Intel® Scalable System Framework (Intel® SSF) Reference Design

Cluster installation for systems based on Intel® Xeon® processor E5-2600 v4 family including Intel® Ethernet.

Based on OpenHPC v1.1

Version 1.0

Summary

This Reference Design is part of the Intel® Scalable System Framework series of reference collateral.

The Reference Design is a verified implementation example of a given reference architecture, complete with hardware and software Bill of Materials information and cluster configuration instructions. It can confidently be used "as is", or be the foundation for enhancements and/or modifications.

Additional Reference Designs are expected in the future to provide example solutions for existing reference architecture definitions and for utilizing additional Intel® SSF elements. Similarly, more Reference Designs are expected as new reference architecture definitions are introduced.

This Reference Design is developed in support of the specification listed below using certain Intel® SSF elements:

- Intel® Scalable System Framework Architecture Specification
- Servers with Intel® Xeon® processor E5-2600 v4 family processors
- Intel® Ethernet
- Software stack based on OpenHPC v1.1

Legal Notices

Table of Contents

Design

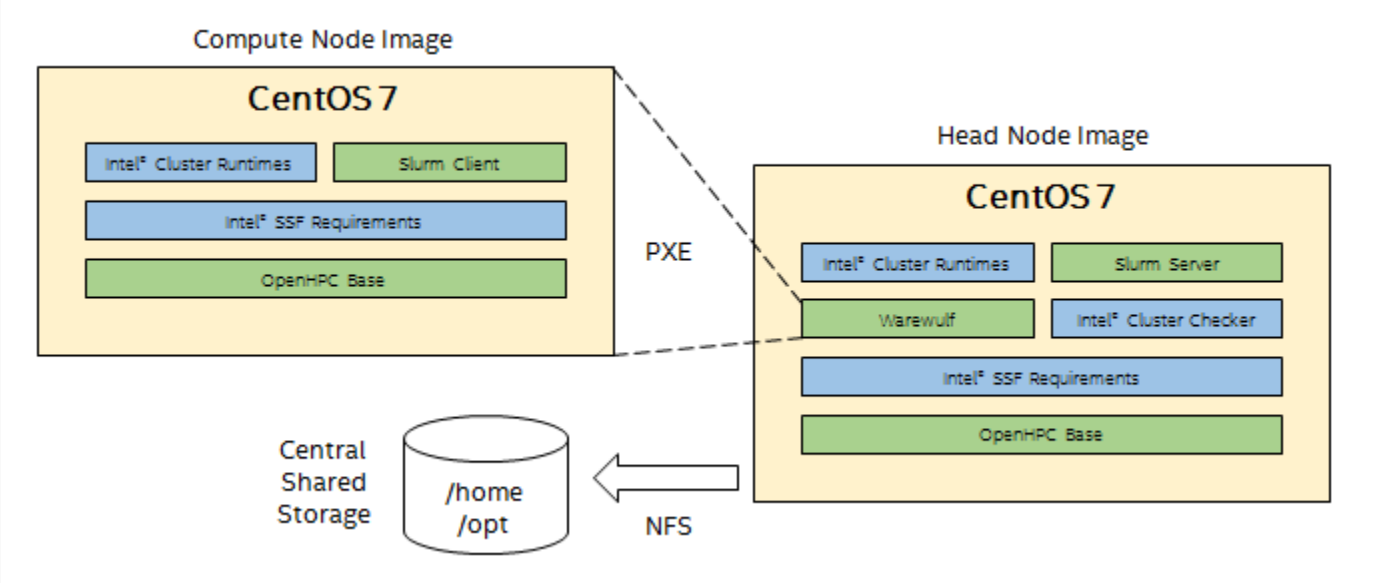
Hardware

Quantity	Item	Manufacturer	Model
5	Intel® Server Chassis	Intel	R2000WT
5	Intel® Server Board (w/ 10Gb Intel® Ethernet Controller)	Intel	S2600WTT
	(2x) Intel® Xeon® Processor	Intel	E5-2600 V4
	(8x) 8GB ECC DDR4 2133Mhz	Micron	MTA18ASF1G72PZ
	Intel® SSD 800GB, 2.5-inch SATA	Intel	S3700 Series
1	Low Latency Gigabit[1] Ethernet Switch	Hewlett-Packard	J2724A ProCurve* Switch

[1] Cluster tested using a 1GbE switch. We highly recommend using a 10GbE switch for optimal performance and will release an updated reference design in the near future.

