

# BYOC: Build Your Own Cluster using CentOS 7.2

Nathan R. Vance, Michael L. Poulton, and William F. Polik  
Hope College, Summer 2016

## I. Introduction

Cluster computers are a standard tool in scientific computing. A cluster computer uses commodity off-the-shelf hardware to solve extremely large or computationally intensive problems. Cluster computing achieves efficiency by computing a single task in parallel or computing multiple tasks simultaneously, leading to an excellent performance-to-price ratio.

In a basic cluster computer, one computer (the head node) relays instructions to the rest of the cluster computers (compute nodes) across an isolated local network. The compute nodes then carry out their assigned tasks and return the results when they are done. This structure can be thought of through the metaphor of a work force: The head node is the manager, which receives tasks from the customer (you) and subcontracts with the compute node workers. When a worker is done, it signals the manager and asks for more work, if any is available. When the job is completed, the manager returns the final product, in this case a calculated result.

## II. Cluster Design

Our design goals include:

- **Robustness and Reliability** - The cluster should be stable after the initial setup.
- **Portability** - The same process and tools can be used on different distributions with little to no change.
- **Scalability** - The procedures should be practical if there are 2, 10, or 100 nodes.
- **No “Magic”** - Common problems are resolved by straightforward, easy-to-follow solutions.
- **Heterogeneity** - Upgrading in the future is still possible when the hardware may be different.
- **Simplicity** - The approach outlined below should allow a person with limited Linux/unix experience to build a cluster of their own.
- **Low-Cost** - Uses readily available hardware and free software (Linux)

## III. Decisions and Setup

### A. Cluster Hardware

There are many hardware components that go into building your cluster. These can generally be broken down into four general categories: the computers themselves, networking supplies, an apparatus to physically store the computers, and a console to access the computers.

- **Computers** - Commercial computers can be purchased from suppliers such as Dell or HP, or you can assemble your own. There are also a number of retailers online that sell computers specifically designed for use in a cluster environment. These usually are systems that are designed to work well with Linux (the OS that nearly every cluster runs). Often, these systems come in high-density packages, such as 1 or 2 U rack-mountable cases (a U is 1.75” of vertical rack space). To make life easier, the compute nodes should support PXE booting and IPMI control.

- **Networking** - Your network switch needs to have at least as many ports as you have compute nodes. A few extra ports are always handy in case you decide to add to your cluster in the future. Network cables are also essential.
- **Physical Configuration** - A cluster can work with tower PCs on a shelf, however, a professional job will typically use special rack-mounted computers. Often, computers will be on tracks allowing them to be slid out far enough to remove the lid without physically detaching them from the rack. It is advisable to leave enough slack in the cables on the backs of the computers so that they can be running while pulled out for diagnostic purposes.
- **Access** - You can have the nodes be headless (operate without a keyboard, mouse, and monitor) or you can purchase a keyboard, video, mouse (KVM) switch. The more nodes involved, the pricier the KVM will be. It is also a good idea to have a local monitor, mouse, and keyboard hooked up to the head-node. This guarantees that you will be able to access your cluster even if the network is down. There are specialty products such as a rack-mountable LCD monitor and keyboard that can serve this purpose well.

## B. Network setup

- **Name Your Cluster** - Names are used as aliases for IP addresses. Two names must be configured, one for the external network and one for the internal network.
  - **External Network** - This is typically formatted as hostname.domain.suffix, where the hostname is whatever you want, and the domain.suffix pertains to the organization the cluster is used by. The example used in this guide, a cluster in Hope College's chemistry department, is ernst.chem.hope.edu.
  - **Internal Network** - The external hostname is typically used in conjunction with a numbering scheme as the base for head and compute node hostnames inside the cluster. For example, we append two digits to the end of the hostname for each node: ernst00 (head node), ernst01 (first compute node), etc. This scheme limits us to 100 nodes, but can easily be expanded to accommodate future upgrades.
- **IP Addresses** - Two options exist for the internal network: static and dynamic assignment.
  - **Static Assignment** - Each compute node is configured individually with its own IP address. This contradicts the scalability goal of this guide because manually configuring IP addresses for a large number of nodes is not practical.
  - **Dynamic Assignment** - Each node can have an identical configuration and receive its IP through the network. This guide will use the dynamic networking solution.

## C. Disk partitioning

As opposed to Windows where partitions are referred to as lettered drives, in Linux they are mounted under directories called 'mount points' in the file system. Partitions are useful for keeping data separate for easy backups or preserving data between installations. This section highlights several useful Linux partitions.

- **root (/)** - This partition is where the actual operating system resides, and in its file system other partitions will be mounted.
- **/admin** - Disk images, software distributions, kickstart files and backups are stored here. This is vital for the installation of all compute nodes in a scalable way.
- **/home** - User files are located here. While not absolutely required to be kept separate from root, it is strongly recommended as this makes a good place to store computational software.

- **swap** - Swapping is the process where, should Linux run out of memory, it writes pages of memory to the swap partition on the hard disk. This can result in a performance gain for systems with too little RAM, however, it is not to be relied upon as swapping is orders of magnitude slower than using RAM.

If your nodes use multiple disks, you will have the choice of which ones to use for which partitions. By convention, the root partition should go on the first hard disk, but the rest is up to you. The following is an example of a single disk partition scheme for a 1 TB hard drive.

<b>Head node</b>		<b>Compute Node</b>	
/	200 GB	/	200 GB
/admin	200 GB	/scratch	rest of space
/home	rest of space		

## D. RAID Devices

When storing large amounts of data it is highly recommended to utilize a RAID device. This may be a separate component connected to the head node, integrated directly into the head node, or part of a separate storage node.

A RAID device works by combining several small drives to form one larger virtual drive and/or to add data redundancy, allowing a drive to fail while still saving the data. There are several commonly used RAID levels that achieve one, the other, or both of these ends.

- **RAID 0** provides a storage size and performance increase by ‘striping’ data across two or more drives. This means that as after a predetermined allotment of data is written to a drive (typically 128k) the RAID will switch to the other drive. This may significantly improve read time in some applications; however, one failed drive causes all the data to be lost. A RAID 0 drive is as large as the size of its smallest drive times the number of drives.
- **RAID 1** provides redundancy with no storage size or performance increase by mirroring data writes to two or more disks, allowing one to go down while preserving the data. The size of a RAID 1 drive is the same as its smallest drive.
- **RAID 5** is similar to RAID 0 except that it includes redundant parity information spread across the 3 or more disks. This allows any one disk to fail without the loss of data. The RAID as a whole will only store as much as the smallest drive times one less than the number of drives.
- **RAID 6** is similar to RAID 5 except that it has two disks worth of redundant parity information spread across 4 or more drives. This allows for two disks to fail without the loss of data. Storage will be limited to the storage of the smallest drive times two less than the number of drives.

