Midterm Exam Distribution

Mean: 85

Median: 88

Check my addition!

Questions most missed

Q. What does a bootloader such as GRUB2 do?

A. Starts a kernel. (might present the user with a menu of kernel options to choose from)

Q. Why is TCP more robust than UDP

A. TCP checks that packets arrive at their destination

Q. Fix this command: srun hostname --nodes 2 ...

A. The program to run goes at the end of the command

srun --nodes 2 ... hostname

Plan

- The Homework 4 team assignment will be live on Friday.
- Your team has 1 week to setup your cluster following the slides.
- We will **NOT** wipe the servers 1- 7.
- We **WILL** wipe servers 9-16.
- First set all the iDRAC interface passwords and root passwords to “GobbledegookCS4!591” on your OLD servers.
- Team leads login to your new Team server, change the root password and setup user accounts for your team mates.
- Undergraduates login to iDRAC on your new *head node* (129.24.245.teamID+17) and your new *compute node* (129.24.245.teamID+25) and change the password.
- Team leads and undergraduates share the new iDRAC and head node root passwords in your team slack channel.
- Team leads work with your team mates to set their user account passwords.
- Team members login to the iDRAC interface for your head node
- Ryan the project assistant will meet with each team to check progress and help you. He has office hours and has posted a booking link in Slack under #hw4.

Give your users sudo access the right way this time

```
[root@localhost ~]# usermod --append -G wheel matthew
[root@localhost ~]# sudo su - matthew
Last login: Sun Oct 20 18:48:09 MDT 2024 from 50.193.238.94 on pts/1
[matthew@localhost ~]$ sudo -i
[sudo] password for matthew:
```

Add user to the wheel group (because then they are a big wheel) and then test that the user can use sudo commands.

Lecture 14: Warewulf

Installation tools (yum and dnf)

- Dnf is the successor to yum.
- So in my examples why do I use “yum”?

```
[matthew@moonshine ~]$ ls -l $(which dnf)
lrwxrwxrwx. 1 root root 5 Oct 31 20:53 /usr/bin/dnf -> dnf-3

[matthew@moonshine ~]$ ls -l $(which yum)
lrwxrwxrwx. 1 root root 5 Oct 31 20:53 /usr/bin/yum -> dnf-3
```

Goal Configuration

Public Network

129.24.245.0

eno1
129.24.245.x
TCP/IP

Head Node

Services:
SLURM
WAREWULF

Goal Configuration

Public IP Network

129.24.245.0

Head Node

eno1
129.24.245.x
TCP/IP

eno2
10.0.0.1
TCP/IP

ibp65s0
RDMA
Infiniband

Services:
SLURM
WAREWULF

Private* IP Network

192.168.1.0

Compute Node 1

eno1
10.0.0.2
TCP/IP

ibp65s0
RDMA
Infiniband

*Non-routable

Configuring a cluster

- Setup the head node with Linux
- Install the services we need to support the cluster on the head node
- Setup an external network interface that leads to the internet on the head node
- Setup two internal network interface that leads to the compute nodes (an ethernet admin network and a Infiniband high speed network)
- Configure a disk image containing Linux that’s stored on the head node.
- Configure the compute node to boot using a disk image it gets from the head node over the network

Services we need

- Trivial File Transfer service
  - This will serve the disk boot image to the compute nodes.
- Network File Server
  - After the compute node boots this is how it will access files stored on the head nodes.
- DHCP
  - Provides an initial IP address and network configuration to the compute nodes.
- Warewulf
  - The warwulf service provides tools for remote management of the compute nodes.

Warewulf “provisioning” process. Assigning IP to compute node.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.y

Private static IP
10.x.x.x

Compute node

PXE BIOS

No IP

iDRAC

Public static IP
129.24.245.z

Warewulf “provisioning” process. Assigning IP to compute node.

Warewulf “provisioning” process. Assigning IP to compute node.

Warewulf “provisioning” process.
Getting boot image.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.x

Private static IP
10.x.x.x

Compute node

PXE
BIOS

Private
dynamic IP
10.0.1.x

iDRAC

Public static IP
129.24.245.x

Warewulf “provisioning” process. Getting boot image.

Warewulf “provisioning” process. Getting boot image.

Warewulf “provisioning” process.
NFS sharing.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.y

Private static IP
10.x.x.x

Compute node

Linux

Private
static IP
10.0.0.x

iDRAC

Public static IP
129.24.245.z

Warewulf “provisioning” process.
NFS sharing.

Warewulf “provisioning” process.
NFS sharing.

Configure Internal Network Interface

- Decide who on your team will perform the installations.
- Make sure everyone on the team understands what is going on.

Configure Internal Network Interface

```
[root@moonshine warewulf]# nmtui
```

Configure Internal Admin Network Interface

[root@moonshine warewulf]# nmtui

```
Edit Connection

        Profile name Internal
              Device eno2 (D4:AE:52:A0:DE:54)

= ETHERNET                                                    <Show>
= 802.1X SECURITY                                             <Show>

= IPv4 CONFIGURATION <Manual>                                 <Hide>
          Addresses 10.0.0.1/16                <Remove>
                    <Add...>
            Gateway 129.24.245.8
        DNS servers <Add...>
      Search domains <Add...>

            Routing (No custom routes) <Edit...>
  [ ] Never use this network for default route
  [ ] Ignore automatically obtained routes
  [ ] Ignore automatically obtained DNS parameters

  [ ] Require IPv4 addressing for this connection

= IPv6 CONFIGURATION <Automatic>                              <Show>

[ ] Automatically connect
[X] Available to all users
```

Internal Interface IP
/16 means 255.255.0.0

External Interface IP
for ***your*** cluster

Configure Internal Network Interface

```
Edit Connection

= ETHERNET                                                  <Show>
= 802.1X SECURITY                                           <Show>

= IPv4 CONFIGURATION <Manual>                               <Hide>
        Addresses 10.0.0.1/16              <Remove>
                  <Add...>
          Gateway 129.24.245.8
      DNS servers <Add...>
   Search domains <Add...>

          Routing (No custom routes) <Edit...>
  [ ] Never use this network for default route
  [ ] Ignore automatically obtained routes
  [ ] Ignore automatically obtained DNS parameters

  [ ] Require IPv4 addressing for this connection

= IPv6 CONFIGURATION <Automatic>                            <Show>

[ ] Automatically connect
[X] Available to all users

                                                  <Cancel> <OK>
```

Make sure to save

Configure Internal Network Interface

matthew — matthew@moonshine:~ — ssh moonshine — 88×23

```
Edit Connection

        Addresses 10.0.0.1/16                <Remove>
                  <Add...>
          Gateway 129.24.245.8
      DNS servers
   Search domains <Add...>

          Routing (No custom routes) <Edit...>
[X] Never use this network for default route
[ ] Ignore automatically obtained routes
[ ] Ignore automatically obtained DNS parameters

[ ] Require IPv4 addressing for this connection

= IPv6 CONFIGURATION <Automatic>                          <Show>

[X] Automatically connect
[X] Available to all users
```

We don't want to use this interface to get to the internet

Configure Internal Network Interface

matthew — matthew@moonshine:~ — ssh moonshine — 88×23

```
Edit Connection

        Addresses 10.0.0.1/16                <Remove>
                  <Add...>
          Gateway 129.24.245.8
      DNS servers <Add...>
   Search domains <Add...>

          Routing (No custom routes) <Edit...>
[X] Never use this network for default route
[ ] Ignore automatically obtained routes
[ ] Ignore automatically obtained DNS parameters

[ ] Require IPv4 addressing for this connection
```

We do want this interface to come up on boot

```
= IPv4 CONFIGURATION <Automatic>                    <Show>

[X] Automatically connect
[X] Available to all users
```

Configure Internal Network Interface

Configure Internal Network Interface

```
[matthew@moonshine ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         summit.carc.unm 0.0.0.0         UG    100    0        0 eno1
10.0.0.0        0.0.0.0         255.255.0.0     U     101    0        0 eno2
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
```

Now we have a route to the internal network (10.0.0.0)

Configure Internal Network Interface

```
[matthew@moonshine ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         summit.carc.unm 0.0.0.0         UG    100    0        0 eno1
10.0.0.0        0.0.0.0         255.255.0.0     U     101    0        0 eno2
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
```

These 0.0.0.0 entries means no gateway address is needed. The corresponding destination network is connected directly to the network card.

