



niklas.boelter@gwdg.de

Hendrik Nolte, Niklas Bölter

Node Provisioning Using Warewulf

Part I - Basics

Table of contents

- 1 Introduction to Compute Clusters
- 2 Introduction to Cluster Management
- 3 Deep Dive: Warewulf

Reminder: What is a Cluster?

- A cluster consists of a large number (even thousands) of nodes
- Usually you can group nodes into sub-clusters which are then only made of by nodes of the same type or/and purpose
- All nodes are typically connected via a certain network topology which takes the individual performance and security requirements into account



Figure: Image of Emmy, one of the systems of NHR@Göttingen.

Simple Example: Beowulf

- Original design dates back to 1996 at NASA
- Basic idea: Connect commodity hardware inside a private network
- Goal: Being able to run parallel programs, e.g. using a Message Passing Interface (MPI), to bring down wall-clock times
- One has two
 - ▶ types of nodes
 - ▶ types of network

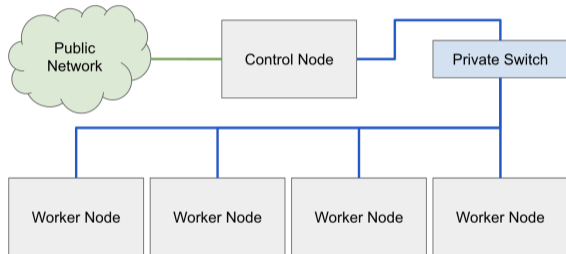


Figure: Simple sketch of the beowulf cluster.

<https://warewulf.org/docs/development/contents/background.html>

Interactive: What is the Challenge for Administrators

- What do you think is the challenge for administrators when you scale out such a cluster?
- How would you solve this?

Problem and Solution for Admins

- The problem is to manage all software stacks on all nodes when scaling out
- Solution: Unlike on your Laptop one does not use a local installation of the Kernel, OS and other software
- Instead all nodes fetch their images and related software from a single Control Node

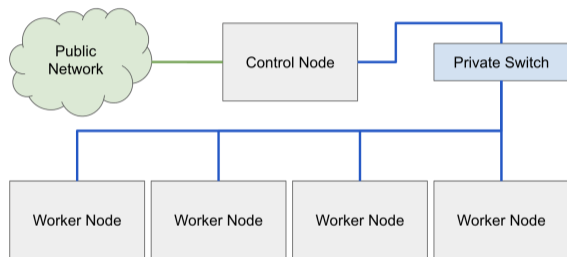


Figure: Simple sketch of the beowulf cluster.

<https://warewulf.org/docs/development/contents/background.html>

Problem and Solution for Admins

- This ensures a homogeneous state across all nodes
- And a admin only has to work on a single machine, not on thousands

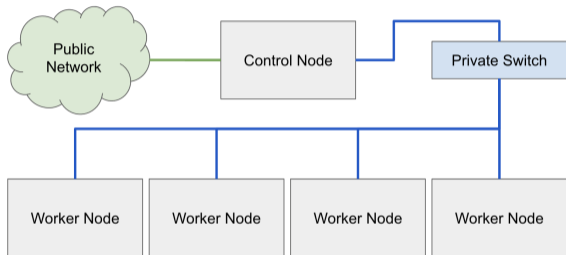


Figure: Simple sketch of the beowulf cluster.

<https://warewulf.org/docs/development/contents/background.html>

PXE Boot

- In order to enable the distribution of the system image from a single control node, nodes need to be able to fetch the OS before the actual boot process
- This is different from your laptop, where your UEFI loads a boot loader which then loads your locally installed OS
- The solution is called Preboot Execution Environment (PXE) boot

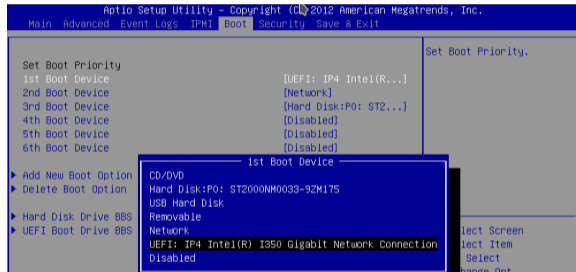


Figure: UEFI boot priorities needed to enable PXE boot.