## IV.Assembly

After you have gathered all your hardware you can begin the fun part of cluster work: actually assembling your cluster. If your compute nodes have more than one network interface, you will have to figure out which one the system recognizes as the first interface and which is the second (eth0 and eth1 under Linux, respectively). Keep this in mind when running your cables. It can be both time consuming and frustrating to track down a problem, only to find that it was caused by a swapped network cable. To

help prevent this, use differently colored cables for different networks and make sure to label cables on both ends. If things are kept consistent between nodes, your life will be much easier when it comes to managing your cluster. Ideally, when you start out, your compute nodes should all be identical, both in terms of internal hardware and external cable configuration.

## V. Basic Linux Commands and Tasks

This guide assumes limited Unix/Linux experience. Refer to this section if unsure how to complete any command line tasks given later in this guide.

1. Directory Navigation and File Manipulation - View the man pages or research online
  - a. `man cp`
  - b. `man mv`
  - c. `man rm`
  - d. `man mkdir`
2. Editing a File with vi
  - a. To open a file,  
`# vi /path/to/file`
  - b. Navigate with arrow keys (or, if feeling adventurous, hjkl keys).
  - c. To edit the file, hit 'i'. This changes you from command mode to insert mode.
  - d. To exit, hit Esc to return to command mode, then type ':wq' without the quotes and hit enter. w stands for write and q stands for quit.
  - e. To exit without writing (saving), in the previous step substitute ':q!'.
3. Discovering a Device Name
  - a. Type the command `lsblk`
  - b. This command outputs a table showing basic information about every connected drive. For example:

NAME	MAJ:MIN	RM	SIZE	RO	TYPE	MOUNTPOINT
sr0	11:0	1	1024M	0	rom	
sda	8:32	0	5.5T	0	disk	
sda1	8:33	0	100G	0	part	/
sda2	8:34	0	200G	0	part	/admin
sda3	8:35	0	5.2T	0	part	/home
sdb	8:0	0	3.7G	0	disk	
sdb1	8:1	0	3.7G	0	part	

The important columns here for our purposes are NAME, SIZE, TYPE and MOUNTPOINT.

    - sr0 is the CD/DVD drive (sr stands for scsi-rom or sata-rom).
    - sda is the hard drive (scsi or sata disk) with partitions sda1, sda2 and sda3 mounted in the file system.
    - sdb is, in this case, a 4GB usb flash drive with a single partition, sdb1.
4. Mounting External Media
  - a. Once you have discovered your device name, mount a partition from the device:  
`# mount /dev/PARTITION_NAME /path/to/mountpoint/`  
See Discovering a Device Name for more information.
  - b. To unmount the device:  
`# umount /dev/PARTITION_NAME OR`  
`# umount /path/to/mountpoint/`  
Notice that these commands are `umount` rather than `unmount`!
5. Timesaving Hints
  - a. When typing a path or filename, try pressing tab to autocomplete names.
  - b. Use the up arrow to access your history to avoid retyping similar commands.

## VI.Head Node

### A. *Manual Installation*

1. Boot the installation media.
2. At the welcome screen select "Install CentOS 7."
3. Configuration:
  - a. Select your language and keyboard layout.
  - b. Network configuration:
    - i. Enter the machine's hostname in the box given.
    - ii. For each connection, click Configure.
    - iii. Check the box to Connect Automatically.
    - iv. In the IPv4 Settings tab and change Method to Manual.
    - v. Under addresses click Add and, if this is the external connection, set the fields according to how you and your network administrator choose. Otherwise, the interface connected to the subnet that the compute nodes are on should only be supplied with an address and netmask. An example network configuration is as follows:

Address: 192.168.1.100  
Netmask: 255.255.255.0

Numbering starts at 100 rather than 1 so that when making your hosts file all the addresses have the same number of digits.
  - c. Time Zone Configuration – Choose a city in your time zone and enable network time.
  - d. Package Selection: Select Server With GUI, and choose E-mail Server and Development Tools as Add-Ons.
  - e. Disk Partitioning:
    - i. Select the installation destination(s) and select "I will configure partitioning."
    - ii. Partition disks and assign mount points as was decided in the planning stage.
  - f. About to install – this is the last step where you can safely abort the installation procedure. Click Begin Installation to continue.
  - g. While the installer runs, set a root password and create a user. The user will be for logging into the desktop; root will be used for everything else.
  - h. Reboot and remove the DVD when finished.
  - i. Upon booting for the first time you will be asked to accept the license. Follow the on-screen prompts to accept it.

### B. *Automated Installation*

To achieve reliability, one needs to have a reproducible installation method that provides consistent results. Luckily, Red Hat's installer, anaconda, has such a method called "kickstart." Kickstarting means that the installer uses a configuration file to automatically install Linux. Anaconda generates a kickstart file after every installation found at `/root/anaconda-ks.cfg`.

#### **Editing the ks.cfg File**

The kickstart file can be edited to include all desired installation options and post-installation configurations. The kickstart file must stay in the Unix file format because Windows, Mac and Unix/Linux handle end of line characters differently. If you are going to edit the file on a

Windows machine, you need to use an editor that can handle the difference in format (Notepad++ is one such editor).

### Important Kickstart Features

- **Comments** - any line with a leading #. These are ignored by the installer. They are used to comment on code or to disable small sections of code.
- **Partitioning** - Several lines with partition info commented out. You should un-comment all of the partition lines. Add the option `--noformat` to the partition entries that you don't want to be formatted by the installer such as `/admin` and `/home`, but not the `/` partition as this contains the previously installed OS.
- **Repository** - This tells the installer where to find the repository to install from. Currently, it is set to install from a cd:

```
# Use CDROM installation media
cdrom
```

This line is the repository line and will be modified several times during this guide.

- **Miscellaneous Settings** - Settings may be applied such as disabling selinux, changing bootloader options, and much more. Modify the commands to disable selinux:

```
selinux --disabled
```

Visit Red Hat's Kickstart Options guide for an exhaustive list of kickstart options at

[https://access.redhat.com/documentation/en-US/Red\\_Hat\\_Enterprise\\_Linux/7/html/Installation\\_Guide/sect-kickstart-syntax.html](https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Installation_Guide/sect-kickstart-syntax.html).

