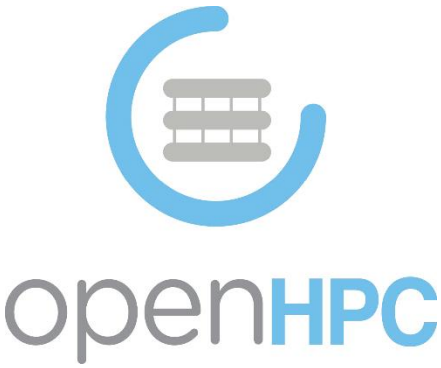
# Installation Guide

OpenHPC* @ Intel
High Performance Computing

# Build an Intel®-Based Cluster with OpenHPC* 2.0 on CentOS* 8

**Step-by-step instructions to install a high performance computing (HPC) cluster on Intel® Server Products using CentOS* Linux* 8.2 and OpenHPC* 2.0.**

February 9, 2021




## Introduction

Use this installation guide to install an HPC cluster using OpenHPC and Warewulf open source software. Based on the core installation recipes provided by OpenHPC, additional procedures have been added to integrate Intel HPC hardware and software products. This document is also restructured for direct migration into a configuration manager. For example, you can easily translate these installation steps directly into Ansible Playbook tasks.

A hardware and software Bill of Materials (BOM) and cluster configuration is included for reference only. The BOM describes the cluster used for development and verification of these instructions, but they should work on any similar hardware configuration. They can also be modified or enhanced as needed for other cluster designs.

Components Included in this Solution

The resulting Beowulf-type HPC cluster is conformant to the HPC Platform Specification, and it includes:

- Intel® Xeon® Scalable Processor-based servers
- Intel® Omni-Path Fabric Software (IFS)
- Intel® oneAPI HPC Toolkit
- Slurm workload manager

# Table of Contents



# Legal Notices



This document is adapted from Install Guide: CentOS 8.2 Base OS Warewulf/SLURM Edition for Linux* (x86 64), by OpenHPC, a Linux Foundation Collaborative Project, under CC-BY-4.0
https://github.com/openhpc/ohpc/releases/download/v2.0.GA/
Install_guide-CentOS8-Warewulf-SLURM-2.0-x86_64.pdf

Cover page: OpenHPC Logo, by OpenHPC under CC-BY-4.0:
https://github.com/openhpc/ohpc/tree/2.x/docs/recipes/install/common/figures

# Section 1: Reference Design

This reference design is validated internally on a specific hardware and software configuration but is presented here more generically. When instructions are followed exactly as written, will install and work correctly on the listed hardware configuration.

Even though other configurations are not validated, the use of similar hardware and software is expected to work correctly. Changes to the chassis, memory size, number of nodes, and processor or storage models should have no impact on cluster operation, but it may impact performance. This recipe may not work as written with changes made to the server board, fabric, or software stack.

## Hardware Bill of Materials

The validation cluster consists of 1 head node and 4 compute nodes, with a 1-Gbps Ethernet interface connected to an external network, and all nodes connected through a 10-Gbps Ethernet network. MPI messaging is performed over Intel® Omni-Path fabric. The Bill of Materials (BOM) specifies the minimum configuration only.

Ethernet and fabric interfaces may be on-board or add-on PCI Express interfaces.

| QTY | Item | Configuration |
|---|---|---|
| 1 | Head Node | 2 Intel® Xeon® Scalable Processors<br>Minimum 32 GB ECC memory<br>Intel® SSD Data Center Family, SATA, 800 GB<br>1 GbE Intel® Ethernet network adapter<br>10 GbE Intel® Ethernet network adapter<br>Intel® Omni-Path Host Fabric Interface (HFI) |
| 4 | Compute node | 2 Intel® Xeon® Scalable Processors<br>Minimum 96 GB ECC memory<br>Intel® 10GB Ethernet interface<br>Intel® Omni-Path Host Fabric Interface (HFI) |
| 1 | Ethernet Switch | Low Latency 10-Gbps Ethernet Switch |
| 1 | Fabric Switch | Intel® Omni-Path Edge Switch |

Table 0-1. Hardware Bill of Materials for the cluster

## Software Bill of Materials

The following software stack has been validated. Changes to the versions shown below may result in installation, operational, or performance issues.

Software required for optional components are covered separately in those sections.

| Software | Version |
|---|---|
| CentOS* Linux* OS installation DVD (minimal or full) | 8 build 2004 |
| Linux* kernel update from CentOS | 4.18.0-193.14.2.el8_2 |
| Intel® oneAPI HPC Toolkit | 2021.1.2 |
| Intel® Cluster Checker | 2019.9 |
| HPC Platform RPM packages for EL8 | 2018.0 |
| OpenHPC* distribution | 2.0 |
| Intel® Omni-Path Fabric Software (includes Intel® Omni-Path HFI Driver) | 10.10.3.1.1 |

## Obtaining Software

Intel® oneAPI Toolkits (including Intel® Cluster Checker tool), Intel® HPC Platform, and OpenHPC* packages are available through online repositories. Instructions are provided in this document to configure them. Intel® Omni-Path Fabric Software is downloaded from Intel; however, this product will move to Cornelis Networks. CentOS installation media is downloaded from CentOS.

# Document Conventions

Certain conventions used in this reference design are intended to make the document easier to use and provide flexibility to support different cluster designs.

## Configuration Manager Migration

This document is structured for direct migration into a configuration manager; for example, translating to an Ansible Playbook, with sections into Ansible plays and steps into Ansible tasks. To facilitate this, steps are written to accomplish a single task or command line operation. The intent is that each step will be easily understood for migration into configuration manager modules. At the same time, clear directions are provided for building the same cluster manually.

Much of the configuration normally performed during head node operating system installation has been moved to later command line steps. This is done to provide a bare-minimum system to the configuration manager

Ansible configuration will be covered in a separate, future document.

## Configurable Options

Values that are configurable by you are distinguished by yellow underline text throughout this document.

Many values used for software configuration depend on the hardware and software listed in the bill of materials, the document date of release, and developer preference. For example, IP addresses and hostnames are expected to be different in the final cluster installation. Depending on your cluster requirements, other suitable values may be used in place of the ones in this document.

Values that are configurable and their default values, include:

### Cluster/Subcluster

| | |
|---|---|
| Management Subnet (Ethernet): | 192.168.1.0 |
| Management Netmask: | 255.255.255.0 (/24) |
| Messaging Subnet (Intel® Omni-Path): | 192.168.5.0 |
| Messaging Netmask: | 255.255.255.0 (/24) |
| IPMI Subnet (Ethernet): | 192.168.2.0 |
| IPMI Netmask: | 255.255.255.0 (/24) |

### Head Node

| | |
|---|---|
| Hostname: | frontend |
| Domain Name: | cluster |
| Management IP Address: | 192.168.1.100 |
| Management Network Device (Ethernet): | eno2 |
| External Network Device (Ethernet): | eno1 |
| Messaging Hostname: | frontend-ib0 |
| Messaging IP Address: | 192.168.5.100 |
| Messaging Fabric Device (Intel® Omni-Path): | ib0 |

### Compute Node

| | |
|---|---|
| Hostnames: | cXX |
| Management IP Addresses: | 192.168.1.XX |
| Management Network Device (Ethernet): | eth0 |
| Messaging IP Addresses: | 192.168.5.XX |
| Messaging Network Device (Intel® Omni-Path): | ib0 |
| IPMI IP Addresses: | 192.168.2.XX |

IPMI Network Device (IPMI): bmc

### Network Device IP Address

Where devices and special nodes are required on the management or messaging fabric, IP addresses above 200 in the last IP address octet will be assigned to these devices.

## Hostname and IP Series

Hostname representation in this document varies to best fit the situation. When used in hostnames, *NNN* and *XXX* are meant represent a strictly 3-digit, zero-padded number (e.g. c001, c002...c200). When used in IP addresses, however, leading zeroes are not used (e.g. 192.168.1.1). As a means of disambiguation, XXX used to denote a single host, while NNN is used to specify the last host in the range.

Padding compute node hostnames is helpful to maintain sort order, but it is not required.

The class C network used in this document supports up to 254 hosts including the head node. These settings allow for a convenient pairing of node hostnames to IP addresses – matching cXXX to 192.168.1.XXX –at the cost of scalability past 253 compute nodes. To build a cluster larger than 253 nodes, different IP conventions must be used.

# 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.

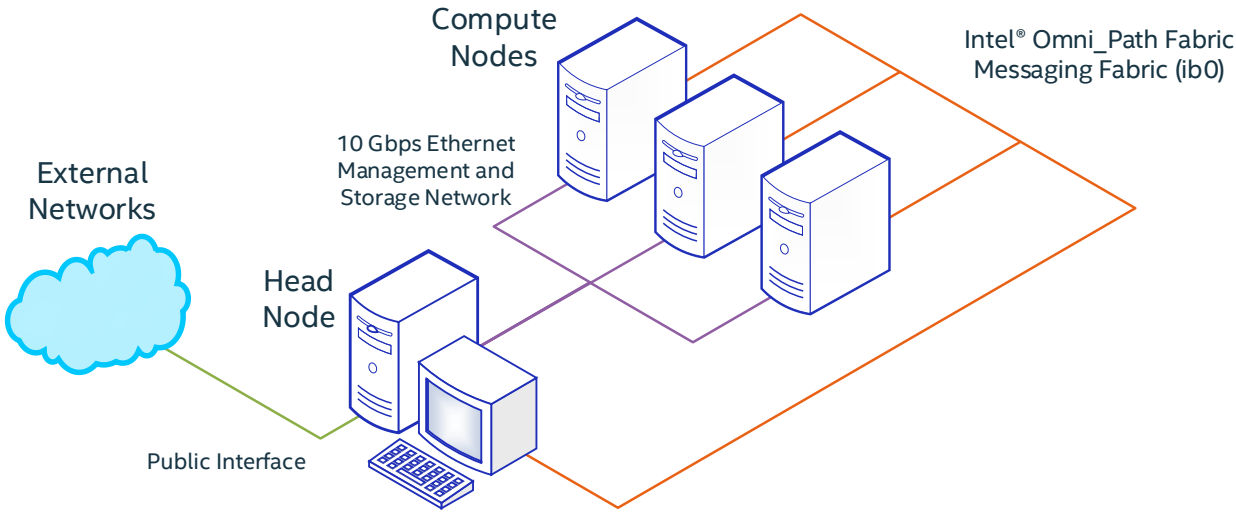



Figure 1-1: Cluster layout

## 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 to collect MAC addresses beforehand. In order to complete these instructions, the MAC addresses on the management fabric are required. For each compute node, identify the Ethernet port connected to the management network and record the interface's MAC address.