Software

Software	Version
CentOS* Linux* Installation DVD	7.2.1511
Intel® Cluster Checker	3.1.2
Intel® Cluster Runtimes	3.8
Intel® SSF meta-RPMS for EL7	2016.0
OpenHPC	1.1



Node conventions

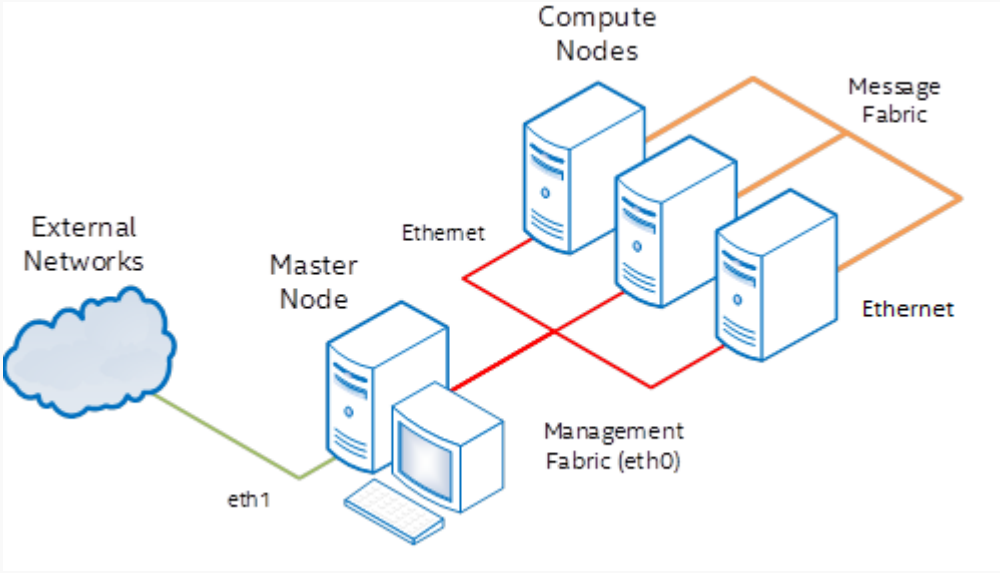
The following node conventions are used in the document as defaults. In general, other values may be substituted for these without affecting the installation. Node numbering, represented by 'XX' in the list below, begins at 01, supporting up to 99 nodes using these default conventions.

- Domain Name: .localdomain
- Subcluster name: default
- Head node name: frontend
- Network masks: 255.255.255.0 (/24)
- Head node IP address: 192.168.1.100
- Compute node names: nXX
- Compute node IP addresses: 192.168.1.XX
- VNFS Image Name: compute-centos72

Preparation

Assembly

This cluster is simple Beowulf style, consisting of a single head node managing all cluster functions and one or more compute nodes for processing.



Collect MAC Addresses

Warewulf provides a method to gather MAC addresses as they are booted. However, this method is inefficient for large clusters. The preferred method to identify nodes is pre-collection of MAC addresses. In order to complete these instructions, the MAC addresses for the first Ethernet port on each system must be recorded.

Procedure: Linux* Operating System Installation

Install the Linux* Operating System

1. **Insert the CentOS* 7.2 DVD. Boot from the CD and select "Install CentOS 7".**
2. **Select "English" as the language and click "Continue".**
3. **Select "DATE & TIME".**

   Select your timezone using the geographical location dropdown menus.

   Click "Done" to return home.

4. **Select "NETWORK & HOST NAME".**

   Enter "frontend.localdomain" as the host name.

   Select "Ethernet (enp3s0f0)" and click "Configure" to setup the internal cluster interface.

   - From the "General" section, check "Automatically connect to this network when it is available".
   - From the "IPv4 Settings" tab, select the "Manual" method and add the address 192.168.1.100 with netmask 24. Save and exit.

   Select "Ethernet (enp3s0f1)" and click "Configure" to setup the external cluster interface.

   - From the "General" section, check "Automatically connect to this network when it is available".
   - Configure the external interface as necessary for your use. Save and exit.

   Set the toggle is set to "ON" for both interfaces.

   Click "Done" to return home.

5. **If necessary, select "INSTALLATION DESTINATION".**

   Select the automatic partitioning option.

   Click "Done" to return home. Accept all defaults for the partitioning wizard if prompted.

