhcpd.conf.template` file, then the cluster administrator needs to edit that file to remove the default `option routers <GATEWAY>;` line, and 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.

### 3.8.6 Capturing and Importing PXEboot Images

Cluster administrators can also modify the files on a booted compute node and use the `scyld-imgctl` command to capture those changes into the image. To avoid overwriting the original image, create an empty image before capturing the node contents. For example, to capture node n0:

```
scyld-imgctl create name=NewImage
scyld-imgctl -iNewImage capture --node n0
```

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-imgctl` command that then uploads it to the head node, replacing any 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 shared storage is not accidentally captured.

The returned content is `cwtar` formatted and cannot be directly uploaded to SELinux enforcing head nodes. Capturing an image in this configuration requires an additional step:

```
scyld-imgctl create name=NewImage
scyld-imgctl -iNewImage capture --node n0 --save temp.cwtar
scyld-modimg --import temp.cwtar --set-name NewImage --upload
```

The `scyld-modimg` tool will perform the necessary repacking before uploading. This same `scyld-modimg --import` command can also be used to import images exported from this or other clusters:

```
scyld-imgctl -iDefaultImage download content
scyld-modimg --import ./DefaultImage.content --set-name NewImage --upload
```

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.

### 3.8.7 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 will ask 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.

### Repos and Distros

One of the earliest steps in this script is to run the `scyld-clusterctl` tool to define a distro prior to creating the first image. In ClusterWare `scyld-modimg` can only create images based on defined distros. Distros associate one or more repos together with their package manager (`yum` is the only package manager currently supported) and an optional release string. If no release string is provided, then any supplied URLs should not include “$releasever”, as the variable will not be defined during image creation. On a CentOS 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
```

Along 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 containing blocks referring to the system’s ClusterWare repository, as well as the repos referenced by the distro database object.

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 `repos create` command, or multiple repos to the `distros create` command. Distros can also be imported from existing yum repo files:

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

The `import` command 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.

Additional details of the `scyld-clusterctl repos` and `distros` commands, as well as the `scyld-modimg` command that is used to actually create and modify images, can be found in the *Reference Guide*.

### 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. 7.6.1810 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. 7.6.1810 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:
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...
elapsed: 0:00:01.0
ERROR: One or more repositories in the newly created image are invalid. This can happen when installing older versions of Linux distributions such as CentOS. Please consult the Administrator’s Guide for more information.
WARNING: The command will be retried with unknown repositories disabled.
elapsed: 0:02:39.9
fixing SELinux file labels... done.
step completed in 0:04:13.6
```

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.
Using Releases on ISO

Some distributions are distributed in ISO form. To create an image from an ISO, mount the ISO on a system, use Apache or another HTTP server to serve the mount point, and use the local server’s URL as the url for the repo.

A quick example using a file server available at 192.168.32.10:

```bash
sudo mount -o loop CentOS-7-x86_64-DVD-1810.iso /mnt/
cd /mnt/
python -m SimpleHTTPServer  # serve local directory on 0.0.0.0:8000 until stopped
```

On the head node:

```bash
scyld-clusterctl repos create name=centos_1810 urls=http://192.168.32.10:8000/
scyld-clusterctl distros create name=centos repos=centos_1810 release=''
scyld-modimg --create centos
```

In this case the centos_1810 repo should be removed after image creation and replaced with a more permanent repo location.

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
```

3.8.8 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 downloaded-tarball.tgz has been downloaded and unpacked on the head node as `/usr/src/downloaded-tarball/`. 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
```

then enter the new UpdatedImage in a chroot environment, now as user `root`:

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

Inside the chroot, copy the unpacked tarball from the outside as user `admin` (and you are asked for the `admin` password) to inside the chroot (owned by `root`), e.g.,

```bash
scp -r admin@localhost:/usr/src/downloaded-tarball /tmp
```

Now examine the `/tmp/downloaded-tarball/` contents, which will typically contain a script that manages the tarball-specific software installation.
There are various 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 manually `yum install` it, naming the specific kernel version, e.g.,

```
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 install scripts use the `uname` command by default to determine the kernel version against which to link a new kernel module. When the `uname` command executes inside a chroot, it actually reports the kernel that is installed on the machine which has executed the `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 `chroot`-executing machine. Accordingly, the 3rd-party install script should support an optional argument that specifies the intended kernel version, e.g.,

```
install-script -k 3.10.0-957.27.2.el7.x86_64.
```

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

- Some 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 installation 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 package(s) that replace the ClusterWare package(s). In other cases you can reinstall these ClusterWare package(s) after the install script successfully completes.

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

### 3.9 Kickstarting

The *Boot Configurations and Images* 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.

#### 3.9.1 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:

```
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 -iCentOS ls -l
Repos
CentOS
iso
  checksum: 6b71f450513c92e2ab6ea663a69989f8f3680f01
  mtime: 2020-03-05 17:19:41 UTC (0:01:39 ago)
  size: 4.3 GiB (4664066048 bytes)
```
The mounted contents of the ISO are accessible through each head node’s webserver at /repo/<REPONAME>/repo/. 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 licences.

The entire ISO can also be downloaded using the scyld-clusterctl command:

```
scyld-clusterctl repos -iCentOS download iso
```

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

### 3.9.2 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 -iCentOS_iso ls -l
Boot Configurations
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= cmdline:

```
scyld-bootctl -iCentOS_iso update cmdline=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

### 3.9. Kickstarting
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 -iCentOS_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 any other boot configuration through the update action. Updating them
with an empty string will reset them back to the detected paths:

```
scyld-bootctl -iCentOS update kernel= initramfs=
```

### 3.9.3 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]>
```

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 inform-
ation 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
managable hunks that can then be included into a top-level kickstart file, much like a C or C++ #include.

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

### 3.10 Booting Diskfull Compute Nodes

In addition to booting diskless clients, ClusterWare can also integrate with existing so-called “diskfull” compute nodes
that boot from full installations on local disk drives. (See Appendix: Creating Nodes With Kickstart and Appendix:
Creating Nodes With Preseed for examples.)
Add locally booted compute nodes (by MAC) to the system using the same mechanisms as diskless nodes. Further, if the compute node has a static IP address, then that address should be in the ClusterWare DHCP range and should also be set in the node database entry. For example:

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

On the compute node the cluster administrator should additionally install the `clusterware-node` package and its dependencies. Since the system will not be using the initramfs provided by a ClusterWare boot configuration for diskless nodes, the administrator will also need to create a configuration file at `/opt/scyld/clusterware-node/etc/node.sh` on the node to instruct the node how to communicate with a head node. This file must define the `base_url` for a head node and the `iface` network interface name that is assigned the MAC address used in the ClusterWare database entry for the node. For example:

```
[admin@virthead]$ cat /opt/scyld/clusterware-node/etc/node.sh
base_url=http://head-01/api/v1
iface=eth0
```

Once the database entry has been created, the packages are installed, and the `node.sh` configuration file is in place, the compute node needs to be rebooted. When the compute node boots, it begins sending status information to the head node.

The newly booted node can be controlled through the customary `scyld-nodectl reboot`, `shutdown`, and `exec` commands. To support `--hard` mode, the administrator must configure the nodes `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 `Database Objects Fields and Attributes` for details.

### 3.11 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.

#### 3.11.1 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` argument 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`. 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.:
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.

3.11.2 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 11 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 11 DHCP server which does not recognize the incoming MAC will by default ignore the incoming DHCP client request. To override this default:

```
scyld-clusterctl --set-accept-nodes True
```

and then any head node that shares the same database will add that new MAC to the shared ClusterWare 11 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 11 cluster, then the cluster administrator should again reenable the default functionality with:

```
scyld-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 update commands, e.g.,

```
scyld-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.

3.11.3 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
```

### 3.11.4 Node Name Resolution

During the installation process a package called `clusterware-dnsmasq` is installed which provides resolution services for head node names. Similar to the `clusterware-iscdhcp`, this package depends on a standard OS provided service, but 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 be 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`, but 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 to map node ids to IP addresses.

### 3.11.5 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 /
```
The tool will pass the command, i.e. `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:

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

This sort of 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 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 16 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 `settings.ini` or can be set in the `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. For example:

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

directly in the option:

```
scyld-nodectl -i n0 exec --stdin=’Hello World’ dd of=/root/output.txt
```

or piped to the command (optionally using redirection on the compute node):

```
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:

```
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:

```
[admin@virthead]$ scyld-nodectl -i[n0-1] exec ls -l
n0: total 4
  n0: -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 a file path via the
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 pscp 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.

3.11.6 Node Attributes

The names and uses of the fields associated with each database object are fixed, but nodes may be augmented with attribute lists for more flexible management. These attribute lists are stored in the attributes field of the 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 it belongs to, 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, 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, but 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: []
```

3.11. Interacting with Compute Nodes
By default no attributes are defined at the node level, but all nodes inherit the _boot_config value from the “DefaultAttribs” group. If an administrator created a new boot configuration (possibly by using the scyld-add-boot-config script mentioned earlier) and called it “AlternateBoot”, then she could assign a single node to that configuration using the scyld-nodectl tool:

```bash
csyld-nodectl -i n0 set _boot_config=AlternateBoot
```

Examining the same nodes after this change would show:

```bash
[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 would create a group and assign nodes to that group:

```bash
csyld-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:

```bash
csyld-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 group nodes, 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.