In order to facilitate automatic or remote restart using IPMI (Intelligent Platform Management Interface) or Redfish, the IP address of the Baseboard Management Controller (BMC) is required. For each compute node record the BMC IP address for the Ethernet port connected to the management network.

On most serverboards, MAC addresses for LAN-on-Motherboard Ethernet interfaces are consecutive. The MAC address of the first BMC is usually the MAC address after the MAC address of the last onboard NIC. For example:

| | |
|---|---|
| NIC1: A4:BF:01:DD:24:D8 | MAC address for provisioning. |
| NIC2: A4:BF:01:DD:24:D9 | |
| IPMI on NIC1: A4:BF:01:DD:24:DA | Used for Warewulf node reboot |
| IPMI on NIC2: A4:BF:01:DD:24:DB | |
| IPMI on dedicated NIC: A4:BF:01:DD:24:DC | Used for remote KVM |

You must identify the correct order of enumerated network devices on the head node.

## Setup Command Line Tools

The remaining steps in this document are all designed to be completed from a command line interface, unless otherwise stated. The command line interface is the default local login method since no GUI (X Window Server) is provided in the OS installation instructions, although there is no restriction against installing a GUI if needed. Remote SSH login is also available by default.

It is recommended to use the method that is easiest to copy and paste commands from this document.

# Installation

# Section 2: Install the Linux* Operating System

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

## Install the Base Operating System

1. Boot from the CentOS* install media (DVD).

   Intel servers will allow you to select the desired boot option by pressing **F6** after power-on.

2. At the CentOS 8 boot menu, Select “Install CentOS 8” and press **Enter**.

   By default, “Test this media” is selected.

   If prompted, press **Enter** again to continue. Installation will begin shortly if **Enter** is not pressed.

   It will take about one minute for the Linux* kernel and the Anaconda installer to load.

3. Select the language and dialect to use during installation.

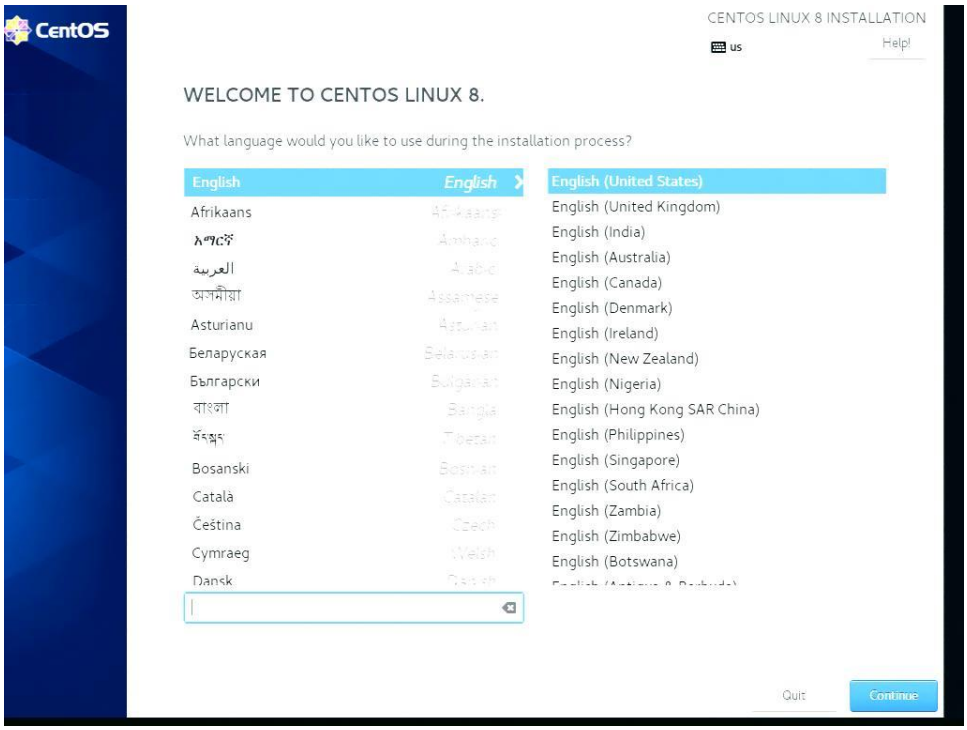



Figure 2-1: CentOS* 8 Welcome Screen

4. Select the correct language for your installation.

   This implementation has not been evaluated in any language other than "English (United States)". Problems with later steps in these instructions may occur if another language is used.

5. Click **Continue**.

   The **Installation Summary** menu will appear next.

   Note than an SMT warning may be present at the bottom of the screen. If you do not want SMT active on your head node, you can change BIOS settings after installation is complete.

## Configure the Network

If you are not installing the recipe on the exact hardware configuration in this recipe, your network interface names may be different. Different firmware versions may also result in name changes. You will need to substitute the interface names on your system with the ones used in this document.

The network interface(s) will be configured after installation.

## Configure Partitioning

For this implementation, “Automatically configure partitioning” will be used on a single system SSD.

Partitioning can be configured to meet your design requirements. For example, if you use a second SSD or HDD for /home directories, it should be configured here.

6. Click the **Installation Destination** icon.

   The **INSTALLATION DESTINATION** screen must be opened and confirmed to remove the warning symbol on the **INSTALLATION SUMMARY** screen.

   If there is more than 1 storage device, you must select a storage device for the OS. When selected for installation, a checkmark will appear over the drive icon.

7. Click **Done**.

   If the drive isn’t empty, the **INSTALLATION OPTIONS** dialog will appear.

8. If the **INSTALLATION OPTIONS** dialog appears, reclaim drive space.

   Otherwise, ignore this step.

   a. Click **Reclaim Space**.

      The **RECLAIM DISK SPACE** dialog will appear.

   b. Click **Delete all**

   c. Click **Reclaim Space**

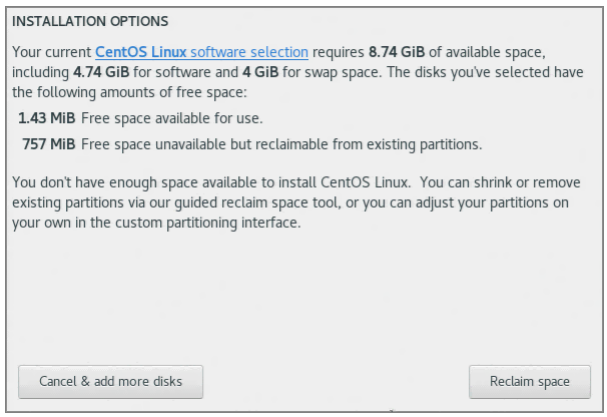



Figure 2-2 **INSTALLATION OPTIONS** dialog

## Configure Software

9. Confirm that the text under **Software Selection** reads “Minimal Install”.

10. If **Software Selection** is not "Minimal Install", update it.

a. Click the **Software Selection** icon.

b. In the **Base Environment** list box on the left side, select “Minimal Install”.

c. Click **Done**.

## Check the configuration.

After completing configuration options, the installation should appear with no red text or warning indicators.

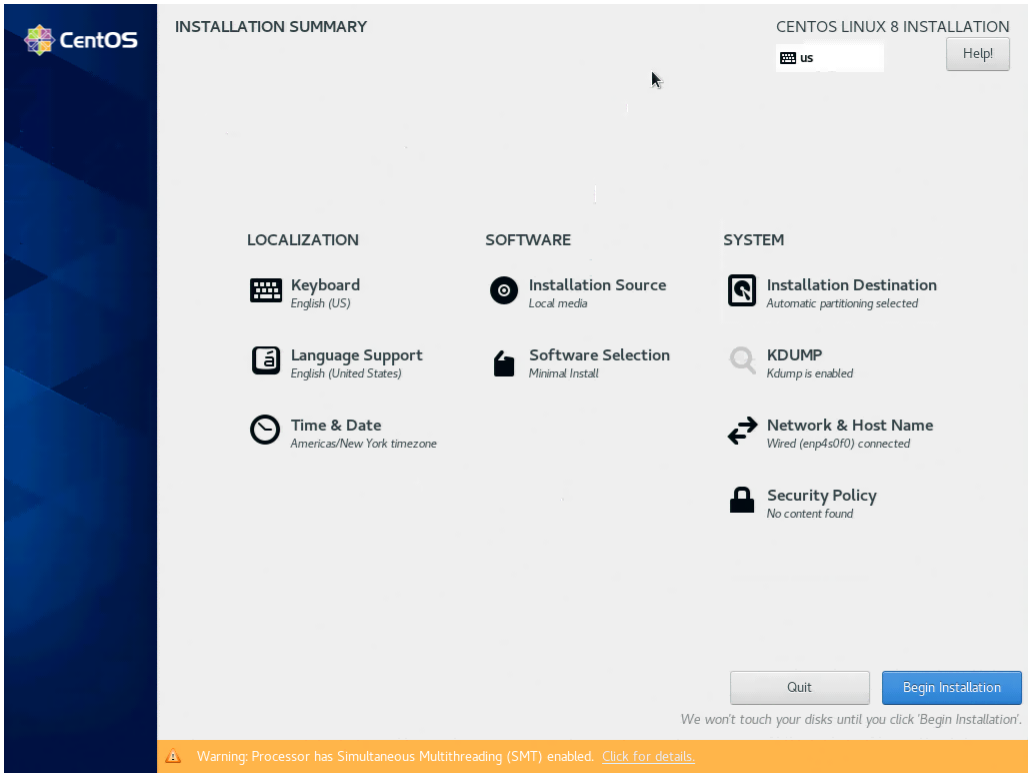



Figure 2-3 Installation Summary

11. Confirm that all options have been correctly configured

    The remaining settings will be configured after the OS is installed.