6. **Click "Begin Installation".**
7. **While waiting for the installation to finish, set the root password. User creation is optional.**

Configure Software Repositories

The OpenHPC community provides a release package that includes GPG keys for package signing and YUM repository configuration. This package can be installed directly from the OpenHPC build server. In addition, the head node must have access to the standard CentOS 7.2 and EPEL repositories. Mirrors are readily available for both repositories.

The public EPEL repository is enabled automatically by the ohpc-release package. This requires that the CentOS Extras repository is enabled, which is default.

8. **Install the ohpc-release package.**

```
yum install http://build.openhpc.community/\
OpenHPC:/1.1/CentOS_7.2/x86_64/ohpc-release-1.1-1.x86_64.rpm
```

9. **Install the CentOS-7.2 (1511) repository.**

```
yum install http://mirror.centos.org/centos/7/os/x86_64/Packages/\
centos-release-7-2.1511.el7.centos.2.10.x86_64.rpm
```

Procedure: Configure the Head Node

The head node is configured as the primary node in the cluster, and set up to manage and install all compute nodes.

Temporarily disable selinux

SELinux can interfere with head node configuration. It is disabled until configuration is complete.

1. **Run setenforce.**

   ```
   setenforce Permissive
   ```

Add the head node to the hosts file.

The head node is not added automatically to the hosts file, so it is done manually

2. **Update /etc/hosts.**

   Open the /etc/hosts file for editing and add the following line to the end of the file.

   ```
   192.168.1.100 frontend.localdomain frontend
   ```

   Save and exit the file.

```
echo "192.168.1.100 frontend.localdomain frontend" >> /etc/hosts
```

Install packages

Packages that were part of the minimum installation are upgraded to assure that they are the latest version. Since new packages are downloaded directly from the online repository, they will be the latest version.

To add support for provisioning services, install the base OpenHPC group package followed by the Warewulf provisioning system packages.

3. **Upgrade all packages.**

   ```
   yum upgrade
   ```

   If necessary, accept any update requests by entering **Y** followed by **Enter**.

4. **Install OpenHPC base.**

   ```
   yum groupinstall ohpc-base
   ```

5. **Install Warewulf.**

   ```
   yum groupinstall ohpc-warewulf
   ```

Configure Warewulf

Warewulf will need to be updated to match settings for this cluster. In addition, hybridizing the compute node significantly reduces its image size without impacting performance, if rarely accessed directories are specified.

6. **Set Warewulf provisioning interface.**

   The provisioning interface is the internal device configured for the 192.168.1.100 IP address.

   a. Open the /etc/warewulf/provision.conf file.
   b. Replace “network device = eth1” with “network device = enp3s0f0”.
   c. Save and close the file.

```
sed "s/device = eth1/device = enp3s0f0/" /etc/warewulf/provision.conf
```

7. **Set the default subnet mask for all nodes.**

   Edit the /etc/warewulf/defaults/node.conf file and add the following lines.

```
cluster = default
groups = computenetmask = 255.255.255.0
network = 192.168.1.0
```

8. **Update node provisioning settings.**

   Edit the /etc/warewulf/defaults/provision.conf file and update the following lines.

```
bootstrap = 3.10.0-327.18.2.el7.x86_64
vnfs = compute-centos72
files = dynamic_hosts, passwd, group, shadow, gshadow
```

9. **Update VNFS defaults.**

   Edit the /etc/warewulf/vnfs.conf file so that the following lines are modified, added, and/or uncommented.

```
exclude += /tmp/*
exclude += /var/log/*
exclude += /var/tmp/*
exclude += /var/cache/*
exclude += /home/*
exclude += /opt/*

hybridpath = /opt/ohpc/admin/images/%{name}

hybridize += /usr/src
hybridize += /usr/lib/locale
hybridize += /usr/lib64/locale
hybridize += /usr/include
hybridize += /usr/share
```

Update Warewulf-required services

Enable TFTP service for compute node image distribution.

Enable HTTP access for Warewulf, in order to support the new Apache web server syntax.

10. **Update tftp.**

    Edit /etc/xinetd.d/tftp and replace “disable = yes” with “disable = no”.

```
sed "s/^\s+disable\s+= yes/ disable = no/" /etc/xinetd.d/tftp
```

11. **Restart the xinetd service.**

```
systemctl restart xinetd
```

12. **Update Web services configuration.**

    a. Open /etc/httpd/conf.d/warewulf-httpd.conf for editing.

    b. Locate “<Directory /usr/libexec/warewulf/cgi-bin>” and insert “Require all granted" as the next line.

    c. Locate “<Directory /usr/share/warewufl/www>”, and replace “Allow from all” with “Require all granted”.

    d. Delete “Order allow,deny”.

