

OpenHPC (v4.1) Cluster Building Recipes

Ubuntu 24.04 LTS Base OS

Warewulf4/SLURM Edition for Linux* (x86_64)



Document Last Update: 2026-04-25

Document Revision: 184207b66a8a684c7d2099670e939ce9d833fe9b

Legal Notice

Copyright © 2016-2023, OpenHPC, a Linux Foundation Collaborative Project. All rights reserved.



This documentation is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0>.

Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation in the U.S. and/or other countries.
*Other names and brands may be claimed as the property of others.

Contents

1	Introduction	5
1.1	Target Audience	5
1.2	Requirements/Assumptions	5
1.3	Inputs	6
2	Install Base Operating System (BOS)	7
3	Install OpenHPC Components	8
3.1	Enable OpenHPC repository for local use	8
3.2	Installation template	8
3.3	Add provisioning services on <i>master</i> node	9
3.4	Add resource management services on <i>master</i> node	9
3.5	Optionally add InfiniBand support services on <i>master</i> node	10
3.6	Optionally add Omni-Path support services on <i>master</i> node	11
3.7	Complete basic Warewulf setup for <i>master</i> node	11
3.8	Define <i>compute</i> image for provisioning	12
3.8.1	Build initial BOS image	12
3.8.2	Add OpenHPC components	12
3.8.3	Customize system configuration	13
3.8.4	Additional Customization (<i>optional</i>)	13
3.8.4.1	Enable InfiniBand drivers	13
3.8.4.2	Enable Omni-Path drivers	14
3.8.4.3	Increase locked memory limits	14
3.8.4.4	Enable ssh control via resource manager	14
3.8.4.5	Add GPU driver	14
3.8.4.6	Enable forwarding of system logs	14
3.8.4.7	Add ClusterShell	15
3.8.4.8	Add <i>genders</i>	15
3.8.4.9	Add Magpie	15
3.8.4.10	Add ConMan	16
3.8.4.11	Add NHC	16
3.8.5	Import files	16
3.9	Finalizing provisioning configuration	17
3.9.1	Two-Stage Provisioning (<i>optional</i>)	17
3.9.2	Build image image and overlays	18
3.9.3	Register nodes for provisioning	18
3.9.4	Optional kernel arguments	19
3.10	Boot compute nodes	19
4	Install OpenHPC Development Components	20
4.1	Development Tools	20
4.2	Compilers	20
4.3	MPI Stacks	20
4.4	Performance Tools	21
4.5	Setup default development environment	21
4.6	3rd Party Libraries and Tools	21
4.7	Optional Development Tool Builds	22
5	Resource Manager Startup	24

6	Post-boot compute node configuration	24
7	Run a Test Job	24
7.1	Interactive execution	25
7.2	Batch execution	26
	Appendices	27
A	Installation Template	27
B	Upgrading OpenHPC Packages	28
C	Integration Test Suite	29
D	Customizing system configuration	30
E	Package Manifest	32
F	Package Signatures	49

1 Introduction

This guide presents a simple cluster installation procedure using components from the OpenHPC software stack. OpenHPC represents an aggregation of a number of common ingredients required to deploy and manage an HPC Linux* cluster including provisioning tools, resource management, I/O clients, development tools, and a variety of scientific libraries. These packages have been pre-built with HPC integration in mind while conforming to common Linux distribution standards. The documentation herein is intended to be reasonably generic, but uses the underlying motivation of a small, 4-node stateless cluster installation to define a step-by-step process. Several optional customizations are included and the intent is that these collective instructions can be modified as needed for local site customizations.

Base Linux Edition: this edition of the guide highlights installation without the use of a companion configuration management system and directly uses distro-provided package management tools for component selection. The steps that follow also highlight specific changes to system configuration files that are required as part of the cluster install process.

1.1 Target Audience

This guide is targeted at experienced Linux system administrators for HPC environments. Knowledge of software package management, system networking, and PXE booting is assumed. Command-line input examples are highlighted throughout this guide via the following syntax:

```
[sms]# echo "OpenHPC hello world"
```

Unless specified otherwise, the examples presented are executed with elevated (root) privileges. The examples also presume use of the BASH login shell, though the equivalent commands in other shells can be substituted. In addition to specific command-line instructions called out in this guide, an alternate convention is used to highlight potentially useful tips or optional configuration options. These tips are highlighted via the following format:

Tip

Life is a tale told by an idiot, full of sound and fury signifying nothing. –Willy Shakes

1.2 Requirements/Assumptions

This installation recipe assumes the availability of a single head node *master*, and four *compute* nodes. The *master* node serves as the overall system management server (SMS) and is provisioned with Ubuntu 24.04 LTS and is subsequently configured to provision the remaining *compute* nodes with Warewulf4 in a stateless configuration. The terms *master* and SMS are used interchangeably in this guide. For power management, we assume that the compute node baseboard management controllers (BMCs) are available via IPMI from the chosen master host. For file systems, we assume that the chosen master server will host an NFS file system that is made available to the compute nodes. Installation information is also discussed to optionally mount a parallel file system and in this case, the parallel file system is assumed to exist previously.

An outline of the physical architecture discussed is shown in Figure 1 and highlights the high-level networking configuration. The *master* host requires at least two Ethernet interfaces with *eth0* connected to the local data center network and *eth1* used to provision and manage the cluster backend (note that these interface names are examples and may be different depending on local settings and OS conventions). Two

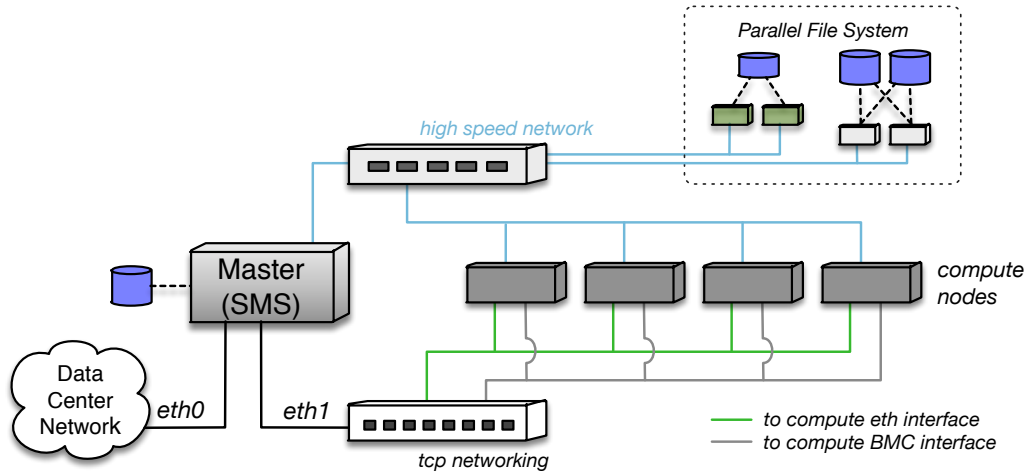


Figure 1: Overview of physical cluster architecture.

logical IP interfaces are expected to each compute node: the first is the standard Ethernet interface that will be used for provisioning and resource management. The second is used to connect to each host's BMC and is used for power management and remote console access. Physical connectivity for these two logical IP networks is often accommodated via separate cabling and switching infrastructure; however, an alternate configuration can also be accommodated via the use of a shared NIC, which runs a packet filter to divert management packets between the host and BMC. Independent of the actual networking configuration it is recommended to have additional security boundaries like a firewall to protect the network interfaces from the Internet.

In addition to the IP networking, there is an optional high-speed network (InfiniBand or Omni-Path in this recipe) that is also connected to each of the hosts. This high speed network is used for application message passing and optionally for parallel file system connectivity as well (e.g. to existing Lustre or BeeGFS storage targets).

1.3 Inputs

As this recipe details installing a cluster starting from bare-metal, there is a requirement to define IP addresses and gather hardware MAC addresses in order to support a controlled provisioning process. These values are necessarily unique to the hardware being used, and this document uses variable substitution (`${variable}`) in the command-line examples that follow to highlight where local site inputs are required. A summary of the required and optional variables used throughout this recipe are presented below. Note that while the example definitions above correspond to a small 4-node compute subsystem, the compute parameters are defined in array format to accommodate logical extension to larger node counts.

- `${sms_name}` # Hostname for SMS server
- `${sms_ip}` # Internal IP address on SMS server
- `${sms_eth_internal}` # Internal Ethernet interface on SMS
- `${eth_provision}` # Provisioning interface for computes
- `${internal_network}` # Subnet network address for internal network
- `${internal_netmask}` # Subnet netmask for internal network
- `${ipv4_gateway}` # Default gateway for the internal network
- `${dns_servers}` # DNS resolver for the internal network
- `${ntp_server}` # Local ntp server for time synchronization
- `${bmc_username}` # BMC username for use by IPMI
- `${bmc_password}` # BMC password for use by IPMI
- `${num_computes}` # Total # of desired compute nodes
- `${c_ip[0]}, ${c_ip[1]}, ...` # Desired compute node addresses
- `${c_bmc[0]}, ${c_bmc[1]}, ...` # BMC addresses for computes
- `${c_mac[0]}, ${c_mac[1]}, ...` # MAC addresses for computes
- `${c_name[0]}, ${c_name[1]}, ...` # Host names for computes
- `${compute_regex}` # Regex matching all compute node names (e.g. "c*")
- `${compute_prefix}` # Prefix for compute node names (e.g. "c")
- `${node_disk}` # Device path for disk to be used for provision-to-disk

Optional:

- `${sysmgmt_host}` # BeeGFS System Management host name
- `${mgs_fs_name}` # Lustre MGS mount name
- `${sms_ipoib}` # IPoIB address for SMS server
- `${ipoib_netmask}` # Subnet netmask for internal IPoIB
- `${c_ipoib[0]}, ${c_ipoib[1]}, ...` # IPoIB addresses for computes
- `${kargs}` # Kernel boot arguments

2 Install Base Operating System (BOS)

In an external setting, installing the desired BOS on a *master* SMS host typically involves booting from a DVD ISO image on a new server. With this approach, insert the Ubuntu 24.04 LTS DVD, power cycle the host, and follow the distro provided directions to install the BOS on your chosen *master* host. Alternatively, if choosing to use a pre-installed server, please verify that it is provisioned with the required Ubuntu 24.04 LTS distribution.

While it is theoretically possible to enable SELinux on a cluster provisioned with Warewulf4, doing so is beyond the scope of this document. Even the use of permissive mode can be problematic and we therefore recommend disabling SELinux on the *master* SMS host. If SELinux components are installed locally, the `selinuxenabled` command can be used to determine if SELinux is currently enabled. If enabled, consult the distro documentation for information on how to disable.

Finally, provisioning services rely on DHCP, TFTP, and HTTP network protocols. Depending on the local BOS configuration on the SMS host, default firewall rules may prohibit these services. Consequently, this recipe assumes that the local firewall running on the SMS host is disabled (it is still recommended to have additional security boundaries like a firewall to protect the cluster from the Internet). If installed, the default firewall service can be disabled as follows:

```
[sms]# systemctl disable --now ufw
```

3 Install OpenHPC Components

With the BOS installed and booted, the next step is to add desired OpenHPC packages onto the *master* server in order to provide provisioning and resource management services for the rest of the cluster. The following subsections highlight this process.

