

2024-07-17

OpenHPC: Beyond the Install Guide

OpenHPC: Beyond the Install Guide
for PEARC24

Sharon Colson Jim Moroney Mike Renfro

Tennessee Tech University

2024-07-22

2024-07-17

OpenHPC: Beyond the Install Guide

- └ Introduction

- └ Acknowledgments and shameless plugs

- └ Acknowledgments and shameless plugs

[Acknowledgments and shameless plugs](#)

[OpenHPC](#) especially Tim Middelkoop (Internet2) and Chris Simmons (Massachusetts Green High Performance Computing Center). They have a BOF at 1:30 Wednesday. You should go to it.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └ Introduction

- └ Acknowledgments and shameless plugs

- └ Acknowledgments and shameless plugs

[Acknowledgments and shameless plugs](#)

[OpenHPC](#) especially Tim Middelkoop (Internet2) and Chris Simmons (Massachusetts Green High Performance Computing Center). They have a BOF at 1:30 Wednesday. You should go to it.

[Jetstream2](#) especially Jeremy Fischer, Mike Lowe, and Julian Pistorius. [Jetstream2](#) has a tutorial at the same time as this one. Please stay here.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Acknowledgments and shameless plugs

└─ Acknowledgments and shameless plugs

Acknowledgments and shameless plugs

[OpenHPC](#) especially Tim Middelkoop (Internet2) and Chris Simmons (Massachusetts Green High Performance Computing Center). They have a BOF at 1:30 Wednesday. You should go to it.

[Jetstream2](#) especially Jeremy Fischer, Mike Lowe, and Julian Pistorius. Jetstream2 has a tutorial at the same time as this one. Please stay here.

[NSF CC*](#) for the equipment that led to some of the lessons we're sharing today (award #2127188).

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Acknowledgments and shameless plugs

└─ Acknowledgments and shameless plugs

x

Acknowledgments and shameless plugs

[OpenHPC](#) especially Tim Middelkoop (Internet2) and Chris Simmons (Massachusetts Green High Performance Computing Center). They have a BOF at 1:30 Wednesday. You should go to it.

[Jetstream2](#) especially Jeremy Fischer, Mike Lowe, and Julian Pistorius. Jetstream2 has a tutorial at the same time as this one. Please stay here.

[NSF CC*](#) for the equipment that led to some of the lessons we're sharing today (award #2127188).

[ACCESS](#) current maintainers of the project formerly known as the XSEDE Compatible Basic Cluster.

2024-07-17

OpenHPC: Beyond the Install Guide

└ Introduction

└└ Where we're starting from

└└└ Where we're starting from

x

Where we're starting from



Figure 1: Two example HPC networks for the tutorial

You:

- ▶ have installed OpenHPC before
- ▶ have been issued a (basically) out-of-the-box OpenHPC cluster for this tutorial

Cluster details:

- ▶ Rocky Linux 9 (x86_64)
- ▶ OpenHPC 3.1, Warewulf 3, Slurm 23.11.6
- ▶ 2 non-GPU nodes
- ▶ 0 GPU nodes (due to technical and licensing conflicts)
- ▶ 1 management node (SMS)
- ▶ 1 unprovisioned login node

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Where we're starting from

└─ Where we're starting from

We used the OpenHPC automatic installation script from Appendix A with a few variations:

1. Installed `s-mail` to have a valid `MailProg` for `slurm.conf`.
2. Created `user1` and `user2` accounts with password-less `sudo` privileges.
3. Changed `CHROOT` from `/opt/ohpc/admin/images/rocky9.3` to `/opt/ohpc/admin/images/rocky9.4`.
4. Enabled `slurmd` and `nsight` in `CHROOT`.
5. Added `nano` and `yum` to `CHROOT`.
6. Removed a redundant `NetworkService` line from `/etc/slurm/slurm.conf`.
7. Stored all compute/GPU nodes' SSH host keys in `/etc/ssh/ssh_known_hosts`.
8. Globally set an environment variable `CHROOT` to `/opt/ohpc/admin/images/rocky9.4`.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Introduction

└─ Where we're going

└─ Where we're going

Where we're going

1. A login node that's practically identical to a compute node (except for where it needs to be different)
2. A slightly more secured SMS and login node
3. GPU drivers on the GPU nodes
4. Using node-local storage for the OS and/or scratch
5. De-coupling the SMS and the compute nodes (e.g., independent kernel versions)
6. Easier management of node differences (GPU or not, diskless/single-disk/multi-disk, Infiniband or not, etc.)
7. Slurm configuration to match some common policy goals (fair share, resource limits, etc.)

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Assumptions

Assumptions

1. We have a VM named `login`, with no operating system installed.
2. The `eth0` network interface for `login` is attached to the internal network, and `eth1` is attached to the external network.
3. The `eth0` MAC address for `login` is known—check the **Login server** section of your handout for that. It's of the format `aa:bb:cc:dd:ee:ff`.
4. We're logged into the SMS as `user1` or `user2` that has `sudo` privileges.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Create a new login node

x

Create a new login node

Working from section 3.9.3 of the install guide:

```
[user@node ~]$ sudo wush -y node new login --netdev eth0 \
--ipaddr=172.16.0.2 --hwaddr=__:__:__:__:__:__
[user@node ~]$ sudo wush -y provision set login \
--node=rocky.4 --bootstrap=$(uname -r) \
--files=dynamic_hosts,passwords,group,shadow,munge.key,network
```

Make sure to replace the __ with the characters from your login node's MAC address!

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the iPXE project).

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the iPXE project).
3. The network card gets the bootloader over TFTP and executes it.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a filename option (a bootloader from the iPXE project).
3. The network card gets the bootloader over TFTP and executes it.
4. iPXE makes a second DHCP request and this time, it gets a URL (by default, [http://SMS_IP/Win/pxe/cfg/\\${client_mac}](http://SMS_IP/Win/pxe/cfg/${client_mac})) for an iPXE config file.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

Ever since login was powered on, it's been stuck in a loop trying to PXE boot. What's the usual PXE boot process for a client in an OpenHPC environment?

1. The client network card tries to get an IP address from a DHCP server (the SMS) by broadcasting its MAC address.
2. The SMS responds with the client's IP and network info, a next-server IP (the SMS again), and a `tftpname` option (a bootloader from the iPXE project).
3. The network card gets the bootloader over TFTP and executes it.
4. iPXE makes a second DHCP request and this time, it gets a URL (by default, `http://SMS_IP/VA/tftp/cfg/${client_mac}`) for an iPXE config file.
5. The config file contains the URL of a Linux kernel and initial ramdisk, plus multiple kernel parameters available after initial bootup for getting the node's full operating system contents.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ What'd we just do?

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ What'd we just do?

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the PXE configuration.
3. The node name, `--swdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.
3. The node name, `--swdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.
4. During the initial bootup, the `--baddr` parameter is passed to a CGI script on the SMS to identify the correct VNFS for the provisioning software to download (set by the `--vzfs` parameter).

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ What'd we just do?

x

What'd we just do?

1. The node name, `--baddr`, and `--ipaddr` parameters go into the SMS DHCP server settings.
2. The `--bootstrap` parameter defines the kernel and ramdisk for the IPXE configuration.
3. The node name, `--rootdev`, `--ipaddr`, `--baddr` parameters all go into kernel parameters accessible from the provisioning software.
4. During the initial bootup, the `--baddr` parameter is passed to a CGI script on the SMS to identify the correct VNFS for the provisioning software to download (set by the `--vzfs` parameter).
5. After downloading the VNFS, the provisioning software will also download files from the SMS set by the `--files` parameter.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Did it work? So far, so good.