- **Postscript** - At the end of the installation some additional commands may be executed. Add the following to the end of the file:

```
%post
%end
```

Between these tags we will add post installation scripts that will be executed once the installation completes. In the DHCP configuration section later in this guide a sample script to automatically configure DHCP is given. It is highly recommended to include all similar system modifications here as well for both documentation and backup purposes.

Creating an automated installation is an iterative process in which one modifies the kickstart file, reinstalls the system, and verifies that the changes took hold as intended. For example, to check that `/admin` and `/home` (and any others that you specified) don't get formatted during the install, create a file on the partition and see if it exists after anaconda finishes reinstalling. Also check that selinux is indeed disabled.

The kickstart file can be located on an ext2 or fat32 formatted usb drive, or a partition on the hard drive from where it will be read by the installer. If you are unsure which to use, skip down to the section titled DVD Based Booting with Kickstart on Hard Drive. It may useful, however, to test the kickstart on removable media first for convenience.

Table 1. Installation Methods for Head Node

Installation Method	Boot/Installer Location	Kickstart Location	Distro Location
Manual	DVD	None	DVD
Automated DVD	DVD	USB or Hard Drive	DVD
Automated Hard	DVD	Hard Drive	Hard Drive

Drive			
Sans Removable Media	Hard Drive	Hard Drive	Hard Drive

## DVD Based Booting with Kickstart on USB

Before we start, this method has one caveat: depending on the hardware configuration of your system, when the installer boots, the usb *may* show up as a different drive than normal. For example, in a configuration with two hard drives sda and sdb and a usb drive sdc, upon booting the usb may be sda while the hard drives are sdb and sdc. This scenario would require the kickstart file to be edited so that all references to any sdX are shifted one letter.

1. Insert a blank ext2 or fat32 formatted usb drive into the head node.
2. Mount the usb (if necessary, see the section on Basic Linux Commands and Tasks).
3. Copy the kickstart file to the usb drive. Make sure the repository line is set to  
`cdrom`
4. You can now use this kickstart file while booting from the DVD by pressing tab at the initial welcome screen and appending the following to the boot options:

```
inst.ks=hd:sdX1:/ks.cfg
```

Note that the drive letter X for the usb might not be the same as under the running operating system. The installer will allow you to edit this line if it fails to find the kickstart file on the specified device.

## DVD Based Booting with Kickstart on Hard Drive

Before we begin, I mentioned before that you don't want to reformat the `/admin` directory. Now you really don't want to reformat it. Refer to the section above on editing the `ks.cfg` file.

1. Make sure you know what the device name of the partition mounted as `/admin` is.
2. Create the following folder hierarchy in `/admin`  

```
# mkdir -p /admin/ks/headnode
```
3. Copy the kickstart file to newly created folder. Verify that the repository line is set to  
`cdrom`
4. You can now use this kickstart file while booting from the DVD by pressing tab at the initial welcome screen and appending the following to the boot options:

```
inst.ks=hd:sdXY:/ks/headnode/ks.cfg
```

The drive sdXY is the same as determined in step 1.

After you get the DVD based solution with a kickstart on the hard drive to work like you want (with anaconda using `ks.cfg` and the `/admin` and `/home` partitions not being formatted), you can begin migrating away from external media.

## DVD Based Booting with Kickstart and Distro on Hard Drive

1. If you have not done so already, perform the DVD Based Booting with Kickstart on Hard Drive installation.

2. Make a directory for the ISOs to reside in.  

```
# mkdir -p /admin/iso/centos7/
```
3. Copy the installation media to the hard drive. If you have the original ISO files floating around you can simply copy them across the network. If not, you can use the dd command to create the ISO files from the DVD.  

```
# dd if=/dev/cdrom of=/admin/iso/centos7/CentOS-7-x86_64-DVD-1511.iso
```
4. In the ks.cfg file, comment out the repository line and replace it with the following:  

```
harddrive --partition=sdXY --dir=/iso/centos7/
```

Note that sdXY is the partition that /admin is located on.
5. Insert the DVD and reboot. Press tab at the initial welcome screen and append the following to the options:  

```
inst.ks=hd:sdXY:/ks/headnode/ks.cfg
```

Note that sdXY is the partition that /admin is located on. The installation should now proceed using both ISOs and ks.cfg from the hard drive. This makes the installation go much faster.

At this point it is possible to completely eliminate external media by setting up the /admin partition to actually be the installer. It is then possible to configure the bootloader to boot directly into the installer, pass it the boot options for the kickstart file, and install CentOS completely automatically.

## Installation Sans Removable Media

1. If you have not done so already, perform a Kickstart and Distro on Hard Drive installation.
2. Make a directory to house boot files.  

```
# mkdir /admin/boot
```
3. Mount the iso and copy the internal files to the boot directory.  

```
# mount -o loop /admin/iso/centos7/CentOS-7-x86_64-DVD-1511.iso /mnt
# cp -a /mnt/* /admin/boot
```
4. At the bottom of /etc/grub.d/40\_custom insert the following:  

```
menuentry "Install" {
    set root=(hdW,msdosZ)
    linux /boot/images/pxeboot/vmlinuz ks=hd:sdXY:/ks/headnode/ks.cfg
    initrd /boot/images/pxeboot/initrd.img
}
```

Note the (hdW,msdosZ) and ks=hd:sdXY lines. These correspond to the drive and partition /admin is on. Use the command lsblk to find the partition in sdXY format. W then corresponds to the drive number; sda is 0, sdb is 1, and so forth. Z is the partition number /admin is on. So, if /admin is located at sda2, use (hd0,msdos2).
5. Regenerate grub.cfg.  

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```
6. At the end of the ks.cfg file, between %post and %end, insert the following:  

```
#grub configuration
cp -p /boot/grub/grub.conf /boot/grub/grub.conf.000
cat >> /etc/grub.d/40_custom << EOF
menuentry "Install" {
    set root=(hdW,msdosZ)
    linux /boot/images/pxeboot/vmlinuz ks=hd:sdXY:/ks/headnode/ks.cfg
    initrd /boot/images/pxeboot/initrd.img
```

```

}
EOF
grub2-mkconfig -o /boot/grub2/grub.cfg

```

This script will now automatically modify your grub.cfg file as done in steps 4 and 5. Make sure to modify the `(hdw,msdosz)` and `sdXY` lines as in step 4.

7. Reboot. At Grub's splash screen, arrow down to the entry titled "Install" and press enter. The system should now install.