3.1 Enable OpenHPC repository for local use

To begin, enable use of the VersatusHPC Ubuntu 24.04 LTS repository by installing the OpenHPC release package. This mirrors the upstream release-package workflow: the release package installs the repository configuration and signing key, then APT metadata is refreshed from the enabled repository.

```
[sms]# export OHPC_REPO_ROOT=https://repos.versatushpc.com.br/openhpc/versatushpc-4
[sms]# export OHPC_UBUNTU_REPO=${OHPC_REPO_ROOT}/updates/Ubuntu_24.04
[sms]# export OHPC_RELEASE_DEB=ohpc-release_4.1-1ohpc1~noble_all.deb
[sms]# apt -y install ca-certificates curl
[sms]# curl -fsSLO ${OHPC_UBUNTU_REPO}/all/${OHPC_RELEASE_DEB}
[sms]# apt -y install ./${OHPC_RELEASE_DEB}
[sms]# rm -f ${OHPC_RELEASE_DEB}
[sms]# apt update
```

Tip

Many sites may find it useful or necessary to maintain a local copy of the OpenHPC repositories. To facilitate this need, standalone tar archives are provided containing the signed APT repository, binary packages, and metadata for local mirrors. The tar files also contain a simple bash script to configure the package manager to use the local repository after download. To use, simply unpack the tarball where you would like to host the local repository and execute the `make_repo.sh` script. Tar files for this release can be found at https://repos.versatushpc.com.br/openhpc/versatushpc-4/update.4.1/Ubuntu_24.04/dist/4.1

In addition to the OpenHPC package repository, the *master* host also requires access to the standard base OS distro repositories in order to resolve necessary dependencies. For Ubuntu 24.04 LTS, this recipe assumes a standard installation with the `main`, `universe`, and `multiverse` components enabled.

```
[sms]# apt -y install ca-certificates curl gnupg software-properties-common
[sms]# add-apt-repository -y universe
[sms]# add-apt-repository -y multiverse
[sms]# apt update
```

3.2 Installation template

The collection of command-line instructions that follow in this guide, when combined with local site inputs, can be used to implement a bare-metal system installation and configuration. The format of these commands is intended to be usable via direct cut and paste (with variable substitution for site-specific settings). Alternatively, the OpenHPC documentation package (`docs-ohpc`) includes a template script which includes a summary of all of the commands used herein. This script can be used in conjunction with a simple text file to define the local site variables defined in the previous section (§ 1.3) and is provided as a convenience for administrators. For additional information on accessing this script, please see Appendix A.

3.3 Add provisioning services on *master* node

With the OpenHPC repository enabled, we can now begin adding desired components onto the *master* server. This repository provides a number of aliases that group logical components together in order to help aid in this process. For reference, a complete list of available package aliases and Debian packages available via OpenHPC are provided in Appendix E. To add support for provisioning services, the following command adds a common base package followed along with the Warewulf provisioning system. Then the main Warewulf configuration files are edited to reflect the environment.

```
# Make sure required variables are set
[sms]# : ${sms_eth_internal:?ERROR: sms_eth_internal not set}
[sms]# : ${internal_network:?ERROR: internal_network not set}

# Install base packages
[sms]# apt -y install ohpc-base yq

# Install Warewulf
[sms]# apt -y install warewulf-ohpc
```

Tip

Many server BIOS configurations have PXE network booting configured as the primary option in the boot order by default. If your compute nodes have a different device as the first in the sequence, the `ipmitool` utility can be used to enable PXE.

```
[sms]# ipmitool -E -I lanplus -H ${bmc_ipaddr} -U root chassis bootdev pxe options=persistent
```

HPC systems rely on synchronized clocks throughout the system and the NTP protocol can be used to facilitate this synchronization. To enable NTP services on the SMS host with a specific server `${ntp_server}`, and allow this server to serve as a local time server for the cluster, issue the following:

```
[sms]# apt -y install chrony
[sms]# systemctl enable chrony.service
[sms]# echo "local stratum 10" >> /etc/chrony/chrony.conf
[sms]# echo "server ${ntp_server}" >> /etc/chrony/chrony.conf
[sms]# echo "allow all" >> /etc/chrony/chrony.conf
[sms]# systemctl restart chrony
```

Tip

Note that the “allow all” option specified for the chrony time daemon allows all servers on the local network to be able to synchronize with the SMS host. Alternatively, you can restrict access to fixed IP ranges and an example config line allowing access to a local class B subnet is as follows:

```
allow 192.168.0.0/16
```

3.4 Add resource management services on *master* node

OpenHPC provides multiple options for distributed resource management. The following command adds the Slurm workload manager server components to the chosen *master* host. Note that client-side components

will be added to the corresponding compute image in a subsequent step.

```
# Install slurm server meta-package
[sms]# apt -y install ohpc-slurm-server

# Use ohpc-provided file for starting SLURM configuration
[sms]# cp /etc/slurm/slurm.conf.ohpc /etc/slurm/slurm.conf
# Setup default cgroups file
[sms]# cp /etc/slurm/cgroup.conf.example /etc/slurm/cgroup.conf

# Identify resource manager hostname on master host
[sms]# perl -pi -e "s/SlurmctldHost=\S+/SlurmctldHost=${sms_name}/" /etc/slurm/slurm.conf
```

There are a wide variety of configuration options and plugins available for Slurm and the example config file illustrated above targets a fairly basic installation. In particular, job completion data will be stored in a text file (`/var/log/slurm_jobcomp.log`) that can be used to log simple accounting information. Sites who desire more detailed information, or want to aggregate accounting data from multiple clusters, will likely want to enable the database accounting back-end. This requires a number of additional local modifications (on top of installing `slurm-slurmdbd-ohpc`), and users are advised to consult the online [documentation](#) for more detailed information on setting up a database configuration for Slurm.

Tip

SLURM requires enumeration of the physical hardware characteristics for compute nodes under its control. In particular, three configuration parameters combine to define consumable compute resources: *Sockets*, *CoresPerSocket*, and *ThreadsPerCore*. The default configuration file provided via OpenHPC assumes the nodes are named `cl-c4` and are dual-socket, 8 cores per socket, and two threads per core for this 4-node example. If this does not reflect your local hardware, please update the configuration file at `/etc/slurm/slurm.conf` accordingly to match your nodes names and particular hardware. Be sure to run `scontrol reconfigure` to notify SLURM of the changes. Note that the SLURM project provides an easy-to-use online configuration tool that can be accessed [here](#).

Other versions of this guide are available that describe installation of alternate resource management systems, and they can be found in the `docs-ohpc` package.

3.5 Optionally add InfiniBand support services on *master* node

The following command adds InfiniBand support using base distro-provided drivers to the chosen *master* host.

```
[sms]# apt -y install rdma-core ibverbs-providers infiniband-diags perftest opensm
# Load IB services
[sms]# udevadm trigger --type=devices --action=add
[sms]# systemctl restart rdma-load-modules@infiniband.service
```

Tip

InfiniBand networks require a subnet management service that can typically be run on either an administrative node, or on the switch itself. The optimal placement and configuration of the subnet manager is beyond the scope of this document, but Ubuntu 24.04 LTS provides the `opensm` package should you choose to run it on the *master* node.

With the InfiniBand drivers included, you can also enable (optional) IPoIB functionality which provides a mechanism to send IP packets over the IB network. If you plan to mount a Lustre file system over InfiniBand (see §?? for additional details), then having IPoIB enabled is a requirement for the Lustre client. Configure the IPoIB interface according to local network policy and the interface names assigned by Ubuntu.

3.6 Optionally add Omni-Path support services on *master* node

Ubuntu 24.04 does not currently provide the Omni-Path package set used by the upstream EL and openEuler recipes. Sites that require Omni-Path should add vendor-supported Ubuntu packages or a local repository before enabling this option.

3.7 Complete basic Warewulf setup for *master* node

At this point, all of the packages necessary to use Warewulf on the *master* host should be installed. Next, we need to update the configuration to allow Warewulf to work with Ubuntu 24.04 LTS, update the hosts file, and to support local provisioning using a second private interface (refer to Figure 1).

```
# Edit the warewulf.conf file to use appropriate interface and settings
[sms]# yq -y -i '.ipaddr = "${sms_ip}"' /etc/warewulf/warewulf.conf
[sms]# yq -y -i '.netmask = "${internal_netmask}"' /etc/warewulf/warewulf.conf
[sms]# yq -y -i '.network = "${internal_network}"' /etc/warewulf/warewulf.conf
[sms]# yq -y -i '.dhcp["range start"] = "${internal_network}"' /etc/warewulf/warewulf.conf
[sms]# yq -y -i '.dhcp["range end"] = "static"' /etc/warewulf/warewulf.conf
[sms]# yq -y -i '.dhcp.template = "static"' /etc/warewulf/warewulf.conf

# Edit the nodes.conf to mount /opt on boot
[sms]# perl -pi -e "s/defaults,noauto,nofail,ro/defaults,nofail,ro/" /etc/warewulf/nodes.conf

# Turn on debugging messages
[sms]# yq -y -i '.nodeprofiles.default.kernel.args -= ["quiet"]' /etc/warewulf/nodes.conf
[sms]# echo "log-debug" >> /etc/dnsmasq.d/ww4-debug.conf

# Enable and start the warewulfd service before using wwctl.
[sms]# systemctl enable --now warewulfd

# Create a new "nodes" profile and inherit the "default" profile
[sms]# wwctl profile add nodes --profile default --comment "Nodes profile"

# Create a new "nodeconfig" overlay to store node configuration files and use the syncuser overlay.
[sms]# wwctl overlay create nodeconfig
[sms]# wwctl profile set --yes nodes --system-overlays nodeconfig --runtime-overlays syncuser

# Set default network configuration
[sms]# wwctl profile set -y nodes --netname=default --netdev=${eth_provision}
[sms]# wwctl profile set -y nodes --netname=default --netmask=${internal_netmask}
[sms]# wwctl profile set -y nodes --netname=default --gateway=${ipv4_gateway}
[sms]# wwctl profile set -y nodes --netname=default --nettagadd=DNS=${dns_servers}

# Configuring Warewulf will restart/enable relevant services to support provisioning
[sms]# wwctl configure --all

# Generate ssh keys (usually generated on login)
[sms]# bash /etc/profile.d/ssh_setup.sh
```

3.8 Define *compute* image for provisioning

With the provisioning services enabled, the next step is to define and customize a system image that can subsequently be used to provision one or more *compute* nodes. The following subsections highlight this process.

3.8.1 Build initial BOS image

Warewulf 4 supports using container images as the base file system for provisioning, and it can import these directly from an OCI registry like Docker Hub. In this example we import the official Ubuntu container image and install the kernel and init-system packages needed for Warewulf provisioning.

The `wwctl image exec` command runs the commands below it, these commands also be run interactively one a time with the command `wwctl image shell ubuntu-24.04`. You can add `/bin/false` as the last command to prevent the image from rebuilding (it will show an error) and rebuild later with the `wwctl image build` command.

```
# Import the base image
[sms]# wwctl image import docker://ubuntu:24.04 ubuntu-24.04 --syncuser

# Define chroot location
[sms]# export CHROOT=$(wwctl image show ubuntu-24.04)

# Enable OpenHPC inside image and update image. Disable image build on exit, rebuild later.
[sms]# wwctl image exec --build=false ubuntu-24.04 -- /bin/bash -ex <<- EOF
apt update
apt -y install ca-certificates curl systemd systemd-sysv \
    udev kmod linux-image-generic initramfs-tools network-manager openssh-server
systemctl enable ssh.service
OHPC_REPO_ROOT=https://repos.versatushpc.com.br/openhpc/versatushpc-4
OHPC_UBUNTU_REPO=${OHPC_REPO_ROOT}/updates/Ubuntu_24.04
OHPC_RELEASE_DEB=ohpc-release_4.1-1ohpc1~noble_all.deb
curl -fsSLO ${OHPC_UBUNTU_REPO}/all/${OHPC_RELEASE_DEB}
apt -y install ./${OHPC_RELEASE_DEB}
rm -f ${OHPC_RELEASE_DEB}
apt update
apt -y upgrade
EOF
```

3.8.2 Add OpenHPC components

The process used in the previous step is designed to provide a minimal Ubuntu 24.04 LTS configuration. Next, we add additional components to include resource management client services, NTP support, and other additional packages to support the default OpenHPC environment. This process modifies the base provisioning image and will access the BOS and OpenHPC repositories to resolve package install requests.

The instructions below are designed to be copied and pasted all at once into a terminal. Alternatively you can run `wwctl image shell ubuntu-24.04` to "run" the node image interactively and run the commands one at a time; this method is recommended for local debugging.