Configure Internal Network Interface

```
[matthew@moonshine ~]$ route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         summit.carc.unm 0.0.0.0         UG    100    0        0 eno1
10.0.0.0        0.0.0.0         255.255.0.0     U     101    0        0 eno2
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
129.24.244.0    0.0.0.0         255.255.252.0   U     100    0        0 eno1
```

The default destination with IP mask 0.0.0.0 matches all IP addresses.

When deciding which interface to use to get to a destination, the genmask with the most networks are chosen first, then if there is a tie the metric value is used.

So, 0.0.0.0 is always matched last and 255.255.255.255 would always match first. 255.255.0.0 beats 0.0.0.0 so packets are sent to eno2 if the destination is 10.0.0.0 even though 0.0.0.0 also matches.

Warewulf Installation

- Decide who on your team will perform the installations.
- Make sure everyone on the team understands what is going on.

```
[matthew@moonshine ~]$ sudo yum groupinstall "Development Tools"
[sudo] password for matthew:
Last metadata expiration check: 0:14:04 ago on Tue 19 Mar 2024 07:53:26 PM CDT.
Dependencies resolved
<snip>
Complete!
```

Warewulf Installation

```
[matthew@moonshine ~]$ sudo yum install epel-release

[matthew@moonshine ~]$ sudo yum config-manager --set-enabled crb
```

Enable the Code Builder repository. This is a repository of tools developed by RedHat employees.

Warewulf Installation

```
[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server
nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs
```

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

golang: programming language developed by Google

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

TFTP: trivial file transfer protocol server

Warewulf Installation

```
[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server
nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs
```

DHCP: dynamic host configuration protocol server

(Review DHCP from the Networking lecture)

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

NFS: Network File System server

(Review NFS from Filesystems lecture)

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

GPG: GNU Privacy Guard (Encryption)

Devel after a package name means the package provides headers for developers to use.

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

Assuan: Package that allows processes to talk to one another through GPG (it’s purpose is to prevent front end server developers from accidently exposing encrypted data)

Warewulf Installation

[matthew@moonshine ~]$ sudo yum install golang tftp-server dhcp-server nfs-utils gpgme-devel libassuan-devel ipxe-bootimgs

ipxe boot images: Provides some standard PXE bootloaders

Warewulf Installation

```
[matthew@moonshine ~]$ sudo -i
[root@moonshine ~]# git clone https://github.com/hpcng/warewulf.git
Cloning into 'warewulf'...
remote: Enumerating objects: 18703, done.
remote: Counting objects: 100% (659/659), done.
remote: Compressing objects: 100% (356/356), done.
remote: Total 18703 (delta 341), reused 521 (delta 278), pack-reused 18044
Receiving objects: 100% (18703/18703), 24.07 MiB | 4.90 MiB/s, done.
Resolving deltas: 100% (10004/10004), done.
```

HPCng: High Performance Computing Next Generation is an open community of people and organizations interested in the broad modernization of HPC capabilities.

Warewulf Installation

Interactive sudo.

```
[matthew@moonshine ~]$ sudo -i
[root@moonshine ~]# git clone https://github.com/hpcng/warewulf.git
Cloning into 'warewulf'...
remote: Enumerating objects: 18703, done.
remote: Counting objects: 100% (659/659), done.
remote: Compressing objects: 100% (356/356), done.
remote: Total 18703 (delta 341), reused 521 (delta 278), pack-reused 18044
Receiving objects: 100% (18703/18703), 24.07 MiB | 4.90 MiB/s, done.
Resolving deltas: 100% (10004/10004), done.
```

HPCng: High Performance Computing Next Generation is an open community of people and organizations interested in the broad modernization of HPC capabilities.

Warewulf Installation

```
[root@moonshine ~]# cd warewulf
[root@moonshine ~]# git checkout tags/v4.5.8
[root@moonshine warewulf]# make clean defaults \
 PREFIX=/usr \
 BINDIR=/usr/bin \
 SYSCONFDIR=/etc \
 DATADIR=/usr/share \
 LOCALSTATEDIR=/var/lib \
 SHAREDSTATEDIR=/var/lib \
 MANDIR=/usr/share/man \
 INFODIR=/usr/share/info \
 DOCDIR=/usr/share/doc \
 SRVDIR=/var/lib \
 TFTPDIR=/var/lib/tftpboot \
 SYSTEMDDIR=/usr/lib/systemd/system \
 BASHCOMPDIR=/etc/bash_completion.d/ \
 FIREWALLDDIR=/usr/lib/firewalld/services \
 WWCLIENTDIR=/warewulf
```

Protip: Copy and paste!

Protip 2: Be careful of Unicode.

Warewulf Installation

```
[root@moonshine ~]# cd warewulf
[root@moonshine ~]# git checkout tags/v4.5.8
[root@moonshine warewulf]# make clean defaults
<snip>
printf " OS := rocky\n WAREWULF := warewulf\n VERSION := 4.5.0rc3+453.g1ca2dd1d\n RELEASE := 1\n PREFIX :=
/usr\n BINDIR := /usr/bin\n SYSCONFDIR := /etc\n SRVDIR := /var/lib\n DATADIR := /usr/share\n MANDIR :=
/usr/share/man\n DOCDIR := /usr/share/doc\n LOCALSTATEDIR := /var/lib\n RELEASE := 1\n TFTPDIR :=
/var/lib/tftpboot\n FIREWALLDDIR := /usr/lib/firewalld/services\n SYSTEMDDIR := /usr/lib/systemd/system\n
BASHCOMPDIR := /etc/bash_completion.d/\n LOGROTATEDIR := /etc/logrotate.d\n DRACUTMODDIR :=
/usr/lib/dracut/modules.d\n WWCLIENTDIR := /warewulf\n WWCONFIGDIR := /etc/warewulf\n WWPROVISIONDIR
:= /var/lib/warewulf/provision\n WWOVERLAYDIR := /var/lib/warewulf/overlays\n WWCHROOTDIR :=
/var/lib/warewulf/chroots\n WWTFTPDIR := /var/lib/tftpboot/warewulf\n WWDOCDIR :=
/usr/share/doc/warewulf\n WWDATADIR := /usr/share/warewulf\n IPXESOURCE := /usr/share/ipxe\n" >Defaults.m
```

Protip: Check make picked up the right variables.