Did it work? So far, so good.

```
[user@node ~]$ sudo ash login
[root@login ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
devtmpfs              2.0G   0   2.0G   0% /dev
tmpfs                 2.0G  843M  1.1G  20% /
tmpfs                 2.0G   0   2.0G   0% /dev/shm
tmpfs                 1.2G  6.5M  1.2G   1% /run
172.16.0.1:/home      10G   12G  7.4G  61% /home
172.16.0.1:/opt/ohpc/pub 100G   6.0G   94G   6% /opt/ohpc/pub
tmpfs                 592M   0   592M   0% /run/user/0
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Did it work? Not entirely.

x

Did it work? Not entirely.

```
[root@login ~]# sinfo
sinfo: error: resolve_ctls_from_dns_srv: res_nsearch error:
Unknown host
sinfo: error: fetch_config: DNS SRV lookup failed
sinfo: error: _establish_config_source: failed to fetch config
sinfo: fatal: Could not establish a configuration source
```

systemctl status slurm is more helpful, with
fatal: Unable to determine this slurm's NodeName. So how do we fix this one?

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the `SMS azure.conf`. To fix that:

1. Run `slurmstat -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.
4. Reload the new Slurm configuration everywhere (well, everywhere functional) with `sudo scontrol reconfigure` on the SMS.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Option 1: take the error message literally

x

Option 1: take the error message literally

So there's no entry for login in the SMS `slurm.conf`. To fix that:

1. Run `slurm -C` on the login node to capture its correct CPU specifications. Copy that line to your laptop's clipboard.
2. On the SMS, run `nano /etc/slurm/slurm/slurm.conf` and make a new line of all the `slurm -C` output from the previous step (pasted from your laptop clipboard).
3. Save and exit nano by pressing `Ctrl-X` and then `Enter`.
4. Reload the new Slurm configuration everywhere (well, everywhere functional) with `sudo scontrol reconfigure` on the SMS.
5. `ssh` back to the login node and restart `slurmd`, since it wasn't able to respond to the `scontrol reconfigure` from the previous step (`sudo ssh login ssystemctl restart slurmd` on the SMS).

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Option 1: take the error message literally

x

Option 1: take the error message literally

Now an `sinfo` should work on the login node:

```
[root@login ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*    up   1-00:00:00        1   idle c[1-2]
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Option 2: why are we running slurmd anyway?

Option 2: why are we running `slurmd` anyway?

The `slurmd` service is really only needed on systems that will be running computational jobs, and the login node is not in that category.

Running `slurmd` like the other nodes means the login node can get all its information from the SMS, but we can do the same thing with a very short customized `slurm.conf` with two lines from the SMS' `slurm.conf`:

```
ClusterName=cluster  
SlurmctldHost=sms
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Interactive test

x

Interactive test

1. On the login node as root, temporarily stop the slurm service with `systemctl stop slurm`
2. On the login node as root, edit `/etc/slurm/slurm.conf` with `nano /etc/slurm/slurm.conf`
3. Add the two lines to the right, save and exit `nano` by pressing `Ctrl-X` and then `Enter`.

Verify that `sinfo` still works without `slurm` and with the custom `/etc/slurm/slurm.conf`.

```
[root@login ~]# sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*    up   1:00:00:00       1  idle c[1-2]
```

`/etc/slurm/slurm.conf` on login node

```
ClusterName=cluster
SlurmctldHost=ans
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Make permanent changes from the SMS

Make permanent changes from the SMS

Let's reproduce the changes we made interactively on the login node in the Warewulf settings on the SMS.

For the customized `slurm.conf` file, we can keep a copy of it on the SMS and add it to the Warewulf file store.

We've done that previously for files like the shared `munge.key` for all cluster nodes (see section 3.8.5 of the OpenHPC install guide).

We also need to make sure that file is part of the login node's provisioning settings.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make permanent changes from the SMS

x

Make permanent changes from the SMS

On the SMS:

```
[user@node ~]$ sudo scp login:/etc/slurm/slurm.conf \  
/etc/slurm/slurm.conf.login \  
slurm.conf 100% 40 87.7KB/s 00:00  
[user@node ~]$ sudo wush -y file import \  
/etc/slurm/slurm.conf.login --name=slurm.conf.login \  
--path=/etc/slurm/slurm.conf
```

Now the file is available, but we need to ensure the login node gets it. That's handled with wush provision.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ A quick look at wush provision

x

A quick look at wush provision

What are the provisioning settings for compute node c1?

```
[userid@ ~]$ wush provision print c1
#### c1 #####
ci: MASTER           = UNDEF
ci: BOOTSTRAP        = c1.96-1.el9.elrepo.x86_64
ci: VMFS              = rocky9.4
ci: VALIDATE         = FALSE
ci: FILES            = dynamic_hosts.group.munge.key.network,
                        passwd.shadow
...
ci: KARGS             = "net.ifnames=0 biosdevname=0 quiet"
ci: BOOTLOCAL        = FALSE
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ A quick look at wsh provision

x

A quick look at wsh provision

What are the provisioning settings for node login?

```
[userid@node ~]$ wsh provision print login
### login #####
login: MASTER      = UNDEF
login: BOOTSTRAP    = 0.1.50-1.el9.elrepo.x86_64
login: VNF2         = rocky9.4
login: VALIDATE     = FALSE
login: FILES        = dynamic_hosts,group,munge.key,network,
                    passwd,shadow
...
login: XARGS        = "set.ifnames=0 biosdevname=0 quiet"
login: BOOTLOCAL    = FALSE
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ A quick look at `wwsh` provision

x

A quick look at `wwsh` provision

The provisioning settings for `c1` and `login` are identical, but there's a lot to read in there to be certain about it.

We could run the two outputs through `diff`, but every line contains the node name, so **no lines are literally identical**.

Let's simplify and filter the `wwsh` provision output to make it easier to compare.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Filter the `wwsh provision` output

x

Filter the `wwsh provision` output

► I only care about the lines containing `= signs`, so

```
wwsh provision print cl | grep "="
```

is a start.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Filter the `wwsh provision` output

x

Filter the `wwsh provision` output

- I only care about the lines containing `= signs`, so

```
wwsh provision print cl | grep "="
```

is a start.

- Now all the lines are prefixed with `cl::`, and I want to keep everything after that, so

```
wwsh provision print cl | grep "=" | cut -d: -f2-
```

will take care of that.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Filtered result

Filtered result

```
vvah provision print cl | grep = | cut -d: -f2-  
MASTER      = UNDEF  
BOOTSTRAP    = 6.1.06-1.el9.elrepo.x86_64  
YNFS          = rocky9-4  
VALIDATE     = FALSE  
FILES        = dynamic_hosts.group.manage.key.network,  
              passwd,shadow  
...  
KARGES       = "net.ifnames=0 biosdevname=0 quiet"  
BOOTLOCAL    = FALSE
```

Much more useful.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make a function for this

x

Make a function for this

We may be typing that command pipeline a lot, so let's make a shell function to cut down on typing:

```
[user@node ~]$ function pprint() { \
  vssh provision print $@ | grep - | cut -d: -f2- ; }
[user@node ~]$ pprint cl
MASTER                = OMPI
BOOTSTRAP               = 6.1.96-1.el9.elrepo.x86_64
...

```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ diff the outputs

x

diff the outputs

We could redirect a `proprint ci` and a `proprint login` to files and `diff` the resulting files, or we can use the shell's `<()` operator to treat command output as a file:

```
[user@node ~]$ diff -u <($(proprint ci)) <($(proprint login))  
[user@node ~]$
```

Either of those shows there are zero provisioning differences between a compute node and the login node.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Add the custom `slurm.conf` to the login node

Add the custom `slurm.conf` to the login node

Add a file to login's FILES property with:

```
[user@node ~]$ sudo websh -y provision set login \
--fileadd=slurm.conf.login
```

(refer to section 3.9.3 of the install guide for previous examples of `--fileadd`).