Before continuing to the compute nodes, it is recommended to configure the DHCP server. See the section on DHCP configuration below.

## VII. Compute nodes

### A. Manual Installation

4. Boot the installation media.
5. At the welcome screen select "Install CentOS 7."
6. Configuration:
  - a. Select your language and keyboard layout.
  - b. Network configuration:
    - i. For each connection, click Configure.
    - ii. Check the box to Connect Automatically.
    - iii. If you have already set up DHCP on the headnode, you should receive an IP address.
  - c. Time Zone Configuration – Choose a city in your time zone.
  - d. Package Selection: Select Minimal
  - e. Disk Partitioning:
    - i. Select the installation destination(s) and select "I will configure partitioning."
    - ii. Partition disks and assign mount points as was decided in the planning stage.
  - f. About to install – this is the last step where you can safely abort the installation procedure. Click Begin Installation to continue.
  - g. While the installer runs, set the root password to be the same as the head node.
  - h. Reboot and remove the DVD when finished.
  - i. Upon booting for the first time you will be asked to accept the license. Follow the on-screen prompts to accept it.

### B. Automated Installation

For the compute nodes, we are going to automate the installation similarly to what was done for the head node. In the kickstart file, be sure to disable selinux as was done for the headnode, but also disable the firewall because the compute nodes won't be connected to the outside world anyway.

```

selinux --disabled
firewall --disabled

```

Table 2. Installation methods for compute node

Method	Boot Location	Kickstart Location	Media Type
Manual	DVD	None	DVD

Automated DVD	DVD	USB	DVD
Automated NFS	DVD	NFS	NFS
Automated PXE	PXE	NFS	NFS

## DVD Based Booting with Kickstart on USB

Before we start, this method has one caveat: depending on the hardware configuration of your system, when the installer boots, the usb *may* show up as a different drive than normal. For example, in a configuration with two hard drives sda and sdb and a usb drive sdc, upon booting the usb may be sda while the hard drives are sdb and sdc. This scenario would require the kickstart file to be edited so that all references to any sdX are shifted one letter.

1. Insert a blank ext2 or fat32 formatted usb drive into the head node.
2. Mount the usb.
3. Copy the kickstart file to the usb drive. Make sure the repository line is set to `cdrom`

4. You can now use this kickstart file while booting from the DVD by pressing tab at the initial welcome screen and appending the following to the boot options:

```
inst.ks=hd:sdX1:/ks.cfg
```

Note that the drive letter X for the usb might not be the same as under the running operating system. The installer will allow you to edit this line if it fails to find the kickstart file on the specified device.

## DVD Based Booting with Kickstart and Distro on NFS

1. In this section it is assumed that the directory structure on the headnode is configured as follows:
  - `/admin/boot` contains the contents of the DVDs as in Installation Sans Removable Media;
  - `/admin/ks/computenode/ks.cfg` is the kickstart file for the compute nodes.
  - The ip address of the interface of the headnode connected to the internal network is 192.168.1.100
  - The headnode is configured with DHCP (see DHCP below).

If your setup differs, translate the following commands accordingly.

2. On the headnode, open up the file `/etc/exports` in a text editor and add the following:

```
/admin 192.168.1.100/255.255.255.0(ro,sync,no_root_squash)
```

This gives all computers on the 192.168.1.0/24 subnet read only access to the `/admin` directory.

3. Reload the NFS export table.  
# `systemctl restart nfs`
4. Temporarily disable the firewall to allow nfs traffic through. We will fix the firewall later.  
# `systemctl stop firewalld`
5. In the compute nodes' `ks.cfg` file, comment out the repository line and replace it with:

```
nfs --server=192.168.1.100 --dir=/admin/boot
```

6. You can now use the kickstart file while booting from the DVD by pressing tab at the initial welcome screen and appending the following to the boot options:  

```
ks=nfs:192.168.1.100:/admin/ks/computenode/ks.cfg ksdevice=ethX
```
7. To make the changes on the headnode persistent between installs, at the end of the ks.cfg file for the headnode between the %post and %end tags add the following:  

```
#nfs
echo "/admin 192.168.1.100/255.255.255.0(rw, sync, no_root_squash)" >>
/etc/exports
systemctl enable nfs
```

## PXE based NFS

See the PXE section below for more information. This assumes you have already completed DVD Based Booting with Kickstart and Distro on NFS.

# VIII. Basic Service Configuration

## A. DHCP

DHCP allows the compute nodes to receive their network configuration from a central server. This step is vital for a scalable system because it allows for the nodes to be identical in configuration.

1. Install dhcp  

```
# yum install dhcp
```
2. add the following to /etc/dhcp/dhcpd.conf  

```
#dhcpd config options
authoritative;
default-lease-time -1;
option broadcast-address 192.168.1.255;
ddns-update-style none;
next-server 192.168.1.100;
filename "pxelinux.0";

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.101 192.168.1.199;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 8.8.8.8;
    option routers 192.168.1.100;
}
```

Change the range option to include enough addresses for all of your nodes. Note that bolded options may need to be adjusted if the headnode's internal IP is not 192.168.1.100. The domain-name-servers option may point to whatever DNS you desire, in this example we use google's at 8.8.8.8.

3. To start the DHCP service on the head node at boot time, execute the command  

```
# systemctl enable dhcpd
```

And to start the service immediately, execute the command  

```
# systemctl start dhcpd
```
4. To automate this process, at the end of the ks.cfg file, append dhcp to the %packages list, and between the %post and %end tags add the following:  

```
#dhcp
cp -p /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.000
cat > /etc/dhcp/dhcpd.conf << EOF
#dhcpd config options
authoritative;
```

```

default-lease-time -1;
option broadcast-address 192.168.1.255;
ddns-update-style none;
next-server 192.168.1.100;
filename "pxelinux.0";

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.101 192.168.1.199;
    option subnet-mask 255.255.255.0;
    option domain-name-servers 8.8.8.8;
    option routers 192.168.1.100;
}
EOF
systemctl enable dhcpd

```

The method for writing this text to a file is called a Here File. As opposed to using the echo command, Here Files have far fewer characters that must be escaped, however there are some characters that still need to be. An example is that every \$ or ` symbol must be escaped with \\$ or ` otherwise Bash will attempt to resolve it as a variable or executable script.