We begin by installing a few common base packages:

```
# Install compute node base meta-package
[sms]# wwctl image exec --build=false ubuntu-24.04 -- /bin/bash -ex <<- EOF
apt -y install ohpc-base-compute
EOF
```

Now, we can include additional required components to the compute instance including resource manager client, NTP, and development environment modules support.

For Ubuntu compute images, use `wwctl image shell` or `wwctl image exec` for APT work. These commands mount `/dev`, `/proc`, `/sys`, and `/run`; direct `chroot $CHROOT` can create `/dev/null` as a regular file in minimal OCI images and break later package operations.

```
[sms]# wwctl image exec --build=false ubuntu-24.04 -- /bin/bash -ex <<- EOF
# Add Slurm client support meta-package
apt -y install ohpc-slurm-client

# Enable services to start on boot
systemctl enable munge.service
systemctl enable slurmd.service

# Add Network Time Protocol (NTP) support
apt -y install chrony

# Include modules user environment
apt -y install lmod-ohpc
EOF
```

3.8.3 Customize system configuration

If planning to install the Intel® oneAPI compiler runtime (see §4.7), register the following additional path (`/opt/intel`) to share with computes:

```
# (Optional) Setup NFS mount for /opt/intel if planning to install oneAPI packages
[sms]# mkdir /opt/intel
[sms]# echo "/opt/intel *(ro,no_subtree_check,fsid=12)" >> /etc/exports
[sms]# echo "${sms_ip}:/opt/intel /opt/intel nfs nfsvers=4,nodev 0 0" >> $CHROOT/etc/fstab
```

3.8.4 Additional Customization (*optional*)

This section highlights common additional customizations that can *optionally* be applied to the local cluster environment. These customizations include:

- Add InfiniBand or Omni-Path drivers
- Increase memlock limits
- Restrict ssh access to compute resources
- Add BeeGFS client
- Add Lustre client
- Enable syslog forwarding
- Add ClusterShell
- Add *mrsh*
- Add *genders*
- Add ConMan
- Add GEOPM

Details on the steps required for each of these customizations are discussed further in the following sections.

3.8.4.1 Enable InfiniBand drivers If your compute resources support InfiniBand, the following commands add OFED and PSM support using base distro-provided drivers to the compute image.

```
# Add IB support
[sms]# wwctl image exec --build=false ubuntu-24.04 - apt -y install rdma-core ibverbs-providers \
    infiniband-diags perftest
```

3.8.4.2 Enable Omni-Path drivers If your compute resources support Omni-Path, install the vendor-supported Ubuntu packages in the compute image before enabling this option.

3.8.4.3 Increase locked memory limits In order to utilize InfiniBand or Omni-Path as the underlying high speed interconnect, it is generally necessary to increase the locked memory settings for system users. This can be accomplished by updating the `/etc/security/limits.conf` file and this should be performed within the *compute* image and on all job submission hosts. In this recipe, jobs are submitted from the *master* host, and the following commands can be used to update the maximum locked memory settings on both the master host and the compute image:

```
# Update memlock settings on master
[sms]# perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s' /etc/security/limits.conf
[sms]# perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s' /etc/security/limits.conf

# Update memlock settings within compute image
[sms]# perl -pi -e 's/# End of file/\* soft memlock unlimited\n$&/s' $CHROOT/etc/security/limits.conf
[sms]# perl -pi -e 's/# End of file/\* hard memlock unlimited\n$&/s' $CHROOT/etc/security/limits.conf
```

3.8.4.4 Enable ssh control via resource manager An additional optional customization that is recommended is to restrict `ssh` access on compute nodes to only allow access by users who have an active job associated with the node. This can be enabled via the use of a pluggable authentication module (PAM) provided as part of the Slurm package installs. To enable this feature within the *compute* image, issue the following:

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

3.8.4.5 Add GPU driver To add the optional NVIDIA GPU driver to the compute nodes, an additional external apt repository provided by NVIDIA must be configured. Once the repository is configured, the GPU driver needs to be installed on the compute image and the corresponding toolkit installed on the SMS node.

OpenHPC provides a convenience package to enable the NVIDIA repository locally along with compatibility packages that integrate the NVIDIA HPC SDK within the standard OpenHPC user environment.

```
# Add NVIDIA GPU driver repository to the SMS
[sms]# apt -y install cuda-repo-ohpc
[sms]# apt update
# Install the toolkit on the SMS
[sms]# apt -y install cuda-drivers cuda-devel-ohpc
# Install NVIDIA repository and GPU driver in the compute image
[sms]# wvctl image exec --build=false ubuntu-24.04 -- /bin/bash -ex <<- EOF
    apt -y install cuda-repo-ohpc
    apt update
    apt -y install cuda-drivers
EOF
```

3.8.4.6 Enable forwarding of system logs It is often desirable to consolidate system logging information for the cluster in a central location, both to provide easy access to the data, and to reduce the impact of storing data inside the compute node's memory footprint if it is stateless. The following commands highlight the steps necessary to configure compute nodes to forward their logs to the master node, and to allow the master node to accept these log requests.

```
# Configure master node to receive messages and reload rsyslog configuration
[sms]# echo 'module(load="imudp")' >> /etc/rsyslog.d/ohpc.conf
[sms]# echo 'input(type="imudp" port="514")' >> /etc/rsyslog.d/ohpc.conf
[sms]# systemctl restart rsyslog

# Install rsyslog in the minimal Ubuntu compute image
[sms]# wvctl image exec --build=false ubuntu-24.04-- apt -y install rsyslog
[sms]# wvctl image exec --build=false ubuntu-24.04-- install -d -m 0755 /etc/rsyslog.d

# Define compute node forwarding destination
[sms]# echo '*. * @${sms_ip}:'514' > $CHROOT/etc/rsyslog.d/ohpc-forward.conf

# Disable most local logging on computes. Emergency and boot logs will remain.
[sms]# perl -pi -e "s/^*\.\.info/\#\*\.\.info/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^authpriv/\#authpriv/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^mail/\#mail/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^cron/\#cron/" $CHROOT/etc/rsyslog.conf
[sms]# perl -pi -e "s/^uucp/\#uucp/" $CHROOT/etc/rsyslog.conf
```

3.8.4.7 Add ClusterShell ClusterShell is an event-based Python library to execute commands in parallel across cluster nodes. Installation and basic configuration defining three node groups (*adm*, *compute*, and *all*) is as follows:

```
# Install ClusterShell
[sms]# apt -y install clustershell

# Setup node definitions
[sms]# cd /etc/clusterhell/groups.d
[sms]# mv local.cfg local.cfg.orig
[sms]# echo "adm: ${sms_name}" > local.cfg
[sms]# echo "compute: ${compute_prefix}[1-${num_computes}]" >> local.cfg
[sms]# echo "all: @adm,@compute" >> local.cfg
```

3.8.4.8 Add *genders* *genders* is a static cluster configuration database or node typing database used for cluster configuration management. Other tools and users can access the *genders* database in order to make decisions about where an action, or even what action, is appropriate based on associated types or "*genders*". Values may also be assigned to and retrieved from a *gender* to provide further granularity. The following example highlights installation and configuration of two genders: *compute* and *bmc*.

```
# Install genders
[sms]# apt -y install genders-ohpc

# Generate a sample genders file
[sms]# echo -e "${sms_name}\t sms" > /etc/genders
[sms]# for ((i=0; i<$num_computes; i++)) ; do
    echo -e "${c_name[$i]}\t compute,bmc=${c_bmc[$i]}"
done >> /etc/genders
```

3.8.4.9 Add Magpie Magpie contains a number of scripts to aid in running a variety of big data software frameworks within HPC queuing environments. Examples include Hadoop, Spark, Hbase, Storm, Pig, Mahout, Phoenix, Kafka, Zeppelin, and Zookeeper. Consult the online [repository](#) for more information on using these scripts; basic installation is outlined as follows:

```
# Install magpie
[sms]# apt -y install magpie-ohpc
```

3.8.4.10 Add ConMan ConMan is a serial console management program designed to support a large number of console devices and simultaneous users. It supports logging console device output and connecting to compute node consoles via IPMI serial-over-lan. Installation and example configuration is outlined below.

```
# Install conman to provide a front-end to compute consoles and log output
[sms]# apt -y install conman-ohpc

# Configure conman for computes (note your IPMI password is required for console access)
[sms]# for ((i=0; i<$num_compute; i++)) ; do
    echo -n 'CONSOLE name="'${c_name[$i]}'" dev="ipmi:'${c_bmc[$i]}'" '
    echo 'ipmiopts="'U:'${bmc_username},P:'${IPMI_PASSWORD:-undefined},W:solpayloadsize'"
done >> /etc/conman.conf

# Enable and start conman
[sms]# systemctl enable conman
[sms]# systemctl start conman
```

Note that an additional kernel boot option is typically necessary to enable serial console output. This option is highlighted in §3.9.4 after compute nodes have been registered with the provisioning system.

3.8.4.11 Add NHC Resource managers often provide for a periodic "node health check" to be performed on each compute node to verify that the node is working properly. Nodes which are determined to be "unhealthy" can be marked as down or offline so as to prevent jobs from being scheduled or run on them. This helps increase the reliability and throughput of a cluster by reducing preventable job failures due to misconfiguration, hardware failure, etc. OpenHPC distributes NHC to fulfill this requirement.

In a typical scenario, the NHC driver script is run periodically on each compute node by the resource manager client daemon. It loads its configuration file to determine which checks are to be run on the current node (based on its hostname). Each matching check is run, and if a failure is encountered, NHC will exit with an error message describing the problem. It can also be configured to mark nodes offline so that the scheduler will not assign jobs to bad nodes, reducing the risk of system-induced job failures.

```
# Install NHC on master and compute nodes
[sms]# apt -y install nhc-ohpc
[sms]# wvctl image exec -build=false ubuntu-24.04 - apt -y install nhc-ohpc
```

```
# Register as SLURM's health check program
[sms]# echo "HealthCheckProgram=/usr/sbin/nhc" >> /etc/slurm/slurm.conf
[sms]# echo "HealthCheckInterval=300" >> /etc/slurm/slurm.conf # execute every five minutes
```

3.8.5 Import files

The Warewulf system includes functionality to import arbitrary files from the provisioning server for distribution to managed hosts through a system called "overlays". Some files, like `/etc/passwd`, and `/etc/hosts` handled in this way by default. Here we add directories and files to the `nodesconfig` overlay that is applied to all nodes.

```
# Add the following to support unprivileged user namespaces for tools like Apptainer
[sms]# wvctl overlay import --parents nodeconfig /etc/subuid
```



```
[sms]# wwctl overlay import --parents nodeconfig /etc/subgid

# Identify master host as local NTP server, configure it with a template with a Tag
[sms]# wwctl overlay import --parents nodeconfig \
/opt/ohpc/pub/examples/chrony.conf.ww /etc/chrony.conf.ww
[sms]# wwctl profile set --yes nodes --tagadd ntpserver=${sms_ip}

# Configure systemd and NetworkManager to wait for the network to be fully up.
[sms]# wwctl overlay import --parents nodeconfig \
/opt/ohpc/pub/examples/network/NetworkManager-wait-online.service.d/override.conf \
/etc/systemd/system/NetworkManager-wait-online.service.d/override.conf
```

Similarly, we can configure Slurm and import 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:

```
# Configure Slurm server in the overlay (using "configless" option)
# using a tag in a template file (slurmd.ww)
[sms]# wwctl overlay import --parents nodeconfig \
/opt/ohpc/pub/examples/slurm/slurmd.ww /etc/sysconfig/slurmd.ww

# Set the value of the slurmd tag to the $sms_ip for the nodes profile.
[sms]# wwctl profile set --yes nodes --tagadd slurmd=${sms_ip}

# Configure munge
[sms]# wwctl overlay import --parents nodeconfig /etc/munge/munge.key
[sms]# wwctl overlay chown nodeconfig /etc/munge/munge.key $(id -u munge) $(id -g munge)
[sms]# wwctl overlay chown nodeconfig /etc/munge $(id -u munge) $(id -g munge)
[sms]# wwctl overlay chmod nodeconfig /etc/munge 0700
```

Finally, to add *optional* support for controlling IPoIB interfaces (see §3.5), add a site-local NetworkManager or Netplan configuration template to the Warewulf overlay that matches the Ubuntu network naming used by your compute nodes.

```
# Import a site-local Ubuntu IPoIB template if one is used locally
[sms]# echo "Add Ubuntu IPoIB overlay template for ib0 here."
```

3.9 Finalizing provisioning configuration

Warewulf provisions a node with an image then customizes it with overlays. This section highlights creation of the node image and overlays, followed by the registration of desired compute nodes.