12. Click **Begin Installation**.

    If you receive a warning about missing "dmraid", you can safely ignore it.

## Set the root password.

While waiting for the installation to complete, set the root password.

User creation is optional.

13. Click **Root Password**.

    The **ROOT PASSWORD** screen will appear.

14. Enter the same password in **Root Password** and **Confirm** text boxes.

15. Click **Done**.

If the password is considered weak, you will need to click **Done** again to confirm it.

## Complete installation

Wait for the installation to complete. A progress bar estimates the remaining time.

16. If the **Finish Configuration** button appears, click that button.

17. Click **Reboot** when the installation is complete. Remove the CentOS* install media.

    The system should already be configured to boot from the primary drive. Wait until reboot is complete and the new OS is loaded.

18. Login as root.

# Post-Install Configuration

Basic configuration and updates to the operating system is performed before installing any management or application software.

**Tip: Disable confirmation prompts for the** dnf **command line**

The **dnf** command line tool is a primary means of installing software in this reference design. It replaces **yum** used in previous CentOS* releases. By default, **dnf** prompts the user before making software changes. You can also confirm actions by adding either the *--assumeyes* or the *-y* option. To permanently disable the prompt, add the following option to the [main] section of the /etc/dnf.conf file.

```
assumeyes=1
```

## Add network interfaces

The network interfaces were not configured during installation. This cluster network will also be the messaging network on Ethernet-only installations.

**19.** Clear any existing network configuration.

```
rm /etc/sysconfig/network-scripts/ifcfg-*
```

Confirm 'y' to remove files.

**20.** Add the external or public network interface.

```
nmcli connection add type ethernet ifname enp4s0f0 con-name public autoconnect yes
```

**21.** Add the cluster messaging network interface.

```
nmcli connection add type ethernet ifname ens513f0 con-name cluster autoconnect yes
```

**22.** Update the public interface configuration.

The public configuration must be configured to match your network requirements. All of the possible options are not documented here, but they can be found on the **nmcli** manpage https://developer.gnome.org/NetworkManager/stable/nmcli.html.

Examples:

If the public interface uses DHPC:

```
nmcli connection modify public ipv4.method auto ipv6.method disabled
```

If the public interface uses a static IP version 4 address:

```
nmcli connection modify public ipv4.method manual ipv4.addr "10.10.224.56/16" \
ipv6.method disabled
```

**23.** Update the cluster interface configuration

```
nmcli connection modify cluster ipv4.method manual ipv4.addr "192.168.1.100/24" \
ipv6.method disabled
```

**24.** Confirm connection status

```
nmcli -p con show --active
```

You should see output similar to this:

```
================================
  NetworkManager active profiles
================================
```

```
NAME     UUID                                  TYPE      DEVICE
-------------------------------------------------------------------------------
public   a15f4572-9589-4c7a-a130-2bc87865f2e0  ethernet  enp4s0f0
cluster  db1b459d-bb47-4abd-afc8-f2f97344a73a  ethernet  ens513f0
```

## Update security configuration

If you need to disable SELinux, do that now. Instructions are provided in the Appendix. This document provides required steps to install the cluster with SELinux enabled and enforcing.

### Configure the Firewall

Provisioning services for the cluster 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 not secure. The following changes will allow all connections within the cluster while maintaining the default firewall configuration on the external interface.

**25.** Secure the external connection.

```
nmcli connection modify public connection.zone work
```

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

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

```
nmcli connection modify cluster connection.zone trusted
```

Once the network and firewall are configured, you can connect to the server using SSH.

## Set system time

For a cluster to function properly, the date and time must be synchronized on all nodes. It is strongly recommended the a central network time server be available that is synchronized to an authoritative time source.

**27.** Set the time zone.

```
timedatectl set-timezone <region/timezone>
```

You can list all available time zones by running **timedatectl list-timezones**.

**28.** Configure chrony

**a.** Open /etc/chrony.conf for editing.

**b.** For each time source in your network, add the following line to the file

```
server <timeserver_ip>
```

where <timeserver_ip> is the IP address of your timeserver. You can use the "pool" option to reference a pool of timeservers.

**c.** Remove the line " pool 2.centos.pool.ntp.org iburst" if you will not use it.

**d.** Save and exit.

**Advisory**

If the frontend does not have access to a time server, or if the time server is unreliable, it is necessary to allow Chrony to act as the time server for compute nodes This can be done by editing the /etc/chrony.conf file and adding the line:

```
local stratum 10
```

29. Enable Chrony

```
systemctl enable chronyd.service
```

30. Restart the time service

```
systemctl restart chronyd
```

## Update Existing and Install New Packages

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 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.

31. **Only if required**, configure an Internet proxy; otherwise, skip this step.

    If your network implements a proxy server for Internet access, YUM must be configured to use it.

    In some cases, you may need to include a username and password for your proxy server. Including an "`http://`" or "`https://`" prefix is not required for most applications.

    a. Open /etc/environment for editing.

    b. Add the following lines to the file.

```
http_proxy=http://[<username>:<password>@]<proxy server>:<port>
https_proxy=http://[<username>:<password>@]<proxy server>:<port>
HTTP_PROXY=http://[<username>:<password>@]<proxy server>:<port>
HTTPS_PROXY=http://[<username>:<password>@]<proxy server>:<port>
no_proxy=localhost,192.168.1.100
NO_PROXY=localhost,192.168.1.100
JAVA_OPTS="-Dhttps.proxyHost=<proxy server> -Dhttps.proxyPort=<port>
     [-Dhttps.proxyUser=<username> -Dhttps.proxyPassword=<password>]
     -Dhttp.proxyHost=<proxy server> -Dhttp.proxyPort=<port>
     [-Dhttp.proxyUser=<username> -Dhttp.proxyPassword=<password>]"
```

    Where *<proxy server>* is the address or FQHN of the proxy server, and *<port>* is the HTTP or HTTPS port, as appropriate. If necessary, you have to specify a *<username>* and *<password>*.

    Remove options in brackets if username and password are not required.

    The `JAVA_OPTS` variable must exist on a single line without carriage returns.

    If you use an internal YUM repository, make sure to add its hostname or IP address to the NO_PROXY lists.

    c. Save and exit the file.

    d. Logout and log in to update the local environment.

## Disable weak dependencies

By default, CentOS installs recommended dependencies in addition to required dependencies. Changing this option will reduce the number of packages installed on the server.

This step is optional.

32. Update DNF configuration to disable weak dependencies.

```
dnf config-manager --setopt="install_weak_deps=False" --save
```

## Create a local RPM repository

A local repository is used to store additional non-OS rpms for installation by YUM or DNF. It simplifies dependency resolution and allows simplified installation and removal of packages.

33. Install the YUM repository creation and other utilities

```
dnf -y install createrepo dnf-utils
```

34. Create a local repository directory

```
mkdir -p /usr/src/localrepo
```

35. Create the local repository database and configuration files

```
createrepo /usr/src/localrepo
```

36. Create the repository configuration file.

    a. Open a new file /etc/yum.repos.d/local.repo for editing.

    Note the period in the local.repo filename.

    b. Enter the following

```
[localrepo]
name=Local Repository
baseurl=file:///usr/src/localrepo
enabled=1
gpgcheck=0
metadata_expire=1
```

    c. Save and exit the file.

    Many of the RPMs added to the local repo will not have signatures. Signature checking should not be disabled on public repos.

## Add additional YUM repositories

The CentOS PowerTools repo will be needed for development packages.

The EPEL repository is required for additional packages needed for HPC Platform compliance, as well as other components used in the reference design. It can be added directly using a package in the EPEL repository or added through a package in the OpenHPC repository. For this solution, the EPEL repository will be set up using the OpenHPC* package.

37. Enable the PowerTools repo

```
dnf config-manager --set-enabled PowerTools
```

38. Install the OpenHPC* release package.

This will also enable the EPEL repository.

```
dnf -y install \
http://repos.openhpc.community/OpenHPC/2/CentOS_8/x86_64/ohpc-release-2-1.el8.x86_64.rpm
```

39. Confirm repository configuration

```
dnf repolist
```

If correctly configured, the following repositories will be enabled.

```
repo id               repo name
AppStream             CentOS-8 - AppStream
BaseOS                CentOS-8 - Base
OpenHPC               OpenHPC-2.0 - Base
OpenHPC-updates       OpenHPC-2 - Updates
PowerTools            CentOS-8 - PowerTools
epel                  Extra Packages for Enterprise Linux 8 - x86_64
epel-modular          Extra Packages for Enterprise Linux Modular 8 - x86_64
extras                CentOS-8 - Extras
localrepo             Local Repository
```

Packages in the initial installation are updated from the online repositories to ensure they are the latest version.

### Update packages.

40. Update all packages.

Do not use the “upgrade” option.

```
dnf -y update
```

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

### Install HPC Platform packages

It is recommended to make all HPC node installations compliant to the HPC Platform specification written by Intel. This step requires two additional software components not provided by YUM repositories.

The HPC Platform RPMs are installed on the head node to meet both the requirements and advisories.

41. Add the HPC Platform repository.

```
dnf config-manager --add-repo \
http://yum.repos.intel.com/hpc-platform/el8/setup/intel-hpc-platform.repo
```

42. Import the GPG public key for the repository.

```
rpm --import http://yum.repos.intel.com/hpc-platform/el8/setup/PUBLIC_KEY.PUB
```

43. Install the HPC Platform compliance packages.

```
dnf -y install "intel-hpc-platform-*"
```

## Update the frontend hostnames.

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

If there are additional devices on the network, other than nodes to be provisioned by the cluster manager, add them here using the same steps. For example, the Intel® Omni-Path fabric switch or a managed Ethernet switch for the management network may be added here.