Warewulf Installation

```
[matthew@moonshine warewulf]$ make all

[matthew@moonshine warewulf]$ make install
```

Configure Warewulf – it will configure the other services for us

```
[root@moonshine warewulf]# less /etc/warewulf/warewulf.conf
```

```
dhcp:
  enabled: true
  template: default
  range start: 10.0.1.1
  range end: 10.0.1.255
  systemd name: dhcpd
```

When a compute node asks for an IP the DHCP server will reply with one from this range

dhpcd will be configured to serve IP addresses on the interface that matches the first IP in the range.
10.0.0.1/16 matches 10.0.1.1

Configure Warewulf – it will configure the other services for us

```
[root@moonshine warewulf]# less /etc/warewulf/warewulf.conf
```

```
tftp:
  enabled: true
  tftproot: /var/lib/tftpboot
  systemd name: tftp
  ipxe:
    "00:00": undionly.kpxe
    "00:07": ipxe-snponly-x86_64.efi
    "00:09": ipxe-snponly-x86_64.efi
    00:0B: arm64-efi/snponly.efi
```

Where disk images will be stored on the head node.

An MBR bootloader image.

Some efi boot loader images.

The compute node will use the PXE network boot protocol to boot off the disk image. PXE boot is built into HPC network card firmware and the UEFI BIOS.

Configure Warewulf – it will configure the other services for us

```
[root@moonshine warewulf]# less /etc/warewulf/warewulf.conf
```

```
nfs:
  enabled: true
  export paths:
  - path: /home
    export options: rw,sync
    mount options: defaults
    mount: true
  - path: /opt
    export options: ro,sync,no_root_squash
    mount options: defaults
    mount: false
  systemd name: nfs-server
```

The ”home” and “opt*” directories will be exported over the network to the compute nodes.

NFS server options. This is where “optional” software is installed.

Enable Systemd Warewulf unit so it starts on boot

```
[root@moonshine warewulf]# systemctl enable --now warewulfd
Created symlink /etc/systemd/system/multi-user.target.wants/warewulfd.service →
/usr/lib/systemd/system/warewulfd.service.
```

The –-now argument says start the daemon too

Check for startup errors

```
[root@moonshine warewulf]# systemctl status warewulfd
● warewulfd.service - Warewulf cluster management daemon
     Loaded: loaded (/usr/lib/systemd/system/warewulfd.service; enabled; preset: disabled)
     Active: active (running) since Wed 2024-03-20 00:01:05 CDT; 2min 50s ago
       Docs: https://warewulf.org/
   Main PID: 212040 (wwctl)
      Tasks: 11 (limit: 407887)
     Memory: 39.8M
        CPU: 59ms
     CGroup: /system.slice/warewulfd.service
             └─212040 /usr/bin/wwctl server start

Mar 20 00:01:05 moonshine systemd[1]: Started Warewulf cluster management daemon.
Mar 20 00:01:05 moonshine wwctl[212029]: SERV   : Started Warewulf (4.5.x-1.git_2805122c) server at PID: 212040
```

Warewulf will now configure the other services for us

```
[root@moonshine warewulf]# wwctl configure dhcp
Building overlay for moonshine: host
Enabling and restarting the DHCP services
Created symlink /etc/systemd/system/multi-user.target.wants/dhcpd.service →
/usr/lib/systemd/system/dhcpd.service.
```

Warewulf will now configure the other services for us

```
[root@moonshine warewulf]# systemctl status dhcpd
● dhcpd.service - DHCPv4 Server Daemon
   Loaded: loaded (/usr/lib/systemd/system/dhcpd.service; enabled; preset: disabled)
   Active: active (running) since Wed 2024-03-20 00:06:10 CDT; 43s ago
     Docs: man:dhcpd(8)
           man:dhcpd.conf(5)
 Main PID: 212079 (dhcpd)
   Status: "Dispatching packets..."
    Tasks: 1 (limit: 407887)
   Memory: 10.2M
      CPU: 16ms
   CGroup: /system.slice/dhcpd.service
           └─212079 /usr/sbin/dhcpd -f -cf /etc/dhcp/dhcpd.conf -user dhcpd -group dhcpd -no

Mar 20 00:06:10 moonshine dhcpd[212079]:
Mar 20 00:06:10 moonshine dhcpd[212079]: No subnet declaration for eno1 (129.24.245.8).
Mar 20 00:06:10 moonshine dhcpd[212079]: ** Ignoring requests on eno1.  If this is not what
Mar 20 00:06:10 moonshine dhcpd[212079]:    you want, please write a subnet declaration
Mar 20 00:06:10 moonshine dhcpd[212079]:    in your dhcpd.conf file for the network segment
Mar 20 00:06:10 moonshine dhcpd[212079]:    to which interface eno1 is attached. **
Mar 20 00:06:10 moonshine dhcpd[212079]:
Mar 20 00:06:10 moonshine dhcpd[212079]: Sending on   Socket/fallback/fallback-net
Mar 20 00:06:10 moonshine dhcpd[212079]: Server starting service.
Mar 20 00:06:10 moonshine systemd[1]: Started DHCPv4 Server Daemon.
```

Why not? And Why is this good?

Warewulf will now configure the other services for us

```
[root@moonshine warewulf]# wwctl configure tftp
Writing PXE files to: /var/lib/tftpboot/warewulf
ERROR  : Could not open source file /usr/share/ipxe/arm64-efi/snponly.efi: open /usr/share/ipxe/arm64-efi/snponly.efi:
no such file or directory
WARN   : ipxe binary could not be copied, booting may not work: open /usr/share/ipxe/arm64-efi/snponly.efi: no such
file or directory
Enabling and restarting the TFTP services
```

This copies the bootloaders that PXE boot will use (they came from installing pxe-bootimgs). We don’t care that the ARM processor bootloaders are missing.

Warewulf will now configure the other services for us

```
[root@moonshine warewulf]# wwctl configure nfs
Building overlay for moonshine: host
Enabling and restarting the NFS services
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-server.service → /usr/lib/systemd/system/nfs-server.service.
```

Warewulf will now configure the other services for us

```
[root@moonshine warewulf]# systemctl status nfs-server
● nfs-server.service - NFS server and services
     Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; preset: disabled)
    Drop-In: /run/systemd/generator/nfs-server.service.d
             └─order-with-mounts.conf
     Active: active (exited) since Wed 2024-03-20 00:16:39 CDT; 47s ago
   Main PID: 212194 (code=exited, status=0/SUCCESS)
        CPU: 24ms


Mar 20 00:16:38 moonshine systemd[1]: Starting NFS server and services...
Mar 20 00:16:39 moonshine systemd[1]: Finished NFS server and services.
```

Check that the nfs-server is exporting the desired directories

```
[root@moonshine warewulf]# cat /etc/exports


# This file is autogenerated by warewulf
# Host:   moonshine
# Time:   03-20-2024 00:16:38 CDT
# Source: /var/lib/warewulf/overlays/host/rootfs/etc/exports.ww
/home 10.0.0.0/255.255.252.0(rw,sync)
/opt 10.0.0.0/255.255.252.0(ro,sync,no_root_squash)
```

The /opt and /home directories are being shared and can be accessed by machines with IP addresses from the 10.0.0.0/24 subnet.

Hmm this is a problem! Our DHCP range was 10.0.1.0 - 10.0.1.255.

We are accumulating a lot of IP addresses here! It’s getting confusing.

Warewulf “provisioning” process. Assigning IP to compute node.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.y

Private static IP
10.x.x.x

Compute node

PXE BIOS

No IP

iDRAC

Public static IP
129.24.245.z

Revisited

Warewulf “provisioning” process. Assigning IP to compute node.

Warewulf “provisioning” process. Assigning IP to compute node.

Warewulf “provisioning” process.
Getting boot image.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.x

Private static IP
10.x.x.x

Compute node

PXE
BIOS

Private
dynamic IP
10.0.1.x

iDRAC

Public static IP
129.24.245.x

Warewulf “provisioning” process. Getting boot image.

Warewulf “provisioning” process. Getting boot image.

Warewulf “provisioning” process.
NFS sharing.