## 3.12 Securing the Cluster

This Administrator’s Guide section discusses cluster security issues that are exclusive to Scyld ClusterWare 11. 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 RHEL7 functionalities for logging and auditing access to nodes and storage, and managing SELinux.

### 3.12.1 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 installation process 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 cluster administrator should change the `plugins.auth` variable from “dummy” to “appauth”. This plugin executes the command defined in the `appauth.app_path` variable as user `root`. An implementation of that command is provided by `/opt/scyld/clusterware/bin/pam_authenticator`. This provided 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, but 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.

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
```

### 3.12.2 Changing the Database Password

The ClusterWare installation process configures the Couchbase 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.

To change this password, use the Couchbase console available on every head node on port 8091. 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 for details.

Once this password is changed within Couchbase, the `database.admin_pass` variable must be changed and the ClusterWare service restarted on each head node.

### 3.12.3 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:

```
scyld-adminctl -iadmin update keys='ssh-rsa AAAAB3NzaC1yc2EAAAADA....'
```

Cluster administrators are also welcome to add SSH keys to compute node images in small private clusters, but 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.

### 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 DefaultImage, an administrator can run:

```
scyld-modimg -iDefaultImage --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.

### 3.12.4 Encrypting Communications

By default the administrative tools communicate with the head node via HTTP, but 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_*_.conf. The Apache VirtualHost definition can be found in httpd_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.

### 3.12.5 SELinux

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

Scyld ClusterWare 11 supports SELinux on the head nodes and compute nodes.

**SELinux On Compute Nodes**

For 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 /etc/selinux/config configuration file. For details please refer to appropriate distro-provided documentation.

In addition to the default targeted SELinux policy provided by CentOS and RHEL, 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 in 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. On exiting the chroot the system will automatically relabel the file system based on the policy referenced in /etc/selinux/config. 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:

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

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

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

When creating boot configuration for an MLS enabled image please be aware that the enforcing=0 command line option is not used by the MLS system and will be ignored. Another caveat is 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.

Fully configuring a cluster for MLS requires significant work including labelling 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.

### SELinux On Head Nodes

On head nodes, SELinux is detected to be in “enforcing” mode at both installation and service run time. During installation there is no difference in what packages are installed, but the installer will note that NFS Ganesha cannot be used to serve compute node images. Because NFS support for extended attributes is inadequate for serving SELinux contexts, the NFS Ganesha integration provided by ClusterWare is disabled on head nodes with SELinux “enforcing” enabled. For similar permission reasons the repacking daemon thread is similarly disabled.

When the scyld-modimg command connects to an SELinux enforcing head node, it will use the cwsquash squashfs-based image format. This can be slower to pack, but allows for more flexibility in boot options once the file is uploaded to the head node. Any cwtar formatted files uploaded to an SELinux “enforcing” head node can be used for booting compute nodes via the rwram boot style, but not for roram, iscsi, or nfs booting because those formats require the cwsquash format. Head nodes with SELinux in “enforcing” mode cannot convert between different image formats.

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:

```bash
systemctl restart clusterware
```

Any image in cwsquash format will automatically be repacked as cwtar when the service is started on a non-enforcing SELinux system.
3.13 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.

3.13.1 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:

```
# 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
    state: down

  n2: {}
```

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 send by each compute node to its parent head node once every 10 seconds, but this value can be adjusted through 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 but never sent status information informing the head node that it was up. 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 but has never been booted. Additional node status can be viewed with `scyld-nodectl status --l` and 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` display can be unwieldy, so the status command defaults to the `--summary` argument. 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. 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)*.

### 3.13.2 Chronograf GUI

Access the Chronograf GUI either through the ClusterWare GUI’s *Monitoring* tab (see *Graphical Interface*) or point your browser at a head node (or at any node on which the `clusterware-tools` package has been installed, thus allowing it access to the ClusterWare database), e.g., `http://localhost/chronograf` if running on a head node, or `http://cw11headnode/chronograf` to access another head node named `cw11headnode`.

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

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

After setup, 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:

![Chronograf Configuration](image)

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:

3.13. Monitoring the Status of the Cluster
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:
3.14 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:8080 if running on a head node, or http://cw11headnode:8080 to access another head node named cw11headnode. You first see a login screen. If no cluster administrators have yet been declared (see the scyld-adminctl command), then login as root:

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.

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

After adding administrator joe:
Click on the top-level Nodes pulldown menu to expose two choices. The first choice Node List displays all known nodes and their summary state. You can enter various qualifiers in the Selector text box to narrow down the list of nodes.

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:

![Image of Node Images]

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

![Image of Boot Configurations]

And click on a specific Boot Configuration to see details:

![Image of DefaultBoot configuration]

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

### 3.15 Managing Multiple Head Nodes

ClusterWare supports optional active-active(-active,...) configurations of multiple cooperating head nodes to provide High Availability in the face of head node failures. All head nodes within a cluster share a single replicated database implemented using Couchbase. This configuration allows any head node to provider 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.

#### 3.15.1 Adding A Head Node

After installing the first head node as described in *First Installation of Scyld ClusterWare*, additional head nodes are installed and added through the same `scyld-install` tool:

```
scyld-install --yum-repo clusterware.repo --join <IP_OF_EXISTING_HEAD>
```
The database password will be required to add the additional head node and can be found in the `database.admin_pass` variable in `/opt/scyld/clusterware/conf/base.ini`.

As when installing additional head nodes, the administrator needs to provide the ClusterWare yum repository via one of the supported methods: a filename passed as an argument, a file already placed in `/etc/yum.repos.d/`, or with a serial number that is passed on the command line or when asked by the installer. However, a cluster configuration file is not required when adding and joining head nodes because those settings are already loaded into the replicated cluster database.

**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 `Optionally allow 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 RHEL7 default is `chronyd` (found in the `chrony` package), although `ntpd` (found in the `ntp` package) continues to be available.

**Important:** After `scyld-install` completes to add a head node, you **must** restart the `clusterware` service on all head nodes.

After a successful `--join`, all connected head nodes will replicate the functionality of the others, thereby allowing each head node to provide DHCP and boot services to any compute node. Further, any head node can be contacted to alter the system configuration.

Subsequent head node software updates are also accomplished by executing `scyld-install`. We recommend that all cooperating head nodes update to a common ClusterWare 11 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 Changelog`.

### Configuring Support for Database Failover

When planning a multi-head cluster, a cluster administrator should allocate three or more head nodes. In this configuration, if one head node fails, the Couchbase service on the other two 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 recovery scenarios are managed by the cluster administrator interacting with the Couchbase 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 console is available online on the Couchbase website: `https://docs.couchbase.com/home/index.html`. ClusterWare currently uses Couchbase version 5.1.

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.
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 defaults to `/opt/scyld/clusterware/storage/`.

Whenever a ClusterWare head node is asked for a file such as a kernel, it first checks for the file in its storage directory. The expected file size and checksum are retrieved from the database, and if the local file has the correct size and checksum, it will be provided. However, if the file is missing or incorrect, then the head node will attempt to retrieve the correct file from a peer. Note that 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 file is discarded if that checksum is incorrect. Contacted peers will not attempt to download the file from other peers to avoid a completely missing file triggering 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 running `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 are deleted from the system, the deleting head node will remove the underlying file(s) from its local storage. That head node will also temporarily move the files’ 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, that head node will not be able to delete the local copy. When the service is later restarted, it will see the file that 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, but cautious administrators may wish to keep these renamed files until they confirm all node images and boot configurations are working as expected.

### 3.15.2 Booting With Multiple Head Nodes

Since all head nodes are connected to the private cluster network, any compute node issuing a 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 “Booting 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, then the node begins to send its periodic status information to its parent. 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 and sending status messages to the former second, now first, head node on the list. It is possible to force the compute nodes to re-download the head node list once the original head node is responding again.

### 3.15.3 Removing A Head Node

A list of connected head nodes can be seen with:

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

To remove a target head node that is running, `ssh` to it and execute:

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

To remove a target head node that is shut down, `ssh` to another head node in the cluster and execute:

```
/opt/scyld/clusterware/bin/managedb eject <IP_ADDRESS>
```

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

After a head node leaves or is ejected from the system, the REST API on that head node may still operate until the `clusterware` service is restarted. This is expected behavior, although it can be confusing. To avoid relying on a head node that may stop working whenever it reboots, we suggest that cluster administrators stop the clusterware service on the head node shortly after that node leaves the cluster. Once a head node is removed from a cluster, it can be re-added to this or another cluster by retarting the `clusterware` service and perfoming a `--join`:

```
scyld-install --join <IP_OF_EXISTING_HEAD>
```

**Important:** After `scyld-install` completes, you **must** restart the `clusterware` service on all head nodes.

### 3.16 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.

#### 3.16.1 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.
3.17 Troubleshooting ClusterWare

The /var/log/clusterware/ folder contains several log files that may help diagnose problems. Additionally, the Couchbase service may have useful information in its logs. Please see their website https://docs.couchbase.com/home/index.html for details. The Couchbase console is available on port 8091 of any ClusterWare head node.

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 Couchbase 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_293aafdf3f635448e9aaa76fc998ebc0c.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 configuration file that is distributed in the clusterware rpm.

Note: If the local cluster administrator modifies /etc/logrotate.d/clusterware, then a subsequent update of the 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.