5. On the compute nodes, the ks.cfg file can be edited to use dhcp. If you previously used a static configuration, change the network line to read:

```
network --bootproto=dhcp --device=ethX --ipv6=auto --activate
```

## B. firewalld

Firewalld is an abstraction layer for netfilter, and is the default firewall for CentOS. Only the headnode needs to have a firewall configured because it is the only node that has direct contact with the outside world. The compute nodes should already have had theirs disabled in their kickstarts so that their firewalls don't interfere with internal communication.

To configure firewalld execute the following firewall-cmd commands:

```

# firewall-cmd --permanent --zone=internal --add-source=[IP/MASK OF YOUR
INSTITUTION]
# firewall-cmd --permanent --zone=internal --remove-service=dhcpv6-client
# firewall-cmd --permanent --zone=internal --remove-service=ipp-client
# firewall-cmd --permanent --zone=internal --add-service=samba
# firewall-cmd --permanent --zone=internal --add-rich-rule='rule protocol
value="icmp" accept'

# firewall-cmd --permanent --zone=public --remove-service=ssh
# firewall-cmd --permanent --zone=public --remove-service=dhcpv6-client
# firewall-cmd --permanent --zone=public --add-service=http
# firewall-cmd --permanent --zone=public --set-target=DROP
# firewall-cmd --permanent --zone=public --change-interface=[EXTERNAL INTERFACE]
# echo "ZONE=public" >> /etc/sysconfig/network-scripts/ifcfg-[EXTERNAL INTERFACE]

# firewall-cmd --permanent --zone=trusted --change-interface=[NODE INTERFACE]
# echo "ZONE=trusted" >> /etc/sysconfig/network-scripts/ifcfg-[NODE INTERFACE]

# nmcli con reload
# firewall-cmd --reload

```

[IP/MASK OF YOUR INSTITUTION] can be the network address and netmask for your school, business, etc. such as 209.140.200.0/18

[EXTERNAL INTERFACE] is the interface connected to the outside world.

[NODE INTERFACE] is the interface connected to the internal compute node network.

When adding this to your kickstart file, you need to use `firewall-offline-cmd` instead of `firewall-cmd`. In addition, `--remove-service` will have to be changed to `--remove-service-from-zone`.

## C. PXE

Pre-boot eXecution Environment (PXE), called MBA on some BIOSes, allows nodes to retrieve their boot media via the network, making it possible to perform a kickstart installation on the nodes without having to physically load any disks. Here are some basic prerequisites before using PXE:

- Your motherboard supports PXE
- PXE is enabled in your BIOS
- PXE is set before local boot methods on the BIOS boot order
- DHCP is set up on the server. If necessary see the section on DHCP.
- Syslinux, tftp-server, and tftp are installed on the headnode.

1. On the headnode, make and populate the directory `/admin/tftpboot`

```
# mkdir /admin/tftpboot
# cp /usr/share/syslinux/pxelinux.0 /admin/tftpboot/
# cp /usr/share/syslinux/menu.c32 /admin/tftpboot/
# mkdir -p /admin/tftpboot/images/centos7/
```

These next two lines assume the contents of the CentOS iso files are in `/admin/boot` as done in *Installation Sans Removable Media*.

```
# cp /admin/boot/images/pxeboot/vmlinuz /admin/tftpboot/images/centos7/
# cp /admin/boot/images/pxeboot/initrd.img /admin/tftpboot/images/centos7/
# mkdir /admin/tftpboot/pxelinux.cfg
```

2. Create the new file `/admin/tftpboot/pxelinux.cfg/default` containing the following:

```
DEFAULT menu.c32
PROMPT 0
TIMEOUT 100
ONTIMEOUT local
```

```
MENU TITLE PXE Menu
```

```
MENU seperator
LABEL local
LOCALBOOT 0
```

```
MENU seperator
LABEL kickstart
kernel images/centos7/vmlinuz
append initrd=images/centos7/initrd.img
ks=nfs:192.168.1.100:/admin/ks/computenode/ks.cfg ksdevice=ethX
```

Note that the append line is all one line including the `ks` and `ksdevice` arguments. Edit the `ksdevice` as needed.

3. Edit the file `/usr/lib/systemd/system/tftp.service`, changing the line

```
ExecStart/usr/sbin/in.tftpd -s /var/lib/tftpboot
```

to

```
ExecStart/usr/sbin/in.tftpd -s /admin/tftpboot
```

and restart the service so that these changes take effect.

4. The first time you test it, make sure that the firewall is disabled on the headnode.

```
# systemctl stop firewalld
```

If selinux hadn't been disabled before through the kickstart file, disable it now.

```
# setenforce 0
```

When `pxelinux.0` boots on a machine, it tftp's back to the boot server and tries to find a configuration file in the directory `pxelinux.cfg`. The filename is determined by converting the IP address given it by the DHCP server to hex. An example is below:

```
192      168    1     101
C0       A8    01    65    → C0A80165
```

`pxelinux.0` attempts to find files in `pxelinux.cfg` in the following order:

```
C0A80165
C0A8016
C0A801
C0A80
C0A8
C0A
C0
C
default
```

This PXE feature is useful for giving specific ks instructions to different sets of compute nodes because of hardware differences; for example, it can supply different partitioning schemes for different hard disk sizes.

## D. *Advanced DHCP*

Sooner or later a compute node will have problems. To associate physical devices with IP addresses one must configure DHCP to give out IP addresses based on hardware MAC addresses.

1. Start `dhclient` so that addresses are logged.  
# `dhclient`
2. Power off all compute nodes.
3. Power on the nodes in order.
4. On the head node view the file `/var/lib/dhcpd/dhcpd.leases`. This should contain entries with the IP and MAC addresses of recently connected compute nodes.
5. Append entries to the very bottom of `/etc/dhcp/dhcpd.conf` for each node. For example:

```
host ernst01 {
    hardware ethernet 00:04:23:c0:d5:5c;
    fixed-address 192.168.1.101;
}
```

## E. */etc/hosts*

This file is used to pair hostnames with IP addresses. The general format of the file is:

```
xxx.xxx.xxx.xxx      hostname.domain      shorthostname
```

The shorthostname field is optional, but saves typing in many situations. We will be adding at least one line for each compute node in the cluster. Make sure that a line containing the following is also in the file as it is required for proper function of certain linux programs.

```
127.0.0.1      localhost.localdomain      localhost
```

An example of `/etc/hosts` on a headnode with 2 compute nodes

```
127.0.0.1          localhost.localdomain localhost
209.140.200.205   ernst.chem.hope.edu  ernst

192.168.1.100     ernst00
192.168.1.101     ernst01
192.168.1.102     ernst02
```

The hosts file for the compute nodes should be identical to that of the headnode.