    When complete, the updated section will be this.

```
<Directory /usr/libexec/warewulf/cgi-bin>
    Require all granted
    SetHandler perl-script
    PerlResponseHandler ModPerl::Registry
    PerlOptions +ParseHeaders
    Options +ExecCGI
</Directory>

<Directory /usr/share/warewulf/www>
    Options Indexes MultiViews
    AllowOverride None
    Require all granted
</Directory>
```

13. **Enable the database and web services to start automatically.**

```
systemctl enable mariadb.service
systemctl enable httpd.service
```

14. **Restart the database and web services.**

```
systemctl restart mariadb
systemctl restart httpd
```

Configure files for distribution

The Warewulf system includes functionality to import files from the head node and distribute these to other nodes. This will be used to synchronize user and group information with compute nodes. Other services, such as Slurm, will use this method to distribute configuration files.

15. **Import files**

```
wwsh file import /etc/passwd
wwsh file import /etc/group
wwsh file import /etc/shadow
wwsh file import /etc/gshadow
```

Create the node image

The OpenHPC build of Warewulf includes enhancements and enabling for CentOS7.2. The wwmkchroot command creates a minimal chroot image for use with Warewulf, which will be created in /opt/ohpc/admin/images/compute-centos72.

To access the remote repositories by hostname (and not IP addresses), the **chroot** environment also needs to be updated to enable DNS resolution. If the head node has a working DNS configuration in place, the **chroot** environment can use this configuration file.

16. **Define chroot location.**

    For ease of use, this is made permanent by adding it to the root bashrc file.

    a. Add the following line to the /root/.bashrc file.

    ```
    export CHROOT=/opt/ohpc/admin/images/compute-centos72
    ```

    b. Source .bashrc

    ```
    . /root/.bashrc
    ```

```
echo "export CHROOT=/opt/ohpc/admin/images/compute-centos72" >> /root/.bashrc
. /root/.bashrc
```

17. **Build the initial chroot image.**

```
wwmkchroot centos-7 $CHROOT
```

18. **Update chroot name resolution.**

```
cp -p /etc/resolv.conf $CHROOT/etc/resolv.conf
```

Procedure: Configure the Cluster

The next task is to customize both the head node and compute node image, by adding additional components, including authentication, log file consolidation, shared storage, and time synchronization. Resource management and fabric support are added in their own sections. Additional software components are added to the compute image directly using yum

Install additional packages

The OpenHPC base package is added to the compute node image. In addition, the Intel® SSF meta-RPMs are installed on all systems, to meet Intel® SSF software stack requirements, as well as software advisories.

Kernel packages are also included. The kernel-devel package will be required for other installs.

The yum-utils package is included, as it provides a simple method to remove old packages when needed.

1. **Add the OpenHPC base package to the node image.**

```
yum --installroot=$CHROOT groupinstall ohpc-base
```

2. **Install Intel® SSF compliance packages.**

```
yum install ssf-meta-el7-2016.0-0.9.x86_64.rpm
yum install ssf-meta-el7-optional-2016.0-0.9.x86_64.rpm
yum --installroot=$CHROOT install ssf-meta-el7-2016.0-0.9.x86_64.rpm
yum --installroot=$CHROOT install ssf-meta-el7-optional-2016.0-0.9.x86_64.rpm
```

3. **Add yum utilities.**

```
yum install yum-utils
```

4. **Add kernel development source.**

```
yum install kernel-devel
```

5. **Add the modules user environment.**

   This is normally installed by default, but should be confirmed.

```
yum install lmod-ohpc
yum --installroot=$CHROOT install lmod-ohpc
```

Configure SSH keys

This will add the cluster key to the node image, allowing passwordless root access to all nodes.

6. **Create SSH keys.**

```
wwinit ssh_keys
```

7. **Authorize the new root key.**

```
cat /root/.ssh/cluster.pub >> $CHROOT/root/.ssh/authorized_keys
```

Configure the Firewall

Provisioning services for Warewulf use DHCP, TFTP, and HTTP network protocols and default firewall rules may block these services. However, disabling the firewall on an Internet-accessible device is insecure. The following changes will allow all connections within the cluster while maintaining the firewall on the external interface.

8. **Secure the external connection.**

```
nmcli connection modify enp3s0f1 connection.zone work
```

   The zone can be set to 'internal' or 'public' if more security is required.