<https://docs.mirantis.com/mcp/q4-18/mcp-deployment-guide/deployment-customizations-guidelines/boot-uefi-pxe.html>

Basic PXE Workflow

■ Prerequisites:

- ▶ PXE requires a DHCP and a TFTP server
- ▶ The client requires a PXE-capable network card and BIOS/UEFI setting
- ▶ Modern UEFI based systems can also boot via HTTP(S)

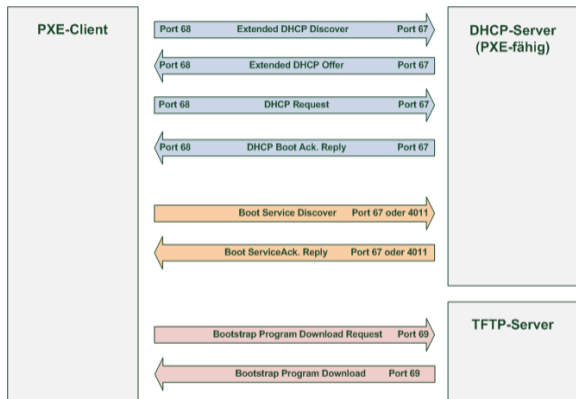


Figure: Simplified workflow of the PXE boot process. https://de.wikipedia.org/wiki/Preboot_Execution_Environment

Basic PXE Workflow

■ Workflow:

- 1 PXE-code on client configures client network setting, including
 - ip address
 - netmask
- 2 and additional PXE-specific information regarding the TFTP server and the bootloader
- 3 The downloaded bootloader is executed in RAM

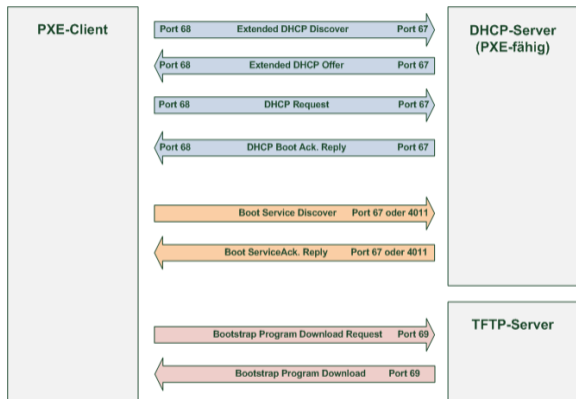


Figure: Simplified workflow of the PXE boot process. https://de.wikipedia.org/wiki/Preboot_Execution_Environment

Warewulf and iPXE-Boot

- Once the previously discussed PXE workflow ended, the Warewulf specific parts start:
 - 1 Using TFTP the iPXE stack is downloaded
 - 2 The iPXE is preconfigured, so that it can download the correct container, kernel modules, and system overlay via http
 - 3 iPXE executes the kernel and processes the system overlay
 - 4 The containers `/sbin/init` is being executed

Recap: What are we going to do

- We have stateless compute nodes (disk optional)
- And a stateful control node to manage the compute nodes
- We are doing a PXE-boot to fetch an iPXE stack
 - ▶ This step is necessary since pure PXE only allows you to download a **single** file via tftp
 - ▶ The overall complexity and fine tune capability of warewulf requires a bit more

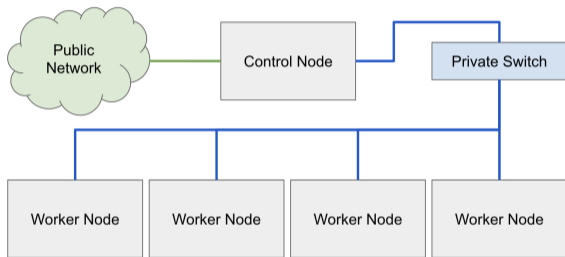


Figure: Simple sketch of the beowulf cluster.

<https://warewulf.org/docs/development/contents/background.html>

What Does Warewulf Do?