3.9.1 Two-Stage Provisioning (optional)

Warewulf optionally supports two-stage provisioning to ramfs or disk on the compute nodes. The nodes are still stateless, but the image is copied to disk during the provisioning process and is reprovisioned on every boot. This can be useful for cases when the compute node images are large, for example GPU drivers, or the compute nodes have limited memory.

To configure two-stage provisioning, install Dracut in the compute node image. The example below provisions to ramfs, which is the default. To provision to disk, see the next section.

```
## Add rd.shell to kernel arguments in the "default" profile.
[sms]# yq -y -i '.nodeprofiles.default.kernel.args += ["rd.shell"]' \
/etc/warewulf/nodes.conf

## Install Dracut in the image
[sms]# wwctl image exec --build=false ubuntu-24.04-- \
```

```

/usr/bin/apt -y install warewulf-dracut-ohpc

## Configure Dracut. Note the leading and trailing spaces in the quotes in "add_dracutmodules"
[sms]# echo 'hostonly="no"' > $CHROOT/etc/dracut.conf.d/wwinit.conf
[sms]# echo 'add_dracutmodules+=" wwinit "' >> $CHROOT/etc/dracut.conf.d/wwinit.conf

## Enable Dracut boot for compute nodes that use the "nodes" profile
[sms]# wwctl profile set --yes nodes --tagadd IPXEMenuEntry=dracut

```

Optionally, configure the compute node to provision to disk. Two stage provisioning (previous step) must also be enabled. The compute node will use, **and erase**, the disk specified by `node_disk`.

```

## Install disk tools
[sms]# wwctl image exec --build=false ubuntu-24.04-- /usr/bin/apt -y install ignition gdisk

## Add ignition Dracut module.
[sms]# echo 'add_dracutmodules+=" ignition "' >> $CHROOT/etc/dracut.conf.d/wwinit.conf

## Create the target "rootfs" partition and filesystem
[sms]# wwctl profile set --yes nodes \
  --diskname ${node_disk} --diskwipe \
  --partname rootfs --partcreate --partnumber 1 \
  --fsname rootfs --fswipe --fsformat ext4 --fspath /

## Enable provision-to-disk for compute nodes that use the "nodes" profile
[sms]# wwctl profile set nodes --yes --root=/dev/disk/by-partlabel/rootfs

```

3.9.2 Build image and overlays

The bootstrap image includes the runtime kernel and associated modules, as well as some simple scripts to complete the provisioning process. We explicitly rebuild the image at the end because rebuilding the image is disabled in earlier steps. It is good practice to rebuild the overlays when after configuration changes.

```

# Build Dracut initramfs and image
[sms]# wwctl image exec --build=false ubuntu-24.04-- /usr/bin/dracut --force --regenerate-all
[sms]# wwctl image build ubuntu-24.04
[sms]# wwctl overlay build

```

3.9.3 Register nodes for provisioning

Nodes can be registered for provisioning using the following syntax.

```

[sms]# for ((i=0; i<$num_computes; i++)) ; do
  wwctl node add --image=ubuntu-24.04 --profile=nodes --netname=default \
    --ipaddr=${c_ip[$i]} --hwaddr=${c_mac[$i]} ${c_name[$i]}
done

```

```

# Optionally define IPoIB network settings (required if planning to mount Lustre/BeeGFS over IB)
[sms]# for ((i=0; i<$num_computes; i++)) ; do
  wwctl node set --yes ${c_name[$i]} --netname=ib --netdev=ib0 --ipaddr=${c_ipoib[$i]} --netmask=${ipoib_netmask}
done

```

Finally, we reconfigure build the overlays and update the Warewulf configuration. It is necessary to rebuild the overlays whenever a overlay is modified.

```
# build the overlays for all the nodes
[sms]# wwctl overlay build

# Update Warewulf configure
[sms]# wwctl configure --all
```

Now that the nodes are defined, we can start munge and Slurm. This must be done after the nodes are defined and the Warewulf configuration is updated.

```
# Enable and start munge and slurmctld
[sms]# systemctl enable --now munge
[sms]# systemctl enable --now slurmctld
```

3.9.4 Optional kernel arguments

Tip

Typical Charliecloud workflows are based around Docker containers, but it is not strictly necessary to install Docker itself on the HPC resource. A common pattern is to build the Docker container on a laptop or VM and upload the result to the cluster for use with Charliecloud. More information can be found at <https://hpc.github.io/charliecloud/>

If any components have added to the boot time kernel command line arguments for the compute nodes, the following command is required to store the configuration in Warewulf:

```
# Set optional compute node kernel command line arguments.
[sms]# wwctl profile set --yes nodes --kernelargs="${kargs}"
```

3.10 Boot compute nodes

At this point, the *master* server should be able to boot the newly defined compute nodes. Assuming that the compute node BIOS settings are configured to boot over PXE, all that is required to initiate the provisioning process is to power cycle each of the desired hosts using IPMI access. The following commands use the `ipmitool` utility to initiate power resets on each of the four compute hosts. Note that the utility requires that the `IPMI_PASSWORD` environment variable be set with the local BMC password in order to work interactively.

```
[sms]# for ((i=0; i<${num_computes}; i++)) ; do
        ipmitool -E -I lanplus -H ${c_bmc[$i]} -U ${bmc_username} -P ${bmc_password} chassis power reset
    done
```

Once kicked off, the boot process should take less than 5 minutes (depending on BIOS post times) and you can verify that the compute hosts are available via ssh, or via parallel ssh tools to multiple hosts. For example, to run a command on the newly imaged compute hosts using `pdsh`, execute the following:

```
[sms]# pdsh -w ${compute_prefix}[1-${num_computes}] uptime
c1 05:03am up 0:02, 0 users, load average: 0.20, 0.13, 0.05
c2 05:03am up 0:02, 0 users, load average: 0.20, 0.14, 0.06
c3 05:03am up 0:02, 0 users, load average: 0.19, 0.15, 0.06
c4 05:03am up 0:02, 0 users, load average: 0.15, 0.12, 0.05
```

4 Install OpenHPC Development Components

The install procedure outlined in §3 highlighted the steps necessary to install a *master* host, assemble and customize a *compute* image, and provision several compute hosts from bare-metal. With these steps completed, additional OpenHPC-provided packages can now be added to support a flexible HPC development environment including development tools, C/C++/FORTRAN compilers, MPI stacks, and a variety of 3rd party libraries. The following subsections highlight the additional software installation procedures.

4.1 Development Tools

To aid in general development efforts, OpenHPC provides recent versions of the GNU autotools collection, the Valgrind memory debugger, EasyBuild, and Spack. These can be installed as follows:

```
# Install autotools meta-package
[sms]# apt -y install ohpc-autotools

[sms]# apt -y install easybuild-ohpc
[sms]# apt -y install hwloc-ohpc
[sms]# apt -y install spack-ohpc
[sms]# apt -y install valgrind-ohpc
```

4.2 Compilers

OpenHPC presently packages the GNU compiler toolchain integrated with the underlying Lmod modules system in a hierarchical fashion. The modules system will conditionally present compiler-dependent software based on the toolchain currently loaded.

```
[sms]# apt -y install gnu15-compilers-ohpc
```

4.3 MPI Stacks

For MPI development and runtime support, OpenHPC provides pre-packaged builds for a variety of MPI families and transport layers. Currently available options and their applicability to various network transports are summarized in Table 1. The command that follows installs a starting set of MPI families that are compatible with both ethernet and high-speed fabrics.

Table 1: Available MPI variants

	Ethernet (TCP)	InfiniBand	Intel® Omni-Path
MPICH (ofi)	✓	✓	✓
MPICH (ucx)	✓	✓	✓
MVAPICH2		✓	
OpenMPI (ofi/ucx)	✓	✓	✓

```
[sms]# apt -y install openmpi5-gnu15-ohpc mpich-gnu15-ohpc
```

Note that OpenHPC 2.x introduces the use of two related transport layers for the MPICH and OpenMPI builds that support a variety of underlying fabrics: **UCX** (Unified Communication X) and **OFI** (OpenFabrics interfaces). For MPICH, separate builds are provided and users can switch between the two via normal

module command semantics. Alternatively, a site can choose to install the ucx variant instead as a drop-in MPICH replacement:

```
[sms]# apt -y install mpich-ucx-gnu15-ohpc
```

If your system includes InfiniBand and you enabled underlying support in §3.5 and §3.8.4, an additional MVAPICH2 family is available for use:

```
[sms]# apt -y install mvapich2-gnu15-ohpc
```

4.4 Performance Tools

OpenHPC provides a variety of open-source tools to aid in application performance analysis (refer to Appendix E for a listing of available packages). This group of tools can be installed as follows:

```
# Install perf-tools meta-package
[sms]# apt -y install ohpc-gnu15-perf-tools
```

4.5 Setup default development environment

System users often find it convenient to have a default development environment in place so that compilation can be performed directly for parallel programs requiring MPI. This setup can be conveniently enabled via modules and the OpenHPC modules environment is pre-configured to load an `ohpc` module on login (if present). The following package install provides a default environment that enables autotools, the GNU compiler toolchain, and the OpenMPI stack.

```
[sms]# apt -y install lmod-defaults-gnu15-openmpi5-ohpc
```

4.6 3rd Party Libraries and Tools

OpenHPC provides pre-packaged builds for a number of popular open-source tools and libraries used by HPC applications and developers. For example, OpenHPC provides builds for FFTW and HDF5 (including serial and parallel I/O support), and the GNU Scientific Library (GSL). Again, multiple builds of each package are available in the OpenHPC repository to support multiple compiler and MPI family combinations where appropriate. Note, however, that not all combinatorial permutations may be available for components where there are known license incompatibilities. The general naming convention for builds provided by OpenHPC is to append the compiler and MPI family name that the library was built against directly into the package name. For example, libraries that do not require MPI as part of the build process adopt the following RPM name:

```
package-<compiler_family>-ohpc-<package_version>-<release>.rpm
```

Packages that do require MPI as part of the build expand upon this convention to additionally include the MPI family name as follows:

```
package-<compiler_family>-<mpi_family>-ohpc-<package_version>-<release>.rpm
```

To illustrate this further, the command below queries the locally configured repositories to identify all of the available PETSc packages that were built with the GNU toolchain. The resulting output that is included shows that pre-built versions are available for each of the supported MPI families presented in §4.3.

```
[sms]# apt search petsc-gnu15 | grep ohpc
petsc-gnu15-impi-ohpc/noble 3.23.7 amd64
petsc-gnu15-mpich-ohpc/noble 3.23.7 amd64
petsc-gnu15-mvapich2-ohpc/noble 3.23.7 amd64
petsc-gnu15-openmpi5-ohpc/noble 3.23.7 amd64
```

Tip

OpenHPC-provided 3rd party builds are configured to be installed into a common top-level repository so that they can be easily exported to desired hosts within the cluster. This common top-level path (`/opt/ohpc/pub`) was previously configured to be mounted on *compute* nodes in §3.8.3, so the packages will be immediately available for use on the cluster after installation on the *master* host.