44. Set the host and domain names

a. Set the domain

```
domainname cluster
```

b. Set the hostname

```
hostname frontend
```

c. Set the hostname to the short hostname

```
hostnamectl set-hostname frontend
```

45. Update /etc/hosts.

a. Open the /etc/hosts file for editing.

b. Add the following line to the end of the file.

```
192.168.1.100      frontend.cluster frontend
```

c. Save and exit the file.

## Reboot the head node

While not always required, the head node is rebooted so that all updates and changes are activated. It is also necessary if the kernel is updated. This is also done to confirm there are no boot errors resulting from package updates.

46. Reboot the head node

```
init 6
```

Wait for the reboot to complete and the command prompt to be displayed.

47. Log in as root.

# Section 3: Install the Cluster

Warewulf is a software package used to create and control nodes in a cluster. Most packages are installed on a provisioning node and other node images are created offline. Warewulf installs all other nodes in the cluster from the provisioning node. In this guide, the cluster head node is configured as the provisioning node, and then set up to manage and install all other cluster nodes.

The cluster installation is performed through a series of steps:

1. The head node is configured.
2. A base compute node image is created and configured.
3. Special-purpose node images, if needed, are created and configured.
4. Physical node information is added to the cluster manager.
5. The cluster is provisioned.
6. The base cluster is evaluated and validated.
7. Optional components are added to the cluster.
8. The complete cluster is evaluated and validated.

## Install and Configure Warewulf

### Install Warewulf packages

Packages that were part of the minimum installation have been updated 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.

**1.** Install OpenHPC base

```
dnf -y install ohpc-base
```

**2.** Install Warewulf

```
dnf -y install ohpc-warewulf
```

**Advisory**

If you encounter Buzilla bug 1762314, where dnf reports a missing Perl package, you can correct it by using the following command.

```
dnf -y module enable perl-DBI
```

### Configure provisioning options

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 excluded from memory.

**3.** 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 = 'ens513f0'
```

c. Save and close the file.

4. Set the default subnet mask for all nodes.

a. Open the /etc/warewulf/defaults/node.conf file.

b. Comment out the line starting with ‘netdev’ in the file:

```
# netdev = eth0
```

c. Add the following lines at the end of the file:

```
groups = 'compute'
netmask = 255.255.255.0
network = 192.168.1.0
netdev = 'eth0'
```

d. Save and close the file.

5. Update node provisioning settings.

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.

a. Open /etc/warewulf/defaults/provision.conf for editing.

b. Edit the following lines to read:

```
bootstrap = centos8-default
vnfs = centos8
files = dynamic_hosts, passwd, group, shadow, gshadow
```

c. Save and close the file.

## Update VNFS defaults

6. Edit the VNFS global configuration file

a. Open /etc/warewulf/vnfs.conf for editing

b. Update exclusions.

In the EXCLUDE section, modify the file so that the following lines, and only the following lines, are included and uncommented. These directories and files will not be copied to the VNFS image.

```
exclude += /tmp/*
exclude += /var/log/[!m]*
exclude += /var/tmp/*
exclude += /var/cache
exclude += /opt/*
exclude += /home/*
```

c. Update the hybridization path.

In the HYBRIDPATH section, modify the file so that the following line, and only the following line, is included and uncommented.

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

**d.** Set directories to hybridize.

In the HYBRIDIZE section, modify the file so that the following lines, and only the following lines, are included and uncommented.

Inside of the VNFS image, these directories will be created a links to the remotely-shared image, which reduces node memory usage.

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

**e.** Save and close the file

## Update bootstrap configuration

**7.** Edit the bootstrap configuration file.

**a.** Open /etc/warewulf/bootstrap.conf for editing.

**b.** Add the following line at the beginning of the file, after the comment lines:

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

**c.** Save and close the file.

## Configure Data Services

**8.** Enable the database services to start automatically.

```
systemctl enable mariadb.service
```

**9.** Enable web services to start automatically.

```
systemctl enable httpd.service
```

**10.** Restart the database service.

```
systemctl restart mariadb
```

**11.** Restart web services.

```
systemctl restart httpd
```

**12.** Update the Warewulf database password

**a.** Open /etc/warewulf/database-root.conf for editing

**b.** Replace the "changeme" password with a password of your choice.

```
database password    = <new_password>
```

This password should be different than the root superuser password.

**c.** Save and exit the file.

13. Set the database root password

    This password should be different than the root superuser password. Include quotes around the new password.

    ```
    mysqladmin --user=root password "<new_password>"
    ```

14. Initialize the Warewulf database

    ```
    wwinit DATASTORE
    ```

    Enter the database root password when prompted.

## Update File Replication

15. Import Files

    The Warewulf system can import files from the provisioning server and synchronized them with nodes. This is a simple way to distribute user credentials to compute nodes.

    a. Add the dynamically-generated hosts file to the database.

    ```
    wwsh file import /etc/hosts --name=dynamic_hosts
    ```

    b. Add local authentication files to the database.

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

    Confirm to overwrite any existing file objects.

## Complete Warewulf initialization

16. Update DHCP

    ```
    wwsh dhcp update
    ```

17. Complete Warewulf headnode configuration

    ```
    wwinit PROVISION
    ```

# Configure the Head Node

Additional configuration on the head node is needed. The head node will act as an NTP time source, NFS server, and logfile collector for other nodes in the cluster.

## Install additional packages

18. Install the GNU C compiler.

```
dnf -y install gcc
```

19. Install OpenHPC compiler support.

```
dnf -y install gnu9-compilers-ohpc
```

20. Install additional tools used by several of the standard and optional components.

```
dnf -y install dmidecode numactl-libs numactl-devel mlocate nfs-utils rpm-build wget w3m
```

## Configure NFS

The /home directory is shared as read/write, and the /opt directory is shared read-only across the cluster.

21. Update NFS exports to exclude unneeded shares.

    a. Open the /etc/exports file for editing.

    b. Comment the /var/chroots and /usr/local shares.

    The final file should look similar to this. The IP addresses and subnet masks should already match the correct values for the network.

```
#WWEXPORT:/home:192.168.1.0/255.255.255.0
/home 192.168.1.0/255.255.255.0(rw,no_root_squash)

#WWEXPORT:/var/chroots:192.168.1.0/255.255.255.0
#/var/chroots 192.168.1.0/255.255.255.0(ro,no_root_squash)

#WWEXPORT:/usr/local:192.168.1.0/255.255.255.0
#/usr/local 192.168.1.0/255.255.255.0(ro,no_root_squash)

#WWEXPORT:/opt:192.168.1.0/255.255.255.0
/opt 192.168.1.0/255.255.255.0(ro,no_root_squash)
```

    c. Save the file and exit.

22. Enable and restart the NFS server.

```
systemctl restart nfs-server
```

## Configure System Log Forwarding

System logging for the cluster can be consolidated to the head node to provide easy access and reduce the memory requirements on the diskless compute node.

23. Configure the head node to receive messages.

    a. Open the /etc/rsyslog.conf file for editing.

**b.** Uncomment these lines by removing the comment symbol #:

```
module(load="imtcp") # needs to be done just once
input(type="imtcp" port="514")
```

**c.** Save the file and exit.

**24.** Restart the rsyslog service.

```
systemctl restart rsyslog
```

# Build the base compute node image

The OpenHPC* build of Warewulf includes enhancements and enabling for CentOS 8. The wwmkchroot command creates a minimal chroot image in the /opt/ohpc/admin/images/ subdirectory.

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.

## Create a new VNFS image

By default, the CentOS* Base repos point to the latest version of CentOS 8. Since this reference design is not always tested against the latest version, the repos for a specific version of CentOS* and its updates are used instead.

In addition, defining a permanent chroot location will simplify modification of the node image.

You will add new variables to the root .bash_profile file.

**25.** Update environment variables

**a.** Open /root/.bash_profile for editing.

**b.** Set the YUM_MIRROR environment variable.

Add the following line to the end of the file. This must be a single line with no carriage return.

```
export YUM_MIRROR=http://mirror.centos.org/centos/8/BaseOS/x86_64/os,
http://mirror.centos.org/centos/8/AppStream/x86_64/os
```

**c.** Define the chroot location.

Add the following lines to the end of the file.

```
export CHROOT=/opt/ohpc/admin/images/centos8
export CHROOTDIR=$CHROOT
```

**d.** Save and exit the file.

**26.** Reload the environment.

```
. /root/.bash_profile
```

**27.** Verify the environment

```
echo $CHROOT
```

This must produce the line:

```
/opt/ohpc/admin/images/centos8
```

**28.** Build the initial chroot image

```
wwmkchroot centos-8 $CHROOT
```

## Configure package repositories

Since the compute nodes and head node in this recipe use the same OS, DNF configuration is replicated from the head node. If you are creating a node image with a different OS release, then the node image will require its own DNF configuration and repository information.

Best practice for clusters dictates that packages should never be installed or removed directly on running production nodes. Instead, update the node image and synchronize this to running systems.

The force option is not included in commands here, in the event of an error entering the command. Each file deletion will need confirmed. Confirm files are only removed from the node image. You may include the force (-f) option at your own risk.

## Configure DNF

29. Update DNF configuration to disable weak dependencies.

```
dnf --installroot=$CHROOT config-manager --setopt="install_weak_deps=False" --save
```

30. Enable the PowerTools repo

```
dnf --installroot=$CHROOT config-manager --set-enabled PowerTools
```

## Configure a local repository

Cross-mount the local repository from the head node into the chroot directory.

31. Create a localrepo mount point.

```
mkdir /usr/src/localrepo
```

32. Bind mount the local repo.

```
mount --bind /usr/src/localrepo $CHROOT/usr/src/localrepo
```

33. Create the local repository configuration file.

    a. Open a new file /etc/yum.repos.d/local.repo for editing.

    b. Enter the following

```
[localrepo]
name=Local Repository
baseurl=file:///usr/src/localrepo
enabled=1
gpgcheck=0
metadata_expire=1
```

    c. Save and exit the file.

## Add additional repos

34. Install the OpenHPC* release package.