9. **Disable filtering on the internal network.**

```
nmcli connection modify enp3s0f0 connection.zone trusted
```

10. **Commit the changes to the current configuration.**

```
nmcli connection reload
```

11. **Restart the firewall.**

```
systemctl restart firewalld
```

12. **Disable the firewall on the compute nodes, if it exists.**

```
chroot $CHROOT systemctl disable firewalld.service
```

Configure NTP

HPC systems typically rely on synchronized clocks throughout the system and the NTP protocol can be used to facilitate this synchronization.

On the compute node image, add the Network Time Protocol (NTP) support and identify the head node as the NTP server.

13. **Enable NTP services on the head node.**

```
systemctl enable ntpd.service
```

14. **Configure the NTP server.**

    a. Open /etc/ntp.conf for editing.

    b. Add and modify external time servers.

Delete or comment-out any lines to servers that are not used.

For each external time server to be used, add the line:

```
server <hostname or ip address>
```

c. Open the cluster subnets for unrestricted access.

Add the following lines to the file.

```
restrict 192.168.1.0 mask 255.255.255.0
```

d. Save and close the file.

15. Restart the time server.

```
systemctl restart ntpd
```

16. Enable NTP time service on compute nodes.

```
chroot $CHROOT systemctl enable ntpd.service
```

17. Configure the NTP client.

a. Open $CHROOT/etc/ntp.conf for editing.

b. Remove or comment-out all lines starting with “server”

c. Add the following line:

```
server 192.168.1.100
```

d. Save and close the file.

Configure NFS

The /home directory is shared for read and write across the cluster. The /opt directory is shared for read-only access to all nodes.

18. Add NFS client mounts.

Add the following two lines to $CHROOT/etc/fstab:

```
192.168.1.100:/home /home nfs nfsvers=4,rsize=1024,wsize=1024,cto 0 0
192.168.1.100:/opt /opt nfs nfsvers=4 0 0
```

19. Update exports.

Update /etc/exports to include only the following lines:

```
/home 192.168.1.0/24(rw,no_subtree_check,fsid=10,no_root_squash)
/opt 192.168.1.0/24(ro,no_subtree_check,fsid=11)
```

20. Install and enable the NFS server.

```
yum install nfs-utils
systemctl enable nfs-server.service
```

Enable system log forwarding

System logging for the cluster is can be consolidated to the head node, to provide easy access and reduce the memory requirements on the diskless compute node. You will disable most local logging on compute nodes, but emergency and boot logs will remain on them.

21. **Configure the head node to receive messages.**

    Edit /etc/rsyslog.conf and uncomment the following 2 lines by removing the '#':

    ```
    # $ModLoad imudp
    # $UDPServerRun 514
    ```

22. **Restart the service.**

    ```
    systemctl restart rsyslog
    ```

23. **Update the compute node configuration.**

    a. Open $CHROOT/etc/rsyslog.conf for editing.

    b. Add the following line in the file.

    ```
    *.* @192.168.1.100:514
    ```

    c. Comment-out the following lines by adding a '#' at the beginning of the line:

    ```
    *.info;mail.none;authpriv.none;cron.none
    authpriv.*
    mail.*
    cron.*
    uucp,news.crit
    ```

    d. Save and close the file.

Intel® Cluster Runtimes

The Intel® Cluster Runtimes are installed on the head node only, and shared through NFS.

24. **Extract the installer.**

    ```
    tar -zxvf intel_cluster_runtimes_3.8-1.tgz -C /usr/src/
    ```

25. **Install the runtimes.**

    a. Start the installer.

    ```
    /usr/src/intel_cluster_runtimes_3.8-1/install.sh
    ```

b. Press **Enter** to continue.

c. Read the EULA.

Press **Space** to scroll one page and continue until you reach the prompt.

d. Type the word “accept” and press **Enter**.

Intel® Cluster Checker

The Intel® Cluster Checker provides a convenient suite of diagnostics that can be used to aid in isolating hardware and software problems on an installed cluster.

A valid license file will need to be installed in order to use this software.

26. Create the Intel® Cluster Checker user account.

```
useradd clck
```

27. Install the Intel® Cluster Checker license file.

a. Create the directory.

```
mkdir -p /opt/intel/licenses
```

b. Install the file.

```
cp <license_file> /opt/intel/licenses
```

c. Set the permissions.

```
chmod 755 /opt/intel/licenses/*
```

28. Extract the Intel® Cluster Checker installation.

```
tar -zxvf l_clck_p_3.1.2.006.tgz -C /usr/src/
```

29. Install Intel® Cluster Checker.

a. Launch the installer.

```
/usr/src/l_clck_p_3.1.2.006/install.sh
```

b. At the Welcome screen, press **Enter** to continue.