For convenience, OpenHPC provides package aliases for these 3rd party libraries and utilities that can be used to install available libraries for use with the GNU compiler family toolchain. For parallel libraries, aliases are grouped by MPI family toolchain so that administrators can choose a subset should they favor a particular MPI stack. Please refer to Appendix E for a more detailed listing of all available packages in each of these functional areas. To install all available package offerings within OpenHPC, issue the following:

```
# Install 3rd party libraries/tools meta-packages built with GNU toolchain
[sms]# apt -y install ohpc-gnu15-serial-libs
[sms]# apt -y install ohpc-gnu15-io-libs
[sms]# apt -y install ohpc-gnu15-python3-libs
[sms]# apt -y install ohpc-gnu15-runtimes
```

```
# Install parallel lib meta-packages for available default MPI toolchains
[sms]# apt -y install ohpc-gnu15-mpich-parallel-libs
[sms]# apt -y install ohpc-gnu15-openmpi5-parallel-libs
```

4.7 Optional Development Tool Builds

In addition to the 3rd party development libraries built using the open source toolchains mentioned in §4.6, OpenHPC also provides *optional* compatible builds for use with the compilers and MPI stack included in newer versions of the Intel® oneAPI HPC Toolkit. These packages provide a similar hierarchical user environment experience as other compiler and MPI families present in OpenHPC.

To take advantage of the available builds, install the compatibility packages that integrate oneAPI-generated compiler and MPI modulefiles within the standard OpenHPC user environment. On Ubuntu these compatibility packages configure the Intel APT repository and install the required oneAPI packages through APT.

```
# Enable Intel oneAPI and install OpenHPC compatibility packages
[sms]# apt -y install intel-compilers-devel-ohpc intel-mpi-devel-ohpc
```

Tip

As noted in §3.8.3, the default installation path for OpenHPC (`/opt/ohpc/pub`) is exported over NFS from the *master* to the compute nodes, but the Intel® oneAPI HPC Toolkit packages install to a top-level path of `/opt/intel`. To make the Intel® compilers available to the compute nodes one must add an additional NFS export for `/opt/intel` that is mounted on desired compute nodes.

The Ubuntu package manifest in Appendix E lists the Intel-family library builds available for local selection.

5 Resource Manager Startup

In section §3, the Slurm resource manager was installed and configured for use on both the *master* host and *compute* node instances. With the cluster nodes up and functional, we can now startup the resource manager services in preparation for running user jobs. Generally, this is a two-step process that requires starting up the controller daemons on the *master* host and the client daemons on each of the *compute* hosts. Note that Slurm leverages the use of the *munge* library to provide authentication services and this daemon also needs to be running on all hosts within the resource management pool. The following commands can be used to startup the necessary services to support resource management under Slurm.

```
# Start munge and slurm controller on master host
[sms]# systemctl enable munge
[sms]# systemctl enable slurmctld
[sms]# systemctl start munge
[sms]# systemctl start slurmctld

# Start slurm clients on compute hosts
[sms]# pdsh -w ${compute_prefix}[1-${num_computes}] systemctl start munge
[sms]# pdsh -w ${compute_prefix}[1-${num_computes}] systemctl start slurmd
```

6 Post-boot compute node configuration

```
# Generate NHC configuration file based on compute node environment
[sms]# pdsh -w c1 "/usr/sbin/nhc-genconf -H '*' -c -" | dshbak -c
```

7 Run a Test Job

With the resource manager enabled for production usage, users should now be able to run jobs. To demonstrate this, we will add a “test” user on the *master* host that can be used to run an example job.

```
[sms]# useradd -m test
```

Warewulf installs a utility on the compute nodes to automatically synchronize overlay files from the provisioning server at one minute intervals. To rebuild the overlay, run the following:

```
[sms]# wwctl overlay build
```

After re-syncing to notify Warewulf of file modifications made on the *master* host, it should take approximately one minute for the changes to propagate.

OpenHPC includes a simple “hello-world” MPI application in the `/opt/ohpc/pub/examples` directory that can be used for this quick compilation and execution. OpenHPC also provides a companion job-launch utility named `prun` that is installed in concert with the pre-packaged MPI toolchains. This convenience script provides a mechanism to abstract job launch across different resource managers and MPI stacks such that a single launch command can be used for parallel job launch in a variety of OpenHPC environments. It also provides a centralizing mechanism for administrators to customize desired environment settings for their users.

7.1 Interactive execution

To use the newly created “test” account to compile and execute the application *interactively* through the resource manager, execute the following (note the use of **prun** for parallel job launch which summarizes the underlying native job launch mechanism being used):

```
# Switch to "test" user
[sms]# su - test

# Compile MPI "hello world" example
[test@sms ~]$ mpicc -O3 /opt/ohpc/pub/examples/mpi/hello.c

# Submit interactive job request and use prun to launch executable
[test@sms ~]$ salloc -n 8 -N 2

[test@c1 ~]$ prun ./a.out

[prun] Master compute host = c1
[prun] Resource manager = slurm
[prun] Launch cmd = mpiexec.hydra -bootstrap slurm ./a.out

Hello, world (8 procs total)
--> Process # 0 of 8 is alive. -> c1
--> Process # 4 of 8 is alive. -> c2
--> Process # 1 of 8 is alive. -> c1
--> Process # 5 of 8 is alive. -> c2
--> Process # 2 of 8 is alive. -> c1
--> Process # 6 of 8 is alive. -> c2
--> Process # 3 of 8 is alive. -> c1
--> Process # 7 of 8 is alive. -> c2
```

Tip

The following table provides approximate command equivalences between SLURM and OpenPBS:

Command	OpenPBS	SLURM
Submit <i>batch</i> job	qsub [job script]	sbatch [job script]
Request <i>interactive</i> shell	qsub -I /bin/bash	salloc
Delete job	qdel [job id]	scancel [job id]
Queue status	qstat -q	sinfo
Job status	qstat -f [job id]	scontrol show job [job id]
Node status	pbsnodes [node name]	scontrol show node [node id]

7.2 Batch execution

For batch execution, OpenHPC provides a simple job script for reference (also housed in the `/opt/ohpc/pub/examples` directory). This example script can be used as a starting point for submitting batch jobs to the resource manager and the example below illustrates use of the script to submit a batch job for execution using the same executable referenced in the previous interactive example.

```
# Copy example job script
[test@sms ~]$ cp /opt/ohpc/pub/examples/slurm/job.mpi .

# Examine contents (and edit to set desired job sizing characteristics)
[test@sms ~]$ cat job.mpi
#!/bin/bash

#SBATCH -J test                # Job name
#SBATCH -o job.%j.out          # Name of stdout output file (%j expands to %jobId)
#SBATCH -N 2                   # Total number of nodes requested
#SBATCH -n 16                  # Total number of mpi tasks #requested
#SBATCH -t 01:30:00           # Run time (hh:mm:ss) - 1.5 hours

# Launch MPI-based executable

prun ./a.out

# Submit job for batch execution
[test@sms ~]$ sbatch job.mpi
Submitted batch job 339
```

Tip

The use of the `%j` option in the example batch job script shown is a convenient way to track application output on an individual job basis. The `%j` token is replaced with the Slurm job allocation number once assigned (job #339 in this example).

Appendices

A Installation Template

This appendix highlights the availability of a companion installation script that is included with OpenHPC documentation. This script, when combined with local site inputs, can be used to implement a starting recipe for bare-metal system installation and configuration. This template script is used during validation efforts to test cluster installations and is provided as a convenience for administrators as a starting point for potential site customization.

Tip

Note that the template script provided is intended for use during initial installation and is not designed for repeated execution. If modifications are required after using the script initially, we recommend running the relevant subset of commands interactively.

The template script relies on the use of a simple text file to define local site variables that were outlined in §1.3. By default, the template installation script attempts to use local variable settings sourced from the `/opt/ohpc/pub/doc/recipes/vanilla/input.local` file, however, this choice can be overridden by the use of the `OHPC_INPUT_LOCAL` environment variable. The template install script is intended for execution on the SMS *master* host and is installed as part of the `docs-ohpc` package into `/opt/ohpc/pub/doc/recipes/vanilla/recipe.sh`. After enabling the OpenHPC repository and reviewing the guide for additional information on the intent of the commands, the general starting approach for using this template is as follows:

1. Install the `docs-ohpc` package

```
[sms]# apt -y install docs-ohpc
```

2. Copy the provided template input file to use as a starting point to define local site settings:

```
[sms]# cp /opt/ohpc/pub/doc/recipes/ubuntu24.04/input.local input.local
```

3. Update `input.local` with desired settings
4. Copy the template installation script which contains command-line instructions culled from this guide.

```
[sms]# cp -p /opt/ohpc/pub/doc/recipes/ubuntu24.04/x86_64/warewulf4/slurm/recipe.sh .
```

5. Review and edit `recipe.sh` to suite.
6. Use environment variable to define local input file and execute `recipe.sh` to perform a local installation.

```
[sms]# export OHPC_INPUT_LOCAL=./input.local  
[sms]# ./recipe.sh
```

B Upgrading OpenHPC Packages

As newer OpenHPC releases are made available, users are encouraged to upgrade their locally installed packages against the latest repository versions to obtain access to bug fixes and newer component versions. This can be accomplished with the underlying package manager as OpenHPC packaging maintains versioning state across releases.

1. (Optional) Ensure repo metadata is current on the SMS node and in the compute image.

```
[sms]# apt update
[sms]# wwctl image exec -build=false ubuntu-24.04 - apt update
```

2. Upgrade master (SMS) node

```
[sms]# apt -y install -only-upgrade "*-ohpc"
[sms]# apt -y install -only-upgrade "ohpc-base"
```

3. Upgrade packages in compute image

```
[sms]# wwctl image exec -build=false ubuntu-24.04 - apt -y install -only-upgrade "*-ohpc"
[sms]# wwctl image exec -build=false ubuntu-24.04 - apt -y install -only-upgrade "ohpc-base-compute"
```

4. Rebuild image(s)

```
[sms]# wwctl image build ubuntu-24.04
```

In the case where packages were upgraded within the compute image, you will need to reboot the compute nodes when convenient to enable the changes.

C Integration Test Suite

This appendix details the installation and basic use of the integration test suite used to support OpenHPC releases. This suite is not intended to replace the validation performed by component development teams, but is instead, devised to confirm component builds are functional and interoperable within the modular OpenHPC environment.

The Ubuntu 24.04 port currently validates the packaged stack with the runtime smoke tests documented in the release validation plan. When a Debian-packaged `test-suite-ohpc` is available, it should be installed and run from this appendix following the same workflow as the upstream recipes.

D Customizing system configuration

Locally installed applications can easily be integrated in to OpenHPC systems by following the Lmod convention laid out by the provided packages. Two sample module files are included in the `examples-ohpc` package—one representing an application with no compiler or MPI runtime dependencies, and one dependent on OpenMPI and the GNU toolchain. Simply copy these files to the prescribed locations, and the lmod application should pick them up automatically.

```
[sms]# mkdir /opt/ohpc/pub/modulefiles/example1
[sms]# cp /opt/ohpc/pub/examples/example.modulefile \
/opt/ohpc/pub/modulefiles/example1/1.0
[sms]# mkdir /opt/ohpc/pub/moduledeps/gnu7-openmpi3/example2
[sms]# cp /opt/ohpc/pub/examples/example-mpi-dependent.modulefile \
/opt/ohpc/pub/moduledeps/gnu7-openmpi3/example2/1.0
[sms]# module avail
```

----- /opt/ohpc/pub/moduledeps/gnu7-openmpi3 -----				
adios/1.12.0	imb/2018.0	netcdf-fortran/4.4.4	ptscotch/6.0.4	sionlib/1.7.1
boost/1.65.1	mpi4py/2.0.0	netcdf/4.4.1.1	scalapack/2.0.2	slepc/3.7.4
example2/1.0	mpiP/3.4.1	petsc/3.7.6	scalasca/2.3.1	superlu_dist/4.2
fftw/3.3.6	mumps/5.1.1	phdf5/1.10.1	scipy/0.19.1	tau/2.26.1
hypre/2.11.2	netcdf-cxx/4.3.0	pnetcdf/1.8.1	scorep/3.1	trilinos/12.10.1

----- /opt/ohpc/pub/moduledeps/gnu7 -----				
R/3.4.2	metis/5.1.0	ocr/1.0.1	pdtoolkit/3.24	superlu/5.2.1
gsl/2.4	mpich/3.2	openblas/0.2.20	plasma/2.8.0	
hdf5/1.10.1	numpy/1.13.1	openmpi3/3.0.0 (L)	scotch/6.0.4	

----- /opt/ohpc/admin/modulefiles -----				
spack/0.10.0				

----- /opt/ohpc/pub/modulefiles -----				
EasyBuild/3.4.1	cmake/3.9.2	hwloc/1.11.8	pmix/1.2.3	valgrind/3.13.0
autotools (L)	example1/1.0 (L)	llvm5/5.0.0	prun/1.2 (L)	
clustershell/1.8	gnu7/7.2.0 (L)	ohpc (L)	singularity/2.4	

Where:
L: Module is loaded

Use "module spider" to find all possible modules.
Use "module keyword key1 key2 ..." to search for all possible modules matching any of the "keys".