```
dnf -y --installroot=$CHROOT install \
http://repos.openhpc.community/OpenHPC/2/CentOS_8/x86_64/ohpc-release-2-1.el8.x86_64.rpm
```

35. Add the HPC Platform repository.

```
dnf --installroot=$CHROOT config-manager --add-repo \
http://yum.repos.intel.com/hpc-platform/el8/setup/intel-hpc-platform.repo
```

36. Import the GPG public key for the repository.

```
rpm --root=$CHROOT --import http://yum.repos.intel.com/hpc-platform/el8/setup/PUBLIC_KEY.PUB
```

# Update the Compute Node Image

## Update packages.

Update the intial packages from the online repositories to ensure they are at the latest version. Note the use of the *installroot* flag in this command. This instructs DNF to install to the specified chroot as opposed to the functional filesystem on the head node.

**37.** Update all packages.

It is unlikely that updated packages exist if the chroot was just created.

```
dnf -y --installroot=$CHROOT update
```

If you receive a message asking if it is OK to import a key, review the message. Enter **Y** and press **Enter** to continue.

## Install Additional OS and Software Packages

**38.** Install a kernel in the chroot image.

The kernel package is not installed to the chroot image. Since the boostrap image will be generated from the compute node image, the kernel is required. The kernel is matched to the head node.

If you want the compute node to use the latest kernel, remove the "uname -r" suffix.

```
dnf -y --installroot=$CHROOT install kernel
```

**39.** Install the OpenHPC* base compute node package.

```
dnf -y --installroot=$CHROOT install ohpc-base-compute
```

**40.** Install HPC Platform compliance packages.

The same HPC Platform packages that were added to the head node are installed on the node image.

```
dnf -y --installroot=$CHROOT install "intel-hpc-platform-*"
```

**41.** Install additional tools.

```
dnf -y --installroot=$CHROOT install dmidecode parted grub2 numactl chrony
```

# Configure the Compute Node Image

Basic client settings on the image are configured to match settings made earlier on the head node, including authentication, log file consolidation, shared storage, and time synchronization. Resource management and fabric support are added in their own sections.

## Confirm security configuration

**42.** Disable the firewall on the compute nodes, if it exists. If the firewall is not installed, this command will report “unit firewalld.service does not exist”.

```
systemctl --root=$CHROOT disable firewalld.service
```

**43.** Ensure that the authorized_keys file has the correct permissions

```
chmod 600 $CHROOT/root/.ssh/authorized_keys
```

## Configure Chrony (NTP)

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

**44.** Enable the chrony service.

```
systemctl --root=$CHROOT enable chronyd.service
```

**45.** Configure the chrony client.

**a.** Open the $CHROOT/etc/chrony.conf file for editing.

**b.** Delete or comment out all existing lines that begin with "pool" or “server”.

**c.** Add the following line to the file.

```
server 192.168.1.100
```

**d.** Save the file and exit.

## 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.

**46.** Add NFS client mounts.

**a.** Open the $CHROOT/etc/fstab file for editing.

**b.** Comment out or remove the mounts for /var/chroots and usr/local. When complete the NFS mounts will be:

```
192.168.1.100:/home  /home  nfs  defaults  0 0
192.168.1.100:/opt   /opt   nfs  defaults  0 0
```

**c.** Save the file and exit.

## Configure Lmod support

**47.** Copy Lmod scripts to the compute node.

```
cp /etc/profile.d/lmod.* $CHROOT/etc/profile.d/
```

## Configure System Log Forwarding

Disable logging on compute nodes except for emergency and boot logs.

**48.** Update the system log forwarding configuration.

**a.** Open the $CHROOT/etc/rsyslog.conf file for editing.

**b.** Update the forwarding configuration.

At the end of the file, uncomment following lines to read:

```
action(type="omfwd"
queue.filename="fwdRule1"       # unique name prefix for spool files
queue.maxdiskspace="1g"         # 1gb space limit (use as much as possible)
queue.saveonshutdown="on"       # save messages to disk on shutdown
queue.type="LinkedList"         # run asynchronously
```

```
action.resumeRetryCount="-1"    # infinite retries if host is down
```

**c.** Update the remote server information.

Update the last line of the file to read:

```
Target="192.168.1.100" Port="514" Protocol="tcp")
```

**d.** Edit the rules.

Locate "#### RULES ####" and comment the following lines by adding a # at the beginning of each line:

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

The following lines remain uncommented:

```
kern.*
*.emerg
local7.*
```

**e.** Save the file and exit.

# Install oneAPI HPC Toolkit

The full toolkit is installed, which includes compilers, libraries, and performance tools. It is also possible to install only the runtime components--instructions for custom installations is found at:

https://software.intel.com/content/www/us/en/develop/documentation/get-started-with-intel-oneapi-hpc-linux/top.html

## Configure the Repository

RPM packages for the oneAPI toolkit are available on a publicly accessible YUM repository. The configured repository will permit simple upgrades to the latest version of the toolkit.

**49.** Install the RPM signature key from the Internet repository.

```
rpm --import https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS-2023.PUB
```

**50.** Add the YUM repository.

**a.** Open /etc/yum.repos.d/OneAPI.repo for editing.

**b.** Enter the following text.

```
[oneAPI]
name=Intel(R) oneAPI repository
baseurl=https://yum.repos.intel.com/oneapi
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://yum.repos.intel.com/intel-gpg-keys/GPG-PUB-KEY-INTEL-SW-PRODUCTS-2023.PUB
```

**c.** Save and close the file.

**51.** Confirm the oneAPI Toolkit packages are available.

This will confirm if the new repo is available and accessible.

```
dnf list "intel-hpckit-*"
```

You should see output similar to this.

```
Available Packages
intel-hpckit-2021.1.0.x86_64                2021.1.0-2684                        oneAPI
```

## Install the Toolkit

The full toolkit is installed using a single metadata package.

**52.** Install Toolkit prerequisites.

```
dnf -y install gcc libstdc++-devel cmake
```

**53.** Install Toolkit prerequisites on the compute node.

```
dnf -y --installroot=$CHROOT install gcc libstdc++-devel cmake
```

**54.** Install the oneAPI Toolkit for HPC

```
dnf install intel-hpckit
```

## Configure environment modules

The oneAPI toolkit includes Tcl-based environment modules that allow users to quickly change their development environment. OpenHPC uses the Lmod environment module system, which is backward compatible with most Tcl environment modules.

To set up oneAPI environment modules, a shell script is provided in the main oneAPI directory.

Lmod does not support absolute environment module paths. Some modules must be updated with a workaround for incompatible features.

**55.** Setup oneAPI environment modules

This will place links to modules under a single directory, /opt/intel/oneapi/modulefiles.

```
/opt/intel/oneapi/modulefiles-setup.sh
```

**56.** Create an Lmod subdirectory for Intel® developer tool modulefiles.

```
mkdir -p /opt/ohpc/pub/modulefiles/intel/
```

**57.** Create an Lmod subdirectory for Intel® MPI Library modulefiles.

```
mkdir -p /opt/ohpc/pub/moduledeps/intel/impi
```

### Compiler toolchain modulefile

**58.** Confirm the compiler version.

At the time of this document, the current compiler version is 2021.1.2. The "latest" link should point to the current version in the directory path.

```
realpath /opt/intel/oneapi/modulefiles/compiler/latest
```

Output will be this or similar to this.

```
/opt/intel/oneapi/compiler/2021.1.2/modulefiles/compiler
```

Note the version number in the path.

**59.** Create an environment module to load the oneAPI toolchain.

**a.** Open /opt/ohpc/pub/modulefiles/intel/2021.1.2 for editing.

**b.** Add the following content to the new file.

Update the version on line 3, if necessary.

```
#%Module1.0#####################################################################

set version "2021.1.2"

proc ModulesHelp { } {
global version
puts stderr "\nThis module loads the Intel compiler environment.\n"
puts stderr "\nSee the man pages for icc, icpc, and ifort for detailed information"
puts stderr "on available compiler options and command-line syntax."
puts stderr "\nVersion $version\n"
}

module-whatis "Name: Intel Compiler"
module-whatis "Version: $version"
module-whatis "Category: compiler, runtime support"
module-whatis "Description: Intel Compiler Family (C/C++/Fortran for x86_64)"
module-whatis "URL: http://software.intel.com/en-us/articles/intel-compilers/"
```

```
# update module path hierarchy
prepend-path    MODULEPATH          /opt/ohpc/pub/moduledeps/intel
prepend-path    MODULEPATH          /opt/intel/oneapi/modulefiles

module load "compiler/$version"

family "compiler"
```

c. Save and exit the file.

60. Create a version file.

a. Open /opt/ohpc/pub/modulefiles/intel/.version.2021.1.2 for editing

b. Add the following content to the new file.

```
#%Module1.0#####################################################################
set     ModulesVersion      "2021.1.2"
```

c. Save and exit the file.

## MPI toolchain modulefile

61. Confirm the MPI Library version.

At the time of this document, the current compiler version is 2021.1.1. The "latest" link should point to the current version in the directory path.

```
realpath /opt/intel/oneapi/modulefiles/mpi/latest
```

Output will be this or similar to this.

```
/opt/intel/oneapi/mpi/2021.1.1/modulefiles/mpi
```

Note the version number in the path.

62. Create an environment module to load the oneAPI MPI Library toolchain.

a. Open /opt/ohpc/pub/moduledeps/intel/impi/2021.1.1 for editing.

b. Add the following content to the new file.

```
#%Module1.0#####################################################################

set version "2021.1.1"

proc ModulesHelp { } {
global version
puts stderr "\nThis module loads the Intel MPI environment.\n"
puts stderr "   mpiifort  (Fortran source)"
puts stderr "   mpiicc    (C   source)"
puts stderr "   mpiicpc   (C++ source)\n"
puts stderr "Version $version\n"
}

module-whatis "Name: Intel MPI"
module-whatis "Version: $version"
module-whatis "Category: library, runtime support"
module-whatis "Description: Intel MPI Library (C/C++/Fortran for x86_64)"
module-whatis "URL: http://software.intel.com/en-us/articles/intel-mpi-library"

module load "mpi/$version"
```

```
family "MPI"
```

c. Save and exit the file.

63. Create a version file.

a. Open /opt/ohpc/pub/moduledeps/intel/impi/.version.2021.1.1 for editing

b. Add the following content to the new file.