The next screen is Step 2 of 6 | License Agreement.

c. Read the EULA.

Press **Space** to scroll each screen until you reach the prompt.

d. Type the word “accept” and press **Enter**.

e. At Step 3 of 6 | Activation, press **Enter**.

f. At Step 4 of 6 | Options, > Configure Cluster Installation, press **Enter**.

g. At Step 4 of 6 | Options > Pre-install Summary, press **Enter**.

h. At Step 5 of 6 | Installation, Press **Enter**.

i. At Step 6 of 6 | Complete, Press **Enter**.

Reboot the head node

The head node is rebooted so that any kernel updates are activated. SELinux will also be temporarily disabled again.

30. **Issue a reboot command.**

```
init 6
```

31. **Login as root.**

32. **Run setenforce.**

```
setenforce Permissive
```

Procedure: Install and Configure SLURM

The Slurm workload manager is added to the cluster. This is a client-server install with server components installed on head node.

Install Slurm on the head node

Slurm requires a system user for the resource management daemons. The default configuration file supplied with the OpenHPC build of Slurm requires that the “slurm" user account exist.

Similarly, to import the global Slurm configuration file and the cryptographic key that is required by the munge authentication library to be available on every host in the resource management pool, issue the following.

1. **Create the slurm user account.**

   ```
   useradd slurm
   ```

2. **Install Slurm server packages.**

   ```
   yum groupinstall ohpc-slurm-server
   ```

3. **Identify resource manager hostname on head node.**

   Open /etc/slurm/slurm.conf for editing and replace the ControlMachine entry with this:

   ```
   ControlMachine=frontend
   ```

```
sed "s/ControlMachine=\S+/ControlMachine=frontend/" /etc/slurm/slurm.conf
```

4. **Import files.**

   ```
   wwsh file import /etc/slurm/slurm.conf
   wwsh file import /etc/munge/munge.key
   ```

5. **Update the /etc/warewulf/defaults/provision.conf file.**

   Locate and replace the “files” line.

   ```
   files = dynamic_hosts, passwd, group, shadow, gshadow, slurm.conf, munge.key
   ```

Install Slurm Client

Slurm will require enumeration of physical hardware characteristics for compute nodes. The following settings are configured on each node by default:

- Sockets 2
- CoresPerSocket 8
- ThreadsPerCore 2

Update the configuration file at $CHROOT/etc/slurm/slurm.conf as needed to match hardware.

The option to restrict SSH access on compute nodes to only those users with active jobs on that node is included. This is enabled with an authentication module (PAM) provided in the Slurm package.

6. **Add Slurm client support.**

```
yum --installroot=$CHROOT groupinstall ohpc-slurm-client
```

7. **Update slurm.conf.**

   a. Open /etc/slurm/slurm.conf for editing.

   b. Update the node resource information. Replace the existing lines as needed. For example:

```
NodeName=n[AA-BB] Sockets=1 CoresPerSocket=68 ThreadsPerCore=2 State=UNKNOWN
```

   or

```
NodeName=n[CC-DD] Sockets=2 CoresPerSocket=12 ThreadsPerCore=2 State=UNKNOWN
```

   c. Update the partition setup. Replace the existing line with:

```
PartitionName=default Nodes=n[CC-DD] Default=YES MaxTime=24:00:00 State=UP
PartitionName=mp Nodes=n[AA-BB] Default=No MaxTime=24:00:00 State=UP
PartitionName=cluster Nodes=n[01-XX] Default=No MaxTime=24:00:00 State=UP
```

8. **Enable SSH control via resource manager.**

```
echo "account required pam_slurm.so" >> $CHROOT/etc/pam.d/sshd
```

Procedure: Install Compute Nodes

Warewulf employs a two-stage boot process for provisioning nodes: A bootstrap image that is used to initialize the kernel and install process and an encapsulated image containing the full system.

Assemble bootstrap image

The bootstrap image includes the runtime kernel and associated modules, as well as simple scripts for provisioning. Locations of all required drivers and firmware to boot the system must be configured.

NOTE: If the environment variable BASH_ENV is not empty, this will conflict with Warewulf scripts that use Perl with the -T switch.