3.17.1 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 -i00:11:22:33:44:55 1s -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 b7412619fe28424ebf7c5f3474009d.

Booting node (MAC=52:54:00:a6:f3:3c) as rwram using boot config f72edc4388964cd919346dfc21cd2c.

If there are no Booting 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.
Verify that ClusterWare services are running. Check the status of ClusterWare services with the command:
```
systemctl status clusterware
```
Restart ClusterWare services from the command line using:
```
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”.

### 3.17.2 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 -icentos8_repo update iso=@CentOS-8.1.1911-x86_64-dvd1.iso
```

### 3.17.3 Failing 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 module to initrd. For example:

```bash
scyld-bootctl -i DefaultBoot download initramfs
scyld-mkramfs --image DefaultImage --kver 3.10.0-957.27.2.el7.x86_64 \
   --modules virtio_blk --output junk.ramfs
scyld-bootctl -i DefaultBoot up initramfs=junk.ramfs
```

### 3.17.4 IP Forwarding

If IP forwarding is desired, but is still not working, then check `/etc/sysctl.conf` to see if it is disabled.

Check for the line “net.ipv4.ip_forward = 1”. If the value is set to 0 (zero) instead of 1, then IP forwarding will be disabled.

### 3.17.5 Soft Power Control Failures

If the `scyld-nodectl reboot` or `shutdown` commands always fall back on hard power control, then this can point to a couple 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 and compute node `power_ureis` in *Boot Configurations and Images*.

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.

### 3.17.6 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`.

### 3.17.7 Finding Further Information

If you encounter a problem installing your Scyld cluster and find that this Administrator’s Guide cannot help you, the following are sources for more information:

- The *Release Notes* contains a Changelog detailing per-release specifics, and a section detailing known problems and workarounds.
- The *Reference Guide* contains a technical reference to Scyld ClusterWare commands.
3.17.8 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 -I` to view this snapshot locally, otherwise execute `scyld-sysinfo` to produce the compressed tarball that can be emailed or otherwise communicated to Penguin Computing.

3.18 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.

3.18.1 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](http://openipmi.sourceforge.net), which includes links to documentation and downloads.

3.19 Updating Software On Your Cluster

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, product documentation, and `Release Notes`.

The `Release Notes` contains a Changelog detailing 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.

**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.

### 3.20 Services, Ports, Protocols

#### 3.20.1 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.

#### 3.20.2 Couchbase

The replicated configuration key/value store is implemented by Couchbase, with 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:

```plaintext
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
```

#### 3.20.3 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*.

#### 3.20.4 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.

#### 3.20.5 NFS Ganesha

NFS Ganesha optionally serves root filesystems to compute nodes across NFS, with a systemd service name **nfs-ganesha** and using port 2049. The log file is */var/log/ganesha.log*. 

Chapter 3. Administrator’s Guide
3.20.6 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`.

3.20.7 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/`.

3.20.8 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`.
4.1 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.

4.2 Important Files on Head Nodes

4.2.1 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.

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 clusterware11]$ bin/scyld-nodectl -vv ls
DEBUG: Starting new HTTP connection (1): localhost:6543
DEBUG: http://localhost:6543 "GET /whoami HTTP/1.1" 200 59
DEBUG: Loaded authentication cookie from previous run.
DEBUG: http://localhost:6543 "GET /nodes HTTP/1.1" 200 63
DEBUG: http://localhost:6543 "POST /mux HTTP/1.1" 200 580
Nodes
   n0
DEBUG: Saved authentication cookie instead of logging out.
```

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.
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.

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.

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.

4.2.2 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.

Tools located in the /opt/scyld/clusterware/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 Administrator’s Guide and the randomize_ini script executed during initial installation.
The `conf/` folder contains the principal configuration files for ClusterWare REST API. In that folder the `httpd_*` 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 multihead configurations. Because of this, future releases may move selected variables from the `base.ini` file into the Couchbase 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:

```
/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.

The `env/`, `modules/`, and `src/` folders contain the Python virtual environment, including the libraries required by the `scyld-*` and other tools.

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 `Backing up and Restoring ClusterWare` for additional discussion of backup up the database contents.
The REST service running on each head node requires a location to mount or unpack root file system images in order to then export their contents to compute nodes via iSCSI or NFS. 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 depending on the size and number of root file systems, 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.

4.3 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 `Boot Configurations and Images`, 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 Couchbase 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:

```ini
[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.

4.4 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 \texttt{mac} with the node’s MAC address, \texttt{name} with the node’s alphanumeric name, and \texttt{power_uri} with a value denoting how to communicate via ipmi to that node. For example, the command \texttt{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 \texttt{attributes}, where an attribute is a collection of one or more attribute name-value pairs. Attribute names that begin with an underscore “\_” are called \texttt{reserved attributes} or \texttt{system attributes}. The cluster administrator uses the \texttt{set} action to change an attribute value. See the following section \texttt{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 \texttt{/etc/clusterware/attributes.ini} and find that node’s attributes. On the node there are helper functions in \texttt{/opt/scyld/clusterware-node/functions.sh} for reading attributes, specifically the function \texttt{attribute_value}.

\section{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 \texttt{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 \texttt{Node Attributes} for more details.

The remainder of this section is a list of system attributes describing their use and allowed values.

\subsection{\texttt{_boot_config}}

Default: none

Values: boot configuration identifier

Depends: none

The \texttt{_boot_config} attribute defines what boot configuration a given node should should use. For a detailed discussion of boot configurations and other database objects, please see \texttt{Boot Configurations and Images}.

A boot configuration identifier may be a, possibly truncated, UID or a boot configuration name.

\subsection{\texttt{_boot_rw_layer}}

Default: overlayfs

Values: overlayfs, rwtab
Depends: _boot_style == roram, nfs, 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”:

```plaintext
READONLY=yes
```

Second, the kernel cmdline in the appropriate boot configuration must include “ro”:

```plaintext
cmdline: enforcing=1 ro
```

### 4.5.3 _boot_style

**Default:** rwram

**Values:** rwram, roram, nfs, iscsi, disked, next, sanboot

**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 nfs and iscsi options use 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. Both of these approaches require a writable overlay, as the images may be shared between multiple nodes.

Note that NFS root file systems are not supported when SELinux is enforcing on the head node(s).

With the disked option, the node boots with images read from local storage. See :ref:`boot_style_disked` 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 command for more details.

### 4.5.4 _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, nfs, or iscsi style, this attribute controls the size of the writable overlay.
4.5.5 _disk_cache

Default: none
Values: local partition
Depends: ignored unless _boot_style == disked

The _disk_cache attribute identifies 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 and formatted with an Ext2/3/4 file system. If the specified location exists, a compute node will keep a copy of the downloaded image, and during later boots will compare the checksum of that file with the expected checksum provided by the head node to avoid unnecessary downloads. If the named partition does not exist, then an error will be logged, although the node will download the image to RAM and still boot. If a cache is present but no _disk_root is provided and 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!

Much like entries in /etc/fstab, partitions can be identified by device file, UUID, or LABEL.

4.5.6 _disk_root

Default: none
Values: local partition
Depends: ignored unless _boot_style == disked

During node booting, the root image will be unpacked into the partition named in the _disk_root attribute. This process will delete the contents of the named partition before unpacking the disk image. If the named partition does not exist, then an error will be logged, although the node will still boot using the image unpacked into RAM instead of on disk.

Important: Any data in the partition specified as a _disk_root will be destroyed at boot time!

Partitions should be identified by device file, UUID, or LABEL.

4.5.7 _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.
4.5.8 _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.

4.5.9 _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.

4.5.10 _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=eno0=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.

4.5.11 _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.

4.5. Reserved Attributes
4.5.12 _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.

4.5.13 _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.

4.5.14 _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.

4.6 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:
4.6.1 --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.

4.6.2 --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
- the --config specified path
- command line arguments

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.
4.6.3 --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.

4.6.4 --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.

4.6.5 --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.

4.7 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.

4.7.1 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.

4.7.2 create (mk)

Create a new object using name-value pairs provided either on the command line or passed using the --content argument described below.
4.7.3 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.

4.7.4 update (up)

Modify existing objects altering individual fields in name-value pairs or a --content argument described below.

4.7.5 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.

4.7.6 delete (rm)

Delete objects.

4.8 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:

```bash
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:

```bash
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:

```json
[
    { "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 }]
```

INI:

```ini
```
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.

### 4.9 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
 TestBoot
 initramfs
  chksum: a1161aa52b98287a3eac4677193c141a3648ebc
  mtime: 2019-02-09 21:13:08 UTC (0:00:52 ago)
  size: 20.0 MiB (20961579 bytes)
  uid: d247e4aa1fde4ac3853e78c0f7683947
 kernel
  chksum: 5a464d2a82839dac21c0fb7350d9cb0d055f8fed
  mtime: 2019-02-09 21:08:34 UTC (0:05:26 ago)
  size: 6.3 MiB (6639808 bytes)
  uid: fc96da5531b94038b38d7ef662b34947
  last_modified: 2019-02-09 21:08:34 UTC (0:05:26 ago)
  name: TestBoot
```
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.

### 4.10 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.

#### 4.10.1 scyld-add-boot-config

**NAME**

scyld-add-boot-config – Add boot information to configuration.

**SYNOPSIS**

```
```

**DESCRIPTION**

Add boot information to cluster configuration.