```
#%Module1.0#####################################################################
set     ModulesVersion      "2021.1.1"
```

c. Save and exit the file.

### Patch the original module files.

Some of the module files include Tcl functions that are incompatible with Lmod compatibility. These need to be modified to provide the same functionality. The link will be replaced by a copy of the original file so that it can be modified without changing the original file.

64. Replace the link with copy of the target.

Enter "y" to confirm to overwrite the link.

```
cp --remove-destination /opt/intel/oneapi/compiler/2021.1.2/modulefiles/compiler \
/opt/intel/oneapi/modulefiles/compiler/2021.1.2
```

65. Update the module file.

a. Open /opt/intel/oneapi/modulefiles/compiler/2021.1.2 for editing.

b. Delete the conditional oclfpga module load.

Locate and delete this entire section.

```
if { [ is-avail $cmplr_root/linux/lib/oclfpga/modulefiles/oclfpga ] } {
    module load $cmplr_root/linux/lib/oclfpga/modulefiles/oclfpga
}
```

c. Add the following section in its place.

```
module try-add oclfpga
```

d. Save and close the file.

66. Re-link the "latest" symlink.

It points to the original, unmodified modulefile. Update it to point to the updated modulefile.

```
ln -sf 2021.1.2 /opt/intel/oneapi/modulefiles/compiler/latest
```

### Other module files.

The same compatibility issue exists for the beta and 32-bit module files. If you need to support these in the Lmod environment, they must also be updated.

To update, repeat the previous steps for the other three modules and "latest" symlinks.

- /opt/intel/oneapi/modulefiles/compiler32/2021.1.2
- /opt/intel/oneapi/modulefiles/compiler32/2021.1.-beta10

- /opt/intel/oneapi/modulefiles/compiler/2021.1-beta10.

# Install and Configure Intel® Omni-Path Fabric Software

New distro releases include Intel Omni-Path drivers and basic tools. For many applications, this is sufficient. However, the Intel® Omni-Path Fabric Software Suite (IFS) provides the latest performance and feature enhancements for your cluster installation.

The RPM packages from the IFS version 10.10.3 tarball will be added to the local repository to simplify installation, dependency resolution, and future updates.

When complete, reboot the head node to confirm that Omni-Path software is active and the subnet manager is running.

## Add the software packages to the local repository.

Only the rpmfiles are required for installation. These are extracted into a single directory which is added as a YUM repository.

**67.** Download the installation tarball.

**a.** Review the license agreement.

```
w3m -dump https://cdrdv2.intel.com/v1/dl/getContent/631010
```

**b.** **If you agree** to the EULA, download the session cookies.

```
wget --save-cookies cookie.txt --keep-session-cookies \
--delete-after https://cdrdv2.intel.com/v1/dl/acceptEula/631010
```

**c.** Download the package.

```
wget --load-cookies=cookie.txt -O IntelOPA-IFS.RHEL82-x86_64.10.10.3.1.1.tgz \
https://cdrdv2.intel.com/v1/dl/getContent/631010
```

**68.** Extract the tarball into the current directory.

```
tar -zxvf IntelOPA-IFS.RHEL82-x86_64.10.10.3.1.1.tgz
```

**69.** Copy rpms to shared local repository, while dereferencing links.

```
cp -L IntelOPA-IFS.RHEL81-x86_64.10.10.2.0.44/repos/OPA_PKGS/RPMS/*.rpm /usr/src/localrepo
```

**70.** Update the repository

```
createrepo --update /usr/src/localrepo
```

## Perform a kernel check

The OPA fabric drivers are provided in IFS for specific kernels. If the driver package for the installed kernel version doesn't exist, it must be created.

To create a new kernel driver package, see Appendix A.

**71.** Verify the head node kernel.

**a.** Get all available package versions.

```
dnf repoquery --showduplicate --qf '%{VERSION}' kmod-ifs-kernel-updates
```

**b.** Get the head node kernel version.

```
uname -r
```

If one of the available package versions matches the head node kernel version, then the driver packages are ready for installation. Otherwise, a kernel driver package must be created.

72. Verify the compute node kernel.

    a. Get all available package versions.

    ```
    dnf --installroot=$CHROOT repoquery --showduplicate --qf '%{VERSION}' kmod-ifs-kernel-updates
    ```

    b. Get the head node kernel version.

    ```
    ls $CHROOT/lib/modules
    ```

    Like the head node, one of the available packages must match the installed kernel.

## Install the software on the head node

### Preparation

73. Remove existing drivers.

Remove any drivers that were installed from the distribution.

```
dnf -y remove opa-*
```

74. Link MPI directory from /usr to /opt.

    This will make the MPI libraries shared across the cluster, if this step is done before the libraries are installed. The purpose is to avoid creating a new NFS export. If the directory already exists, use the **mv** command to relocate it to /opt.

    a. Create directories

    ```
    mkdir /opt/mpi
    ```

    b. Create softlinks in /usr to the new directories

    ```
    ln -s /opt/mpi /usr/mpi
    ```

### Install OPA Software

These packages can be installed in a single **dnf** command. They are only separated here for clarity.

75. Install Intel Omni-Path basic drivers.

    ```
    dnf -y install opameta_intel_hfi
    ```

76. Install FastFabric and tools

    ```
    dnf -y install opameta_fastfabric
    ```

77. Install the MPI libraries

    ```
    dnf -y install opameta_openmpi_gcc_hfi opameta_mvapich2_gcc_hfi
    ```

    The optimized GCC MPI libraries for Intel® Omni-Path fabric are now available in /opt/mpi/gcc.

78. Install the fabric manager

    ```
    dnf -y install opameta_opafm
    ```

### Update services

79. Enable the rdma service.

```
systemctl enable rdma.service
```

80. Enable the fabric manager.

    This step is required if an unmanaged Omni-Path switch is used.

    This step is not required if a managed Omni-Path switch is used. It is recommended in order to enable the head node to be a backup fabric manager.

    It will load after the next reboot of the head node.

```
opaautostartconfig --enable OPAFM
```

## Install the software on the compute node image

81. Remove existing drivers

```
dnf -y --installroot=$CHROOT remove opa-*
```

82. Create softlinks from /usr to /opt.

```
ln -s /opt/mpi $CHROOT/usr/mpi
```

83. Install the Intel Omni-Path drivers.

```
dnf -y --installroot=$CHROOT install opameta_intel_hfi opa-basic-tools
```

84. Enable the rdma service.

```
systemctl --root=$CHROOT enable rdma.service
```

85. Update the boot configuration

    The bootstrap image should be updated to include the new drivers and firmware.

    a. Open the /etc/warewulf/bootstrap.conf file for editing.

    b. Add the following lines to the beginning of the file:

```
drivers += extra/ifs-kernel-updates/*
firmware += updates/hfi1*
```

    c. Save and exit the file.

# Configure IPoIB

OpenHPC* provides a template file for setting up the ib0 interfaces. For IPoIB interfaces on compute nodes, the template is imported and copied to each node during provisioning.

## Configure the head node

**86.** Update the ib0 configuration

**a.** Open a new file /etc/sysconfig/network-scripts/ifcfg-ib0 file for editing.

**b.** Enter the following into the file.

```
DEVICE=ib0
NAME=ib0
BOOTPROTO=static
IPADDR=192.168.5.100
NETMASK=255.255.255.0
ONBOOT=yes
STARTMODE='auto'
NM_CONTROLLED=yes
CONNECTED_MODE=yes
MTU=65520
TYPE=Infiniband
ZONE=trusted
IPV6INIT=no
```

**c.** Save the file and exit.

**87.** Update the hosts file.

**a.** Open /etc/hosts for editing

**b.** Add the following line to the file, immediately after the 192.168.100 entry.

```
192.168.5.100    frontend-ib0.cluster frontend-ib0
```

**c.** Save and close the file.

**88.** Update the IPoIB device settings

This provides optimized IP over Fabric performance on Omni-Path

**a.** Open a new file /etc/sysctl.d/80-opa_ipofabric.conf file for editing.

**b.** Enter the following lines

```
net.ipv4.tcp_rmem = 16384 349520 16777216
net.ipv4.tcp_wmem = 16384 349520 16777216
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.core.somaxconn = 2048
net.ipv4.tcp_mtu_probe = 1
net.core.netdev_max_backlog=250000
```

**c.** Save and close the file

## Update the Compute Nodes

You will import a template network script for Intel® Omni-Path interface that is specifically designed to be deployed by Warewulf* for compute nodes.

**89.** Install Network Manager.

This is needed to bring up the IPoIB interface.

```
dnf -y --installroot=$CHROOT install NetworkManager
```

**90.** Enable connected mode for the compute nodes and enable setting non-default MTUs

**a.** Open the file /opt/ohpc/pub/examples/network/centos/ifcfg-ib0.ww for editing

**b.** Locate the line "NM_CONTROLLED" and change it to:

```
NM_CONTROLLED=yes
```

**c.** Add the following lines

```
NAME=ib0
TYPE=Infiniband
MTU=%{NETDEVS::IB0::MTU}
CONNECTED_MODE=yes
```

**d.** Save and close the file.

**91.** Import the file template into the database.

```
wwsh -y file import /opt/ohpc/pub/examples/network/centos/ifcfg-ib0.ww
```

**92.** Change the distributed file deployment location.

```
wwsh -y file set ifcfg-ib0.ww --path=/etc/sysconfig/network-scripts/ifcfg-ib0
```

**93.** Update the system settings for optimized IP over Fabric performance on Omni-Path

**a.** Open a new file $CHROOT/etc/sysctl.d/80-opa_ipofabric.conf for editing.

**b.** Enter the following lines

```
net.ipv4.tcp_rmem = 16384 349520 16777216
net.ipv4.tcp_wmem = 16384 349520 16777216
net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
```

**c.** Save and close the file

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

**a.** Open /etc/warewulf/defaults/provision.conf for editing.

**b.** Locate the “files” list.

**c.** Add the following entry to the end of the line. The comma is added directly after the last entry; delete any training spaces first.