1. **Edit /etc/warewulf/bootstrap.conf.**

   Add the following line:

   ```
   drivers += updates/kernel
   ```

2. **Clear BASH_ENV.**

   ```
   unset BASH_ENV
   ```

3. **Build bootstrap image for compute nodes.**

   ```
   wwbootstrap --chroot=$CHROOT 3.10.0-327.18.2.el7.x86_64
   ```

Assemble Virtual Node File System (VNFS) image

With the local site customizations in place, the following step uses the wwvnfs command to assemble a VNFS capsule from the chroot environment defined for the compute instance.

4. **Create the VNFS image.**

   ```
   wwvnfs --chroot $CHROOT
   ```

   The capsule is approximately 250MB in size. You can see that the VNFS capsules exists by executing:

   ```
   wwsh vnfs list
   ```

Register nodes for provisioning

In the steps below, the variable “XX” is used in hostnames, node IPs, and MAC addresses. It must be replaced by the node number (01-99).

Associations for nodes, boot images, and vnfs images are configured in the file /etc/warewulf/defaults/provision.conf.

5. **Add compute nodes to Warewulf datastore.**

   Repeat this command for each compute node in the cluster.

   ```
   wwsh node new nXX --ipaddr=192.168.1.XX --hwaddr=<mac_address> -D enp3s0f0
   ```

6. **Review the added nodes.**

   ```
   wwsh node list
   ```

7. **Restart dhcp.**

   ```
   systemctl restart dhcpd
   ```

8. **Update PXE.**

   ```
   wwsh pxe update
   ```

Boot compute nodes

The head node is now able to boot compute nodes remotely.

9. **Power-cycle or power-on each compute node.**

Procedure: Resource Manager Startup

The Slurm resource manager was installed and configured for use on both the head node and compute node instances. Once cluster nodes are operational, Slurm can be started. First, the controller on the head node is started, after which client daemons can be started on compute nodes.

Slurm uses the munge library to provide authentication services; this daemon must be running on all hosts that Slurm manages.

By default, the client hosts are initialized into an unknown state. A command is issued to place the hosts into production.

1. **Start munge and slurm controller on head node.**

```
systemctl enable munge.service
systemctl enable slurmctld.service
systemctl start munge
systemctl start slurmctld
```

2. **Start slurm clients on compute nodes.**

```
pdsh -w n[01-XX] systemctl start slurmd
```

3. **Put compute nodes into production.**

```
scontrol update nodename=n[01-XX] state=idle
```

Procedure: Add users and synchronize files

Warewulf automatically synchronizes imported files from the head node at five minute intervals. If a new user is created, several files will be outdated and the Warewulf database must be updated, after which the compute nodes will need to be updated. This resync process can be accomplished as follows:

```
wwsh file resync
```

After calling the resync command, it will take approximately 5 minutes for changes to propagate. However, you can manually pull the changes from compute nodes using the following command:

```
pdsh -w n[01-XX] /warewulf/bin/wwgetfiles
```

Procedure: Run Intel® Cluster Checker

Switch to clck user.

To run Intel® Cluster Checker, it is advised to switch to the clck user because the tool needs a shared directory.

```
su - clck
```

Create the nodelist.

The nodelist is essentially list of the nodes in the cluster and additional information about the node roles and groupings. It is typical that every node is listed in the Intel® Cluster Checker nodelist.

1. **Open a new file called "node.list" for editing.**

2. **Add nodes to the nodelist.**

   First, add the head node to the nodelist by adding the following line.

   ```
   frontend # role:head
   ```

   Next, add a line for each compute node in the cluster, using the following line as a template:

   ```
   nXX # subcluster:default role:compute
   ```

3. **Save and exit the file.**

   The nodelist should look similar to this when completed.

   ```
   frontend #                    role:head
   n01      # subcluster:default role:compute
   n02      # subcluster:default role:compute
   n03      # subcluster:default role:compute
   n04      # subcluster:default role:compute
   ```

   The file can be parsed directly from the Warewulf database. The head node will still need to be entered manually, but the remaining nodes can be added by a script. For example:

```
#!/usr/bin/python

ROLES=['boot','compute','enhanced','external','head','job_schedule','login','network_address','storag
e']

DATASET=Popen(['wwsh', 'object', 'print', '-t', 'node', '-p' 'NODENAME', '-p', 'ENABLED', '-p',
'CLUSTER', '-p', 'GROUPS'], stdout=PIPE)

LOOP=1
for BUFFER in DATASET.stdout:
   DATA=BUFFER.split()
   ROLE=""
   if LOOP == 1:
```

```
    if DATA[0] <> "NODENAME":
       print 'wwsh output error: column header missing'
       break
if LOOP > 2:
    if DATA[1] == "0":
        print '#',
    print '{:<12}'.format(DATA[0]),'#',
    if DATA[2] not in ['', 'UNDEF']:
        print "subcluster:",'{:<12}'.format(DATA[2]),
    else:
        print ' '*24,
    for GROUP in DATA[3].split(','):
        if GROUP in ROLES:
            ROLE += GROUP+','
    if ROLE <> "":
        print 'role:',ROLE.rstrip(',')
LOOP += 1
```