**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.

- **--auto-first**
  - If there are currently no attribute groups, then automatically build an attribute group referencing a new boot configuration referencing a new image.

- **--no-nodes**
  - Skip asking about assigning the attribute group to nodes.

**ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS**

- **--base-url URL**
  - Specify the base URL of the ClusterWare REST API.

- **-u**, **--user USER[:PASSWD]**
  - Use connection user USER and optional colon-separated password PASSWD.

**EXAMPLES**

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 --auto-first
```

**RETURN VALUES**

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.

### 4.10.2 scyld-adminctl

**NAME**

`scyld-adminctl` – Query and modify administrators for the cluster.

**SYNOPSIS**

```bash
```

**DESCRIPTION**

Query or modify administrators for the cluster. This tool does not control the details of authentication. For that please consult Securing the Cluster in the Administrator’s Guide.

**ACTIONS**

- `list (ls)` List information about administrator(s).
- `create (mk) name=NEWNAME` Add an administrator NEWNAME.
- `clone (cp) name=NEWNAME` Copy administrator to new identifier NEWNAME.
- `update (up)` Modify administrator fields.
- `replace (re)` Replace all administrator fields.
- `delete (rm)` Delete administrators(s).

**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.
- `--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]` Use connection user USER and optional colon-separated password PASSWD.

**FORMATTING ARGUMENTS**
--human Format the output for readability (default).
--json Format the output as JSON.
--pretty Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty Opposite of –pretty.

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.

4.10.3 scyld-attribctl

NAME
scyld-attribctl – Query and modify attribute groups for the cluster.

SYNOPSIS

DESCRIPTION
Query and modify attribute groups for the cluster.

ACTIONS
list (ls) List information about attribute group(s).
create (mk) name=NEWNAME Add an attribute group NEWNAME.
clone (cp) name=NEWNAME Copy attribute group to new identifier NEWNAME.
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.

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.
--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]
Use connection user USER and optional colon-separated password PASSWD.

FORMATTING ARGUMENTS
--human
Format the output for readability (default).
--json
Format the output as JSON.
--pretty
Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty
Opposite of –pretty.

EXAMPLES
scyld-attribctl create name=iScsiCompute
Add a new attribute group.
scyld-attribctl -i iScsiCompute 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.

4.10.4 scyld-bootctl

NAME
scyld-bootctl – Query and modify boot configurations for the cluster.

SYNOPSIS

DESCRIPTION
Query and modify boot configurations for the cluster.

ACTIONS
list (ls) List information about boot configurations.
create (mk) Add a boot configuration.
clone (cp) Copy boot configurations to new identifiers.
update (up) Modify boot configuration fields.
replace (re)  Replace all boot configuration fields.
delete (rm)  Delete boot configurations.
download  Download named files (any of “initramfs”, “kernel”).

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.
--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]  Use connection user USER and optional colon-separated password PASSWD.

FORMATTING ARGUMENTS

--human       Format the output for readability (default).
--json         Format the output as JSON.
--pretty      Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty   Opposite of –pretty.

EXAMPLES

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.

scyld-bootctl -iFed29Boot download kernel

Download the kernel previously uploaded to the Fed29Boot configuration.

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.

scyld-bootctl -i DefaultBoot ls -l

Display details about the DefaultBoot configuration.

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.
RETURN VALUES
Upon successful completion, `scyld-bootctl` returns 0. On failure, an error message is printed to `stderr` and `scyld-bootctl` returns 1.

4.10.5 scyld-cluster-conf

NAME
`sclyd-cluster-conf` — load or save the cluster configuration file.

SYNOPSIS

DESCRIPTION
Save the cluster configuration file, or load a new file.

ACTIONS
load CLUSTER_CONFIG Load CLUSTER_CONFIG as the new configuration file, optionally loading only nodes.

save CLUSTER_CONFIG Save the current configuration file to CLUSTER_CONFIG.

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.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS
--base-url URL Specify the base URL of the ClusterWare REST API.

-u, --user USER[:PASSWD] Use connection user USER and optional colon-separated password PASSWD.

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:

```
1 10.10.24.100/24 10.10.24.199
node 08:00:27:A2:3F:C9
node
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:

```
1 10.10.24.100/24
node 08:00:27:A2:3F:C9
node
node 08:00:27:A2:E4:A2

2 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**

```
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.
4.10.6 scyld-clusterctl

NAME
scyld-clusterctl – Tool for manipulating global cluster settings.

SYNOPSIS

DESCRIPTION
Query and modify global cluster setting.

CLUSTER-WIDE SETTINGS AND COMMANDS
- --set-group ATTRIB_GROUP Set the default attribute group.
- --get-accept-nodes Get whether unknown nodes should be automatically added.
- --set-accept-nodes BOOLEAN Set whether unknown nodes should be automatically added (BOOLEAN=T) or not (=F).
- --distros List distros supported by the head node(s).
- --get-distro Get the current default distro.
- --set-distro DISTRO Set the default distro.
- --image-formats List image formats supported by the head node(s).
- --get-image-format Get the current default image format.
- --set-image-format FORMAT Set the default image format.

DATABASE QUERYING AND MODIFICATION, SELECT A CLASS OF DATABASE OBJECT
repos Manipulate available repos.
distros Manipulate available distros.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS
- --base-url URL Specify the base URL of the ClusterWare REST API.
- -u, --user USER [:PASSWD] Use connection user USER and optional colon-separated password PASSWD.

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.

FORMATTING ARGUMENTS
- --human Format the output for readability (default).
- --json Format the output as JSON.
--pretty
   Indent JSON or XML output, and substitute human readable output for other
   formats.

--no-pretty
   Opposite of –pretty.

EXAMPLES

RETURN VALUES

Upon successful completion, scyld-clusterctl returns 0. On failure, an error message is printed to stderr and scyld-
clusterctl returns 1.

4.10.7 scyld-imgctl

NAME

scyld-imgctl – Query and modify images for compute nodes.

SYNOPSIS

   USER[:PASSWD]] [--human | --json] [--pretty | --no-pretty] [--show-uIds] 
   [-a | -i ADMINS] {list,ls, create,mk, clone,cp, update,up, replace,re, 
   delete,rm, download, stat}

DESCRIPTION

Query or modify images for compute node provisioning.

ACTIONS

list (ls)  List information about node images.

create (mk)  Add node image.

clone (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  Download named files (any of “content”).

stat  Print the recorded file stats for an image.

OPTIONAL ARGUMENTS

-h, --help
   Print the command usage message and exit. If -h is the first option, all other
   options will be ignored. If -h is not the first option, the other options will be
   parsed up to the -h option, but no action will be taken.

-v, --verbose
   Increase verbosity.

-q, --quiet
   Decrease verbosity.

-c, --config CONFIG
   Specify a client configuration file.

--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

4.10. Scyld ClusterWare Administrator Tools 91
--base-url URL Specify the base URL of the ClusterWare REST API.
-u, --user USER[:PASSWD] Use connection user USER and optional colon-separated password PASSWD.

FORMATTING ARGUMENTS
--human Format the output for readability (default).
--json Format the output as JSON.
--pretty Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty Opposite of --pretty.

EXAMPLES
scyld-imgctl -i DefaultImage download content
Download the previously uploaded image named DefaultImage.
scyld-imgctl -i DefaultImage stat
Print the last modified time and size of the previously uploaded image.
scyld-imgctl -i DefaultImage clone name=NewImage
Clone the DefaultImage to a new NewImage.

RETURN VALUES
Upon successful completion, scyld-imgctl returns 0. On failure, an error message is printed to stderr and scyld-imgctl returns 1.

4.10.8 scyld-install

NAME
scyld-install – Install or update ClusterWare.

SYNOPSIS
[-l | --load DATABASE_FILE] [-s | --save DATABASE_FILE] --without-files
[-u | --update] [--database-passwd PASSWD]

DESCRIPTION
Tool to install ClusterWare and perform initial basic configuration, and to update an existing installation.

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.

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

--clear  Clear the ClusterWare database. DELETES ALL IMAGES AND CONFIGURATION!

--clear-all  A last resort more severe reset than --clear that removes all ClusterWare (except for clusterware-installer) and couchbase-server RPMs, and deletes /opt/couchbase, /opt/scyld/clusterware*, and /var/log/clusterware, then reinstalls ClusterWare.

--no-tools  Don’t install the ClusterWare tools. The default is to install the tools.

--join HEAD_IP  Join this head node to existing head node HEAD_IP. After, you must restart clusterware on all head nodes.

--skip-version-check  Use this installer without first checking online for a newer version.

-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-passwd PASSWD  Specify the root password used for the server maintaining the Couchbase database.

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.

4.10.9 scyld-mkramfs

NAME

scyld-mkramfs – Tool to create an initial root file system image.

SYNOPSIS

scyld-mkramfs [-h] [OPTION] ... -o, --output PATHNAME

DESCRIPTION

Create an initial root file system image from either the local system or an existing image. When passed the --image option this tool leverage the scyld-modimg tool to unpack the specified image, install the clusterware-tools package, and execute scyld-mkramfs within the unpacked image.

The kernel version will default to the currently running kernel according to uname -r, but can be overridden using the --kver argument.