```
, ifcfg-ib0.ww
```

**d.** Save and close the file.

# Assemble the node image

Warewulf* employs a two-stage process for provisioning nodes. First, a bootstrap is used to initialize the kernel and install process, then an encapsulated image containing the full system is loaded on the node.

## Create a bootstrap image

The bootstrap image includes the runtime kernel and associated modules, as well as simple scripts for provisioning.

**95.** Get the kernel used in the compute nod image

It should be the same as the head node image.

```
ls $CHROOT/lib/modules
```

This should output a single entry, such as "4.18.0-147.8.1.el8_1.x86_64".

**96.** Build bootstrap image for Intel® Xeon processor nodes

```
wwbootstrap --name=centos8-default --chroot=$CHROOT <kernel_version>
```

## Create a 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.

**97.** Create the VNFS image.

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

The capsule is approximately 380 MB in size. You can see that the VNFS capsules exists by executing

```
wwsh vnfs list
```

## Register cluster nodes

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).

> **Note**
>
> In the step below, XX is used in both IP addresses and hostnames. Omit leading zeroes for IP addresses otherwise Warewulf* will interpret the first zero to mean the number is in octal base. It is safe (and recommended) to include leading zeroes in hostnames.

Associations for nodes, boot images, and vnfs images are configured in the file:

/etc/warewulf/defaults/provision.conf.

**98.** **Repeat** the following command for each compute node in the cluster.

This will add Intel® Xeon server nodes to the Warewulf datastore. Replace <*mac_address*> with the correct value for that node.

```
wwsh -y node new cXX --ipaddr=192.168.1.XX --hwaddr=<mac_address>
```

**99.** Review the added nodes.

```
wwsh node list
```

Output will be similar to this:

```
NAME                GROUPS              IPADDR              HWADDR
==========================================================================
c1.default.cluster  compute             192.168.1.1         a4:bf:01:21:71:03
c2.default.cluster  compute             192.168.1.2         a4:bf:01:a5:dd:0f
c3.default.cluster  compute             192.168.1.3         a4:bf:01:18:7e:4c
c4.default.cluster  compute             192.168.1.4         a4:bf:01:22:12:ab
```

You can use **wwsh node print** to view detailed settings for each node.

**100.** **Repeat** the following command once for each compute node in the cluster with Intel® Omni-Path Fabric.

Add the IPoIB interface to each node. Also increase the MTU of the ib0 interface to 65.52 KB, up from the default 2.044 KB.

```
wwsh -y node set cXXX -D ib0 --ipaddr=192.168.5.XXX --netmask=255.255.255.0 \
--network=192.168.5.0 --mtu 65520
```

**101.** Confirm all nodes have been added.

```
wwsh node print | grep ib0.IPADDR
```

The output should appear similar to this:

```
c1: ib0.IPADDR       = 192.168.5.1
c2: ib0.IPADDR       = 192.168.5.2
c3: ib0.IPADDR       = 192.168.5.3
c4: ib0.IPADDR       = 192.168.5.4
```

**102.** Update DHCP.

```
wwsh dhcp restart
```

**103.** update PXE

```
wwsh pxe update
```

# Provision the cluster

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

**104.** Boot compute nodes

Power cycle each compute node. Methods may include manually resetting the power button or issuing IPMI commands. Each node must boot over the network from the enp3s0f0 interface or the installation will fail.

To monitor the status of compute nodes, forward a command to all compute nodes to assess how many have booted.

```
pdsh -w c[01-XX] uptime
```

# Section 4: Verify Cluster Design with Intel Cluster Checker

Intel® Cluster Checker (CLCK) is one of many tools installed as part of Intel® Parallel Studio XE. It is a powerful tool designed to assess cluster health. It can also be installed independently if Intel Parallel Studio Runtimes are installed.

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.

Cluster Checker was installed as part of the oneAPI HPC Toolkit.

## Configure Intel® Cluster Checker

### Update the configuration file

Changes to the Intel® Cluster Checker configuration file are needed.

Change the `<network_interface>` option to the head node cluster interface.

1. Back up the configuration file.

   ```
   cp /opt/intel/clck/2019.9/etc/clck.xml{,.backup}
   ```

2. Update the main configuration file

   As you edit the file, you will also need to uncomment these elements.

   a. Open /opt/intel/clck/2019.9/etc/clck.xml for editing.

   b. Locate the <network_interface> element under <collector>.

      Uncomment the element by removing the "<!--" and "-->" lines around it.

   c. Change the network interface for the head node. It should be:

      ```
      <network_interface>ens513f0</network_interface>
      ```

   d. Save and exit the file.

### Create a new user

If this user does not already exist, the user is created. While Cluster Checker can be run as the root user, it is more appropriate to evaluate functionality as a normal system user.

3. Add a dedicated Intel Cluster Checker user.

   ```
   useradd -m clck
   ```

4. Set the clck user password.

   Execute the following command. You must enter the desired password once to set and then again to confirm.

   ```
   passwd clck
   ```

   If there are additional configuration options required for new user accounts, such as assignment to groups or changes to permissions, perform those changes now.

5. Synchronize files

```
wwsh file resync
```

6. Login as the new user

```
su - clck
```

## Load Intel oneAPI Developer Tools

Many of the Cluster Checker checks require oneAPI runtime libraries. These need to be added to the user environment to load on login to any node.

7. Set the oneAPI environment to load automatically.

   a. Open /etc/bash_profile.sh for editing.

   b. Add the following lines to the end of the file.

```
module load intel
module load impi
```

   c. Save and exit the file.

8. Logout

```
exit
```

9. Login as the clck user

```
su - clck
```

10. Confirm that the modules are loaded

```
mpiicc --version
```

    This should print the current version of the Intel® MPI Compiler for C.

## Create a nodefile.

The nodefile defines the cluster nodes to check. For each node in the cluster, there is a line in the nodefile that lists the node hostname and defines the role of that node in the cluster. In addition, it also defines the subcluster to which the node belongs.

Each line has the format:

```
hostname   # role: <role_name>
```

The next steps create this list in the clck user home directory. The default role is “compute” and it can be omitted for compute nodes.

11. Create an updated nodefile.

    a. Open a new file named “nodefile” for editing

    b. List every node short hostname in the cluster, one per line, including the headnode.

    c. After the headnode hostname, add on the same line.

```
# role: head
```

    Do not use tabs to space entries.

    d. For each compute node, enter the hostname and add on the same line:

```
# role: compute
```

The completed file should look similar to this:

```
frontend # role: head
c01      # role: compute
c02      # role: compute
c03      # role: compute
c04      # role: compute
```

**e.** Save and exit the file.

# Run Cluster Checker

Intel® Cluster Checker executes in two phases. In the data collection phase, Intel® Cluster Checker collects data from the cluster for use in analysis. In the analysis phase, Intel® Cluster Checker analyzes the data in the database and produces the results of analysis. It is possible to invoke these phases together or separately and to customize their scope.

The **clck** command executes data collection followed immediately by analysis and displays the results of analysis. It is also possible to run data collection and analysis separately, using the commands **clck-collect** and **clck-analyze**.

Ahe following commands are run as the clck user.

**12.** Load the Intel® Cluster Checker environment.

```
module load clck
```

To see a comprehensive list of options, run the command:

```
clck --help
```

**13.** Execute the data collection and analysis with the health framework definition.

This will gather all information required to evaluate the overall health and conformance of the cluster, including consistency, functionality, and performance.

```
clck -f nodefile -F health_extended_user
```

The analyzer returns the list of checks performed, the list of nodes checked, and the results of the analysis into the file clck_results.log. The results are a synopsis of the cluster health.

**14.** Open the file "clck_results.log" to look at the results.

Correct any identified errors and rerun the health check before performing validation in the next step.

**15.** Validate Intel HCP Platform compliance.

Run collection and analysis to validate against the HPC Platform Specification. Validation will be done sections 1.1 to 2.2, from Core to HPC Application Compatibility.

```
clck -f nodefile -F intel_hpc_platform_compat-hpc-2018.0
```

**16.** Open the file clck_results.log to look at the results.

In validation mode, the analyzer results highlight all instances of non-compliance against the HPC Platform Specification. Analysis of this reference design ending in an overall “PASS” result indicative of validation to that specification.

# Optional Components

# Section 5: Intel Parallel Studio XE Runtime

If applications are built with Intel® Compilers or are dynamically linked to Intel® Performance Libraries*, the Intel® Parallel Studio XE runtime should be made available on the target platform. Otherwise, the application must provide all required libraries in its installation. Intel® OneAPI runtimes should provide the required runtimes, but these instructions are provided if earlier versions are required.

The runtime packages are available at no cost and are available through online repositories.

Packages are installed on the head node only and shared through NFS.

## Set up the Internet Repository

Intel Parallel Studio Runtimes are available through a public repository.

1. Install the signature key from the Internet repository.

```
rpm --import \
https://yum.repos.intel.com/2020/setup/RPM-GPG-KEY-intel-psxe-runtime-2020
```

2. Install the repository configuration package.

```
dnf -y install \
https://yum.repos.intel.com/2020/setup/intel-psxe-runtime-2020-reposetup-1-0.noarch.rpm
```

## Install the Runtime packages

3. Install the main package. This will install all required packages and dependencies.

```
dnf -y install intel-psxe-runtime
```

4. Install dependencies on compute nodes.

   These are needed by PSXE runtimes, but not always installed when PSXE is shared from a central location.

```
dnf -y --installroot=$CHROOT install gcc libstdc++-devel
```

# Section 6: Singularity

Singularity is a Linux container solution, with support for both its own and Docker containers.

## Install the package

1. Install the package from the EPEL repo.