Run Intel® Cluster Checker

4. **Import the Intel® Cluster Checker environment.**

   ```
   source /opt/intel/clck/3.1.2.006/bin/clckvars.sh
   ```

5. **Run the collector.**

   Intel® Cluster Checker separates the tasks of system information collection and analysis into two separate tools. Using the nodelist generated from above, run this command to collect the data:

   ```
   clck-collect –a –f node.list
   ```

   This command instructs Intel® Cluster Checker to collect information for all checks for every node in the nodelist. To see a more comprehensive list of collector options, run the command:

   ```
   clck-collect –-help
   ```

6. **Run the analyzer.**

   Now that the data has been collect, we may use Intel® Cluster Checker to analyze the cluster by running the command:

   ```
   clck-analyze –f node.list
   ```

   The analyzer will return the list of checks performed, the list of nodes that were checked, and also the results of the analysis. Similar to the collector, to see a more comprehensive list of analyzer options, run the command:

   ```
   clck-analyze –-help
   ```

   The following are examples of the Intel® Cluster Checker analyzer output you may see.

   a. List of checks performed.

   ```
   Reading the database for the following checks:
      all_to_all...  done (0.0183 seconds)
   ```

```
cpu...  done (0.00552 seconds)
datconf...  done (0.00321 seconds)
dgemm...  done (0.00615 seconds)
environment...  done (0.073 seconds)
...
```

b. List of nodes.

```
Nodes being tested:
node[00-03], frontend.localdomain
```

c. Diagnoses.

```
1. Data for one or more checks are not available or could not be
   parsed correctly.
   [ Id: no-data ]
   [ Severity: 90%; Confidence: 90% ]
   [ 5 Nodes: node[00-03], frontend.localdomain ]
   [ Remedy: Disable any checks that are not relevant. Verify the
   correct database is being used and contains valid data. If necessary,
   collect the missing data. ]
```

d. Diagnosed signs.

```
1. Intel(R) MPI Benchmarks data is not available for analysis.
   [ Id: imb_pingpong-data-missing ]
   [ Severity: 90%; Confidence: 90% ]
   [ 4 Nodes: node[00-03] ]
   [ Remedy: Run the 'imb_pingpong' data provider. ]
```

e. Undiagnosed signs.

```
1. The High Performance Linpack benchmark run failed.
   [ Id: hpl-cluster-failed ]
   [ Severity: 90%; Confidence: 90% ]
   [ 4 Nodes: (node01, node02, node03, node00) ]
```

Procedure: Run a Test Job

Once the resource manager is enabled for production use, users should be able to run jobs. To test this, create and use a “test" user on the head node. Then compile and execute an application interactively through the resource manager.

Note the use of **prun** for parallel job execution, which summarizes the underlying native job launch mechanism being used.

1. **Add a test user.**

   ```
   useradd -m test
   ```

2. **Synchronize files.**

   ```
   wwsh file resync
   ```

3. **Switch to the "test" user account.**

   ```
   su - test
   ```

4. **Compile the MPI "hello world" example.**

   ```
   mpicc -O3 /opt/ohpc/pub/examples/mpi/hello.c
   ```

5. **Submit the job.**

   This is an interactive job request using **prun** to launch the executable

   f. Start a Slurm job in a pseudo-terminal.

   ```
   srun -n 6 -N 2 --pty /bin/bash
   ```

   g. Execute the binary using prun.

   ```
   prun ./a.out
   ```

   This should produce output similar to the following:

   ```
   [prun] Master compute host = n01
   [prun] Launch cmd = mpiexec.hydra -bootstrap slurm ./a.out
   Hello, world (6 procs total)
   --> Process # 0 of 8 is alive. -> n01
   --> Process # 4 of 8 is alive. -> n02
   --> Process # 1 of 8 is alive. -> n01
   --> Process # 5 of 8 is alive. -> n02
   --> Process # 2 of 8 is alive. -> n01
   --> Process # 3 of 8 is alive. -> n01
   ```