Positional arguments
--kver VER Specify the kernel version. Defaults to current running kernel.
--image IMGID Use scyld-modimg to execute this command inside an image.
--no-selinux Do not include SELinux support.
--modules Space-separated list of additional modules to include.
--output, -o FILE Where to write the initramfs.

OPTIONAL ARGUMENTS

-h, --help Print usage message and exit. Ignore trailing args, parse and ignore preceding args.

EXAMPLES

scyld-mkramfs --image DefaultImage --kver 3.10.0-957.5.1.el7.x86_64 --output fresh.ramfs

Create an initramfs based on version 3.10.0-957.5.1.el7.x86_64 of the kernel from the DefaultImage. Note that that kernel version must be installed inside the image for this command to work.

RETURN VALUES

Upon successful completion, scyld-mkramfs returns 0. On failure, an error message is printed to stderr and scyld-mkramfs returns 1.

4.10.10 scyld-modimg

NAME

scyld-modimg – Tool for manipulating image contents.

SYNOPSIS


DESCRIPTION

Manipulate image contents.

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.
--all Select all local images (default).
-i, --image IMAGE Select an image by uid.
--show-uids Do not try to make the output more human readable.
--image-format FMT Format for new images. Detected from the server, but can be forced to cwtar or cwsquash.
--shell SHELL Select the shell to use in the image for the mutation operations (default /bin/bash).
--freshen Discard any cached changes and download a fresh copy of the image.
--download-only PATHNAME Download a new local copy and then exit. If the optional PATHNAME is provided, then it will be overwritten; and if none is provided, then any cached changes will be lost.
--overwrite Keep the same UID after modifications, and overwrite any existing image on upload.
--no-overwrite Opposite of –overwrite.
--upload Upload the final version.
--no-upload Opposite of –upload.
--discard Discard image changes.
--no-discard Opposite of –discard.

IMAGE MUTATIONS

The following steps can be performed on the selected image. Any failures will terminate execution.

–create [DISTRO] Create a new image from scratch.

--install PKGS Install packages into the image.

–update [PKGS] Update specified packages in the image. (default=ALL)

--uninstall PKGS Uninstall packages from the image.

--query PKGS Query package versions from the image.

--unpack TARGZ Unpack a tar.gz file into the image.

--execute CMD Execute a command in the unpacked image.

--chroot Chroot into the unpacked image to allow for manual modifications.

–copyout SRC DEST Copy files or directories SRC out of the image to destination DEST.

–copyin SRC DEST Copy files or directories from SRC into the image as DEST.

--delete Delete the selected image(s) from the local cache.

--mount PATH Unpack the image into PATH and bind-mount various folders as if preparing for –chroot. After an image is mounted it can be customized by other commands, such as ansible, before being repacked.

--unmount PATH Repack the image from a previously mounted PATH.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

--base-url URL Specify the base URL of the ClusterWare REST API.

-u, –user USER [: PASSWD] Use connection user USER and optional colon-separated password PASSWD.

FORMATTING ARGUMENTS

4.10. Scyld ClusterWare Administrator Tools
--human Format the output for readability (default).
--json Format the output as JSON.
--pretty Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty Opposite of --pretty.

EXAMPLES

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.

4.10.11 scyld-nodectl

NAME

scyld-nodectl – Query and modify nodes for the cluster.

SYNOPSIS


DESCRIPTION

Query or modify cluster nodes.

ACTIONS

list (ls) List information about nodes.
create (mk) Add a node.
clone (cp) Copy node to new identifiers.
update (up) Modify node fields.
replace (re) Replace all node fields.
del (rm) Delete nodes.
set Set attribute value.
clear Delete attribute values.
join Append groups to the node group lists.
leave Remove groups from the node group lists.
status [-refresh] Check node status. If the optional --refresh is added, then the command continues to execute and the displayed status gets updated if and when anything changes.
hardware Check node hardware.
boot (poweron) Boot nodes.
reboot [-soft] [-hard] Reboot node 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 reboot; if after a short delay the node does not appear to begin a reboot, then perform a hard power cycle.
**shutdown** [--soft] [--hard]  Shutdown node 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.

**power**  Check or control the node power state through the plugin defined by the node’s power_uri, usually ipmi. The possible arguments are on, off, cycle, or status and correspond to the those options to ipmitool.

**sol**  Start a serial-over-lan connection using the local ipmitool.

**ssh**  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.

**script**  Execute scripts from the clusterware-node package on the selected machines. The available scripts can be listed by running `scyld-nodectl script list`. Generally these scripts are used at node boot time and do not need to be called directly.

**exec**  Execute command on nodes. Note that when running exec, the scyld-nodectl command will pass stdin to the remote command. A string or file can also be provided as stdin using the --stdin argument. This also means that 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 by including the exact text in /opt/scyld/clusterware-tools/banner.txt. The text will still print when running the scyld-nodectl ssh command, though will be filtered out when using scyld-nodectl exec, thereby making results of parallel commands much easier to read. Increasing the command verbosity with --vv will stop the banner filtering.

Commands executed on multiple nodes within a single run of scyld-nodectl exec will be executed 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. In order to make such output easier to parse, the exec action takes two optional arguments. The --grouped argument will buffer data on the client side to ensure no output is interleaved. The --in-order option performs the same way, but additionally ensures that output from each node appears in node index order, i.e. node n1 output will appear before node n2 output.

**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.
- **--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 to query or modify.
- **--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] Use connection user USER and optional colon-separated password PASSWD.

**FORMATTING ARGUMENTS**

--human Format the output for readability (default).
--json Format the output as JSON.
--pretty Indent JSON or XML output, and substitute human readable output for other formats.
--no-pretty Opposite of –pretty.

**EXAMPLES**

scyld-nodectl status

Shows the status of each node.

scyld-nodectl list

List all node names.

scyld-nodectl ls -l

Shows an expanded information available for each of the nodes.

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 n10 update mac=00:25:90:0C:D9:3C

Replace the current MAC address for node n10 with a new MAC address.

**RETURN VALUES**

Upon successful completion, **scyld-nodectl** returns 0. On failure, an error message is printed to stderr and **scyld-nodectl** returns 1.

### 4.10.12 scyld-sysinfo

**NAME**

**scyld-sysinfo** – Capture the system state information.

**SYNOPSIS**

```
```

**DESCRIPTION**

Capture the current system state into a subdirectory in the current working directory with the name “sysinfo-hostname-YY-MM-DD” (using a 2-digit Year-Month-Day). Unless the optional --no-tar argument is specified, this subdirectory is compressed by default into a gzip’ed tarball.

This command works best when executed by a non-root cluster administrator who has sudo rights.

The command also optionally captures sysinfo state for compute nodes, either all up nodes or a specific node or list of nodes. This functionality also works best if the head node has passwordless access to compute nodes.

If the optional -d alphanum_str is specified, then the directory name contains that alphanumeric alphanum_str string, e.g., “sysinfo-alphanum_str-hostname-YY-MM-DD.tar.gz”.

---

**Chapter 4. Reference Guide**
If `-m message` is specified, then the `message` string is retained as the contents of the file DESCRIPTION at the top of the output directory. If `-m message` is not specified, then the script queries the user for optional multi-line input that is retained as file DESCRIPTION in the output directory.

In the event that `scyld-sysinfo` aborts while capturing data, note that a partial capture is still available as the subdirectory “sysinfo-<hostname>-YY-MM-DD” in the current working directory.

**OPTIONAL ARGUMENTS:**

- `-h, --help` Print usage message and exit. Ignore trailing args, parse and ignore preceding args.
- `-V` Print the version number.
- `--no-tar` Leave the output as a subdirectory, not as a gzip’ed tarball.
- `--up` Optionally capture the state of all `up` compute nodes.
- `-i <NODES>` Optionally capture the state of a specific node or nodes.
- `-d alphanum_str` Insert the alphanumeric string `alphanum_str` into the output directory/tarball name.
- `-d message` If specified, then the `message` string is retained as the contents of file DESCRIPTION at the top of the output directory.