## F. RSH

Rsh (remote shell) offers similar functionality to ssh (secure shell) without the encryption and corresponding hassle of keys. While a poor solution for logging in over a modern external network, it is an ideal technology for communication within the cluster. On each computer (using kickstarts, you don't want to type it all),

1. Install rsh

```
# yum install rsh rsh-server
```

2. Append the following lines to `/etc/securetty`

```
rsh
rexec
rsync
rlogin
```

3. Create the file `/root/.rhosts` where each line follows the pattern `[HOSTNAME] root`.

For example,

```
ernst00 root
ernst01 root
ernst02 root
```

4. Create the file `/etc/hosts.equiv`, in which each line is a hostname. For example,

```
ernst00
ernst01
ernst02
```

5. Enable and start the sockets

```
# systemctl enable rsh.socket
# systemctl enable rexec.socket
# systemctl enable rlogin.socket
# systemctl start rsh.socket
# systemctl start rexec.socket
# systemctl start rlogin.socket
```

## G. btools

btools are a set of scripts used to automate the execution of commands and tasks across all compute nodes in a cluster. While "atools" would be the commands themselves typed by some poor system administrator, and ctools (which actually exist) are complex gui cluster managements suites, btools fit somewhere in the middle. What follows is the list of btools and required support files. Feel free to modify them as needed.

- `/usr/local/sbin/bhosts` contains the hostnames of all compute nodes. For example:

```
ernst01
ernst02
```
- `/usr/local/sbin/brsh` loops through the hosts in `bhosts`, executing `rsh <some command>` for each.

```
#!/bin/sh
```

```

#
#Broadcast rsh
for host in `cat /usr/local/sbin/bhosts`
do
    echo "*****${host}*****"
    rsh ${host} $*
done

```

- /usr/local/sbin/bexec is similar to brsh in that it executes commands over all nodes. The difference is that bexec is designed to execute things that take a while in a timely manner. Where brsh waits for the first node to finish before moving on to the second, bexec gets them all started, then collects and displays the logs for the operations.

```

#!/bin/sh
# The total number of nodes (determined dynamically)
let nhost=0
# Run the command on each node, logging output
for host in `cat /usr/local/sbin/bhosts`
do
    logfile="/tmp/${host}.$$log"
    echo "***** ${host} *****" > $logfile
    rsh ${host} $* >> $logfile &
    pids[nhost]=$!
    let nhost=nhost+1
done

# Wait for all processes to finish
for ((i=0; i < nhost; i++))
do
    wait ${pids[$i]}
done

# Concatenate the results and cleanup
for host in `cat /usr/local/sbin/bhosts`
do
    logfile="/tmp/${host}.$$log"
    cat $logfile
    rm $logfile
done

```

- /usr/local/sbin/bpush copies a file to all nodes. Similarly to bexec, it executes simultaneously, then retrieves and displays logs.

```

#!/bin/sh
# The total number of nodes (determined dynamically)
let nhost=0
# Run the command on each node, logging output
for host in `cat /usr/local/sbin/bhosts`
do
    logfile="/tmp/${host}.$$log"
    echo "***** ${host} *****" > $logfile
    rcp $1 ${host}:%2 >> $logfile &
    pids[nhost]=$!
    let nhost=nhost+1
done

# Wait for all processes to finish
for ((i=0; i < nhost; i++))
do
    wait ${pids[$i]}
done

# Concatenate the results and cleanup
for host in `cat /usr/local/sbin/bhosts`
do

```

- ```

        logfile="/tmp/${host}.$$$.log"
        cat $logfile
        rm $logfile
    done

```
- /usr/local/sbin/bfiles contains:

```

/etc/passwd
/etc/group
/etc/shadow
/etc/gshadow

```
  - /usr/local/sbin/bsync copies the files defined in bfiles to all compute nodes. This causes all users on the headnode to be on the compute nodes as well.

```

#!/bin/sh
#
for host in `cat /usr/local/sbin/bhosts`
do
    echo "Synching ${host}"
    for file in `cat /usr/local/sbin/bfiles`
    do
        echo "Copying ${file}"
        rsync --rsh=rsh ${file} ${host}:${file}
    done
done

```

## H. NFS

NFS, or Network File System, is used to share files between the head node and the compute nodes. If you haven't done so already, complete the section on firewalld first. NFS uses dynamic port mappings so the firewall must be set up properly on the head node to allow all communication coming from the compute nodes.

1. The general format of the configuration file, /etc/exports, is as follows:

```

/exportdir ipaddr/netmask(options)

```

Modify yours to share /home and /admin. For example:

```

/home 192.168.1.100/255.255.255.0(rw,sync,no_root_squash)
/admin 192.168.1.100/255.255.255.0(ro,sync,no_root_squash)

```

2. Restart nfs and enable it at boot time

```

# systemctl restart nfs
# systemctl enable nfs

```

3. Import the shares on the compute nodes by appending the following lines to /etc/fstab.

Example entries for the above two nfs shares:

```

ernst:/home /home nfs rw,hard,intr 0 0
ernst:/admin /admin nfs ro,hard,intr 0 0

```

ernst is the name of the head node, and /home is one of the shares defined in

/etc/exports on the head node. The second /home tells the OS where to mount the share on the compute node, and nfs lets the OS know that it should use nfs to mount the share. The remaining items on the line are options that specify how the mountpoint is treated.

4. The shares will now be mounted automatically on boot up, but may be mounted manually as follows:

```

# mount /home
# mount /admin

```

## I. NTP (chrony)

NTP is a protocol used by chrony to synchronize the time on the head node with some external source.

1. Find a working timeserver (if your institution has its own, that's the best!) and add it to /etc/chrony.conf on the headnode in the following format:

```
server 192.43.244.18 #time.nist.gov
```

The comment at the end of the server line is purely optional but can be helpful when looking at the file. If you have chosen more than one server you may add them in a similar fashion.

2. Comment out the `server X.centos.pool.ntp.org iburst` lines as they will interfere with your chosen server.
3. Allow your internal compute nodes to be clients of this ntp server by uncommenting:  

```
allow 192.168/16
```
4. Set the nodes to use the headnode as their time server by adding the following line to `/etc/chrony.conf` on the compute nodes.  

```
server 192.168.1.100 # headnode
```
5. Enable and start chrony on both head and compute nodes.  

```
# systemctl enable chronyd  
# systemctl start chhronyd
```
6. After a few minutes have passed and NTP has had a chance to synchronize, execute the following commands to test your ntp configuration.  

```
# chronyc tracking  
# chronyc sources -v
```

The `-v` option on the `sources` command prints cleverly formatted explanations for each column in the output. Together, these commands print out information about the time server connection that is useful for debugging. They can be run on either the head node to show info about your external time server(s) or on a compute node to print info about the time server on your head node.