Head node

0

iDRAC

Public static IP
129.24.245.x

Linux

Public static IP
129.24.245.y

Private static IP
10.x.x.x

Compute node

Linux

Private
static IP
10.0.0.x

iDRAC

Public static IP
129.24.245.z

Warewulf “provisioning” process.
NFS sharing.

Warewulf “provisioning” process.
NFS sharing.

Configure ssh keys for the cluster so services and users can access compute nodes without having to enter a password.

```
[root@moonshine warewulf]# wwctl configure ssh
Updating system keys
Setting up key: ssh_host_rsa_key
Setting up key: ssh_host_dsa_key
Setting up key: ssh_host_ecdsa_key
Setting up key: ssh_host_ed25519_key
Setting up: /root/.ssh/authorized_keys
[root@moonshine warewulf]#

[root@moonshine ~]# cat /root/.ssh/authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC3dqNIErTAMN8+nT4ejn3OTnltq7sc+BqJpvmpgw3+o7KUnbPMJUMpyZeqvrGlC6
pT0MX7Wogd0Iszcsv6G2oyafnRJA4Q4gmRxGiQaIAG9kDpTczfsGYdcEDZEOgXN1rwTn4/rGUZcL9bS2YlXg94rDKDhEjM5Vbf
YLmE4UQnAMeppyoe4vPxuxzbb5wLiCkigmFd7bQXauH9j7/HbtdcFmTOyQoyRA7usYXr/ervFfSPCsRbzIw+Yv9872ALFqVGA
209xqM8iNz7jUY1EsgUerL5hMWDqqjYvJC2UmdyIxnuFUl50T6KpXOAG+oKHbOhCReIm4MM/vOus0GR4XjcswDejfHzMNT
QWmpSGFPhrzIU3Jj24wfDap3Srls0fnTPFMlolFsVkv8HIP1Wl46n2vgkRpEEXbSqCd+9v5Mj5SOo/7k9WhKQ4REEWOaVgJ0f
N6gDxy6J0bkBlVnyd/Lw5FHDK5Vz75Ht9jB8QngoulZB9N9KnjQdkLCPN+zIpkE=
```

Download a Rocky 9 Linux Boot Image

```
[root@moonshine ~]# wwctl container import docker://ghcr.io/hpcng/warewulf-rockylinux:9 rocky-9
Copying blob 489e1be6ce56 done
Copying blob 280c15a49d01 done
Copying blob 39f01640b517 done
Copying blob 99b4942e1205 done
Copying config 46aa9f0cce done
Writing manifest to image destination
Storing signatures
2024/03/20 00:31:37  info unpack layer: sha256:489e1be6ce56f590a5a31bdf814671cac006421930c1175cb62e1763bf51a3f9
2024/03/20 00:31:40  info unpack layer: sha256:280c15a49d01a6159a231e325ada76e79d9d972bd128bb0abe4d8b80bba4fbb5
2024/03/20 00:31:50  info unpack layer: sha256:39f01640b5175b07c8525a1dbfe980293b64b39ad7a76c146d5f189ba9f830b3
2024/03/20 00:31:50  info unpack layer: sha256:99b4942e1205be66808b8588de5f81b1a46957d85c51101972f01bfed05e66cd
uid/gid not synced: run `wwctl container syncuser --write rocky-9`
```

Containers are whole filesystems and possibly Linux installations stored in a single file. With Docker you can run these operating systems on top of the host operating system. For now we are just using it as a file from which our compute node will boot using PXE.

We can modify this boot image – for example we can write the headnode's user information into the image so when the compute node boots from the image it knows about the users already.

```
[root@moonshine ~]# wwctl container syncuser --write rocky-9 --build
uid/gid synced for container rocky-9
Created image for VNFS container rocky-9: /var/lib/warewulf/provision/container/rocky-9.img
Compressed image for VNFS container rocky-9: /var/lib/warewulf/provision/container/rocky-9.img.gz
```

Now we set the “default” warewulf compute node profile to use the Rocky 9 image we created.

```
[root@moonshine ~]# wwctl profile set --yes --container rocky-9 "default”
```

Print the warewulf profile to make sure the Container Name is set correctly.

```
[root@moonshine ~]# sudo wwctl profile list -a
 PROFILE  FIELD         VALUE
 default  Id            default
 default  Comment       This profile is automatically included for each node
 default  ContainerName  rocky-9
```

Disable selinux and reboot.

Note

If you just installed the system fresh and have SELinux enforcing, you may need to reboot the system at this stage to properly set the contexts of the TFTP contents. After rebooting, you might also need to run `$ sudo restorecon -Rv /var/lib/tftpboot/` if there are errors with TFTP still.

Set SELINUX=disabled in /etc/selinux/config

Setting up a Compute Node

- Now we are ready to move on from a single server to building a cluster.
- We will start by building a cluster with just two nodes, a head node (which you have already configured) and a compute node.
- The tools we use here could just as well be used to build a cluster with 1000 compute nodes.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setup IP routing on the head node so the compute node can use it to reach the internet.

1) Enable Kernel IP routing

```
[matthew@moonshine ~]$ cat /proc/sys/net/ipv4/ip_forward
0
[matthew@moonshine ~]$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[matthew@moonshine ~]$
```

This zero means the kernel doesn't allow packets that arrive on one interface to leave on another.

sysctl is a tool for modifying kernel settings

Setup IP routing on the head node so the compute node can use it to reach the internet.

1) Enable Kernel IP routing

```
[matthew@moonshine ~]$ cat /proc/sys/net/ipv4/ip_forward
0
[matthew@moonshine ~]$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[matthew@moonshine ~]$
```

We enable that capability so our compute node can use this head node as an IP gateway.

Setup IP routing on the head node so the compute node can use it to reach the internet.

1) Enable Kernel IP routing

Edit /etc/sysctl.conf

matthew — matthew@moonshine:~ — ssh moonshine — 88×23

```
File Edit Options Buffers Tools Conf Help
# sysctl settings are defined through files in
# /usr/lib/sysctl.d/, /run/sysctl.d/, and /etc/sysctl.d/.
#
# Vendors settings live in /usr/lib/sysctl.d/.
# To override a whole file, create a new file with the same in
# /etc/sysctl.d/ and put new settings there. To override
# only specific settings, add a file with a lexically later
# name in /etc/sysctl.d/ and put new settings there.
#
# For more information, see sysctl.conf(5) and sysctl.d(5).
net.ipv4.ip_forward = 1
```

Add this line to make turning on IP forwarding permanent.

```
-UU-:**--F1  sysctl.conf    All L11    (Conf[Unix]) ----------------------------------
```

Setup IP routing on the head node so the compute node can use it to reach the internet.

1) Enable Kernel IP routing

```
[matthew@moonshine ~]$ sudo sysctl -p
net.ipv4.ip_forward = 1
```

Apply the contents of /etc/sysctl.conf immediately.

Setup IP routing on the head node so the compute node can use it to reach the internet.

2) Setup firewall rules

```
[matthew@moonshine ~]$ sudo firewall-cmd --zone=external --add-interface=eno1 --permanent
The interface is under control of NetworkManager, setting zone to 'external'.
Success
[matthew@moonshine ~]$ sudo firewall-cmd --zone=internal --add-interface=eno2 --permanent
The interface is under control of NetworkManager, setting zone to 'external'.
Success
```

Add firewall zones. An external one for the internet facing network interface (eno1) and an internal one for the compute node network interface (eno2).

Setup IP routing on the head node so the compute node can use it to reach the internet.