**EXAMPLES**

```bash
scyld-sysinfo
  Capture the state of the current node into a gzip’ed tarball, executed as user root.

scyld-sysinfo --no-tar
  Capture the state of the current node into a human-readable subdirectory of the current working directory.

scyld-sysinfo -I -d UMich
  The output directory name for the head node “headnode1” is “sysinfo-UMich-headnode1-YY-MM-DD”.

scyld-sysinfo -m "dhcpd fails with network error"
  The output directory contains the file DESCRIPTION that contains the specified string.

scyld-sysinfo --up
  Capture the state of the current head node and all the `up` compute nodes.

scyld-sysinfo -i n0-10
  Capture the state of the current head node and compute nodes n0 through n10.

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.

### 4.10.13 scyld-tool-config

**NAME**

`scyld-tool-config` – “Command line tool” for ClusterWare11
SYNOPSIS


DESCRIPTION

The generic command line tool for ClusterWare11.

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.

- **--yes**
  Answer yes or to defaults to all questions.

ARGUMENTS TO OVERRIDE BASIC CONFIGURATION DETAILS

- **--base-url URL**
  Specify the base URL of the ClusterWare REST API.

- **-u, --user USER [ : PASSWD]**
  Use connection user USER and 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.

### 4.11 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.

#### 4.11.1 managedb

**NAME**

managedb – Directly manipulate the database.

**SYNOPSIS**

{join, leave, eject NODE, clear, save ARCHIVE, load ARCHIVE, merge ARCHIVE}

**DESCRIPTION**

This is a low-level tool that directly manipulates the database, typically only executed by other scyld-* tools. The tool resides in /opt/scyld/clusterware/bin/.

**ACTIONS**

- **join**
  Join this head node to an existing cluster.

- **leave**
  Remove this head node from the cluster.

- **eject NODE**
  Remove the specified head node NODE from the cluster.
clear  Reset the data back to a fresh, empty state.

save ARCHIVE  Save the database to an ARCHIVE file or directory.

load ARCHIVE  Load the database from an ARCHIVE file or directory.

merge ARCHIVE  Merge the contents of an ARCHIVE file or directory into 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.
- --print-options  Print all backend options, then exit.
- --as-ini  Use ini format when printing options.

EXAMPLES

RETURN VALUES

Upon successful completion, managedb returns 0. On failure, an error message is printed to stderr and managedb returns 1.
5.1 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 OR THAT THE OPERATION OF THE SOFTWARE WILL BE ERROR FREE OR APPEAR PRECISELY AS DESCRIBED IN THE ACCOMPANYING DOCUMENTATION.

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.

5.2 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.

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 HOLDERS ‘‘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 HOLDERS 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

https://github.com/phdeniel/nfs-ganesha/blob/master/src/cidr/LICENSE

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:

5.2. Third-Party License Agreements
(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
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:

www.OpenPBS.org +1 650 967-4675 sales@OpenPBS.org
877 902-4PBS (US toll-free)

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

---

**5.2. Third-Party License Agreements**
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.
APPENDIX: CLUSTERWARE 7 VS. 11 COMMANDS

**ClusterWare 7**
- `bpsh 0-127 date`

**ClusterWare 11**
- `pdsh -w n[0-127] date`
  
**ClusterWare 11**
- `scyld-nodectl -i n[0-127] exec date`

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 only for cluster administrators.

**ClusterWare 7**
- `bpsh -a date`

**ClusterWare 11**
- `scyld-nodectl –up exec date`

Perform the same remote execution of the `date` command, although this time just for nodes in the “up” state.

**ClusterWare 7**
- `bpcp`

**ClusterWare 11**
- `rsync, scp, pdcp`

`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.

**ClusterWare 7**
- `bpstat`

**ClusterWare 11**
- `scyld-nodectl status`

**ClusterWare 7**
- `bpstat -U`

**ClusterWare 11**
- `scyld-nodectl status –refresh`

**ClusterWare 7**
- `bpstat –long`

**ClusterWare 11**
- `scyld-nodectl status –long`

`bpstat` only executes on the master node.

`scyld-nodectl` executes on any node that has the `clusterware-tools` package installed and only for cluster administrators.

**ClusterWare 7**
- `beostatus`

**ClusterWare 11**
- `Chronograf GUI`

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 7</th>
<th>bpctl -S 0-127 -R</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare 11</td>
<td>scyld-nodectl -i n[0-127] reboot</td>
</tr>
<tr>
<td>ClusterWare 7</td>
<td>bpctl -S all -R</td>
</tr>
<tr>
<td>ClusterWare 11</td>
<td>scyld-nodectl -a reboot</td>
</tr>
<tr>
<td>ClusterWare 7</td>
<td>bpctl -S allup -R</td>
</tr>
<tr>
<td>ClusterWare 11</td>
<td>scyld-nodectl –up 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 only for cluster administrators.

<table>
<thead>
<tr>
<th>ClusterWare 7</th>
<th>beoconfig node</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClusterWare 11</td>
<td>scyld-nodectl ls -l</td>
</tr>
<tr>
<td>ClusterWare 7</td>
<td>beoconfig kernelimage</td>
</tr>
<tr>
<td>ClusterWare 11</td>
<td>scyld-bootctl -a ls -l</td>
</tr>
<tr>
<td>ClusterWare 7</td>
<td>beoconfig kernelcommandline</td>
</tr>
<tr>
<td>ClusterWare 11</td>
<td>scyld-imgctl -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 have the *clusterware-tools* package installed and only for cluster administrators.
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 ClusterWare has dependencies on various base distribution packages, and a subsequent ClusterWare update may have dependencies on new or updated base distribution packages.

Next, obtain a Scyld ClusterWare 11 ISO file from Penguin Computing that contains the desired software, then mount that ISO locally. For example, suppose the ISO `clusterware-11.0.8.iso` contains Scyld ClusterWare 11 release 11.0.8:

```
# Mount the ClusterWare ISO, if not already mounted.
mount -o loop clusterware-11.0.8.iso /var/www/html/cw11.0.8
```

For an initial install, use a cluster configuration file (e.g., named `cluster-conf`) that is described in First Installation of Scyld ClusterWare, then execute the `scyld-install` script that is embedded in the ISO:

```
/var/www/html/cw11.0.8/scyld-install --config cluster-conf
```

That performs the basic first install of ClusterWare and creates `/etc/yum.repos.d/clusterware.repo`, which points at the software in the ISO.

**Important:** For ClusterWare v11.1.0 and earlier, you must edit that file to change file:///var/www/html/cw11.0.8 to http://localhost/cw11.0.8.

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>`:

```
scyld-clusterctl repos create name=<baseOSrepo> urls=http://<baseOSserver>/<baseOSiso>
```

```
scyld-clusterctl distros create name=<baseOSdistro> repos=<baseOSrepo> release=<baseOSrelease>
```

Now finish the setup. The following expects to find a single base distribution `repo` and single `distro`:

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

**For a software update of an existing Scyld ClusterWare 11,** rename the current `/etc/yum.repos.d/clusterware.repo`, then execute the script (which recreates `clusterware.repo` with the appropriate values):

```
(cd /etc/yum.repos.d; sudo mv -f clusterware.repo clusterware.repo.bak)
/var/www/html/cw11.0.8/scyld-install
```
8.1 Singularity

Singularity is available in Scyld ClusterWare 11 by installing the singularity-scyld 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.
sclyd-imgctl -i DefaultImage clone name=SingularityImage

# Install needed packages inside the new image.
sclyd-modimg -i SingularityImage --freshen --overwrite --upload --no-discard --install singularity-sclyld,openmpi3.1-gnu

# Now get into the chroot of the Singularity image.
sclyd-modimg -i SingularityImage --overwrite --upload --no-discard --chroot

# 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:

clyd-bootctl -i DefaultBoot clone name=SingularityBoot
clyd-bootctl -i SingularityBoot update image=SingularityImage
clyd-nodectl -i n[0-1] set _boot_config=SingularityBoot
# Now reboot nodes n0 and n1
clyd-nodectl -i n[0-1] reboot

When the nodes are up, then initialize passphrase-less key-based access, as described in Optionally install OpenMPI. 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:

yum install singularity-sclyld openmpi3.1-gnu  # if not already installed
module load openmpi/gnu/3.1.3
mpirun -np 2 --host n0,n1 singularity exec openmpi.sif /usr/bin/ring

8.2 Docker

Scyld ClusterWare 11 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.
sclyd-imgctl -i DefaultImage clone name=DockerImage

# Install needed packages inside the new image.
sclyd-modimg -i DockerImage --freshen --overwrite --upload --no-discard --install docker --exec "systemctl enable docker"

8.2. Docker
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 *Optionally allow IP Forwarding* for details.

Now boot node n0 with the new DockerImage:

```
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 *Optionally install OpenMPI*, 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.
APPENDIX: CREATING ARBITRARY CENTOS IMAGES

The earlier section Creating PXEboot Images 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. This involves using scyld-clusterctl to create custom distros and repos.

First, create a valid repo that accesses the CentOS http://vault.centos.org/ repositories. Note that the two ‘$’ dollar sign characters in the urls= argument need a preceding backslash “escape” character.

```
[admin@virthead] scyld-clusterctl repos create name=CentOS_vault \
    urls=http://vault.centos.org/\$releasever/os/\$basearch/
```

Created Repo

CentOS_vault

Then create a new distro that references the new CentOS_vault repo. For this example use the specific name of the CentOS 7.5 folder in the CentOS vault:

```
[admin@virthead] scyld-clusterctl distros create \n    name=CentOS_7.5.1804 release=7.5.1804 repos=CentOS_vault
```

View the result:

```
[admin@virthead]$ scyld-clusterctl distros ls -l
Distros
    CentOS
        name: CentOS
        package_manager: yum
        release: 7
        repos
            CentOS_base
    CentOS_7.5.1804
        name: CentOS_7.5.1804
        package_manager: yum
        release: 7.5.1804
        repos
```

CentOS_vault

Create a new CentOS 7.5 image that uses this new distro:

```
[admin@virthead] scyld-modimg --create CentOS_7.5.1804 --set-name CentOS_7.5
Executing step: Create CentOS_7.5.1804
Preparing the chroot...
[sudo] password for admin:
...done.
Initializing the chroot...
elapsed: 0:00:53.9
...initialized.
```
Installing core packages...
  elapsed: 0:00:00.6
ERROR: One or more repositories in the newly created image are invalid or unreachable. This can happen when installing older versions of Linux distributions such as CentOS where the included yum .repo files point to invalid locations.
  Please consult the Administrator's Guide for more information.
WARNING: The command will be retried with unknown repositories disabled.
  elapsed: 0:07:07.7
[sudo] password for admin:
  fixing SELinux file labels...
  ...done.
  step completed in 0:09:35.9
Repacking 1cf04a7790e24c9087d704b71ce7ea9e
  88.8% complete, elapsed: 0:01:37.8, remaining: 0:00:12.4 (58.6% compression)
Cleaning up.
Upload image? [yn] n

Note the “ERROR:” during image creation. This is entirely expected and describes what went wrong on the first pass. After the retry, because there is no subsequent error reported, the administrator can assume that the resulting CentOS_7.5.1804 image is useable.
APPENDIX: CREATING ARBITRARY RHEL IMAGES

The Appendix: Creating Arbitrary CentOS Images describes how to create PXEboot images using arbitrary CentOS repos. This appendix describes how to create arbitrary PXEboot RHEL images.