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Check for provisioning differences

Check for provisioning differences

```
[user@node ~]$ diff -u <($(preprint ci) <($(preprint login))
--- /dev/fd/63 2024-07-06 11:11:07.682959677 -0400
+++ /dev/fd/62 2024-07-06 11:11:07.683959681 -0400
@@ -2,7 +2,7 @@
BOOTSTRAP      = 6.1.96-1.el9.elrepo.x86_64
SWS            = rocky9.4
VALIDATE       = FALSE
FILES          = dynamic_hosts.group,munge.key,network,
+passwd,shadow
+passwd,shadow,slurm.conf,login
PRESSHELL      = FALSE
POSTSHELL      = FALSE
POSTRETDOWN    = FALSE
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Ensure slurmd doesn't run on the login node

x

Ensure `slurmd` doesn't run on the login node

To disable the `slurmd` service on just the login node, we can take advantage of conditions in the `systemd` service file. Back on the login node as root:

```
[user@node ~]$ sudo ash login
[root@login ~]# systemctl edit slurmd
```

Insert these lines between the lines of `## Anything between here... and` and `## Lines below this comment...:`

```
[Unit]
ConditionHost=!c*
ConditionHost=!g*
```

This will only run the service on nodes whose hostnames start with `c` or `g` (we don't have any `g` nodes here, but this is how you can handle multiple name types).

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Ensure slurmd doesn't run on the login node

x

Ensure slurmd doesn't run on the login node

Once that file is saved, try to start the slurmd service with `systemctl start slurmd` and check its status with `systemctl status slurmd`.

```
s slurmd.service - Slurm node daemon
..
Condition: start condition failed at Sat 2024-07-06 18:12:17
EDT; 4min 22s ago
..
Jul 06 17:14:16 login systemd[1]: Stopped Slurm node daemon.
Jul 06 18:12:17 login systemd[1]: Slurm node daemon was skipped
because of an unset condition check (ConditionNot=c*).
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make the changes permanent

x

[Make the changes permanent](#)

The `systemctl edit` command resulted in a file `/etc/systemd/system/slurmd.service.d/override.conf`. Let's:

- make a place for it in the cheat on the SMS, and
- copy the file over from the login node.

```
[user@lana ~]$ sudo mkdir -p \
${CHROOT}/etc/systemd/system/slurmd.service.d/
[user@lana ~]$ sudo scp \
login:/etc/systemd/system/slurmd.service.d/
${CHROOT}/etc/systemd/system/slurmd.service.d/
override.conf 100% 23 36.7KB/s 00:00
```

(Note: we globally pre-set the `CHROOT` environment for any account that logs into the SMS so that you didn't have to.)

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make the changes permanent

x

Make the changes permanent

Finally, we'll:

- rebuild the VNFs, and
- reboot both the login node and a compute node to test the changes.

```
[user@ana ~]$ sudo uvvns --chroot=${CHROOT}
Using 'rocky9.4' as the VNFs name
...
Total elapsed time
: 04:45 s
[user@ana ~]$ sudo azh login reboot
[user@ana ~]$ sudo azh ci reboot
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Verify the changes on the login node

[Verify the changes on the login node](#)

Verify that the login node doesn't start slurm, but can still run sinfo without any error messages.

```
[user@node ~]$ sudo ash login systemctl status slurm
s slurm.service - Slurm node daemon
...
Jul 06 18:26:23 login systemd[1]: Slurm node daemon was
skipped because of an unset condition check
(ConditionHost=c*).
[user@node ~]$ sudo ash login sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*    up  1-00:00:00      1  idle c[1-2]
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Verify the changes on a compute node

x

Verify the changes on a compute node

Verify that the compute node still starts slurmd (it can also run sinfo).

```
[user@node ~]$ sudo ssh ci systemctl status slurmd
● slurmd.service - Slurm node daemon
...
Jul 06 19:03:22 ci slurmd[1082]: slurmd: CPUs=2 Boards=1
Sockets=2 Cores=1 Threads=1 Memory=5012 TopDisk=2956
Options=DS CPUOptions=(null) FeaturesAvail=(null)
FeaturesActive=(null)
[user@node ~]$ sudo ssh ci sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE MODELIST
normal*    up 1:00:00:00      1  idle  ci2
normal*    up 1:00:00:00      1  down  ci
```

(Yes, ci is marked down—we'll fix that shortly.)

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A dedicated login node

- └─ Problem: the login node doesn't let users log in

x

Problem: the login node doesn't let users log in

What if we ssh to the login node as someone other than root?

```
[user@node ~]$ ssh login
Access denied: user user1 (uid=1001) has no active jobs on this
node.
Connection closed by 172.16.0.2 port 22
```

which makes this the exact opposite of a login node for normal users. Let's fix that.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Make the login node function as a login node

x

Make the login node function as a login node

- ▶ The Access denied is caused by the `pan_sshd.no` entry at the end of `/etc/pan.d/sshd`, which is invaluable on a normal compute node, but not on a login node.
- ▶ On the SMS, you can also do a `diff -u /etc/pan.d/sshd $(CWDOT)/etc/pan.d/sshd`
- ▶ You'll see that the `pan_sshd.no` line is the only difference between the two files.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A dedicated login node

└─ Test a PAM change to the login node

Test a PAM change to the login node

- ▶ Temporarily comment out the last line of the login node's `/etc/pam.d/ssh` and see if you can ssh into the login node as a normal user (i.e., `ssh user1@login`).
- ▶ Your user should be able to log in now.
- ▶ In case the PAM configuration won't let root log in, **don't panic!** Instructors can reboot your login node from its console to put it back to its original state.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make the change permanent

x

Make the change permanent

- We want to ensure that the login node gets the same `/etc/pam.d/sshd` that the SMS uses.
- We'll follow the same method we used to give the login node a custom `slurm.conf`:

```
[user@ana ~]$ sudo wvuh -y file import /etc/pam.d/sshd \
--name=sshd.login
[user@ana ~]$ wvuh file list
...
sshd.login : rw-r--r-- 1 root root 727 /etc/pam.d/sshd
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Make the change permanent

x

Make the change permanent

```
[username ~]$ sudo wush -y provision set login \
--fileadd=sashd.login
[username ~]$ diff -u <($(proprint cl) <($(proprint login)
...
VALIDATE      = FALSE
- FILES        = dynamic_hosts.group,munge.key,network,
passwd,shadow
+ FILES        = dynamic_hosts.group,munge.key,network,
passwd,shadow,elurm.conf.login,sashd.login
...

```

(refer to section 3.9.3 of the install guide for previous examples of --fileadd).

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A dedicated login node
 - └─ Test the change

x

[Test the change](#)

Reboot the login node and let's see if we can log in as a regular user.

```
[user@node ~]$ sudo ssh login reboot  
[user@node ~]$ ssh login  
[user@login ~]$
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ A bit more security for the login node

A bit more security for the login node

Not too long after your SMS and/or login nodes are booted, you'll see messages in the SMS /var/log/secure like:

```
Jul 11 11:24:08 sms sshd[162636]: Invalid user evilmike from  
68.66.205.120 port 1028  
...  
Jul 11 11:24:08 sms sshd[162636]: Failed password for invalid  
user evilmike from 68.66.205.120 port 1028 ssh2  
...
```

because people who want to break into computers for various reasons have Internet connections.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A bit more security for the login node

└─ A bit more security for the login node

A bit more security for the login node

There's a lot of things that can be done to secure things, including:

1. Placing the SMS and login node external interfaces on protected network segment.
2. Allowing only administrative users to SSH into the SMS.
3. Replacing password-based authentication with key-based authentication.