2) Setup firewall rules

```
[matthew@moonshine ~]$ sudo firewall-cmd --zone=external --add-interface=eno1 --permanent
The interface is under control of NetworkManager, setting zone to 'external'.
Success
[matthew@moonshine ~]$ sudo firewall-cmd --zone=internal --add-interface=eno2 --permanent
The interface is under control of NetworkManager, setting zone to ‘internal'.
Success
[matthew@moonshine ~]$ sudo firewall-cmd --set-default-zone=external
Success
```

Add firewall zones. An external one for the internet facing network interface (eno1) and an internal one for the compute node network interface (eno2).

Make the external interface the default firewall zone (all connections not in a zone use the external zone rules – they should be restrictive)

Setup IP routing on the head node so the compute node can use it to reach the internet.

2) Setup firewall rules

```
[matthew@moonshine ~]$ sudo firewall-cmd --new-policy internal-external --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --policy internal-external --add-ingress-zone=internal --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --policy internal-external --add-egress-zone=external --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --policy internal-external --set-target=ACCEPT --permanent
success
```

Setup a routing policy so the firewall can send packets from the internal zone to the internal zone and vice versa.

ACCEPT tells the firewall daemon to accept packets

Setup IP routing on the head node so the compute node can use it to reach the internet.

2) Setup firewall rules

```
[matthew@moonshine ~]$ sudo firewall-cmd --info-zone external
external (active)
  target: default
  icmp-block-inversion: no
  interfaces: eno1
  sources:
  services: ssh
  ports:
  protocols:
  forward: yes
  masquerade: yes
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

What did those commands actually do? They enabled something called IP Network Address Translation (NAT) Masquerading.

This is what makes private IP addresses so useful. NAT makes it so a single public IP address can pretend to be the address of thousands of computers with private addresses.

The firewall keeps track of which internal IP addresses were talking to computers on the external networks (including the internet – that way those external computers can reply to computers on the internal network)

This is how you can have multiple devices on your home network for example.

Since we are dealing with the firewall lets set the internal zone to allow the warewulf services we installed to send and receive data.

```
[matthew@moonshine ~]$ sudo firewall-cmd --zone internal --add-service warewulf --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --zone internal --add-service nfs --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --zone internal --add-service tftp --permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --zone internal --add-service dhcp –permanent
success
[matthew@moonshine ~]$ sudo firewall-cmd --reload
success
```

Add firewall exceptions

```
[root@moonshine warewulf]# firewall-cmd --list-all --zone internal
internal (active)
  target: default
  icmp-block-inversion: no
  interfaces: eno2
  sources:
  services: cockpit dhcp dhcpv6-client mdns nfs samba-client ssh tftp warewulf
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Check that the services are allowed by the firewall daemon on the internal zone.

Make triple sure this is the Compute node's iDRAC not the Head node!!!

Wipe your Compute Nodes Harddrive

Integrated Dell Remote Access Controller 7 | Enterprise | Support | Dell TechCen

System
PowerEdge R620
root , Admin

- Overview
  - Server
    - Logs
    - Power / Thermal
    - Virtual Console
    - Alerts
    - Setup
    - Troubleshooting
    - Licenses
    - Intrusion
  - iDRAC Settings
  - Hardware
  - Storage
    - Physical Disks
    - **Virtual Disks**
    - Controllers
    - Enclosures
  - Host OS

Virtual Disks

Properties | Create | **Manage** | Identify

Manage Virtual Disks

Jump To : Manage Virtual Disks

Manage Virtual Disks

Controller: PERC H310 Mini (Embedded)

Status	Name	RAID Level	Dedicated Hotspares	Virtual Disk Actions
✅	Virtual Disk 0	RAID-0	Not Applicable	✓ Action

- ✓ Action
- Delete
- Edit Policy: Read Cache
- Edit Policy: Write Cache
- Edit Policy: Disk Cache
- Initialize: Fast

lode: At Next Reboot | Cance

Make triple sure this is the Compute node's iDRAC not the Head node!!!

Wipe your Compute Nodes Harddrive

Integrated Dell Remote Access Controller 7 | Enterprise | Support | Dell TechCen

System
PowerEdge R620
root , Admin

- Overview
  - Server
    - Logs
    - Power / Thermal
    - Virtual Console
    - Alerts
    - Setup
    - Troubleshooting
    - Licenses
    - Intrusion
  - iDRAC Settings
  - Hardware
  - Storage
    - Physical Disks
    - Virtual Disks
    - Controllers
    - Enclosures
  - Host OS

Properties | Attached Media | vFlash | Service Module | **Job Queue**

Job Queue

		ID	Job	Status
+		JID_112460810334	Configure: RAID.Integrated.1-1	Scheduled (0%)
+		JID_053846261033	Check Update From Repo	Failed
+		JID_053845425754	Check Update From Repo	Failed

iDRAC adds the initialization request to a job queue.
It doesn’t happen until you reboot the node.

Configure the node to try PXE boot as the first boot option.

Uncheck this little box to make the change stick

Chat | Keyboard | Screen Capture | Refresh | Full Screen | Connect Virtual Media | Disconnect Viewer | Console

```
Initializing firmware interfaces...
Initialization complete.

Lifecycle Controller: Applying Updates or Setting System Configuration.
```

Restart your node... once it boots it reports that it is applying changes.

```
iPXE 1.0.0+ (4bd064de) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP HTTPS iSCSI TFTP VLAN AoE EFI Menu

net0: d4:ae:52:8b:72:8c using NII on NII-0000:01:00.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 d4:ae:52:8b:72:8c)................... ok
net0: 10.0.1.2/255.255.252.0
net0: fe80::d6ae:52ff:fe8b:728c/64
Next server: 10.0.0.1
Filename: http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c
http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c... ok
d4-ae-52-8b-72-8c : 333 bytes [script]

==============================================================================

Warewulf v4

MESSAGE: This node is unconfigured. Please have your system administrator add a
configuration for this node with HW address: d4:ae:52:8b:72:8c

Rebooting in 1 minute...
```

notice at this stage the IP address assigned to PXE is from the DHCP pool

```
iPXE 1.0.0+ (4bd064de) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP HTTPS iSCSI TFTP VLAN AoE EFI Menu

net0: d4:ae:52:8b:72:8c using NII on NII-0000:01:00.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 d4:ae:52:8b:72:8c)................... ok
net0: 10.0.1.2/255.255.252.0
net0: fe80::d6ae:52ff:fe8b:728c/64
Next server: 10.0.0.1
Filename: http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c
http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c... ok
d4-ae-52-8b-72-8c : 333 bytes [script]

==============================================================================

Warewulf v4

MESSAGE: This node is unconfigured. Please have your system administrator add a
configuration for this node with HW address: d4:ae:52:8b:72:8c

Rebooting in 1 minute...
```

And that the compute node trying to load from your head nodes internal IP address.

```
iPXE 1.0.0+ (4bd064de) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP HTTPS iSCSI TFTP VLAN AoE EFI Menu

net0: d4:ae:52:8b:72:8c using NII on NII-0000:01:00.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 d4:ae:52:8b:72:8c)................... ok
net0: 10.0.1.2/255.255.252.0
net0: fe80::d6ae:52ff:fe8b:728c/64
Next server: 10.0.0.1
Filename: http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c
http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c... ok
d4-ae-52-8b-72-8c : 333 bytes [script]

==============================================================================

Warewulf v4

MESSAGE: This node is unconfigured. Please have your system administrator add a
configuration for this node with HW address: d4:ae:52:8b:72:8c

Rebooting in 1 minute...
```

So far so good, but we haven’t added this node to warewulf yet – so it doesn’t get configured.

```
iPXE 1.0.0+ (4bd064de) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP HTTPS iSCSI TFTP VLAN AoE EFI Menu

net0: d4:ae:52:8b:72:8c using NII on NII-0000:01:00.0 (open)
  [Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 d4:ae:52:8b:72:8c)................... ok
net0: 10.0.1.2/255.255.252.0
net0: fe80::d6ae:52ff:fe8b:728c/64
Next server: 10.0.0.1
Filename: http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c
http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c... ok
d4-ae-52-8b-72-8c : 333 bytes [script]

==============================================================================

Warewulf v4

MESSAGE: This node is unconfigured. Please have your system administrator add a
configuration for this node with HW address: d4:ae:52:8b:72:8c

Rebooting in 1 minute...
```

Make a note of your node's MAC address (yours will be different). We will need it to add the node to warewulf.