```
yum install singularity
```

## Update the compute node image

2. Install dependencies

```
yum --installroot=$CHROOT install singularity
```

3. Update the image

```
wwvnfs --chroot=$CHROOT --hybridize
```

4. Reboot the compute nodes

```
pdsh -w c[01-04] init 6
```

# Section 7: Slurm Workload Manger

This section is optional.

The Slurm workload manager is added to the cluster.

The Slurm Workload Manager provides a means of scheduling jobs and allocating cluster resources to those jobs. This is a client-server install with server components installed on the head node.

## Install Slurm Server

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.

Normal user SSH access to compute nodes is restricted by PAM (pluggable authentication module) via Slurm. Since this may not be ideal in certain instances, a user group is created for which these restrictions do not apply.

The global Slurm configuration file and the cryptographic key that is required by the munge authentication library have to be available on every host in the resource management pool.

1. Create a user group that is granted unrestricted SSH access.

   ```
   groupadd ssh-allowed
   ```

2. Add the dedicated Intel Cluster Checker user to the whitelist.

   This user account should be able to run Cluster Checker both inside and outside of a resource manager job.

   ```
   usermod -aG ssh-allowed clck
   ```

3. Create the Slurm user account

   ```
   useradd --system --shell=/sbin/nologin slurm
   ```

4. Install Slurm server packages.

   ```
   dnf -y install ohpc-slurm-server prun-ohpc munge
   ```

5. Update the Warewulf files

   ```
   wwsh file sync
   ```

## Update node resource information.

This procedure depends on the hardware configuration of the cluster. The Slurm configuration file needs to be updated with the node names of the compute nodes, the properties of their processors and the Slurm partitions, or queues, they belong to.

"NodeName" reflects the compute nodes names with their respective capabilities defined.

"Sockets" defines the number of sockets in a compute node.

"CoresPerSocket" defines the number of CPU cores per socket.

"ThreadsPerCore" defines the number of threads that can be executed in parallel on a core.

"PartitionName" defines the Slurm partitions, or queues, that compute names are assigned to and which users can use to access the resources; the name of the partition will be visible to the user.

"Nodes" defines the compute nodes that belong to a given partition.

"Default" defines which partition is the default partition, *i.e.* the partition that is used when a user doesn't explicitly specify a partition.

For a cluster based on the Bill of Materials of this Reference Design, the Slurm configuration file needs to be updated. This will require two changes.

9. Add the host name of the Slurm controller server to the Slurm configuration file.
10. Update the NodeName definition to reflect the hardware capabilities.

Other configuration changes will also be made..

## Update Slurm configuration.

**6.** Copy the example file.

```
cp /etc/slurm/slurm.conf.ohpc /etc/slurm/slurm.conf
```

**7.** Open /etc/slurm/slurm.conf for editing.

**8.** Update the SLURM server hostname

**a.** Locate the line

```
ControlMachine=
```

**b.** Replace it with

```
ControlMachine=frontend
```

**9.** Update return to service.

By default, SLURM will reinstate a node that was in DOWN state for being non-responsive. This change will make a compute node available again when it registers a valid configuration with the SLURM server.

**a.** Locate the line

```
ReturnToService=1
```

**b.** Update the value to read

```
ReturnToService=2
```

**10.** Update server CPU and core parameters.

**a.** Locate the line beginning with

```
NodeName=
```

**b.** Replace the line with

```
NodeName=c[01-04] Sockets=2 CoresPerSocket=24 ThreadsPerCore=2 State=UNKNOWN
```

**11.** Update the parition information.

Prepare the system with a partition that includes all cluster nodes regardless of type. This is useful for administrative and validation jobs, such as Intel® Cluster Checker. Compute partitions are created for each node type (subcluster).

**a.** Locate the the line beginning with

```
PartitionName=
```

b. Replace the single line with these two lines:

```
PartitionName=xeon Nodes=c[01-04] Default=YES MaxTime=24:00:00 State=UP
Oversubscribe=EXCLUSIVE
PartitionName=cluster Nodes=c[01-04] Default=NO MaxTime=24:00:00 State=UP
```

12. Save and close the file.

## Update Warewulf

13. Import Slurm configuration files into Warewulf.

```
wwsh -y file import /etc/slurm/slurm.conf
```

14. Change munge key owner.

```
chown munge:munge $CHROOT/etc/munge/munge.key
```

15. Import munge security keys into Warewulf.

```
wwsh -y file import /etc/munge/munge.key
```

16. Update the /etc/warewulf/defaults/provision.conf file.

a. Open the /etc/warewulf/defaults/provision.conf file for editing

b. Locate the line starting with:

```
files = dynamic_hosts, passwd…
```

c. Append the following to the end of the line:

```
, munge.key
```

Note the comma at the beginning of the added text. Do not change any part of the existing line; only append to the end.

d. Save and close the file

17. Enable the MUNGE service.

```
systemctl enable munge.service
```

18. Enable the Slurm controller

```
systemctl enable slurmctld.service
```

# Install Slurm Client

19. Add Slurm client support to the compute node image

```
dnf -y --installroot=$CHROOT install ohpc-slurm-client
```

20. Create a munge.key file.

    It will be replaced with the latest copy at node boot time.

```
cp -f /etc/munge/munge.key $CHROOT/etc/munge/munge.key
```

    Agree to overwrite the existing file, if necessary.

21. Change munge key owner.

```
chown munge:munge $CHROOT/etc/munge/munge.key
```

22. Create the munge log directory.

```
mkdir -p $CHROOT/var/log/munge
```

23. Update the client so that configuration is pulled from the server.

```
echo SLURMD_OPTIONS="--conf-server 192.168.1.100" > $CHROOT/etc/sysconfig/slurmd
```

24. Enable the MUNGE service.

```
systemctl --root=$CHROOT enable munge.service
```

25. Enable the Slurm client.

```
systemctl --root=$CHROOT enable slurmd.service
```

26. Update the initial Slurm configuration on the compute nodes.

    The Warewulf synchronization mechanism does not synchronize during early boot.

```
cp -f /etc/slurm/slurm.conf $CHROOT/etc/slurm/
```

    Agree to overwrite the existing file, if necessary.

27. Allow unrestricted SSH access for the users in the sshwhitelist group.

    Users not belonging to sshwhitelist will be subject to SSH restrictions.

    a. Open $CHROOT/etc/security/access.conf for editing.

    b. Append the following lines, in order, to the end of the file.

```
+ : root : ALL
+ : ssh-allowed : ALL
- : ALL : ALL
```

    c. Save and close the file.

28. Enable SSH control via Slurm workload manager.

    Enabling PAM (pluggable authentication module) in the chroot environment limits SSH access to only those nodes on which the user has active jobs.

    a. Open $CHROOT/etc/pam.d/sshd for editing

    b. Append the following lines, in order, to the end of the file

```
account sufficient pam_access.so
account required pam_slurm.so
```

c. Save and close the file

29. Update the VNFS image.

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

# Finalize Slurm configuration

30. Start the MUNGE service on the head node.

```
systemctl restart munge
```

31. Start the Slurm controller on the head node.

```
systemctl restart slurmctld
```

32. Reboot the compute nodes.

```
pdsh -w c[01-XX] init 6
```

Wait until compute nodes have been re-provisioned and are online again.

## Ensure that the compute nodes are available in Slurm

33. Make compute nodes available in Slurm.

In order to ensure that the compute nodes are available for resource allocation in Slurm, they need to be put into the “idle” state.

a. Update configuration state

```
scontrol reconfig
```

b. Move nodes to idle.

```
scontrol update NodeName=c[01-XX] State=idle
```

34. Check the Slurm status.

All nodes should have the state “idle”.

```
sinfo
```

# 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.

**35.** Add a test user.

```
useradd -m test
```

**36.** Synchronize files

```
wwsh file resync
```

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

```
su - test
```

**38.** Source the environment for the MPI implementation that will be used

```
source /usr/mpi/gcc/openmpi-4.0.3-hfi/bin/mpivars.sh
```

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

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

**40.** Submit the job.

Start a Slurm job on all nodes. You may adjust the numbers "4" in the command below for the number of nodes in your cluster. If all clocks are not synchronized, the job will fail (with an expired credentials error).

```
srun --partition=cluster -N 4 -n 4  /home/test/hello
```

This should produce output similar to the following

```
Hello, world (1 procs total)
   --> Process #   0 of   1 is alive. -> c001.default.cluster

Hello, world (1 procs total)
   --> Process #   0 of   1 is alive. -> c002.default.cluster

Hello, world (1 procs total)
   --> Process #   0 of   1 is alive. -> c003.default.cluster

Hello, world (1 procs total)
   --> Process #   0 of   1 is alive. -> c004.default.cluster
```

# Appendices

# Appendix A: Compile Intel® Omni-Path Fabric kernel modules

In order to install Omni-Path fabric software on updated kernels, kernel module packages will need to be created. This procedure is not required when using the Intel Fabric Suite INSTALL script.

The source RPMs will require additional development tools and the kernel source for the version to be installed. In order to keep the compute node image small, all additional tools and kernel sources will be installed on the head node and cross-compiled for any compute node images.

This document assumed that you have already updated the local repository with packages from the IFS tarball.

1. Install prerequisite packages for kernel module development

   ```
   dnf install gcc    rpm-build redhat-rpm-config kernel-rpm-macros
   ```

2. Install the kernel source for the required kernel.

   For the head node running kernel, **uname -r** will display the kernel version. Replace *<kernel_version>* with the version to be compiled.

   ```
   dnf -y install kernel-devel-<kernel_version>
   ```

3. Install dependencies for Omni-Path driver builds.

   ```
   dnf -y install elfutils-libelf-devel kernel-abi-whitelists
   ```

4. Rebuild the kernel RPMs

   ```
   rpmbuild --rebuild IntelOPA-IFS.RHEL81-x86_64.10.10.2.0.44/repos/\
   OPA_PKGS/SRPMS/ifs-kernel-updates-4.18.0_193.el8.x86_64-2123.src.rpm
   ```

   The resulting RPMs will be available in /root/rpmbuild/RPMS/x86_64.

5. Copy the new RPMs to the local repository

   ```
   cp ~/rpmbuild/RPMS/x86_64/*.rpm /srv/localrepo
   ```

6. Update the repository

   ```
   createrepo --update /usr/src/localrepo
   ```