Users of OpenHPC may find it desirable to rebuild one of the supplied packages to apply build customizations or satisfy local requirements. On Ubuntu, this can be accomplished by enabling source repositories, installing build dependencies, and rebuilding the Debian source package. A brief example using the FFTW library is highlighted below.

```
# Enable source entries for the OpenHPC repository
[test@sms ~]$ sudo sed -i 's/~Types: deb$/Types: deb deb-src/' \
/etc/apt/sources.list.d/versatushpc-openhpc.sources
[test@sms ~]$ sudo apt update

# Install Debian package build tools
[test@sms ~]$ sudo apt -y install build-essential debhelper devscripts fakeroot

# Install FFTW's build dependencies and download source
[test@sms ~]$ sudo apt -y build-dep fftw-gnu15-openmpi5-ohpc
[test@sms ~]$ apt source fftw-gnu15-openmpi5-ohpc

# Modify packaging as desired
```

```
[test@sms ~]$ cd fftw-gnu15-openmpi5-ohpc-*
[test@sms ~/fftw-gnu15-openmpi5-ohpc-*]$ perl -pi -e "s/enable-static=no/enable-static=yes/" debian/rules

# Increment Debian revision and rebuild binary packages
[test@sms ~/fftw-gnu15-openmpi5-ohpc-*]$ dch --local +static "Enable static FFTW build"
[test@sms ~/fftw-gnu15-openmpi5-ohpc-*]$ dpkg-buildpackage -us -uc -b

# Install the new package
[test@sms ~/fftw-gnu15-openmpi5-ohpc-*]$ sudo apt -y install \
    ../fftw-gnu15-openmpi5-ohpc-*_x86_64.deb
```

E Package Manifest

This appendix provides a summary of available meta-package groupings and all of the individual RPM packages that are available as part of this OpenHPC release. The meta-packages provide a mechanism to group related collections of RPMs by functionality and provide a convenience mechanism for installation. A list of the available meta-packages and a brief description is presented in [Table 2](#).

Table 2: **Available OpenHPC Meta-packages**

Group Name	Description
ohpc-autotools	Collection of GNU autotools packages.
ohpc-base	Collection of base packages.
ohpc-base-compute	Collection of compute node base packages.
ohpc-gnu15-io-libs	Collection of IO library builds for use with GNU compiler toolchain.
ohpc-gnu15-mpich-io-libs	Collection of IO library builds for use with GNU compiler toolchain and the MPICH runtime.
ohpc-gnu15-mpich-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the MPICH runtime.
ohpc-gnu15-mpich-perf-tools	Collection of performance tool builds for use with GNU compiler toolchain and the MPICH runtime.
ohpc-gnu15-mvapich2-io-libs	Collection of IO library builds for use with GNU compiler toolchain and the MVAPICH2 runtime.
ohpc-gnu15-mvapich2-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the MVAPICH2 runtime.
ohpc-gnu15-mvapich2-perf-tools	Collection of performance tool builds for use with GNU compiler toolchain and the MVAPICH2 runtime.
ohpc-gnu15-openmpi5-io-libs	Collection of IO library builds for use with GNU compiler toolchain and the OpenMPI runtime.
ohpc-gnu15-openmpi5-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain and the OpenMPI runtime.
ohpc-gnu15-openmpi5-perf-tools	Collection of performance tool builds for use with GNU compiler toolchain and the OpenMPI runtime.
ohpc-gnu15-parallel-libs	Collection of parallel library builds for use with GNU compiler toolchain.
ohpc-gnu15-perf-tools	Collection of performance tool builds for use with GNU compiler toolchain.
ohpc-gnu15-python-libs	Collection of python related library builds for use with GNU compiler toolchain.
ohpc-gnu15-python3-libs	Collection of python3 related library builds for use with GNU compiler toolchain.
ohpc-gnu15-runtimes	Collection of runtimes for use with GNU compiler toolchain.
ohpc-gnu15-serial-libs	Collection of serial library builds for use with GNU compiler toolchain.
ohpc-intel-impi-io-libs	Collection of IO library builds for use with Intel(R) oneAPI Toolkit and Intel(R) MPI runtime.
ohpc-intel-impi-parallel-libs	Collection of parallel library builds for use with Intel(R) oneAPI Toolkit and the Intel(R) MPI Library.
ohpc-intel-impi-perf-tools	Collection of performance tool builds for use with Intel(R) oneAPI Toolkit compiler toolchain and the Intel(R) MPI runtime.
ohpc-intel-io-libs	Collection of IO library builds for use with Intel(R) oneAPI Toolkit.
ohpc-intel-mpich-io-libs	Collection of IO library builds for use with Intel(R) oneAPI Toolkit and MPICH runtime.
ohpc-intel-mpich-parallel-libs	Collection of parallel library builds for use with Intel(R) oneAPI Toolkit and the MPICH runtime.
ohpc-intel-mpich-perf-tools	Collection of performance tool builds for use with Intel(R) oneAPI Toolkit compiler toolchain and the MPICH runtime.
ohpc-intel-mvapich2-io-libs	Collection of IO library builds for use with Intel(R) oneAPI Toolkit and MVAPICH2 runtime.

Table 2 (cont): **Available OpenHPC Meta-packages**

Group Name	Description
ohpc-intel-mvapich2-parallel-libs	Collection of parallel library builds for use with Intel(R) oneAPI Toolkit and the MVAPICH2 runtime.
ohpc-intel-mvapich2-perf-tools	Collection of performance tool builds for use with Intel(R) oneAPI Toolkit compiler toolchain and the MVAPICH2 runtime.
ohpc-intel-openmpi5-io-libs	Collection of IO library builds for use with Intel(R) oneAPI Toolkit and OpenMPI runtime.
ohpc-intel-openmpi5-parallel-libs	Collection of parallel library builds for use with Intel(R) oneAPI Toolkit and the OpenMPI runtime.
ohpc-intel-openmpi5-perf-tools	Collection of performance tool builds for use with Intel(R) oneAPI Toolkit compiler toolchain and the OpenMPI runtime.
ohpc-intel-perf-tools	Collection of performance tool builds for use with Intel(R) oneAPI Toolkit.
ohpc-intel-python3-libs	Collection of python3 related library builds for use with Intel(R) oneAPI Toolkit.
ohpc-intel-serial-libs	Collection of serial library builds for use with Intel(R) oneAPI Toolkit.
ohpc-slurm-client	Collection of client packages for SLURM.
ohpc-slurm-server	Collection of server packages for SLURM.
ohpc-warewulf	Collection of base packages for Warewulf provisioning.

What follows next in this Appendix is a series of tables that summarize the underlying RPM packages available in this OpenHPC release. These packages are organized by groupings based on their general functionality and each table provides information for the specific RPM name, version, brief summary, and the web URL where additional information can be obtained for the component. Note that many of the 3rd party community libraries that are pre-packaged with OpenHPC are built using multiple compiler and MPI families. In these cases, the RPM package name includes delimiters identifying the development environment for which each package build is targeted. Additional information on the OpenHPC package naming scheme is presented in §4.6. The relevant package groupings and associated Table references are as follows:

- Administrative tools (Table 3)
- Provisioning (Table 4)
- Resource management (Table 5)
- Compiler families (Table 6)
- MPI families (Table 7)
- Development tools (Table 8)
- Performance analysis tools (Table 9)
- Lustre (Table ??)
- IO Libraries (Table 10)
- Runtimes (Table 11)
- Serial/Threaded Libraries (Table 12)
- Parallel Libraries (Table 13)

Table 3: **Administrative Tools**

RPM Package Name	Version	Info/URL
autoconf-ohpc	2.71	GNU autoconf for OpenHPC. https://www.gnu.org/software/autoconf
automake-ohpc	1.16.5	GNU automake for OpenHPC. https://www.gnu.org/software/automake
conman-ohpc	0.3.1	ConMan: The Console Manager. http://dun.github.io/conman
docs-ohpc	4.0.0	OpenHPC documentation. https://github.com/openhpc/ohpc
examples-ohpc	2.0	Example source code and templates for OpenHPC. https://github.com/openhpc/ohpc
genders-compat-ohpc	1.32	Genders compatibility API. https://github.com/chaos/genders
genders-ohpc	1.32	Static cluster configuration database. https://github.com/chaos/genders
hpc-workspace-ohpc	1.5.0	Temporary workspace management for HPC. https://github.com/holgerBerger/hpc-workspace
libtool-ohpc	2.4.6	GNU libtool for OpenHPC. https://www.gnu.org/software/libtool
lmod-defaults-gnu15-openmpi5-ohpc	2.0	OpenHPC default login environment (gnu15/openmpi5). https://github.com/openhpc/ohpc
lmod-ohpc	9.0.5	Lmod based Modules - OpenHPC. https://github.com/TACC/Lmod
losf-ohpc	0.56.0	A Linux operating system framework for managing HPC clusters. https://github.com/hpcsi/losf
mdtoc-ohpc	1.4.0	Markdown table-of-contents generator for OpenHPC. https://github.com/kubernetes-sigs/mdtoc
mrsh-ohpc	2.12	Remote shell program that uses munge authentication. https://github.com/chaos/mrsh
nhc-ohpc	1.4.3	LBNL Node Health Check. https://github.com/mej/nhc
ohpc-release	4.1	OpenHPC release files and repository configuration. https://github.com/VersatusHPC/ohpc
pdsh-mod-genders-ohpc	2.36	Pdsh genders module. https://github.com/chaos/pdsh
pdsh-mod-slurm-ohpc	2.36	Pdsh Slurm module. https://github.com/chaos/pdsh
pdsh-ohpc	2.36	Parallel remote shell program. https://github.com/chaos/pdsh
prun-ohpc	2.2	Convenience utility for parallel job launch. https://github.com/openhpc/ohpc

Table 4: **Provisioning**

RPM Package Name	Version	Info/URL
warewulf-dracut-ohpc	4.6.5	Dracut module for loading a Warewulf image. https://github.com/warewulf/warewulf
warewulf-ohpc	4.6.5	A provisioning system for large clusters of bare metal and/or virtual systems. https://github.com/warewulf/warewulf