129.24.245.33

idrac-HR3SLV1 - iDRAC7 - Network Devices | EL9 Quickstart (Rocky Linux and RHEL) — Warewulf User Guide development documentation | Warewulf 4 » ADMIN Magazine

Integrated Dell Remote Access Controller 7 | Enterprise | Support | Dell TechCenter | About | Logout

System
PowerEdge R620
root , Admin

- Overview
  - Server
    - Logs
    - Power / Thermal
    - Virtual Console
    - Alerts
    - Setup
    - Troubleshooting
    - Licenses
    - Intrusion
  - iDRAC Settings
  - Hardware
    - Batteries
    - Fans
    - CPU
    - Memory
    - Front Panel
    - **Network Devices**
    - Power Supplies
    - Removable Flash Media
  - Storage
  - Host OS

Network Devices | **Integrated NIC 1**

Integrated NIC 1: BRCM GbE 4P 5720-t rNDC

Port Properties

Vendor Name	Broadcom Corp
Number of Ports	4

Ports and Partitioned Ports

Link Status	Port	Partition	Protocol

Integrated NIC 1 Port 1 Partition 1

Link connection

Link Status	Up
Link Speed	1000 Mbps
OS Driver State	Operational
Auto Negotiation	Enabled

You can also find the MAC address here

MAC Addresses

MAC Addresses	✔ D4:AE:52:8B:72:8C	Virtual MAC Addresses	D4:AE:52:8B:72:8C

Port Properties

Family Firmware Version	7.0.48
Family Driver Version	15.0.19.0
PCI Device ID	165F

Settings and Capabilities

Wake On LAN	Capable
Management Pass Through	Capable
Energy Efficient Ethernet	Capable
Supported Boot Protocol	iSCSI, PXE

Warewulf Node Settings – Default Config

```
[matthew@moonshine ~]$ sudo wwctl profile set --yes --netdev eno1 --netmask 255.255.0.0 --gateway 10.0.0.1 "default"
```

These are the settings that will be applied to all compute nodes. (I know, you only have one)

All nodes will use interface eno1, the address of the head node's internal interface as their gateway to other networks (10.0.0.1), and a subnet mask of 255.255.0.0

Warewulf Node Settings – Specific Node Config

```
[matthew@moonshine ~]$ sudo wwctl node add --hwaddr D4:AE:52:8B:72:8C --ipaddr 10.0.0.2 moonshine01
Added node: moonshine01
```

Now we setup a profile for a particular node.

The node is identified by its MAC address (this is the address you noted previously).

We give it a name. Name yours with the cluster name followed by 01.

Warewulf Node Settings – Specific Node Config

```
[matthew@moonshine ~]$ sudo wwctl node add --hwaddr D4:AE:52:8B:72:8C
--ipaddr 10.0.0.2 moonshine01
Added node: moonshine01
```

The IP address, MAC address, and node name are the only things that will vary from node to node.

We are setting the node name and IP address. We are matching the MAC address.

- If you made a mistake you can delete the node with

wwctl node delete {node name}

Warewulf Node Settings – Specific Node Config

```
[matthew@moonshine warewulf]$ cat /etc/warewulf/nodes.conf
WW_INTERNAL: 45
nodeprofiles:
  default:
    comment: This profile is automatically included for each node
    container name: rocky-9
    network devices:
      default:
        device: eno1
        netmask: 255.255.0.0
        gateway: 10.0.0.1
nodes:
  moonshine01:
    profiles:
    - default
    network devices:
      default:
        hwaddr: d4:ae:52:8b:72:8c
        ipaddr: 10.0.0.2
```

These commands populate a text file (as with most configs in Linux)

Warewulf Node Settings – Specific Node Config

```
[matthew@moonshine warewulf]$ cat /etc/warewulf/nodes.conf
WW_INTERNAL: 45
nodeprofiles:
  default:
    comment: This profile is automatically included for each node
    container name: rocky-9
    network devices:
      default:
        device: eno1
        netmask: 255.255.0.0
        gateway: 10.0.0.1
nodes:
  moonshine01:
    profiles:
    - default
    network devices:
      default:
        hwaddr: d4:ae:52:8b:72:8c
        ipaddr: 10.0.0.2
```

Notice we added this to the default profile back when we built the boot image.

Example of modifying the Warewulf boot image

This is just an example – you can skip doing this. We will use the shell later.

```
[root@moonshine ~] wwctl container shell rocky-9
[rocky-9] Warewulf> yum install passwd
Rocky Linux 9 - BaseOS                                          463 kB/s | 2.2 MB     00:04
Rocky Linux 9 - AppStream                                       2.5 MB/s | 7.4 MB     00:02
Rocky Linux 9 - Extras                                           23 kB/s |  14 kB     00:00
Last metadata expiration check: 0:00:01 ago on Sat Mar 23 21:59:20 2024.
Package passwd-0.80-12.el9.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[rocky-9] Warewulf> passwd root
Changing password for user root.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[rocky-9] Warewulf> exit
exit

The Rocky Linux community provides updates for the latest point release of
Rocky Linux 9. If you need to remain on a specific point release (e.g., Rocky
Linux 9.2) you may want to engage with a commercial support provider for
long-term support.

https://rockylinux.org/support

+ dnf clean all
25 files removed
Rebuilding container...
```

As an example, we install the passwd program and use it to set the root password in the image.

Build the Warewulf boot image overlays

```
[root@moonshine warewulf]# wwctl overlay build
Building system overlays for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]: /var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img
Compressed image for overlay moonshine01/[wwinit]: /var/lib/warewulf/provision/overlays/moonshine01/__SYSTEM__.img.gz
Building runtime overlays for moonshine01: [generic]
Created image for overlay moonshine01/[generic]: /var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img
Compressed image for overlay moonshine01/[generic]: /var/lib/warewulf/provision/overlays/moonshine01/__RUNTIME__.img.gz
[root@moonshine warewulf]#
```

Containers are flexible because you can add layers of configuration.

The runtime layer gets reapplied every couple of minutes. You can use it to make configuration changes that will be picked up by all the compute nodes.