- Warewulf will be used for the basic PXE-boot and provides
 - ▶ the dhcp server
 - ▶ the tftp server
- and a stateful control node to manage the compute nodes
- We are doing a PXE-boot to fetch a iPXE stack
- Network settings, image, kernel are all managed by Warewulf

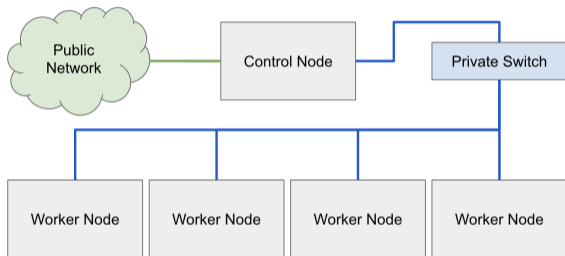


Figure: Simple sketch of the beowulf cluster.

<https://warewulf.org/docs/development/contents/background.html>

Warewulf Uses Containers?!?

- Yes, but not the containers you (probably) know
- We are still in HPC, thus the containers are booted on bare metal
- Most containers from Docker or Singularity have a lightweight systemd
 - ▶ That works, because your container is executed within a container runtime
- If it has a full fledged systemd, you can download and use the Container
- Same holds for core-utils
- You can still work with docker, buildah, kiwi, and many more tools to build your containers
- You can also fetch a base image and open a shell and modify/customize an image as you need
- It is basically an uncompressed chroot

Creating Containers from Scratch

- You can bootstrap a mini chroot directory

```
$ sudo yum install -installroot /tmp/newroot basesystem bash
chkconfig coreutils e2fsprogs ethtool filesystem findutils
gawk grep initscripts iproute iputils net-tools nfs-utils pam
psmisc rsync sed setup shadow-utils rsyslog tzdata util-linux
words zlib tar less gzip which util-linux openssh-clients
openssh-server dhclient pciutils vim-minimal shadow-utils
strace cronie crontabs cpio wget rocky-release ipmitool yum
NetworkManager
```

- Afterwards you can import it into warewulf:

```
$ sudo wwctl container import /tmp/newroot containername
```

Building a Container Using Singularity

- You can build a container "normal" with a recipe
- Warewulf offers a few templates, where one can add their own stuff in the post section
- The image modification then takes place using the sandbox-model:

```
$sudo aptainer build --sandbox /tmp/newroot  
/path/to/Singularity/recipe.def  
$sudo wwctl container import /tmp/newroot containername
```


Stateless Provisioning

- Provisioning: Putting an OS onto a system
- As mentioned, directly booting your OS without any installation
- In this case, this is done stateless, i.e. it is provisioned to memory
- Ongoing discussion about statefulness, e.g. having it on disk
 - ▶ Currently there is no need for non-volatile storage

Profiles

- In order to boot a node with warewulf, you need to configure it
- To help with that, you can group nodes which share the same attributes
- These groups are called Profiles

```
=====
NODE          FIELD          PROFILE      VALUE
-----
n1            Id             --           n1
n1            comment        default      This profile is automatically included for each
node
n1            cluster        --           --
n1            container      --           --
n1            ipxe            --           (default)
n1            runtime        --           (generic)
n1            wwinitt        --           (wwinitt)
n1            root           --           (lntramsf)
n1            discoverable   --           --
n1            init           --           (/sbin/init)
n1            asset          --           --
n1            kerneloverride --           --
n1            kernelargs     --           (quiet crashkernel=no vga=791 net.naming-scheme=
y238)
n1            ipniaddr       --           --
n1            ipninetmask    --           --
n1            ipniport       --           --
n1            ipnigateway    --           --
n1            ipnuser        --           --
n1            ipnlpass       --           --
n1            ipniinterface  --           --
n1            ipniwrite      --           --
n1            profile        --           default
n1            default:type   --           (ethernet)
n1            default:onboot --           --
n1            default:netdev --           (eth0)
n1            default:hwaddr --           --
n1            default:ipaddr --           --
n1            default:ipaddr6 --           --
n1            default:netmask --           (255.255.255.0)
n1            default:gateway --           --
n1            default:mtu    --           --
n1            default:primary --          true
```

Figure: Screenshot showing the node attributes which one can set

Live Demo

Any Questions?
Live Demo