Table 5: **Resource Management**

RPM Package Name	Version	Info/URL
magpie-ohpc	3.2	Scripts for running Big Data software in HPC environments. https://github.com/LLNL/magpie
munge-devel-ohpc	0.5.13	MUNGE development files. https://github.com/dun/munge
munge-libs-ohpc	0.5.13	MUNGE shared libraries. https://github.com/dun/munge
openpbs-client-ohpc	23.06.06	OpenPBS client. http://www.openpbs.org
openpbs-devel-ohpc	23.06.06	OpenPBS development files. http://www.openpbs.org
openpbs-execution-ohpc	23.06.06	OpenPBS execution host. http://www.openpbs.org
openpbs-server-ohpc	23.06.06	OpenPBS server. http://www.openpbs.org
pmix-ohpc	4.2.9	Process Management Interface for Exascale (PMIx). https://github.com/openpmix/openpmix
slurm-contribs-ohpc	25.05.6	Slurm contributed scripts. https://slurm.schedmd.com
slurm-devel-ohpc	25.05.6	Slurm development files. https://slurm.schedmd.com
slurm-example-configs-ohpc	25.05.6	Slurm example configuration files. https://slurm.schedmd.com
slurm-libpmi-ohpc	25.05.6	Slurm PMI libraries. https://slurm.schedmd.com
slurm-ohpc	25.05.6	Slurm Workload Manager. https://slurm.schedmd.com
slurm-openlava-ohpc	25.05.6	Slurm OpenLava/LSF compatibility wrappers. https://slurm.schedmd.com
slurm-pam-slurm-ohpc	25.05.6	Slurm PAM module. https://slurm.schedmd.com
slurm-perlapi-ohpc	25.05.6	Slurm Perl API. https://slurm.schedmd.com
slurm-sackd-ohpc	25.05.6	Slurm auth and cred kiosk daemon. https://slurm.schedmd.com
slurm-slurmctld-ohpc	25.05.6	Slurm controller daemon. https://slurm.schedmd.com
slurm-slurmd-ohpc	25.05.6	Slurm compute node daemon. https://slurm.schedmd.com
slurm-slurmdbd-ohpc	25.05.6	Slurm database daemon. https://slurm.schedmd.com
slurm-sview-ohpc	25.05.6	Slurm graphical interface. https://slurm.schedmd.com
slurm-torque-ohpc	25.05.6	Slurm Torque/PBS compatibility wrappers. https://slurm.schedmd.com

Table 6: **Compiler Families**

RPM Package Name	Version	Info/URL
gnu15-compilers-ohpc	15.2.0	GNU Compiler Collection 15.2.0. https://gcc.gnu.org
intel-compilers-devel-ohpc	2025.0	OpenHPC compatibility package for Intel(R) oneAPI HPC Toolkit. https://github.com/openhpc/ohpc
intel-oneapi-toolkit-release-ohpc	2025.0	Intel(R) oneAPI HPC Toolkit Repository Setup. https://github.com/openhpc/ohpc
llvm10-compilers-ohpc	10.0.1	LLVM compiler infrastructure for OpenHPC. https://llvm.org

Table 7: **MPI Families / Communication Libraries**

RPM Package Name	Version	Info/URL
intel-mpi-devel-ohpc	2025.0	OpenHPC compatibility package for Intel(R) oneAPI MPI Library. https://github.com/openhpc/ohpc
mpich-gnu15-ohpc mpich-intel-ohpc mpich-ucx-gnu15-ohpc	5.0.0	MPICH MPI implementation (gnu15, OpenHPC). https://www.mpich.org
mvapich2-gnu15-ohpc mvapich2-intel-ohpc	2.3.7	MVAPICH2 MPI implementation (gnu15, OpenHPC). http://mvapich.cse.ohio-state.edu
openmpi5-gnu15-ohpc openmpi5-intel-ohpc	5.0.10	Open MPI 5 (gnu15, OpenHPC). https://www.open-mpi.org
ucx-cma-ohpc	1.20.0	UCX CMA transport support. https://www.openucx.org
ucx-devel-ohpc	1.20.0	UCX development files. https://www.openucx.org
ucx-ib-ohpc	1.20.0	UCX InfiniBand transport support. https://www.openucx.org
ucx-ohpc	1.20.0	Unified Communication X (UCX) framework. https://www.openucx.org
ucx-rdmacm-ohpc	1.20.0	UCX RDMA CM transport support. https://www.openucx.org

Table 8: **Development Tools**

RPM Package Name	Version	Info/URL
cmake-ohpc	4.3.0	Cross-platform family of build tools. https://cmake.org
cuda-devel-ohpc	25.9	OpenHPC compatibility package for CUDA. https://github.com/openhpc/ohpc
cuda-repo-ohpc	25.9	CUDA toolkit repository configuration. https://github.com/openhpc/ohpc
easybuild-ohpc	5.2.1	Software build and installation framework. https://easybuilders.github.io/easybuild
hwloc-ohpc	2.13.0	Portable Hardware Locality (hwloc). https://www.open-mpi.org/projects/hwloc
python3-mpi4py-gnu15-impi-ohpc python3-mpi4py-gnu15-mpich-ohpc python3-mpi4py-gnu15-mvapich2-ohpc python3-mpi4py-gnu15-openmpi5-ohpc python3-mpi4py-intel-impi-ohpc python3-mpi4py-intel-mpich-ohpc python3-mpi4py-intel-mvapich2-ohpc python3-mpi4py-intel-openmpi5-ohpc	4.1.1	Python bindings for the Message Passing Interface (gnu15 + impi, OpenHPC). https://github.com/mpi4py/mpi4py
python3-numpy-gnu15-ohpc python3-numpy-intel-ohpc	2.4.3	NumPy array processing for numbers, strings, records and objects (gnu15, OpenHPC).
python3-scipy-gnu15-openmpi5-ohpc	1.14.1	Scientific Tools for Python (gnu15 + openmpi5, OpenHPC).
spack-ohpc	1.1.1	OpenHPCware package management. https://github.com/conda-forge/spack
valgrind-ohpc	3.26.0	An instrumentation framework for building dynamic analysis tools. https://valgrind.org

Table 9: Performance Analysis Tools

RPM Package Name	Version	Info/URL
dimemas-gnu15-impi-ohpc dimemas-gnu15-mpich-ohpc dimemas-gnu15-mvapich2-ohpc dimemas-gnu15-openmpi5-ohpc dimemas-intel-impi-ohpc dimemas-intel-mpich-ohpc dimemas-intel-mvapich2-ohpc dimemas-intel-openmpi5-ohpc	5.5.0	Dimemas network simulator tool (gnu15 + impi, OpenHPC). https://tools.bsc.es
extrae-gnu15-impi-ohpc extrae-gnu15-mpich-ohpc extrae-gnu15-mvapich2-ohpc extrae-gnu15-openmpi5-ohpc extrae-intel-impi-ohpc extrae-intel-mpich-ohpc extrae-intel-mvapich2-ohpc extrae-intel-openmpi5-ohpc	5.0.3	Extrae instrumentation framework for parallel applications (gnu15 + impi, OpenHPC). https://tools.bsc.es
imb-gnu15-impi-ohpc imb-gnu15-mpich-ohpc imb-gnu15-mvapich2-ohpc imb-gnu15-openmpi5-ohpc imb-intel-impi-ohpc imb-intel-mpich-ohpc imb-intel-mvapich2-ohpc imb-intel-openmpi5-ohpc	2021.10	Intel MPI Benchmarks (gnu15 + impi, OpenHPC). https://github.com/intel/mpi-benchmarks
likwid-gnu15-ohpc likwid-intel-ohpc	5.5.1	Performance monitoring and benchmarking suite (gnu15, OpenHPC). https://github.com/RRZE-HPC/likwid
msr-safe-ohpc	1.7.0	Safer MSR access for OpenHPC. https://github.com/LLNL/msr-safe
omb-gnu15-impi-ohpc omb-gnu15-mpich-ohpc omb-gnu15-mvapich2-ohpc omb-gnu15-openmpi5-ohpc omb-intel-impi-ohpc omb-intel-mpich-ohpc omb-intel-mvapich2-ohpc omb-intel-openmpi5-ohpc	7.5.2	OSU Micro-Benchmarks (gnu15 + impi, OpenHPC). https://mvapich.cse.ohio-state.edu/benchmarks
papi-ohpc	7.2.0	Performance Application Programming Interface. https://icl.utk.edu/papi
paraver-ohpc	4.12.0	Paraver parallel program visualization and analysis tool. https://tools.bsc.es

Table 9 (cont): Performance Analysis Tools

RPM Package Name	Version	Info/URL
pdtoolkit-gnu15-ohpc pdtoolkit-intel-ohpc	3.25.1	Program Database Toolkit (gnu15, OpenHPC). http://www.cs.uoregon.edu/Research/pdt
scalasca-gnu15-impi-ohpc scalasca-gnu15-mpich-ohpc scalasca-gnu15-mvapich2-ohpc scalasca-gnu15-openmpi5-ohpc scalasca-intel-impi-ohpc scalasca-intel-mpich-ohpc scalasca-intel-mvapich2-ohpc scalasca-intel-openmpi5-ohpc	2.6.2	Toolset for performance analysis of large-scale parallel applications (gnu15 + impi, OpenHPC). http://www.scalasca.org
scorep-gnu15-impi-ohpc scorep-gnu15-mpich-ohpc scorep-gnu15-mvapich2-ohpc scorep-gnu15-openmpi5-ohpc scorep-intel-impi-ohpc scorep-intel-mpich-ohpc scorep-intel-mvapich2-ohpc scorep-intel-openmpi5-ohpc	9.4	Scalable Performance Measurement Infrastructure for Parallel Codes (gnu15 + impi, OpenHPC). http://www.vi-hps.org/projects/score-p
tau-gnu15-impi-ohpc tau-gnu15-mpich-ohpc tau-gnu15-mvapich2-ohpc tau-gnu15-openmpi5-ohpc tau-intel-impi-ohpc tau-intel-mpich-ohpc tau-intel-mvapich2-ohpc tau-intel-openmpi5-ohpc	2.35.1	Tuning and Analysis Utilities profiling package (gnu15 + impi, OpenHPC). http://www.cs.uoregon.edu/research/tau/home.php

Table 10: IO Libraries

RPM Package Name	Version	Info/URL
adios2-gnu15-impi-ohpc adios2-gnu15-mpich-ohpc adios2-gnu15-mvapich2-ohpc adios2-gnu15-openmpi5-ohpc adios2-intel-impi-ohpc adios2-intel-mpich-ohpc adios2-intel-mvapich2-ohpc adios2-intel-openmpi5-ohpc	2.11.0	The Adaptable IO System v2 (gnu15 + impi, OpenHPC). https://adios2.readthedocs.io/en/latest/index.html
cubew-gnu15-ohpc cubew-intel-ohpc	4.9.1	CUBE high performance write library (gnu15, OpenHPC). https://www.scalasca.org/scalasca/software/cube-4.x
hdf5-gnu15-ohpc	2.1.0	A general purpose library and file format for storing scientific data (gnu15, OpenHPC). http://www.hdfgroup.org/HDF5
netcdf-cxx-gnu15-impi-ohpc netcdf-cxx-gnu15-mpich-ohpc netcdf-cxx-gnu15-mvapich2-ohpc netcdf-cxx-gnu15-openmpi5-ohpc netcdf-cxx-intel-impi-ohpc netcdf-cxx-intel-mpich-ohpc netcdf-cxx-intel-mvapich2-ohpc netcdf-cxx-intel-openmpi5-ohpc	4.3.1	C++ Libraries for the Unidata network Common Data Form (gnu15 + impi, OpenHPC). https://www.unidata.ucar.edu/software/netcdf
netcdf-fortran-gnu15-impi-ohpc netcdf-fortran-gnu15-mpich-ohpc netcdf-fortran-gnu15-mvapich2-ohpc netcdf-fortran-gnu15-openmpi5-ohpc netcdf-fortran-intel-impi-ohpc netcdf-fortran-intel-mpich-ohpc netcdf-fortran-intel-mvapich2-ohpc netcdf-fortran-intel-openmpi5-ohpc	4.6.2	Fortran Libraries for the Unidata network Common Data Form (gnu15 + impi, OpenHPC). https://www.unidata.ucar.edu/software/netcdf
netcdf-gnu15-impi-ohpc netcdf-gnu15-mpich-ohpc netcdf-gnu15-mvapich2-ohpc	4.10.0	C Libraries for the Unidata network Common Data Form (gnu15 + impi, OpenHPC). https://www.unidata.ucar.edu/software/netcdf
netcdf-gnu15-ohpc netcdf-gnu15-openmpi5-ohpc netcdf-intel-impi-ohpc netcdf-intel-mpich-ohpc netcdf-intel-mvapich2-ohpc netcdf-intel-openmpi5-ohpc	4.10.0	C Libraries for the Unidata network Common Data Form (gnu15, OpenHPC). https://www.unidata.ucar.edu/software/netcdf
otf2-gnu15-impi-ohpc otf2-gnu15-mpich-ohpc otf2-gnu15-mvapich2-ohpc otf2-gnu15-openmpi5-ohpc otf2-intel-impi-ohpc otf2-intel-mpich-ohpc otf2-intel-mvapich2-ohpc otf2-intel-openmpi5-ohpc	3.1.1	Open Trace Format 2 library (gnu15 + impi, OpenHPC). https://www.vi-hps.org/projects/score-p
phdf5-gnu15-impi-ohpc phdf5-gnu15-mpich-ohpc phdf5-gnu15-mvapich2-ohpc phdf5-gnu15-openmpi5-ohpc phdf5-intel-impi-ohpc phdf5-intel-mpich-ohpc phdf5-intel-mvapich2-ohpc phdf5-intel-openmpi5-ohpc	2.1.0	A general purpose library and file format for storing scientific data - parallel (gnu15 + impi, OpenHPC). http://www.hdfgroup.org/HDF5