Check the node settings…

```
[matthew@moonshine warewulf]$ wwctl node list -a
 NODE        FIELD                    PROFILE  VALUE
 moonshine01 Id                       --       moonshine01
 moonshine01 Comment                  default  This profile is automatically included for each node
 moonshine01 ContainerName            default  rocky-9
 moonshine01 Ipxe                     --       (default)
 moonshine01 RuntimeOverlay           --       (generic)
 moonshine01 SystemOverlay            --       (wwinit)
 moonshine01 Root                     --       (initramfs)
 moonshine01 Init                     --       (/sbin/init)
 moonshine01 Kernel.Args              --       (quiet crashkernel=no vga=791 net.naming-scheme=v238)
 moonshine01 Profiles                 --       default
 moonshine01 PrimaryNetDev            --       (default)
 moonshine01 NetDevs[default].Type    --       (ethernet)
 moonshine01 NetDevs[default].OnBoot  --       (true)
 moonshine01 NetDevs[default].Device  default  eno1
 moonshine01 NetDevs[default].Hwaddr  --       d4:ae:52:8b:72:8c
 moonshine01 NetDevs[default].Ipaddr  --       10.0.0.2
 moonshine01 NetDevs[default].Netmask default  255.255.0.0
 moonshine01 NetDevs[default].Gateway default  10.0.0.1
 moonshine01 NetDevs[default].Primary --       (true)
[matthew@moonshine warewulf]$
```

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Setting up a Compute Node

Our tasks are:

- Setup IP routing on the head node so the compute node can use it to reach the internet.
- Wipe the Compute Node's harddrive. (We want to boot from the filesystem image stored on the head node – not the compute node's internal harddrive)
- Set the Compute Node to boot using PXE.
- Get the MAC address of the Compute Node's ethernet network card (so we know which compute node is ours)
- Configure the Warewulf settings for our compute node.
- Modify the Compute Node filesystem image and rebuild it.
- Boot the Compute Node over the network, login, and test the network.

Try rebooting the Compute Node

Try Booting the Compute Node

129.24.245.33

Chat | Keyboard | Screen Capture | Refresh | Full Screen | Connect Virtual Media | Disconnect Viewer | Console Controls

```
Configuring (net0 d4:ae:52:8b:72:8c)................... ok
net0: 10.0.1.2/255.255.252.0
net0: fe80::d6ae:52ff:fe8b:728c/64
Next server: 10.0.0.1
Filename: http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c
http://10.0.0.1:9873/ipxe/d4-ae-52-8b-72-8c... ok
d4-ae-52-8b-72-8c : 2638 bytes [script]

================================================================================

Warewulf v4 now booting: moonshine01 (d4:ae:52:8b:72:8c)

Container: rocky-9
Kernel: rocky-9 (container default)
KernelArgs: quiet crashkernel=no vga=791 net.naming-scheme=v238

Warewulf Controller: 10.0.0.1
Downloading Kernel Image:
http://10.0.0.1:9873/provision/d4%3Aae%3A52%3A8b%3A72%3A8c... ok
Downloading Container Image:
imgextract: command not found

Image extract not supported in this iPXE, using standard initrd mode
Downloading Container Image:
http://10.0.0.1:9873/provision/d4%3Aae%3A52%3A8b%3A72%3A8c...
```

Try Booting the Compute Node

```
Warewulf Node:        moonshine01
Container:            rocky-9
Kernelargs:           quiet crashkernel=no vga=791 net.naming-scheme=v238

Network:
    default: eno1
    default: 10.0.0.2/16
    default: d4:ae:52:8b:72:8c
moonshine01 login:
```

Try logging in over SSH

```
[matthew@moonshine ~]$ sudo -i
[sudo] password for matthew:

[root@moonshine ~]# ssh moonshine01
Last login: Mon Mar 25 08:15:51 2024 from 10.0.0.1

[root@moonshine01 ~]# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=8.87 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=56 time=9.20 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=56 time=8.88 ms
^C
--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 8.865/8.981/9.204/0.157 ms
[root@moonshine01 ~]#
```

Modifying Overlays

- We will add DNS nameservers to our Compute Node so we can use friendly host names

- First we will remove the warewulf template file (seems to be broken)
- Then import our own resolve.conf file into the wwinit overlay

- Then reboot the compute node so it picks up the change

Setup DNS with Warewulf Overlay

```
[root@moonshine etc]# cd /var/lib/warewulf/overlays/wwinit/rootfs/etc


[root@moonshine etc]# mv resolv.conf.ww resolve.conf.ww.backup

[root@moonshine etc]# wwctl overlay import wwinit /etc/resolv.conf
Building overlay for moonshine01: [wwinit]
Created image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img
Compressed image for overlay moonshine01/[wwinit]:
/var/lib/warewulf/provision/overlays/moonshine01/wwinit.img.gz
```

Setup DNS with Warewulf Overlay

```
[root@moonshine etc]# ssh moonshine01
Last login: Mon Mar 25 09:50:25 2024 from 10.0.0.1
[root@moonshine01 ~]# reboot
[root@moonshine01 ~]# Connection to moonshine01 closed by remote host.
Connection to moonshine01 closed.

[root@moonshine etc]# ping moonshine01
From moonshine (10.0.0.1) icmp_seq=133 Destination Host Unreachable
From moonshine (10.0.0.1) icmp_seq=134 Destination Host Unreachable
From moonshine (10.0.0.1) icmp_seq=135 Destination Host Unreachable
From moonshine (10.0.0.1) icmp_seq=136 Destination Host Unreachable
From moonshine (10.0.0.1) icmp_seq=137 Destination Host Unreachable
<a couple of minutes later>
64 bytes from moonshine01 (10.0.0.2): icmp_seq=138 ttl=64 time=183 ms
64 bytes from moonshine01 (10.0.0.2): icmp_seq=139 ttl=64 time=0.217 ms
64 bytes from moonshine01 (10.0.0.2): icmp_seq=140 ttl=64 time=0.270 ms
64 bytes from moonshine01 (10.0.0.2): icmp_seq=141 ttl=64 time=0.197 ms
```

Setup DNS with Warewulf Overlay

```
[root@moonshine ~]# ssh moonshine01
[root@moonshine01 ~]# cat /etc/resolv.conf
# Generated by NetworkManager
search hpc.unm.edu
nameserver 129.24.246.110
nameserver 129.24.246.118
[root@moonshine01 ~]# ping google.com
PING google.com (142.250.191.238) 56(84) bytes of data.
64 bytes from ord38s32-in-f14.1e100.net (142.250.191.238): icmp_seq=1 ttl=56 time=30.9 ms
64 bytes from ord38s32-in-f14.1e100.net (142.250.191.238): icmp_seq=2 ttl=56 time=30.8 ms
64 bytes from ord38s32-in-f14.1e100.net (142.250.191.238): icmp_seq=3 ttl=56 time=30.9 ms
64 bytes from ord38s32-in-f14.1e100.net (142.250.191.238): icmp_seq=4 ttl=56 time=31.0 ms
```

IP Addresses for the IB Network

- IB doesn't support VLANS so you are all on the same network.

- Questions:

1) Should you run opensm? No! "There can only be one!"
2) What should your IP addresses be for InfiniBand interfaces?

    Head node: 192.168.0.TeamID

    Compute Node: 192.168.1.TeamID

Setup Infiniband Network for the Cluster

1. Review the slides at the end of the HPC Networking lecture to make sure InfiniBand drivers are installed and there is basic connectivity.
2. Assign an IP address to the internal InfiniBand interface on the head node.
3. Configure warewulf to setup an InfiniBand interface on the compute node.
4. Test and benchmark the connection between the head node and the compute node using iperf.