Though #3 will eliminate brute-force password guessing attacks, it's usually not practical for a login node. So let's mitigate that differently with `fail2ban`.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A bit more security for the login node

└─ How fail2ban works (by default)

How fail2ban works (by default)

1. Monitor `/var/log/secure` and other logs for indicators of brute-force attacks (invalid users, failed passwords, etc.)
2. If indicators from a specific IP address happen often enough over a period of time, use `firewalld` to block all access from that address for a period of time.
3. Once that period has expired, remove the IP address from the block list.

This reduces the effectiveness of brute-force password guessing by orders of magnitude (~10 guesses per hour versus ~100 or ~1000 guesses per hour).

Including `firewalld` could mean that some necessary services get blocked by default when `firewalld` starts. Let's see what those could be.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ See what processes are listening on the login node

See what processes are listening on the login node

We'll use the `netstat` command to look for sockets that are udp or tcp, listening, and what process the socket is attached to. We omit anything only listening for localhost connections.

```
[user@node ~]$ sudo ssh login netstat -tulp | grep -v localhost
Active Internet connections (only servers)
Proto ... Local Address ... State PID/Program name
tcp      0.0.0.0:ssh      LISTEN 1034/sshd: /usr/abi
tcp      0.0.0.0:swarp    LISTEN 1/init
tcp6     [::]:ssh        LISTEN 1034/sshd: /usr/abi
tcp6     [::]:swarp      LISTEN 1/init
udp      0.0.0.0:swarp    0.0.0.0:* 1/init
udp      0.0.0.0:37036    0.0.0.0:* 1143/rpylogd
udp6     [::]:swarp      [::]:* 1/init
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ See what processes are listening on the login node

x

See what processes are listening on the login node

`sshd` secure shell daemon, the main thing we want to protect against brute force attempts

`init` the first process started during booting the operating system. Effectively, this shows up when you participate in NFS file storage, as a server or a client (and login is a client).

`rsyslogd` message logging for all kinds of applications and services

Of these, `sshd` is the only one that we need to ensure `firewalld` doesn't block by default. In practice, the `ssh` port (22) is always in the default list of allowed ports.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ Test installing fail2ban on the login node

Test installing fail2ban on the login node

Install the fail2ban packages into the CHROOT with

```
[user@hpc ~]$ sudo yum install --installroot=${CHROOT} \
fail2ban
[user@hpc ~]$ sudo chroot ${CHROOT} systemctl enable \
fail2ban firewalld
```

(the yum command will also install firewalld as a dependency of fail2ban).

Add the following to the chroot's sashd.local file with
sudo nano \${CHROOT}/etc/fail2ban/jail.d/sashd.local:

```
[sashd]
enabled = true
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Should I run `fail2ban` everywhere?

x

Should I run `fail2ban` everywhere?

`fail2ban` is probably best to keep to the login node, and not the compute nodes:

- ▶ Nobody can SSH into your compute nodes from outside.
- ▶ Thus, the only things a compute node could ban would be your SMS or your login node.
- ▶ A malicious or unwitting user could easily ban your login node from a compute node by SSH'ing to it repeatedly, which would effectively be a denial of service.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Test installing fail2ban on the login node

Test installing fail2ban on the login node

```
[user@node ~]$ sudo mkdir -p \
${CHROOT}/etc/systemd/system/fail2ban.service.d/ \
${CHROOT}/etc/systemd/system/firewalld.service.d/
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ Test installing fail2ban on the login node

Test installing fail2ban on the login node

```
[user@node ~]$ sudo nano \
${CHROOT}/etc/systemd/system/fail2ban.service.d/override.conf
```

Add the lines

```
[Unit]
ConditionRoot=/login*
```

save and exit with Ctrl-X.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Test installing fail2ban on the login node

x

Test installing fail2ban on the login node

Finally, duplicate the override file for firewalld:

```
[user@node ~]$ sudo cp \
${CHROOT}/etc/systend/system/fail2ban.service.d/override.conf \
${CHROOT}/etc/systend/system/firewalld.service.d/override.conf
```


2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Test installing fail2ban on the login node

x

Test installing fail2ban on the login node

Before we go further, check if there's anything in `/var/log/secure` on the login node:

```
[User@node ~]$ sudo cat login |> /var/log/secure
-rw-r----- 1 root root 0 Jul 7 03:14 /var/log/secure
```

Nope. Let's fix that, too.

- Looking in `/etc/rsyslog.conf`, we see a bunch of things commented out, including the line `#authpriv.* /var/log/secure`.
- Rather than drop in an entirely new `rsyslog.conf` file that we'd have to maintain, `rsyslog` will automatically include any `*.conf` files in `/etc/rsyslog.d`.
- Let's make one of those for the chroot.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ Make an rsyslog.d file, rebuild the VNFS, reboot the login node

Make an rsyslog.d file, rebuild the VNFS, reboot the login node

```
[user@hpc ~]$ echo "authpriv.* /var/log/secure" | \
sudo tee ${CHROOT}/etc/rsyslog.d/authpriv-local.conf
authpriv.* /var/log/secure
[user@hpc ~]$ cat \
${CHROOT}/etc/rsyslog.d/authpriv-local.conf
authpriv.* /var/log/secure
[user@hpc ~]$ sudo mvnfs --chroot=${CHROOT}
[user@hpc ~]$ sudo ash login reboot
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ Post-reboot, how's fail2ban and firewalld on the login node?

Post-reboot, how's fail2ban and firewalld on the login node?

```
[user@node ~]$ sudo ssh login systemctl status firewalld
[root@login ~]# systemctl status firewalld
x firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service;
   enabled; preset)
   Active: failed (Result: exit-code) since Thu 2024-07-11
   16:40:47 EDT; 46ms>
...
Jul 11 16:40:47 login systemd[1]: firewalld.service: Main
   process exited, code=exited, status=3/NOTIMPLEMENTED
Jul 11 16:40:47 login systemd[1]: firewalld.service: Failed
   with result "exit-code".
```

Not great.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

Diagnosing 3/NOTIMPLEMENTED

► So many Google results amount to "reboot to get your new kernel", but we've just booted a new kernel.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- So many Google results amount to "reboot to get your new kernel", but we've just booted a new kernel.
- Red Hat has an article telling you to verify that you haven't disabled module loading by checking `sysctl -s | grep modules_disabled`, but that's not disabled either.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- So many Google results amount to "reboot to get your new kernel", but we've just booted a new kernel.
- Red Hat has an article telling you to verify that you haven't disabled module loading by checking `sysctl -a | grep modules_disabled`, but that's not disabled either.
- The Red Hat article does tell you that packet filtering capabilities have to be enabled in the kernel, and that gets us closer.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- So many Google results amount to "reboot to get your new kernel", but we've just booted a new kernel.
- Red Hat has an article telling you to verify that you haven't disabled module loading by checking `zypper --n | grep modules_disabled`, but that's not disabled either.
- The Red Hat article does tell you that packet filtering capabilities have to be enabled in the kernel, and that gets us closer.
- It is possible to install and start `firewalld` on the SMS (you don't have to verify this right now), and that's using the same kernel as the login node.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- So many Google results amount to "reboot to get your new kernel", but we've just booted a new kernel.
- Red Hat has an article telling you to verify that you haven't disabled module loading by checking `zypper --s | grep modules_disabled`, but that's not disabled either.
- The Red Hat article does tell you that packet filtering capabilities have to be enabled in the kernel, and that gets us closer.
- It is possible to install and start `firewalld` on the SMS (you don't have to verify this right now), and that's using the same kernel as the login node.
- Or is it?

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

Diagnosing 3/NOTIMPLEMENTED

- How did we get the kernel that the login node is using?