## J. Traffic Forwarding

Traffic forwarding allows compute nodes to access networks outside of the cluster. This can be useful for downloading updates from a central server directly to the compute nodes. This step is purely optional, especially since we will be setting up the headnode to be an update server later. Traffic forwarding can be permanently enabled or can be switched on and off.

To make traffic forwarding available, verify that `/etc/dhcp/dhcpd.conf` contains the following two lines:

```
option domain-name-servers 8.8.8.8;  
option routers 192.168.1.100;
```

Where 8.8.8.8 is the DNS the nodes will connect to and 192.168.1.100 is the internal network IP of the headnode.

### Temporary Forwarding:

Echo a 1 into `/proc/sys/net/ipv4/ip_forward`

```
# echo 1 > /proc/sys/net/ipv4/ip_forward - Enables
```

```
# echo 0 > /proc/sys/net/ipv4/ip_forward - Disables
```

and punch a hole in the current firewall session

```
# firewall-cmd --zone=public --add-masquerade - Enables
```

```
# firewall-cmd --zone=public --remove-masquerade - Disables
```

### Permanent forwarding:

So that changes from the above commands persist across reboots, in the file `/etc/sysctl.conf` add the line

```
net.ipv4.ip_forward = 1
```

and modify the firewall:

```
# firewall-cmd --permanent --zone=public --add-masquerade
```

## K. HTTP

Every project will have its own http requirements. A basic webserver can be set up simply by installing and starting httpd:

```
# yum install httpd
# systemctl enable httpd
# systemctl start httpd
```

The default webserver will display an apache page when you enter your IP (or a domain name if you can bribe your network administrator to set it up) in a web browser. To display your own content, populate `/var/www/html/` with the file `index.html`. This will be the main webpage for your site. Other pages may be included in the same directory or subdirectories.

## L. Samba

Samba is software that allows Windows computers to access files on your linux server.

1. Install samba:

```
# yum install samba
```

2. Back up `/etc/samba/smb.conf` and replace it with the following:

```
[global]
    workgroup = [NAME OF WORKGROUP]
    server string = [MY SERVER STRING]
    unix password sync = yes
    pam password change = yes
    interfaces = lo [OTHER INTERFACES]
    security = user
    passdb backend = tdbsam
    load printers = no
[homes]
    comment = Home Directories
    browseable = no
    writable = yes
    csc policy = disable
    inherit permissions = yes
```

In the above configuration,

- `[NAME OF WORKGROUP]` corresponds to the workgroup on the windows computers that will be connecting. To discover this, at the windows cmd prompt enter `net config workstation` and one of the outputted lines should resemble:  
Workstation domain                      WORKGROUP
- The server string in the config file is completely arbitrary.
- Make sure that `[OTHER INTERFACES]` includes the interface that connects to the outside world.

3. Enable and start samba:

```
# systemctl enable smb.service
# systemctl start smb.service
```

4. Setup existing system users with samba passwords:

```
# smbpasswd -a USERNAME
```

If the firewall is configured as it is in the FirewallD section, you should now be able to access user home directories from your local network.

If the system user wishes to keep their unix and samba passwords in sync, then when the user changes passwords he/she should use `smbpasswd` rather than `passwd` because `smbpasswd` is configured to change both the samba and linux passwords.

## M. Yum local repository

### 1. Clone CentOS repositories

Prepare a good spot to store the repository.

```
# mkdir -p /admin/software/repo/
```

Sync with a mirror.

```
# rsync -azHhv --delete some-mirror.org::CentOS/7* /admin/software/repo/
```

Note the syntax for specifying the folder to be synchronized. If the folder is

```
rsync://mirrors.liquidweb.com/CentOS/7, it would be written as
```

```
mirrors.liquidweb.com::CentOS/7
```

This should use about 40G of disk space and take several hours. You can use the same command to update your local copy of the repo. Updates will proceed much more quickly than the first copy.

### 2. Edit Yum configuration on Head Node and Compute Nodes

The files in `/etc/yum.repos.d/` need to be edited so that the head node will use itself as the update server, and the compute nodes will use their nfs mount of `/admin` as the update server. To do this, edit all the files with your favorite text editor and change the line

```
#baseurl=http://mirror.centos.org/centos/$releasever....
```

to

```
baseurl=file:/admin/software/repo/$releasever....
```

and comment out all mirrorlist lines. This can be automated using the following sed commands:

```
# sed -i.000 "s#baseurl=http://mirror.centos.org/centos&baseurl=file:/admin/software/repo&" /etc/yum.repos.d/*.repo
```

```
# sed -i "s#mirrorlist&#mirrorlist&" /etc/yum.repos.d/*.repo
```

If you have installed additional repos (such as epel on the headnode for ganglia) make sure not to modify their .repo files.

### 3. Update nodes

Updating the nodes involves running the following as the root user on each node.

```
# yum upgrade
```

### 4. Create your own repo

There are times when you will need to install software to your compute nodes that is not available in the repo that you cloned, for example, ganglia from epel. In this case, it is not efficient to clone the entire epel repository for only a few software packages. If you have many compute nodes, it can slow the network down immensely to allow forwarding in your firewall and download the software from epel to each node individually. Thus, the possible solutions then are to download the rpm files to the headnode, `push` them to the compute nodes, and `bexec rpm -ivh /path/to/software.rpm`, or to create your own specialized repository.

a. Create a directory to house the repo in the headnode

- ```
# mkdir /admin/software/localrepo
```
- b. Download the rpms desired to your local repo, in this example, the required ones for compute nodes for ganglia
 

```
# cd /admin/software/localrepo
```

 (If you haven't already, install epel-release, a 3<sup>rd</sup> party repository containing ganglia, amongst many others)
 

```
# yum install epel-release
# yumdownloader ganglia ganglia-gmond ganglia-gmetad libconfuse
```
  - c. Create the repo metadata
 

```
# createrepo /admin/software/localrepo
```
  - d. Create `/etc/yum.repos.d/local.repo` and populate it with the following:
 

```
[local]
name=CentOS local repo
baseurl=file:///admin/software/localrepo
enabled=1
gpgcheck=0
protect=1
```
  - e. Push `local.repo` to your compute nodes
 

```
# bpush /etc/yum.repos.d/local.repo /etc/yum.repos.d/
```
  - f. Install the desired software on compute nodes like normal
 

```
# bexec yum -y install ganglia-gmond ganglia-gmetad
```
  - g. If you want to add software to the local repo, simply repeat steps b and c.