For this example we build a RHEL 7.7 image. First, download the RHEL 7.7 ISO file and mount it at var/www/html/rh7.7. We suggest using the “Red Hat Enterprise Linux for Scientific Computing” ISO file such as rhel-computenode-7.7-x86_64-dvd.iso. Compute node images can also be made from Red Hat Workstation or Server ISOs using the steps outlined below, though those will require appropriate licenses from Red Hat. In the case of full server installs, be aware of known issue related to the NetworkManager-config-server package, as noted in Known Issues And Workarounds.

Once the appropriate ISO has been downloaded, then create a “RHEL7_7” repo:

```
[admin@virthead]$ scyld-clusterctl repos create name=RHEL7_7 urls=http://localhost/rh7.7
Created Repo RHEL7_7
```

Then create a new distro that references the new RHEL7_7 repo:

```
[admin@virthead]$ scyld-clusterctl distros create name=RHEL7 repos=RHEL7_7 release=7
Created Distro RHEL7
```

And optionally view the result:

```
[admin@virthead]$ scyld-clusterctl distros ls -l
Distros
    CentOS
        name: CentOS
        package_manager: yum
        release: 7
        repos
            CentOS_base
    RHEL7
        name: RHEL7
        package_manager: yum
        release: 7
        repos
            RHEL7_7
```

**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`.

Now create a new RHEL 7.7 image that uses this distro:
Executing step: Create RHEL7
Preparing the chroot...
...done.
Initializing the chroot...
elapsed: 0:02:21.6
...initialized.
Installing core packages...
elapsed: 0:09:04.2
fixing SELinux file labels...
...done.
step completed in 0:11:36.4
Executing step: Set-Name RHEL7.7test
step completed in 0:00:00.0
Repacking RHEL7.7test
89.3% complete, elapsed: 0:01:39.8, remaining: 0:00:12.0 (62.1% compression)
Cleaning up.
Upload image? [yn] y
Checksumming image RHEL7.7test
elapsed: 0:00:01.5
Created a new image object.
Uploading new image content.
100.0% complete, elapsed: 0:00:02.7 remaining: 0:00:00.0
Please wait while the server processes the upload:

Optionally validate the new image using chroot to check that RHEL7.7 is indeed installed:

Extract the kernel from the image and build the ramfs:
elapsed: 0:00:01.6
Executing step: Copyout ['/boot/vmlinuz-3.10.0-1062.el7.x86_64', '.']
step completed in 0:00:01.3
Cleaning up.

[admin@virthead]$ scyld-mkramfs --image RHEL7.7test --kver 3.10.0-1062.el7.x86_64 --output rhel7.ramfs
    tar: Removing leading '/' from member names
This script will execute the following command to construct the initramfs:
    scyld-modimg --image RHEL7.7test --install iscsi-initiator-utils,nfs-utils,dhclient,sudo \
        --unpack mkramfs.tar.gz.mtio9 \ 
        --execute '/opt/scyld/clusterware-tools/mkramfs/scyld-mkramfs \ 
        --kver "3.10.0-1062.el7.x86_64" --output "/scyld-mkramfs-rhel7.ramfs"' \ 
        --export /scyld-mkramfs-rhel7.ramfs rhel7.ramfs --discard
Checksumming image RHEL7.7test
elapsed: 0:00:01.6
Unpacking image RHEL7.7test
  100.0% complete, elapsed: 0:00:28.4 (62.1% compression)1 (61.7% compression)
Checksumming...
  elapsed: 0:00:01.6
Executing step: Install iscsi-initiator-utils,nfs-utils,dhclient,sudo
  elapsed: 0:00:19.3
    fixing SELinux file labels...
step completed in 0:00:29.0
Executing step: Unpack mkramfs.tar.gz.mtio9
  100.0% complete, elapsed: 0:00:00.3 (58.7% compression)n)
step completed in 0:00:00.3
Executing step: Execute /opt/scyld/clusterware-tools/mkramfs/scyld-mkramfs \
    --kver "3.10.0-1062.el7.x86_64" --output "/scyld-mkramfs-rhel7.ramfs"
Packing the ramfs into /scyld-mkramfs-rhel7.ramfs
This ramfs will include kernel modules for 3.10.0-1062.el7.x86_64
Creating the initramfs using dracut.
dracut: Executing: /opt/scyld/clusterware-tools/dracut/bin/dracut --force --no-hostonly \
    --modules "url-lib nfs iscsi base network clusterware" \ 
    --drivers " virtio_net virtio_pci e1000 8139too ata_piix \ 
        sd_mod ata_Generic ahci ext4 overlay loop squashfs" \ 
    --include /tmp/tmp.zWbjibLkfl / /scyld-mkramfs-rhel7.ramfs 3.10.0-1062.el7.x86_64
dracut: *** Including module: network-legacy ***
dracut: *** Including module: network ***
dracut: *** Including module: url-lib ***
dracut: *** Including module: kernel-network-modules ***
dracut: *** Including module: iscsi ***
dracut: *** Including module: nfs ***
dracut: *** Including module: rootfs-block ***
dracut: *** Including module: udev-rules ***
dracut: Skipping udev rule: 91-permissions.rules
dracut: Skipping udev rule: 80-drivers-modprobe.rules
dracut: *** Including module: base ***
dracut: *** Including module: clusterware ***
dracut: *** Including module: fs-lib ***
dracut: *** Including modules done ***
dracut: *** Installing kernel module dependencies ***
dracut: *** Installing kernel module dependencies done ***
dracut: *** Resolving executable dependencies ***
dracut: *** Resolving executable dependencies done***
dracut: *** Hardlinking files ***
dracut: *** Hardlinking files done ***
dracut: *** Stripping files ***
dracut: *** Stripping files done ***
dracut: *** Generating early-microcode cpio image ***
dracut: *** Constructing AuthenticAMD.bin ****
dracut: *** Constructing GenuineIntel.bin ****
dracut: *** Store current command line parameters ***
dracut: *** Creating image file '/scyld-mkramfs-rhel7.ramfs' ***
          fixing SELinux file labels...
          step completed in 0:00:44.9
Executing step: Export ['/scyld-mkramfs-rhel7.ramfs', 'rhel7.ramfs']
          step completed in 0:00:00.1
Cleaning up.

Create a new boot “rhel7Boot” with the kernel and ramfs that was just created:

[admin@virthead]$ scyld-bootctl -i DefaultBoot clone name=rhel7Boot
Cloned Boot Configurations
DefaultBoot: 4c4ddf4dc5874cee9acad9b675e73e6b
[admin@virthead]$ scyld-bootctl -i rhel7Boot update image=RHEL7.7test \ 
              initramfs=@rhel7.ramfs kernel=@vmlinuz-3.10.0-1062.el7.x86_64
Boot Configurations
  rhel7Boot
      success: True
[admin@virthead]$ scyld-bootctl 1s -l
Boot Configurations
  DefaultBoot
    cmdline: enforcing=0
    image: DefaultImage
    initramfs
      chksum: 5a10b0be5a2dd1e73a401ef80796749c66671813
      mtime: 2020-01-03 22:31:58 UTC (5 days, 3:15:30 ago)
      size: 32.4 MiB (33931703 bytes)
    kernel
      chksum: 869caa96db3a3a222f64b15fa8099aeb5e32008
      mtime: 2020-01-03 22:31:59 UTC (5 days, 3:15:30 ago)
      size: 6.4 MiB (6734016 bytes)
      last_modified: 2020-01-03 22:31:59 UTC (5 days, 3:15:30 ago)
      name: DefaultBoot
      release: 3.10.0-1062.9.1.el7.x86_64
  rhel7Boot
    cmdline: enforcing=0
    image: RHEL7.7test
    initramfs
      chksum: ec41f02961a5f94e60d7dc20fd5512ac36f2884c
      mtime: 2020-01-09 01:47:24 UTC (0:00:04 ago)
      size: 32.4 MiB (33934241 bytes)
    kernel
      chksum: 65b4a04a71c32834f4f99ac505b2282900311f0d0
      mtime: 2020-01-09 01:47:25 UTC (0:00:04 ago)
      size: 6.4 MiB (6730032 bytes)
      last_modified: 2020-01-09 01:47:25 UTC (0:00:03 ago)
      name: rhel7Boot
      release: 3.10.0-1062.el7.x86_64

Assign rhel7Boot to node n0 and reboot n0:

[admin@virthead]$ scyld-nodectl -i n0 set _boot_config=rhel7Boot
Results
  n0
      success: True
When the rebooted node n0 returns to an “up” state, then you can optionally verify that it is a RHEL 7.7 environment:

```
[admin@virthead]$ scyld-nodectl status
n[0-5] up

[admin@virthead]$ scyld-nodectl -i n0 ssh

[root@n0 ~]# cat /etc/redhat-release

Red Hat Enterprise Linux Server release 7.7 (Maipo)

[root@n0 ~]# exit

logout

Connection to 10.54.80.100 closed.
```
APPENDIX: CREATING NODES WITH KICKSTART

Scyld ClusterWare 11 supports using Red Hat kickstart to install full distributions on a diskfull compute node. We provide examples for CentOS and Ubuntu.