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

Diagnosing 3/NOTIMPLEMENTED

- ▶ How did we get the kernel that the login node is using?
- ▶ Via vubootstrap `$(uname -r)` on the SMS (section 3.9.1)

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A bit more security for the login node

└─ Diagnosing 3/NOTIMPLEMENTED

Diagnosing 3/NOTIMPLEMENTED

- ▶ How did we get the kernel that the login node is using?
- ▶ Via `vubootstrap $(uname -r)` on the SMS (section 3.9.1)
- ▶ That section **also** had a command that most of us don't pay close attention to:
`echo "drivers += updates/kernel/" >> /etc/warwulf/bootstrap.conf`

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better infrastructure nodes

└─ A bit more security for the login node

└─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- ▶ How did we get the kernel that the login node is using?
- ▶ Via `vubootstrap $(uname -r)` on the SMS (section 3.9.1)
- ▶ That section **also** had a command that most of us don't pay close attention to:
`echo "drivers += updatea/kernel/" >> /etc/warewolf/bootstrap.conf`
- ▶ So though the login node is running the same kernel **version** as the SMS, it may **not** have all the drivers included.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- How did we get the kernel that the login node is using?
- Via vubootstrap `$(uname -r)` on the SMS (section 3.9.1)
- That section **also** had a command that most of us don't pay close attention to:
`echo "drivers += updatea/kernel/" >> /etc/vaerwulf/bootstrap.conf`
- So though the login node is running the same kernel **version** as the SMS, it may **not** have all the drivers included.
- Where are the drivers we care about? `lsmod` on the SMS shows a lot of `nf`-named modules for the Netfilter kernel framework.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

- How did we get the kernel that the login node is using?
- Via vubootstrap `$(uname -r)` on the SMS (section 3.9.1)
- That section **also** had a command that most of us don't pay close attention to:
`echo "drivers += updates/kernel/" >> /etc/varsnullf/bootstrap.conf`
- So though the login node is running the same kernel **version** as the SMS, it may **not** have all the drivers included.
- Where are the drivers we care about? `lsmod` on the SMS shows a lot of `nf`-named modules for the Netfilter kernel framework.
- `find /lib/modules/$(uname -r) -name '*nf*'` shows these modules are largely located in the `kernel/net` folder (specifically `kernel/net/ipv4/netfilter`, `kernel/net/ipv6/netfilter`, and `kernel/net/netfilter`).

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

ls kernel/net in our /etc/warewolf/bootstrap.conf at all?

```
[user@hsm ~]$ grep kernel/net /etc/warewolf/bootstrap.conf
[user@hsm ~]$
```

Nope, let's add it.

```
[user@hsm ~]$ grep kernel/net /etc/warewolf/bootstrap.conf
[user@hsm ~]$ echo "drivers += kernel/net/" | \
sudo tee -a /etc/warewolf/bootstrap.conf
drivers += kernel/net/
[user@hsm ~]$ grep kernel/net /etc/warewolf/bootstrap.conf
drivers += kernel/net/
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Diagnosing 3/NOTIMPLEMENTED

x

Diagnosing 3/NOTIMPLEMENTED

Let's re-run the wubootstrap command and reboot the login node:

```
[user@hans ~]$ sudo wubootstrap $(uname -r)
...
Bootstrap image '6.1.97-1.el9.elrepo.x86_64' is ready
Done.
[user@hans ~]$ sudo ssh login reboot
```


2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Did 3/NOTIMPLEMENTED go away?

x

Did 3/NOTIMPLEMENTED go away?

```
[user@hpc ~]$ sudo nsh login systemctl status firewalld
* firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service;
           enabled; preset: enabled)
   Active: active (running) since Thu 2024-07-11 21:58:18
           EDT; 43s ago
   ...
Jul 11 21:58:18 login systemd[1]: Starting firewalld - dynamic
firewall daemon...
Jul 11 21:58:18 login systemd[1]: Started firewalld - dynamic
firewall daemon.
```

It did.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes
 - └─ A bit more security for the login node
 - └─ Does fail2ban actually work now?

x

Does fail2ban actually work now?

```
[user@node ~]$ sudo ssh login grep 68.66.205.120 \  
/var/log/fail2ban.log  
...  
2024-07-11 22:02:27,030 fail2ban.actions ... [sshd] Ban \  
68.66.205.120
```

It does.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better infrastructure nodes

- └─ A bit more security for the login node

- └─ What does it look like from evilmike's side?

What does it look like from evilmike's side?

```
mike@server:~$ ssh evilmike@149.165.155.235
evilmike@149.165.155.235's password:
Permission denied, please try again.
evilmike@149.165.155.235's password:
Permission denied, please try again.
evilmike@149.165.155.235's password:
evilmike@149.165.155.235: Permission denied (publickey,
gssapi-keyex,gssapi-with-mic,password).
mike@server:~$ ssh evilmike@149.165.155.235
ssh: connect to host 149.165.155.235 port 22: Connection
refused
```

evilmike is thwarted, at least for now.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ More seamless reboots of compute nodes

└─ Why was c1 marked as down?

x

Why was c1 marked as down?

You can return c1 to an idle state by running
`sudo scontrol update node=c1 state=resume` on the SMS:

```
[user@hpc ~]$ sudo scontrol update node=c1 state=resume
[user@hpc ~]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*   up  1-00:00:00      1    idle c[1-2]
```

We should configure things so that we don't have to manually resume nodes every time we reboot them.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ More seamless reboots of compute nodes
 - └─ More seamless reboots of compute nodes

x

More seamless reboots of compute nodes

- Slurm doesn't like it when a node gets rebooted without its knowledge.
- There's an `scontrol reboot` option that's handy to have nodes reboot when system updates occur, but it requires a valid setting for `RebootProgram` in `/etc/slurm/slurm.conf`.
- By default, Slurm and OpenHPC don't ship with a default `RebootProgram`, so let's make one.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ More seamless reboots of compute nodes
 - └─ Adding a valid RebootProgram

x

Adding a valid RebootProgram

```
[user@node ~]$ grep -i reboot /etc/slurm/slurm.conf
#RebootProgram=""
[user@node ~]$ echo 'RebootProgram="/sbin/shutdown -r now"' \
| sudo tee -a /etc/slurm/slurm.conf
[user@node ~]$ grep -i reboot /etc/slurm/slurm.conf
#RebootProgram=""
RebootProgram="/sbin/shutdown -r now"
```

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ More seamless reboots of compute nodes

└─ Informing all nodes of the changes and testing it out

x

Informing all nodes of the changes and testing it out

```
[user@node ~]$ sudo scontrol reconfigure  
[user@node ~]$ sudo scontrol reboot ASAP sxtatstate=RESUME c1
```

- ▶ scontrol reboot will wait for all jobs on a group of nodes to finish before rebooting the nodes.
- ▶ scontrol reboot ASAP will immediately put the nodes in a DRAIN state, routing all pending jobs to other nodes until the rebooted nodes are returned to service.
- ▶ scontrol reboot ASAP sxtatstate=RESUME will set the nodes to accept jobs after the reboot. sxtatstate=DRAIN will leave the nodes in a DRAIN state if you need to do more work on them before returning them to service.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ More seamless reboots of compute nodes
 - └─ Did it work?

Did it work?