## N. Ganglia

Ganglia is a cluster monitoring software suite that reports the status of all the nodes in the cluster over http.

### Prerequisites

- Install httpd
- Create a local repository containing ganglia, ganglia-gmond, ganglia-gmetad, and libconfuse

### 1. Install required software

On the headnode, install ganglia-gmetad, ganglia-gmond, and ganglia-web

```
# yum -y install ganglia ganglia-gmetad ganglia-gmond ganglia-web
```

On compute nodes, install ganglia-gmetad and gnanglia-gmond

```
# bexec yum -y install ganglia ganglia-gmetad ganglia-gmond
```

### 2. Modify `/etc/ganglia/gmond.conf`

On the headnode, backup and edit `/etc/ganglia/gmond.conf` such that

- In the `cluster` block, `name` is something you will recognize
- In the `udp_send_channel` block, `mcast_join` is commented out and the line `host = 192.168.1.100` (internal ip of headnode) is added
- In the `udp_recev_channel` block, `mcast_join` and `bind` are both commented out.

Push the modified `gmond.conf` file to all compute nodes

```
# bpush /etc/ganglia/gmond.conf /etc/ganglia/
```

### 3. Enable and start gmetad and gmond

```
# systemctl enable gmetad
# systemctl start gmetad
# systemctl enable gmond
```

```
# systemctl start gmond
# bexec systemctl enable gmetad
# bexec systemctl start gmetad
# bexec systemctl enable gmond
# bexec systemctl start gmond
```

#### 4. Configure php

In `/usr/share/ganglia/conf.php`, between the `<?php` and `?>` tags, add the line  
`$conf['gweb_confdir'] = "/usr/share/ganglia";`

#### 5. Configure http

Edit `/etc/httpd/conf.d/ganglia.conf` such that it reflects the following:

```
Alias /ganglia /usr/share/ganglia

<Directory "/usr/share/ganglia">
    AllowOverride All
    Require all granted
</Directory>
```

And restart httpd

```
# systemctl restart httpd
```

6. Monitor your cluster from a web browser by going to `http://your.domain.name/ganglia`.

## O. Torque

### Prerequisites

- Setup either rsh or passwordless ssh for users between the headnode and compute nodes in both directions.
- Install btools (or be prepared to do things manually on nodes).
- Install libtool, openssl-devel, libxml2-devel, boost-devel, gcc, gcc-c++, and git.
- Allow all network traffic between headnode and compute nodes.
- Share `/home` with all nodes over nfs.
- Ensure all nodes (including the headnode) have identical `/etc/hosts` files including an entry for each node.
- Ensure all nodes know all users (you can use `bsync`).

#### 1. Download Torque

Clone the source from github:

```
# git clone https://github.com/adaptivecomputing/torque.git -b 6.0.1 6.0.1
# cd 6.0.1
# ./autogen.sh
```

#### 2. Install Torque

- **With rsh:**

Configure, make, and install:

```
# ./configure --with-rcp=rcp
# make
# make install
```

- **Or With rsh:**

Configure, make, and install:

```
# ./configure
# make
# make install
```

### 3. Configure Torque on headnode

Make sure Torque is using the hostname on the external network (e.g., ernst.chem.hope.edu):

```
# echo [correct_hostname] > /var/spool/torque/server_name
```

Configure the library path

```
# echo "/usr/local/lib" > /etc/ld.so.conf.d/torque.conf
# ldconfig
```

Populate /var/spool/torque/server\_priv/nodes. A basic example follows:

```
node01 np=1
node02 np=1
```

Start the trqauthd daemon:

```
# cp contrib/systemd/trqauthd.service /usr/lib/systemd/system/
# systemctl enable trqauthd.service
# systemctl start trqauthd.service
```

Initialize serverdb:

```
# ./torque.setup root
# qterm
```

Start pbs\_server:

```
# cp contrib/systemd/pbs_server.service /usr/lib/systemd/system/
# systemctl enable pbs_server.service
# systemctl start pbs_server.service
```

### 4. Install Torque MOMs on compute nodes

On the server, in the source directory, build packages for the nodes:

```
# make packages
```

Copy contrib/systemd/pbs\_mom.service to /usr/lib/systemd/system/ on all compute nodes. If using btools, this can be accomplished by the following:

```
# bpush contrib/systemd/pbs_mom.service /usr/lib/systemd/system/
```

Install torque-package-mom-linux-x86\_64.sh and torque-package-clients-linux-x86\_64.sh to all compute nodes:

```
# bpush torque-package-mom-linux-x86_64.sh
# bpush torque-package-clients-linux-x86_64.sh
# bexec ./torque-package-mom-linux-x86_64.sh --install
# bexec ./torque-package-clients-linux-x86_64.sh --install
```

Configure the compute node library paths

```
# bpush /etc/ld.so.conf.d/torque.conf /etc/ld.so.conf.d/
# bexec /sbin/ldconfig
```

Make sure the nodes are using the headnode's hostname on the internal node network for the server name (e.g., ernst00):

```
# bexec 'echo [correct_hostname] > /var/spool/torque/server_name'
```

Start the pbs\_mom service:

```
# bexec systemctl enable pbs_mom.service
# bexec systemctl start pbs_mom.service
```

## 5. Configure the scheduler

On the headnode, copy the scheduler service file to the correct location:

```
# cp contrib/systemd/pbs_sched.service /usr/lib/systemd/system/
```

Enable and start the scheduler:

```
# systemctl enable pbs_sched.service
# systemctl start pbs_sched.service
```

## 6. Test the system

Verify that you can ssh or rsh from the compute node to the headnode as the user that is to be running the jobs.

Make sure all nodes are reporting:

```
# pbsnodes -a
```

As a non-root user, run a test interactive job:

```
$ qsub -I
```

Exit from the resulting shell and run a job that returns something:

```
$ echo "date" | qsub
```

If successful, two files `STDIN.oXX` and `STDIN.eXX` should appear in your working directory.

If not, you should receive mail with an error report.

Look at a job while it is running:

```
$ echo "sleep 10" | qsub
$ qstat
```

This should display that the queue has a running job in it.