Steps:

1. Enable clusterware compute node Internet access, which allows for network installation. See Optionally allow IP Forwarding for details.

2. Download the proper netboot installer kernel and initramfs.

   The Ubuntu 18.04 netboot kernel and initramfs links:
   
   

   The CentOS 7 netboot kernel and initramfs links:
   
   http://mirror.centos.org/centos-7/7/os/x86_64/isolinux/initrd.img
   
   http://mirror.centos.org/centos-7/7/os/x86_64/isolinux/vmlinuz

3. Create a kickstart file for the intended installation.

   An Ubuntu 18.04 example:

```
lang en_US
langsupport en_US
keyboard us
mouse
timezone America/New_York
rootpw --disabled
user ubuntu --fullname "Ubuntu User" --password root4me2
poweroff
text
install
url --url http://us.archive.ubuntu.com/ubuntu
bootloader --location=mbr
zerombr yes
clearpart --all --initlabel
part / --fstype ext4 --size 1 --grow --asprimary
part swap --size 1024
part /boot --fstype ext4 --size 256 --asprimary
auth --useshadow --enablemd5
network --bootproto=dhcp
firewall --enabled
```
A CentOS 7 example:

install
text
url --url http://mirror.centos.org/centos-7/7/os/x86_64
lang en_US.UTF-8
keyboard us
network --bootproto=dhcp --device=eth0
rootpw --plaintext root4me
firewall --enabled
authconfig --enablesnapshot --passalgo=sha512
selinux --disabled
timezone America/New_York
bootloader --location=mbr
zerombr
clearpart --all --initlabel
part /boot --fstype=ext4 --asprimary --size=1023
part swap --asprimary --size=1024
part / --fstype=ext4 --grow --asprimary --size=200
poweroff
%packages --nobase
@core
%end

After the Ubuntu or CentOS installation completes, you can install the `clusterware-node` package and the node configuration (see Booting Diskfull Compute Nodes) using the kickstart `%post` script:

An Ubuntu example:

```bash
%post
echo "deb https://<AUTH_TOKEN>:@updates.penguincomputing.com/clusterware/11/deb/ /" \\> /etc/apt/sources.list.d/clusterware.list
apt-get update
apt install -y clusterware-node
echo "base_url=http://<HEADNODE_IP>/api/v1" \\> /opt/scyld/clusterware-node/etc/node.sh
echo "iface=ens3" >> /opt/scyld/clusterware-node/etc/node.sh
%end
```

A CentOS 7 example:

```bash
%post
cat <<"EOF" >/etc/yum.repos.d/clusterware-node.repo
[scyld-el7]
name=Scyld el7
baseurl=https://<AUTH_TOKEN>:@updates.penguincomputing.com/clusterware/11/el7/$basearch/
enabled=1
gpgcheck=1
priority=3
EOF
yum -y install clusterware-node
```
Note: Replace `<HEADNODE_IP>` from the above examples with the appropriate head node’s IP address, and replace `<AUTH_TOKEN>` with the appropriate customer authentication token.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>echo &quot;base_url=http://&lt;HEADNODE_IP&gt;/api/v1&quot; &gt; /opt/scyld/clusterware-node/etc/node.sh</code></td>
<td>Create a file to set the base URL for API requests.</td>
</tr>
<tr>
<td><code>echo &quot;iface=eth0&quot; &gt;&gt; /opt/scyld/clusterware-node/etc/node.sh</code></td>
<td>Configure the network interface.</td>
</tr>
<tr>
<td><code>%end</code></td>
<td>End of configuration section.</td>
</tr>
</tbody>
</table>

4. Host kickstart on a web server. The ClusterWare head node can be used for this purpose. Then simply copy the kickstart file to `/var/www/html/netboot/kickstart`.

5. Create a boot configuration using the downloaded kernel and initramfs, and assign boot configuration to the compute node. For example, for node n10:

```bash
scyld-bootctl create name=NetBoot initramfs=@initrd.gz kernel=@linux image=None
cmdline="enforcing=0 ks=http://<HEADNODE_IP>/netboot/kickstart"
scyld-nodectl -i n10 set _boot_config=NetBoot
```

Again, replace `<HEADNODE_IP>` from the above example with the appropriate head node’s IP address.

6. Set the compute node to network boot, then power on the compute node. The kickstart file should be able to automate all installation, and then shutdown the compute node. Then change the compute node to hard drive boot and power on the compute node again.

If you have not installed `clusterware-node` and created `/opt/scyld/clusterware-node/etc/nodes.sh` in the kickstart `%post` script, you may do so manually now, then reboot compute node again.

These “diskfull” nodes can be managed by the same `scyld-nodectl` command as the PXEboot nodes. See [Booting Diskfull Compute Nodes](#) for details.
APPENDIX: CREATING NODES WITH PRESEED

Scyld ClusterWare 11 supports using Debian preseed to install full distributions on a diskfull compute node.

Steps:

1. Enable clusterware compute node Internet access, which allows for network installation. See *Optionally allow IP Forwarding* for details.

2. Download the proper netboot installer kernel and initramfs.

The Debian 10 netboot kernel and initramfs links:


3. Create a preseed file for the intended installation.

A Debian 10 example:

```plaintext
d-i mirror/country string us
d-i mirror/http/hostname string http.us.debian.org
d-i mirror/http/directory string /debian
d-i mirror/http/proxy string
d-i mirror/suite string buster
d-i netcfg/choose_interface select auto
d-i netcfg/wireless_wep string
d-i passwd/root-login boolean true
d-i passwd/root-password password root4me
d-i passwd/root-password-again password root4me
d-i passwd/user-fullname string Debian User
d-i passwd/username string debian
d-i passwd/user-password password insecure
d-i passwd/user-password-again password insecure
d-i clock-setup/utc boolean true
d-i time/zone string US/Eastern
d-i partman-auto/disk string /dev/sda
d-i partman-auto/method string lvm
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-lvm/device_remove_lvm_span boolean true
d-i partman-auto/purge_lvm_from_device boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-auto-lvm/new vg_name string sys vg
d-i partman-auto/choose_recipe atomic
d-i partman-auto-lvm/guided_size string max
```
After the Debian installation completes, you can install the `clusterware-node` package and the node configuration (see *Booting Diskfull Compute Nodes*) using the preseed late_command script:

A Debian example:

```bash
d-i preseed/late_command string \
    echo "deb https://<AUTH_TOKEN>@updates.penguincomputing.com/clusterware/11/deb/ /" \
    > /target/etc/apt/sources.list.d/clusterware.list; \
    in-target apt-key adv --fetch-keys \ 
    https://updates.penguincomputing.com/DEB-GPG-KEY-PenguinComputing; \
    in-target apt-get update; \
    in-target apt-get install -y clusterware-node; \
    echo "base_url=http://<HEADNODE_IP>/api/v1" \
    > /target/opt/scyld/clusterware-node/etc/node.sh; \
    echo "iface=ens3" >> /target/opt/scyld/clusterware-node/etc/node.sh
```

Note: Replace `<HEADNODE_IP>` from the above examples with the appropriate head node’s IP address, and replace `<AUTH_TOKEN>` with the appropriate customer authentication token.

4. Host preseed on a web server.

The ClusterWare head node can be used for this purpose. Then simply copy the preseed file to `/var/www/html/netboot/preseed`.

5. Create a boot configuration using the downloaded kernel and initramfs, and assign boot configuration to the compute node. For example, for node n10:

```bash
scylld-bootctl create name=NetBoot initramfs=@initrd.gz kernel=@linux image=None
scylld-bootctl -i NetBoot up \
    cmdline="enforcing=0 \ 
    url=http://<HEADNODE_IP>/netboot/preseed debian-installer/locale=en_US \ 
    console-setup/ask_detect=false console-setup/layoutcode=us \ 
    keyboard-configuration/xkb-keymap=us \ 
    netcfg/get_hostname=unassigned-hostname netcfg/get_domain=unassigned-domain"

scylld-nodectl -i n10 set_boot_config=NetBoot
```

Again, replace `<HEADNODE_IP>` from the above example with the appropriate head node’s IP address.

Note: Some configuration options are set by the scylld-bootctl cmdline option because they must be configured before the preseed file is loaded by the Debian installer.

6. Set the compute node to network boot, then power on the compute node. The preseed file should be able to automate all installation, and then shutdown the compute node. Then change the compute node to hard drive boot and power on the compute node again.

If you have not installed `clusterware-node` and created `/opt/scyld/clusterware-node/etc/nodes.sh` in the preseed late_command script, you may do so manually now, then reboot compute node again.
These “diskfull” nodes can be managed by the same `scyld-nodectl` command as the PXEboot nodes. See *Booting Diskfull Compute Nodes* for details.
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 `loram` 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).
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, Revised Edition, published March 16, 2020.