Configure the head node IB interface

```
                    Edit Connection

        Profile name Internal IB
              Device 00:00:00:00:00:00:02:C9:03:00:35:80:91)

 INFINIBAND                                                  <Hide>
     Transport mode <Datagram>
                MTU                (default)

 IPv4 CONFIGURATION <Manual>                                 <Hide>
           Addresses 192.168.0.1/16              <Remove>
                     <Add...>
             Gateway
         DNS servers <Add...>
     Search domains <Add...>

             Routing (No custom routes) <Edit...>
 [X] Never use this network for default route
 [ ] Ignore automatically obtained routes
```

Configure the Head Node IB

Verify the IP address was associated with the IB interface. Masking 16 bits allow us to have the IP host address match the compute node number, i.e. moonshine01 will have address 192.168.1.1 and the head node has address 192.168.0.1

```
[matthew@moonshine ~]$ ip a
<snip>
6: ibp65s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2044 qdisc fq_codel state UP group default qlen 256
    link/infiniband 80:00:02:08:fe:80:00:00:00:00:00:00:00:02:c9:03:00:35:80:91 brd
00:ff:ff:ff:ff:12:40:1b:ff:ff:00:00:00:00:00:00:ff:ff:ff:ff
    inet 192.168.0.1/16 brd 192.168.0.255 scope global noprefixroute ibp65s0
       valid_lft forever preferred_lft forever
    inet6 fe80::202:c903:35:8091/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Configure the Head Node IB

Include the IB interface in the internal firewall zone.

```
[root@moonshine etc]# firewall-cmd --zone=internal --add-interface=ibp65s0 --permanent
The interface is under control of NetworkManager, setting zone to 'internal'.
success
```

Configure the Compute Node IB

Install infiniband drivers into the warewulf image. Notice we don’t install opensm.

```
[matthew@moonshine ~]$ sudo wwctl container shell rocky-9
Container image is rebuilt depending on the exit status of the last command.


Run "true" or "false" to enforce or abort image rebuild.
[rocky-9|discard] Warewulf> yum install rdma-core libibverbs-utils librdmacm librdmacm-utils ibacm infiniband-diags
Rocky Linux 9 - BaseOS                          1.7 MB/s | 2.3 MB     00:01
Rocky Linux 9 - AppStream                       7.4 MB/s | 8.0 MB     00:01
Rocky Linux 9 - Extras                         24 kB/s |  15 kB     00:00
Dependencies resolved.
================================================================================
 Package              Architecture   Version              Repository       Size
================================================================================
Installing:
 ibacm                x86_64         48.0-1.el9           baseos           88 k
 infiniband-diags     x86_64         48.0-1.el9           appstream       313 k

Rebuilding container...
Building container: rocky-9
Created image for VNFS container rocky-9: /var/lib/warewulf/provision/container/rocky-9.img
Compressed image for VNFS container rocky-9: /var/lib/warewulf/provision/container/rocky-9.img.gz
```

Configure the Compute Node IB

Add the network settings we want to apply to all the nodes with the default profile

```
[root@moonshine warewulf]# wwctl profile set --netdev ib0 --netmask 255.255.0.0 --netname=ibp65s0 "default"
Are you sure you want to modify 1 profile(s): y
```

Set the network configuration that is specific to moonshine01

```
[root@moonshine warewulf]# wwctl node set moonshine01 --ipaddr 192.168.1.1 --type=InfiniBand --netname=ibp65s0
Are you sure you want to modify 1 nodes(s)
```

Always check your commands had the intended effect

```
[root@moonshine warewulf]# wwctl node list -a

moonshine01  NetDevs[ib0].Type        --        InfiniBand
 moonshine01  NetDevs[ib0].OnBoot      --        (true)
 moonshine01  NetDevs[ib0].Device      default    ib0
 moonshine01  NetDevs[ib0].Ipaddr      --        192.168.1.1
 moonshine01  NetDevs[ib0].Netmask     default    255.255.0.0
```

Configure the Compute Node IB

```
root@moonshine warewulf]# ssh moonshine01
Last login: Mon Oct 21 08:29:26 2024 from 10.0.0.1
```

The keys we setup earlier and the usersync command allow us to login to the compute note from the root account of the head node without a password.

```
[root@moonshine01 ~]# wall "I'm going to reboot in a couple of minutes, so save your work. Matthew"

Broadcast message from root@moonshine01 (pts/0) (Mon Oct 21 08:31:46 2024):

I'm going to reboot in a couple of minutes, so save your work. Matthew
```

It’s polite to warn the other people on your server that you are going to reboot.

```
[root@moonshine01 ~]# shutdown -r +2
Reboot scheduled for Mon 2024-10-21 08:36:23 UTC, use 'shutdown -c' to cancel.
```

Configure the Compute Node IB

```
[root@moonshine01 ~]# ibstat
CA 'mlx4_0'
CA type: MT4099
Number of ports: 1
Firmware version: 2.10.700
Hardware version: 0
Node GUID: 0x0002c90300f67b70
System image GUID: 0x0002c90300f67b73
Port 1:
State: Active
Physical state: LinkUp
Rate: 40
Base lid: 12
LMC: 0
SM lid: 1
Capability mask: 0x02594868
Port GUID: 0x0002c90300f67b71
Link layer: InfiniBand
```

After reboot you should have a working IB interface.

Recall from the HPC Networking Lecture other ways to check that everything is working as expected.

Start iPerf Server on the Head Node

```
[root@moonshine warewulf]# yum install iperf3
```

Open the firewall ports of iPerf temporarily.

```
[root@moonshine etc]# firewall-cmd --zone=internal --permanent --add-port=5201/udp
success
[root@moonshine etc]# firewall-cmd --zone=internal --permanent --add-port=5201/tcp
success
[root@moonshine etc]# firewall-cmd --reload
success
```

```
[root@moonshine warewulf]# iperf3 -s -B 10.0.0.1
```

Listen on the ethernet interface first.

```
Server listening on 5201
```

Open a second terminal, login to the compute node, and install iPerf.

```
Last login: Mon Oct 21 01:00:03 on ttys004
matthew@Erebus ~ % ssh moonshine
Enter passphrase for key '/Users/matthew/.ssh/ssh_rsa_key':
Last login: Mon Oct 21 02:29:55 2024


[matthew@moonshine ~]$ sudo -i
[sudo] password for matthew:

[root@moonshine ~]# ssh moonshine01
Last login: Mon Oct 21 08:30:58 2024 from 10.0.0.1
[root@moonshine01 ~]#

[root@moonshine01 ~]# yum install iperf3
```

Open a second terminal, login to the compute node, and install iPerf.

```
[root@moonshine01 ~]# iperf3 -c 10.0.0.1
Connecting to host 10.0.0.1, port 5201
[  5] local 10.0.0.2 port 33640 connected to 10.0.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec   114 MBytes   958 Mbits/sec    0    375 KBytes
[  5]   1.00-2.00   sec   112 MBytes   938 Mbits/sec    0    375 KBytes
[  5]   2.00-3.00   sec   113 MBytes   945 Mbits/sec    0    375 KBytes
[  5]   3.00-4.00   sec   112 MBytes   938 Mbits/sec    0    375 KBytes
[  5]   4.00-4.71   sec  79.9 MBytes   941 Mbits/sec    0    375 KBytes
```

Now let’s test the InfiniBand Connection

```
[root@moonshine etc]# iperf3 -s -B 192.168.0.1
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

```
[root@moonshine01 ~]# iperf3 -c 192.168.0.1
Connecting to host 192.168.0.1, port 5201
[  5] local 192.168.1.1 port 35144 connected to 192.168.0.1 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  2.52 GBytes  21.6 Gbits/sec  2383    580 KBytes
[  5]   1.00-2.00   sec  2.53 GBytes  21.7 Gbits/sec  2508    671 KBytes
[  5]   2.00-3.00   sec  2.52 GBytes  21.7 Gbits/sec  2204    436 KBytes
[  5]   3.00-4.00   sec  2.52 GBytes  21.7 Gbits/sec  2255    377 KBytes
[  5]   4.00-5.00   sec  2.51 GBytes  21.5 Gbits/sec  1729    401 KBytes
[  5]   5.00-6.00   sec  2.51 GBytes  21.6 Gbits/sec  2148    589 Kbytes
```