```
[username ~]$ sudo ash ci uptime
15:52:27 up 1 min, 0 users, load average: 0.09, 0.06, 0.02
[username ~]$ winfo
PARTITION AVAIL TIMELIMIT NODES STATE NODELIST
normal*   up 1-00:00:00    1   idle c[1-2]
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Downsides of stateless provisioning

x

Downsides of stateless provisioning

Log into c1 as root, check available disk space and memory, then allocate a 5 GB array in memory:

```
[user@node ~]$ sudo ssh c1
[root@c1 ~]# df -h /tmp
Filesystem      Size  Used Avail Use% Mounted on
tmpfs           2.0G  843M  2.1G  29% /
[root@c1 ~]# free -m
              total        used         free   ...
Mem:           5912          3162          2862   ...
Swap:              0              0              0   ...
[root@c1 ~]# module load python3-numpy
[root@c1 ~]# python3 -c \
'import numpy as np; x=np.full((25000, 25000), 1)'
[root@c1 ~]#
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Downsides of stateless provisioning

Downsides of stateless provisioning

Consume some disk space in /tmp, try to allocate the same 5 GB array again:

```
[root@cl ~]# dd if=/dev/zero of=/tmp/foo bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 0.63492 s, 1.7 GB/s
[root@cl ~]# module load py3-numpy
[root@cl ~]# python3 -c \
'import numpy as np; x=np.full((25000, 25000), 1)'
```

Killed

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Downsides of stateless provisioning

x

Downsides of stateless provisioning

Clean off the disk usage, allocate the 5 GB array in memory once more, and log out from the node:

```
[root@cl ~]# rm /tmp/foo
[root@cl ~]# python3 -c \
'import numpy as np; x=np.full((25000, 25000), 1)'
[root@cl ~]# exit
[user@hpc ~]$
```

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ Summary of the default OpenHPC settings

Summary of the default OpenHPC settings

1. The root filesystem is automatically sized to 50% of the node memory.
2. There's no swap space.
3. Consumption of disk space affects the workloads you can run (since disk space is really in RAM).

Even if we reformat node-local storage every time we reboot, moving file storage from RAM to disk is beneficial.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Strategies

x

Strategies

Typical bare-metal node

- PXE handled by network card, all disks available for node-local storage
- Usually, the default kernel contains all the drivers you need

Jetstream2 instance

- First disk (`/dev/vda`) exists to provide iPXE support, so don't break that
- Some extra steps may be needed to enable storage and filesystem kernel modules

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Examine the existing partition scheme (non-GPU nodes)

Examine the existing partition scheme (non-GPU nodes)

Log back into a compute node as root, check the existing partition table:

```
[user@node ~]$ sudo mkfs -l parted -l /dev/vda
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
1	1049kB	3146kB	2097kB		EFI	boot, esp

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Summary of existing partition scheme

Summary of existing partition scheme

1. GPT (GUID partition table) method on both node types
2. Different amounts of disk space on each node type
3. Each sector is 512 bytes
4. Bootable partition 1 (from 1040 kB = 1 MiB to 3146 kB = 3 MiB) for iPXE

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Plan for new partition scheme

Plan for new partition scheme

1. Non-destructive partitioning of `/dev/sda` once (outside of Warewulf, with a parted script).
2. 512 MiB partition for `/boot`.
3. 2 GiB partition for swap.
4. 2 GiB partition for `/`.
5. remaining space for `/tmp`.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ Define new partition scheme

x

Define new partition scheme

Could make a copy of an OpenHPC-provided example partition scheme (in `/etc/wareulf/filesystems/examples`), but we'll start one from scratch:

```
[user@node ~]$ sudo nano \
/etc/wareulf/filesystems/jetstream.cfs
```

- ▶ The `.cfs` file is a mix of parted commands, `size` parameters, and `/etc/fstab` information.
- ▶ It's typical, **but not 100% required**, to give a full set of `select`, `skipart`, and `name` commands for parted.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ Define new partition scheme

x

Define new partition scheme

Contents of jetstream.cnda (part 1):

```
select /dev/vda
```

On Jetstream2:

- ▶ we leave /dev/vda1 unmodified, since we need it for iPXE booting,
- ▶ we "semi-manually" (i.e., outside of Warewulf, but using a script) partition the rest of /dev/vda to include
 - ▶ 512 MiB for /boot
 - ▶ 2 GiB for swap
 - ▶ 2 GiB for /
 - ▶ remaining space for /tmp

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ Define new partition scheme

Contents of `jetstream.cde` (part 2):

```
# mkpart primary 3MiB 515MiB
# mkpart primary 515MiB 2563MiB
# mkpart primary 2563MiB 4611MiB
# mkpart primary 4611MiB 100%
name 2 boot
name 3 swap
name 4 root
name 5 tmp
```

- Note how to create partitions, and add commands to label them.
- `mkpart` commands are intended to be comments here, so that Warewulf can ignore them, but we can keep everything in one place.

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Define new partition scheme

Define new partition scheme

Contents of jetstream.cnda (part 3):

```
## mkfs NUMBER FS-TYPE [ARGS...]
mkfs 2 ext4 -L boot
mkfs 3 swap
mkfs 4 ext4 -L root
mkfs 5 ext4 -L tmp
## fatlab NUMBER mountpoint type opts freq passno
fatlab 4 /      ext4 defaults 0 0
fatlab 2 /boot ext4 defaults 0 0
fatlab 3 none  swap defaults 0 0
fatlab 5 /tmp  ext4 defaults 0 0
```

- Format partitions and mount them.
- Save and exit nano with Ctrl-X.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Partition the disks outside of Warewulf

Partition the disks outside of Warewulf

- ▶ parted has a `--script` parameter helpful for passing in one or more commands at the command line.
- ▶ We want to pass in the commented `mkpart` commands of our `jetstream.coda` file.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Show the mkpart lines

[Show the mkpart lines](#)

```
[user@node ~]$ grep mkpart \
/etc/warewulf/filesystem/jetstream.cds
# mkpart primary 3MiB 515MiB
# mkpart primary 515MiB 2663MiB
# mkpart primary 2663MiB 4611MiB
# mkpart primary 4611MiB 100%
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Take out the # signs from the `mkpart` lines

Take out the # signs from the `mkpart` lines

```
[user@node ~]$ grep mkpart \  
/etc/warewulf/filesystems/jetstream.cfs | sed 's/#//g'  
mkpart primary 38MiB 515MiB  
mkpart primary 515MiB 2663MiB  
mkpart primary 2663MiB 4611MiB  
mkpart primary 4611MiB 100%
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Put all the commands on one line

Put all the commands on one line

```
[user@node ~]$ echo $(grep mkpart \
/etc/xsawulf/filesystem/jetstream.cmds | sed 's/#//g')
mkpart primary 3MiB 515MiB mkpart primary 515MiB 2663MiB mkpart prim
primary ext4 4611MiB 100%
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Partition the drive

Partition the drive

(all of the below goes on one literal line, no backslashes, line breaks, or anything else)

```
user@node ~]$ sudo ash ci parted --script /dev/vda  
$(echo $(grep mkpart  
/etc/waasult/filesystem/jetstream.cde )  
sed 's/#/g')
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ Check your results

x

[Check your results](#)

```
[user@node ~]$ sudo zsh ci parted -l
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 64.4GB
...


| Number | Start  | End    | Size   | File system | Name    | Flags     |
|--------|--------|--------|--------|-------------|---------|-----------|
| 1      | 1049kB | 3145kB | 2097kB |             | EFI     | boot, esp |
| 2      | 3145kB | 540MB  | 537MB  |             | primary |           |
| 3      | 540MB  | 2688MB | 2147MB |             | primary | swap      |
| 4      | 2688MB | 4835MB | 2147MB |             | primary |           |
| 5      | 4835MB | 21.5GB | 16.6GB |             | primary |           |


```

Now repeat the previous `sudo zsh NODE parted --script` command for the other node (c2).

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Apply the Warewulf filesystem provisioning commands to the nodes

Apply the Warewulf filesystem provisioning commands to the nodes

```
(user@node ~)$ sudo wvuh provision set 'c*' \  
--filesystem=jetstream
```

Do not reboot your nodes yet!

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ What could possibly go wrong?

x

What could possibly go wrong?