Table 10 (cont): **IO Libraries**

RPM Package Name	Version	Info/URL
pnetcdf-gnu15-impi-ohpc pnetcdf-gnu15-mpich-ohpc pnetcdf-gnu15-mvapich2-ohpc pnetcdf-gnu15-openmpi5-ohpc pnetcdf-intel-impi-ohpc pnetcdf-intel-mpich-ohpc pnetcdf-intel-mvapich2-ohpc pnetcdf-intel-openmpi5-ohpc	1.14.1	A Parallel I/O Library for NetCDF File Access (gnu15 + impi, OpenHPC). https://parallel-netcdf.github.io
sionlib-gnu15-impi-ohpc sionlib-gnu15-mpich-ohpc sionlib-gnu15-mvapich2-ohpc sionlib-gnu15-openmpi5-ohpc sionlib-intel-mpi-ohpc sionlib-intel-mpich-ohpc sionlib-intel-mvapich2-ohpc sionlib-intel-openmpi5-ohpc	1.7.7	Scalable I/O Library for Parallel Access to Task-Local Files (gnu15 + impi, OpenHPC). https://apps.fz-juelich.de/jsc/sionlib/docu/index.html

Table 11: **Runtimes**

RPM Package Name	Version	Info/URL
charliecloud-ohpc	0.43	Lightweight user-defined software stacks for HPC. https://charliecloud.io

Table 12: **Serial/Threaded Libraries**

RPM Package Name	Version	Info/URL
cubelib-gnu15-ohpc cubelib-intel-ohpc	4.9.1	CUBE high performance analysis library (gnu15, OpenHPC). https://www.scalasca.org/scalasca/software/cube-4.x
gotcha-gnu15-ohpc gotcha-intel-ohpc	1.0.8	A library for wrapping function calls in shared libraries (gnu15, OpenHPC). https://github.com/LLNL/GOTCHA
gsl-gnu15-ohpc gsl-intel-ohpc	2.8	GNU Scientific Library (gnu15, OpenHPC). http://www.gnu.org/software/gsl
metis-gnu15-ohpc metis-intel-ohpc	5.1.0	Serial Graph Partitioning and Fill-reducing Matrix Ordering (gnu15, OpenHPC).
opari2-gnu15-ohpc opari2-intel-ohpc	2.0.9	An OpenMP runtime performance measurement instrumenter (gnu15, OpenHPC). https://www.vi-hps.org/projects/score-p
openblas-gnu15-ohpc openblas-intel-ohpc	0.3.32	An optimized BLAS library based on GotoBLAS2 (gnu15, OpenHPC). http://www.openblas.net
plasma-gnu15-ohpc plasma-intel-ohpc	25.5.27	Parallel Linear Algebra Software for Multicore Architectures (gnu15, OpenHPC). https://github.com/icl-utk-edu/plasma
ptscotch-gnu15-impi-ohpc ptscotch-gnu15-mpich-ohpc ptscotch-gnu15-mvapich2-ohpc ptscotch-gnu15-openmpi5-ohpc ptscotch-intel-impi-ohpc ptscotch-intel-mpich-ohpc ptscotch-intel-mvapich2-ohpc ptscotch-intel-openmpi5-ohpc	7.0.10	Graph, mesh and hypergraph partitioning library - MPI variant (gnu15 + impi, OpenHPC). https://gitlab.inria.fr/scotch/scotch
r-gnu15-ohpc	4.5.3	R language for statistical computing (gnu15, OpenHPC).
scotch-gnu15-ohpc scotch-intel-ohpc	7.0.10	Graph, mesh and hypergraph partitioning library (gnu15, OpenHPC). https://gitlab.inria.fr/scotch/scotch
superlu-gnu15-ohpc superlu-intel-ohpc	7.0.1	A general purpose library for the direct solution of linear equations (gnu15, OpenHPC). http://crd.lbl.gov/~xiaoye/SuperLU

Table 13: **Parallel Libraries**

RPM Package Name	Version	Info/URL
boost-gnu15-impi-ohpc boost-gnu15-mpich-ohpc boost-gnu15-mvapich2-ohpc boost-gnu15-openmpi5-ohpc boost-intel-impi-ohpc boost-intel-mpich-ohpc boost-intel-mvapich2-ohpc boost-intel-openmpi5-ohpc	1.90.0	Free peer-reviewed portable C++ source libraries (gnu15 + impi, OpenHPC). http://www.boost.org
fftw-gnu15-impi-ohpc fftw-gnu15-mpich-ohpc fftw-gnu15-mvapich2-ohpc fftw-gnu15-openmpi5-ohpc fftw-intel-impi-ohpc fftw-intel-mpich-ohpc fftw-intel-mvapich2-ohpc fftw-intel-openmpi5-ohpc	3.3.10	A Fast Fourier Transform library (gnu15 + impi, OpenHPC). http://www.fftw.org
hypre-gnu15-impi-ohpc hypre-gnu15-mpich-ohpc hypre-gnu15-mvapich2-ohpc hypre-gnu15-openmpi5-ohpc hypre-intel-impi-ohpc hypre-intel-mpich-ohpc hypre-intel-mvapich2-ohpc hypre-intel-openmpi5-ohpc	3.1.0	Scalable algorithms for solving linear systems of equations (gnu15 + impi, OpenHPC). http://www.llnl.gov/casc/hypre
mfem-gnu15-impi-ohpc mfem-gnu15-mpich-ohpc mfem-gnu15-mvapich2-ohpc mfem-intel-impi-ohpc mfem-intel-mpich-ohpc mfem-intel-mvapich2-ohpc mfem-intel-openmpi5-ohpc	4.9	Lightweight, general, scalable C++ library for finite element methods (gnu15 + impi, OpenHPC). http://mfem.org

Table 13 (cont): **Parallel Libraries**

RPM Package Name	Version	Info/URL
mumps-gnu15-impi-ohpc mumps-gnu15-mpich-ohpc mumps-gnu15-mvapich2-ohpc mumps-gnu15-openmpi5-ohpc mumps-intel-impi-ohpc mumps-intel-mpich-ohpc mumps-intel-mvapich2-ohpc mumps-intel-openmpi5-ohpc	5.8.2	A Multifrontal Massively Parallel Sparse direct Solver (gnu15 + impi, OpenHPC). https://mumps-solver.org
opencoarrays-gnu15-mpich-ohpc opencoarrays-gnu15-mvapich2-ohpc opencoarrays-gnu15-openmpi5-ohpc	2.10.3	ABI to leverage the parallel programming features of Fortran 2018 (gnu15 + mpich, OpenHPC). http://www.opencoarrays.org
petsc-gnu15-impi-ohpc petsc-gnu15-mpich-ohpc petsc-gnu15-mvapich2-ohpc petsc-intel-impi-ohpc petsc-intel-mpich-ohpc petsc-intel-mvapich2-ohpc petsc-intel-openmpi5-ohpc	3.24.4	Portable Extensible Toolkit for Scientific Computation (gnu15 + impi, OpenHPC). http://www.mcs.anl.gov/petsc
scalapack-gnu15-impi-ohpc scalapack-gnu15-mpich-ohpc scalapack-gnu15-mvapich2-ohpc scalapack-gnu15-openmpi5-ohpc scalapack-intel-impi-ohpc scalapack-intel-mpich-ohpc scalapack-intel-mvapich2-ohpc scalapack-intel-openmpi5-ohpc	2.2.2	A subset of LAPACK routines redesigned for heterogeneous computing (gnu15 + impi, OpenHPC). https://netlib.org/scalapack
slepc-gnu15-impi-ohpc slepc-gnu15-mpich-ohpc slepc-gnu15-mvapich2-ohpc slepc-intel-impi-ohpc slepc-intel-mpich-ohpc slepc-intel-mvapich2-ohpc slepc-intel-openmpi5-ohpc	3.24.3	Scalable Library for Eigenvalue Problem Computations (gnu15 + impi, OpenHPC). http://slepc.upv.es
superlu-dist-gnu15-impi-ohpc superlu-dist-gnu15-mpich-ohpc superlu-dist-gnu15-mvapich2-ohpc superlu-dist-gnu15-openmpi5-ohpc superlu-dist-intel-impi-ohpc superlu-dist-intel-mpich-ohpc superlu-dist-intel-mvapich2-ohpc superlu-dist-intel-openmpi5-ohpc	9.2.1	A general purpose library for the direct solution of linear equations (gnu15 + impi, OpenHPC). https://portal.nersc.gov/project/sparse/superlu
trilinos-gnu15-impi-ohpc trilinos-gnu15-mpich-ohpc trilinos-gnu15-mvapich2-ohpc trilinos-intel-impi-ohpc trilinos-intel-mpich-ohpc trilinos-intel-mvapich2-ohpc trilinos-intel-openmpi5-ohpc	17.0.0	A collection of libraries of numerical algorithms (gnu15 + impi, OpenHPC). Rev: 184207b66a8a684c7d2099670e939ce9d833fefb https://trilinos.org

Table 13 (cont): **Parallel Libraries**

RPM Package Name	Version	Info/URL
superlu_dist-gnu15-impi-ohpc superlu_dist-gnu15-mpich-ohpc superlu_dist-gnu15-mvapich2-ohpc superlu_dist-gnu15-openmpi5-ohpc superlu_dist-intel-impi-ohpc superlu_dist-intel-mpich-ohpc superlu_dist-intel-mvapich2-ohpc superlu_dist-intel-openmpi5-ohpc	6.4.0	A general purpose library for the direct solution of linear equations. https://portal.nersc.gov/project/sparse/superlu
trilinos-gnu15-impi-ohpc trilinos-gnu15-mpich-ohpc trilinos-gnu15-mvapich2-ohpc trilinos-gnu15-openmpi5-ohpc trilinos-intel-impi-ohpc trilinos-intel-mpich-ohpc trilinos-intel-mvapich2-ohpc trilinos-intel-openmpi5-ohpc	16.1.0	A collection of libraries of numerical algorithms. https://trilinos.org

F Package Signatures

All of the repository metadata provided via the Ubuntu OpenHPC repository is signed with a GPG signature. By default, APT verifies this signature during metadata refresh to ensure that packages have not been altered. The repository key used by the Ubuntu port is installed at `/usr/share/keyrings/versatushpc.gpg`.

```
[sms]# gpg --show-keys --fingerprint /usr/share/keyrings/versatushpc.gpg
```

The following command can be used to verify repository metadata once it has been downloaded locally by confirming that APT accepts the configured source.

```
[sms]# apt update
[sms]# apt-cache policy docs-ohpc
```