- ▶ A lot, if you consider some edge cases and corner cases.
- ▶ This was by far the slowest-progressing and most error-prone section of the tutorial to develop.
- ▶ Using `wash provision set XXXX --preshell=1` and/or `--postshell=1` during debugging was invaluable.
- ▶ Rather than have y'all suffer through this without easy access to a console, I'll take you through what would have gone wrong if we'd rebooted just now.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ What went wrong (part 1)

x

What went wrong (part 1)

```
Starting the provision handler:
+ ansible-playbook auto configuration not activated           SKIPPED
+ filesystem                                                    SKIPPED
+ devices not ready, retrying...
```

Figure 2: Device not ready, retrying

► Running `dsmsg | grep vd` at the postshell command prompt confirmed that no `/dev/vd` devices were found.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ How it got fixed (part 1)

x

How it got fixed (part 1)

- ▶ Comparing the `lsmod` output on the failing node versus the SMS indicated we were missing the `virtio_blk` kernel module.
- ▶ Running `sodprobe virtio_blk` and `dmseg | grep vd` at the postshell command prompt confirmed this.
- ▶ Waresulf fix is to:
 - ▶ `run`
 - `echo modprobe += virtio_blk | sudo tee -a /etc/waresulf/bootstrap.conf`
 - ▶ `run sudo wbootstrap KERNEL_VERSION`
 - ▶ reboot the node and try again.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ What went wrong (part 2)

What went wrong (part 2)

```
• filesystem                                000000
• mounting parted on sda-sdb                00
• formatting sda-sdb1                        00
• formatting sda-sdb2                        00
• formatting sda-sdb3                        00
• formatting sda-sdb4                        00
• formatting sda-sdb5                        00
• mounting /                                CS011
                                          CS011
```

Figure 3: Mounting /, error

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Semi-stateful node provisioning
 - └─ How it got fixed (part 2)

x

How it got fixed (part 2)

```
# parted -l
Model: Hifiso Black Device (virt10k)
Disk /dev/vda: 50.7GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Mount      Flags
 1      1049kB  3106kB  2057kB  xfs          primary/
 2      3106kB  5289kB  2182kB  ext4        boot
 3      5289kB  25329kB 20039kB linux-swap(v1)  swap
 4      25329kB 47329kB 22000kB ext4        root
 5      47329kB 50729kB 3400kB  ext4        /usr
```

Figure 4: parted -l looks ok

► running parted -l showed a valid partition table

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ How it got fixed (part 2)

[How it got fixed \(part 2\)](#)

► Trying to mount the proposed root partition with
`mkdir /mnt ; mount -t xfs /dev/vdb4 /mnt` failed with
mount: mounting /dev/vdb4 as /mnt failed: No such file or directory

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

How it got fixed (part 2)

- Trying to mount the proposed root partition with
`mkdir /mnt ; mount -t xfs /dev/sdb4 /mnt` failed with
`mount: mounting /dev/sdb4 as /mnt failed: No such file or directory`
- But both `/mnt` and `/dev/sdb4` both existed, as seen from `ls -l` on each of them.

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

x

How it got fixed (part 2)

- ▶ Trying to mount the proposed root partition with `addio /mnt ; mount -t auto /dev/sdb4 /mnt` failed with `mount: mounting /dev/sdb4 as /mnt failed: No such file or directory`
- ▶ But both `/mnt` and `/dev/sdb4` both existed, as seen from `ls -l` on each of them.
- ▶ Surprisingly, when I left the root partition as a ramdisk and tried to partition and mount `swap` and `/tmp` from disk partitions, provisioning threw errors, but post-provisioning, both `swap` and `/tmp` **were available** to the node!

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ How it got fixed (part 2)

How it got fixed (part 2)

► What was different? A missing filesystem module in the provisioning kernel (in my case, `ext4`).

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

How it got fixed (part 2)

- What was different? A missing filesystem module in the provisioning kernel (in my case, `ext4`).
- Running `nodprobe ext4` at the `portabell` command prompt and re-running the `mount` command above caused the filesystem to mount.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ How it got fixed (part 2)

How it got fixed (part 2)

- What was different? A missing filesystem module in the provisioning kernel (in my case, ext4).
- Running `nodprobe ext4` at the `portable11` command prompt and re-running the `mount` command above caused the filesystem to mount.
- Warewulf fix is to:

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

x

How it got fixed (part 2)

- What was different? A missing filesystem module in the provisioning kernel (in my case, ext4).
- Running `modprobe ext4` at the `portshell` command prompt and re-running the `mount` command above caused the filesystem to mount.
- Warewulf fix is to:
 - `run echo modprobe += ext4 | sudo tee -a /etc/warewulf/bootstrap.conf`

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

x

How it got fixed (part 2)

- What was different? A missing filesystem module in the provisioning kernel (in my case, ext4).
- Running `nodprobe ext4` at the `portshell` command prompt and re-running the `mount` command above caused the filesystem to mount.
- Warewulf fix is to:
 - `run echo nodprobe += ext4 | sudo tee -a /etc/warewulf/bootstrap.conf`
 - `run sudo wbootstrap REBOOT_VERSION`

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ How it got fixed (part 2)

x

How it got fixed (part 2)

- ▶ What was different? A missing filesystem module in the provisioning kernel (in my case, ext4).
- ▶ Running `nodprobe ext4` at the `portshell` command prompt and re-running the `mount` command above caused the filesystem to mount.
- ▶ Warewulf fix is to:
 - ▶ `run echo nodprobe -- ext4 | sudo tee -a /etc/warewulf/bootstrap.conf`
 - ▶ `run sudo wbootstrap VERBEL_VERBUSTION`
 - ▶ reboot the node and try again.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Semi-stateful node provisioning

- └─ Make the necessary wwbootstrap changes, then reboot your nodes

x

Make the necessary wwbootstrap changes, then reboot your nodes

```
[user@node ~]$ echo modprobe += virtio_blk | \
sudo tee -a /etc/warewulf/bootstrap.conf
[user@node ~]$ echo modprobe += ext4 | \
sudo tee -a /etc/warewulf/bootstrap.conf
[user@node ~]$ sudo wwbootstrap $(uname -r)
[user@node ~]$ sudo ddah -w 'c[1-2].g[1-2]' reboot
```

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Semi-stateful node provisioning

└─ Final result on a compute node (part 2)

x

Final result on a compute node (part 2)

Consume 5 GB of space in /tmp (we only used 1 GiB previously), then allocate 5 GB for an array in memory:

```
[user@node ~]$ sudo mkfs.gl
[root@cl ~]# dd if=/dev/zero of=/tmp/foo bs=1M count=5120
5120+0 records in
5120+0 records out
5368709120 bytes (5.4 GB, 5.0 GiB) copied, 8.97811 s, 598 MB/s
[root@cl ~]# module load py3-numpy
[root@cl ~]# python3 -c \
'import numpy as np; x=np.full((25000, 25000), 1)'
[root@cl ~]# rm /tmp/foo
```

No Killed messages due to running out of memory. We're able to consume much more /tmp space and all practically the RAM without conflict.

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Decoupling kernels from the SMS

└─ Decoupling kernels from the SMS

Decoupling kernels from the SMS

- ▶ If you keep your HPC around for a long period, you might want/need to support different operating systems or releases.
- ▶ Maybe you need to run a few nodes on Rocky 8 while keeping the SMS on Rocky 9 (snkchroot supports that).
- ▶ Maybe you need to use a different kernel version for exotic hardware or new features, but don't want to risk the stability of your SMS.
- ▶ A simple `snkchroot $(uname -r)` won't do that.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Decoupling kernels from the SMS
 - └─ Decoupling kernels from the SMS

x

Decoupling kernels from the SMS

Check wbootstrap --help:

```
[user@node ~]$ wbootstrap --help
USAGE: /usr/bin/wbootstrap [options] kernel_version
...
  OPTIONS:
    -C, --chroot    Look into this chroot directory to find
                    the kernel
...

```

So if we install a kernel into the `$(CHROOT)` like any other package, we can bootstrap from it instead of the SMS kernel.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Decoupling kernels from the SMS

- └─ Install a different kernel into the CHROOT, bootstrap it

Install a different kernel into the CHROOT, bootstrap it

```
[user@node ~]$ sudo yum -y install --installroot=$CHROOT kernel
...
Installing:
kernel x86_64 5.14.0-427.24.1.el9_4 ...
...
Complete!
[user@node ~]$ sudo evmbootstrap --chroot=${CHROOT} \
  5.14.0-427.24.1.el9_4.x86_64
Number of drivers included in bootstrap: 880
...
Bootstrap image '5.14.0-427.24.1.el9_4.x86_64' is ready
Done.
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Decoupling kernels from the SMS
 - └─ Check your nodes' provisioning summary

x

[Check your nodes' provisioning summary](#)

```
[user@node ~]$ wvsh provision list
NODE      VMFS      BOOTSTRAP  ...
-----
c1         rocky9.4   6.1.07~1.el9.elrep...
c2         rocky9.4   6.1.07~1.el9.elrep...
login      rocky9.4   6.1.07~1.el9.elrep...
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Decoupling kernels from the SMS

- └─ Change the default kernel for nodes, reboot them.

Change the default kernel for nodes, reboot them.

```
[user@node ~]$ sudo wush provision set 'e' \  
--bootstrap=5.14.0-427.24.1.el9_4.x86_64  
Are you sure you want to make the following changes to 5  
node(s):  
  
SET: BOOTSTRAP           = 5.14.0-427.24.1.el9_4.x86_64  
  
Yes/No? y  
[user@node ~]$ sudo scontrol reboot ASAP nextstate=RESUME \  
c[1-2]  
[user@node ~]$ sudo pdksh -w 'g[1-2].login' reboot
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes
 - └─ Decoupling kernels from the SMS
 - └─ Verify everything came back up

x

Verify everything came back up

```
[user@node ~]$ sudo pssh -w 'c[1-2].login' uname -r \
| sort
c1: 5.14.0-427.24.1.el9_4.x86_64
c2: 5.14.0-427.24.1.el9_4.x86_64
login: 5.14.0-427.24.1.el9_4.x86_64
[user@node ~]$ minfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
normal*    up 1-00:00:00      2   idle c[1-2]
```

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ Management of GPU drivers

x

Unfortunately, we can't do this one as a live in-class exercise.

- ▶ Jetstream2 uses NVIDIA GRID to split up GPUs
- ▶ GRID drivers are proprietary and the license doesn't allow redistribution
- ▶ Typical bare-metal drivers that **can** be redistributed don't work with GRID

So instead, we'll show you how we do this on a bare-metal installation of OpenHPC 2 and Rocky 8. None of the steps change for OpenHPC 3 or Rocky 9.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ See what we have

See what we have

```
[renfro2@nna ~]$ sudo sh gpunode002 lspci | grep -i nvidia
05:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
06:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
84:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
85:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
[renfro2@nna ~]$ sudo sh gpunode002 nvidia-smi | grep Driver
NVIDIA-SMI 470.199.02 Driver Version: 470.199.02 ...
4.18.0-513.24.1.el8_9.x86_64
[renfro2@nna ~]$ XY=$(sudo sh gpunode002 "uname -r")
```

On a system that's never had NVIDIA drivers installed, the `nvidia-smi` command will return a command not found.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ Download the driver

[Download the driver](#)

```
[renfro2@nms ~]$ NV=470.256.02
[renfro2@nms ~]$ BASEURL=https://us.download.nvidia.com/tesla
[renfro2@nms ~]$ wget \
$(BASEURL)/$(NV)/NVIDIA-Linux-x86_64-$(NV).run
...
... 'NVIDIA-Linux-x86_64-470.256.02.run' saved ...
[renfro2@nms ~]$
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ Prepare to install the driver

Prepare to install the driver

```
[renfr2@node ~]$ sudo install -o root -g root -m 0755 \
NVIDIA-Linux-x86_64-${NV}.run ${CHROOT}/root
[renfr2@node ~]$ sudo mount -o rw,bind /proc ${CHROOT}/proc
[renfr2@node ~]$ sudo mount -o rw,bind /dev ${CHROOT}/dev
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

└─ Making better compute nodes

└─ Management of GPU drivers

└─ Install the driver, clean up, update VNFS

x

Install the driver, clean up, update VNFS

```
[username ~]$ sudo chroot ${CHROOT} \  
/root/NVIDIA-Linux-x86_64-${KV}.run --disable-nouveau \  
--kernel-name=${KV} --no-drm --run-utility-xconfig --silent
```

You'll get up to five harmless warnings from this:

1. You do not appear to have an NVIDIA GPU supported by....
2. One or more modprobe configuration files to disable Nouveau are already present at...
3. The nvidia-drm module will not be installed
4. nvidia-installer was forced to guess the X library path
5. Unable to determine the path to install the libglvnd

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ Clean up, update the VNFS, reboot

Clean up, update the VNFS, reboot

```
[renfro2@nna ~]$ sudo rm \
${CHROOT}/root/NVIDIA-Linux-x86_64-${NV}.run
[renfro2@nna ~]$ sudo rmroot ${CHROOT}/proc ${CHROOT}/dev
[renfro2@nna ~]$ sudo wvsnfs --chroot=${CHROOT}
[renfro2@nna ~]$ vvvah provision print gpunode002 | \
egrep -i 'bootstrapvnfs'
gpunode002: BOOTSTRAP      = 4.19.0-013.24.1.el9_9.x86_64
gpunode002: VNFS           = rocky-8-k80
[renfro2@nna ~]$ sudo nsh gpunode002 reboot
```

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Making better compute nodes

- └─ Management of GPU drivers

- └─ Wait for the reboot and provision, check versions

Wait for the reboot and provision, check versions

```
[renfro2@node ~]$ sudo sash gpunode002 uptime
15:11:05 up 1 min, 0 users, load average: 1.56, 0.51, 0.18
[renfro2@node ~]$ sudo sash gpunode002 lspci | grep -i nvidia
05:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
06:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
84:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
85:00.0 ... NVIDIA Corporation GK210GL [Tesla K80] (rev a1)
[renfro2@node ~]$ sudo sash gpunode002 nvidia-smi | grep Driver
| NVIDIA-SMI 470.256.02 Driver Version: 470.256.02 ...
```

x

OpenHPC: Beyond the Install Guide

- └─ Managing system complexity
 - └─ Configuration settings for different node types
 - └─ Configuration settings for different node types

x

What tools have we used so far to define node settings?

1. `vvsh` node for node name and network information (MACs, IPs, provisioning interface)
2. `vvsh` provision for VMFS, kernel, kernel parameters, files
3. When the files include `systemd` services, other options become possible via `Condition` or similar statements

Manually building these up over time and storing the results in the Warewulf database may be tedious to review, and we might want to easily port our setup to a dev/test environment, a new version of OpenHPC, etc.

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Managing system complexity
 - └─ Automation for Warewulf3 provisioning
 - └─ Automation for Warewulf3 provisioning

Automation for Warewulf3 provisioning

Any kind of automation, scripting, or orchestration is beneficial for managing cluster settings:

- ▶ shell scripts,
- ▶ Python scripts,
- ▶ Ansible playbooks,
- ▶ Puppet manifests,
- ▶ etc.

Mike's used Ansible as part of the Basic Cluster project, Tim's `obpc-jetstream2` repository does the same. TN Tech uses Python scripts for their Warewulf management.

x

2024-07-17

OpenHPC: Beyond the Install Guide

- └─ Configuring Slurm policies

- └─ Configuring Slurm policies

[Configuring Slurm policies](#)

Can adapt a lot of Mike's CaRCC Emerging Centers talk from a couple years ago for this. Fair share, hard limits on resource consumption, QOSes for limiting number of GPU jobs or similar.

x